Sergei Artemov
Anil Nerode (Eds.)

LNCS 7734

Logical Foundations of Computer Science

International Symposium, LFCS 2013
San Diego, CA, USA, January 2013
Proceedings

LF
CS
2013

Springer

Sergei Artemov   Anil Nerode (Eds.)

Logical Foundations of Computer Science

International Symposium, LFCS 2013
San Diego, CA, USA, January 6-8, 2013
Proceedings

Springer

Volume Editors

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

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

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-35721-3 e-ISBN 978-3-642-35722-0
DOI 10.1007/978-3-642-35722-0
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012954010

CR Subject Classification (1998): F.4.1, F.4, F.3, I.2.3, I.2.2, F.2

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

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

Printed on acid-free paper

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

Preface

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

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

The 2013 Symposium on Logical Foundations of Computer Science (LFCS 2013) took place in the Catamaran Resort Hotel, San Diego, California, during January 6–8. This volume contains the extended abstracts of talks selected by the Program Committee for presentation at LFCS 2013.

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

We thank the authors and reviewers for their contributions. We acknowledge the support of the U.S. National Science Foundation, the Graduate Center of the City University of New York, and the University of California, San Diego.

We are grateful to Yu Junhua for preparing this volume for Springer.

October 2012

Anil Nerode
Sergei Artemov

LFCS 2013 Organization

Steering Committee

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

Program Committee

Sergei Artemov	CUNY Graduate Center, New York, USA - Chair
Steve Awodey	Carnegie-Mellon University, Pittsburgh, USA
Alexandru Baltag	University of Amsterdam, The Netherlands
Andreas Blass	University of Michigan, Ann Arbor, USA
Samuel Buss	University of California, San Diego, USA
Walter Dean	University of Warwick, UK
Rod Downey	Victoria University of Wellington, New Zealand
Ruy de Queiroz	Universidade Federal de Pernambuco, Recife, Brazil
Antonio Montalban	University of Chicago, USA
Rosalie Iemhoff	Utrecht University, The Netherlands
Bakhadyr Khoussainov	The University of Auckland, New Zealand
Roman Kuznets	University of Bern, Switzerland
Robert Lubarsky	Florida Atlantic University, Boca Raton, USA
Victor Marek	University of Kentucky, Lexington, USA
Franco Montagna	Università Degli Studi di Siena, Italy
Lawrence Moss	Indiana University Bloomington, USA
Anil Nerode	Cornell University, Ithaca, USA
Mati Pentus	Moscow State University, Russia
Michael Rathjen	University of Leeds, UK
Jeffrey Remmel	University of California, San Diego, USA
Bryan Renne	University of Amsterdam, The Netherlands
Philip Scott	University of Ottawa, Canada
Alex Simpson	University of Edinburgh, UK
Sonja Smets	University of Groningen, The Netherlands
Alasdair Urquhart	University of Toronto, Canada
Michael Zakharyaschev	Birkbeck College, University of London, UK

Additional Reviewers

Laurent Bienvenu
Samuel Bucheli
Adam Chlipala
Alexander Clark
Chris Conidis
Giovanni Curi
Adam Day
Eduardo Fermé
Melvin Fitting
Jörg Flum
Nicola Gambino
William Gasarch
Konstantinos Georgatos
Giorgio Ghelli
Norihiro Kamide
Kohei Kishida
Hidenori Kurokawa
Andrew Lewis
Dan Licata
Iris Loeb
Glyn Morrill
Andre Nies
R. Ramanujam
Jan Reimann
Cody Roux
Kristina Sojakova
Mariya Soskova
Kazushige Terui
Sebastiaan Terwijn
Jan van Eijck

Table of Contents

Compositional Reasoning for Multi-modal Logics*

Luca Aceto[1], Anna Ingólfsdóttir[1], Cristian Prisacariu[2], and Joshua Sack[3]

[1] ICE-TCS, School of Computer Science, Reykjavik University, Reykjavik, Iceland
luca,annai@ru.is

[2] Dept. of Informatics – Univ. of Oslo, P.O.Box 1080 Blindern, 0316 Oslo, Norway
cristi@ifi.uio.no

[3] Institute of Logic, Language, and Computation – University of Amsterdam
joshua.sack@gmail.com

Abstract. We provide decomposition and quotienting results for multi-modal logic with respect to a composition operator, traditionally used for epistemic models, due to van Eijck et al. (Journal of Applied Non-Classical Logics 21(3–4):397–425, 2011), that involves sets of atomic propositions and valuation functions from Kripke models. While the composition operator was originally defined only for epistemic $S5^n$ models, our results apply to the composition of any pair of Kripke models. In particular, our quotienting result extends a specific result in the above mentioned paper by van Eijck et al. for the composition of epistemic models with disjoint sets of atomic propositions to compositions of any two Kripke models regardless of their sets of atomic propositions. We also explore the complexity of the formulas we construct in our decomposition result.

1 Introduction

Decomposition and quotienting techniques [2,9,15,23] have been used for a wide variety of logics, such as Hennessy-Milner logic [10] or modal μ-calculus [13], and much attention has been given to extending and optimizing these [2,14]. Compositional reasoning normally involves a parallel-like *composition operator* over the models of the logic in question. In the cases just cited, the main composition operator of interest is usually some form of parallel composition from process algebras [4,11,18,19]. In these cases, one observes what is called the *state explosion problem*; when a system is built up by composing several processes/components, its state space grows exponentially with the number of components. This is the main drawback of doing model checking of such systems (even for algorithms that are linear in the size of the *model* and the formula). Compositional reasoning has proved useful in tackling the state space explosion problem in several applications.

* Luca Aceto and Anna Ingólfsdóttir were partially supported by the project 'Processes and Modal Logics' (project nr. 100048021) of the Icelandic Research Fund. Joshua Sack was partially supported by the VIDI research programme number 639.072.904, which is financed by the Netherlands Organisation for Scientific Research.

Intuitively, considering some form of composition of models $M_1||M_2$ and a formula φ to check on this composed model, the technique of compositional reasoning provides an alternative to checking $M_1||M_2 \models \varphi$, by instead checking two potentially simpler problems: $M_1 \models \phi_1$ and $M_2 \models \phi_2$. When the two new formulas are not much larger than the original, this method can be very useful. There are also heuristic techniques that aim at keeping the new formulas small [2].

The aim of this paper is to develop a theory of compositionality and quotienting for multi-modal logic with respect to a composition operator that has been recently introduced in [22] for S5^n (epistemic) models. This composition behaves similarly to the well-known synchronous composition; however, while the set of states in a parallel composition is generally the Cartesian product, the composition between epistemic models introduced in [22] eliminates states whose atomic valuations on the components are not, so to speak, compatible.

Arguably, the composition of [22] is the most natural that one would want on S5^n models. This composition behaves similarly to the well-known synchronous composition of labelled transition systems. It is easy to see that the standard asynchronous composition that is normally studied in process algebras and concurrency theory does not preserve S5^n models (see e.g. [1]), whereas the synchronous composition does. Another observation is that unlike other types of frames (i.e., transition systems without a valuation of propositional constants), the S5^n frames are trivial without propositional constants and a valuation attached to their states (i.e., they are bisimilar to a single reflexive point). Therefore, a composition of S5^n models should take valuations and propositional constants into consideration.

Although originally defined for S5^n models, the composition of [22] is also well-defined on other classes of models. For example, the class of Kripke models is closed under it. An example of a class of models that is *not* closed with respect to the composition of [22] is that of KD45 models, often used to model belief. (See Remark 2.6.)

The involvement of valuations and propositional constants in compositions in general has received relatively little attention, and distinguishes the results in this paper from mainstream composition results [6,9,15,23]. There are, however, other compositions that use valuations and propositional constants, and there is work that employs related techniques. One composition that uses valuations is the concurrent program of [16], where two non-epistemic models are composed in such a way that the states of the composition may disagree with the components on the valuation. The composition we employ in this paper eliminates any state where there may be such disagreement between a composite state and its components. Another related composition is the update product from [5], though that composition is not between two Kripke models, but between a Kripke (or epistemic) model and an *action model*, a syntactic structure that differs from a Kripke model in that the valuation is replaced by a function assigning a formula to each point of the model. A composition result in the setting of transition systems that also involves pruning the global state space is that of [20]; however this result does not involve logic as we do. Furthermore, given that modal formulas characterize finite transition systems up to bisimulation, and synchronizing on common actions is similar to compatible states based on common valuations, there are connections between our techniques and the techniques for synchronizing up to bisimulation from [8].

Our most technically involved contribution is the proof strategy of a decomposition result (Th. 3.9) for the composition operator of [22]. This result follows naturally from the relationship between the primary composition of focus and an auxiliary composition (Th. 3.4). We also study the connections between the composition of models with overlapping sets of atomic propositions and compositions of models with disjoint sets of atomic propositions (Th. 5.5). Furthermore, we provide a quotienting theorem (Th. 4.3), which can be used to synthesize missing components in composite models. If we have a model N in the composition and want to construct M in order to achieve property φ for the composition of M and N, we can first compute the quotient formula of φ with respect to N and then synthesize a model for it, if one exists. We show in the proof of Corollary 5.6 that the quotienting result [22, Th. 16] involving only epistemic models with disjoint sets of atomic propositions is an instance of our quotienting result, and in Section 5.2, we discuss how to extend our primary decomposition result to one involving an even more general composition operator. Finally, in Section 6, we provide an analysis of the complexity of the formulas we construct in our main decomposition result. To save space, we omit or abbreviate a number of proofs, but make the full-length proofs available in [1].

2 Preliminaries

In what follows we assume a fixed finite set I of *labels* (also called *agents* in epistemic logic).

Definition 2.1 (Multi-modal Logic). *The* multi-modal logic $\mathcal{L}(\mathsf{P})$*, over a set* P *of propositional constants, is defined by the grammar:*

$$\phi := p\,(p \in \mathsf{P}) \mid \bot \mid \phi \vee \phi \mid \neg\phi \mid \langle i \rangle \varphi\,(i \in I).$$

The set P *is called the* vocabulary *of the logic. The formulas* $\phi_1 \wedge \phi_2$*,* $\phi_1 \leftrightarrow \phi_2$*, and* $[i]\phi$ *for* $i \in I$ *are derived in the standard way from this grammar, empty disjunctions identified with* $\bot$*, and* $\top$ *with* $\neg\bot$*.*

We are especially interested here in epistemic logics where the modality $[i]\varphi$ is usually read as: *agent* i *"knows" formula* φ, and is written $K_i\varphi$. But our work is applicable more generally, to multi-modal logics with propositional constants. We also want our notation to be close to both the epistemic logic community and the works on decomposition techniques.

The logic $\mathcal{L}(\mathsf{P})$ is interpreted over *(multi-modal) Kripke models.*

Definition 2.2 (Multi-modal Kripke Structure and Model).

– *A* (multi-modal) Kripke structure *is a tuple* $K = (W, \rightarrow)$ *where* W *is the set of worlds (also called states), and* $\rightarrow$ *is a family of relations* $\xrightarrow{i} \subseteq W \times W$ *indexed by a fixed set* I*. A* pointed (multi-modal) Kripke structure *is a pair* (K, w)*, where* $K = (W, \rightarrow)$ *and* $w \in W$*.*

- *A* multi-modal Kripke model *is a tuple $M = (W, \to, \mathsf{P}, V)$ where $(W, \to)$ is a Kripke structure,* P *is the set of propositional constants (i.e., the vocabulary of the model), and $V : W \to \mathcal{P}(\mathsf{P})$ is a valuation function. A model is* finite *if W and* P *are both finite. A* pointed (multi-modal) Kripke model *is a pair (M, w), where $M = (W, \to, \mathsf{P}, V)$ and $w \in W$.*

Definition 2.3 (Interpreting Multi-modal Logic). *The formulae in $\mathcal{L}(\mathsf{P})$ are interpreted in a Kripke model $M = (W, \to, \mathsf{P}, V)$ at some $w \in W$ as follows:*

- *$(M, w) \models p$ iff $p \in V(w)$,*
- *$(M, w) \models \phi_1 \vee \phi_2$ iff $(M, w) \models \phi_1$ or $(M, w) \models \phi_2$,*
- *$(M, w) \models \neg\phi$ iff it is not the case that $(M, w) \models \phi$ (abbreviated $(M, w) \not\models \phi$)*
- *$(M, w) \models \langle i \rangle \phi$ iff there exists a $w' \in W$ s.t. $w \xrightarrow{i} w'$ and $(M, w') \models \phi$.*

We read $(M, w) \models \varphi$ as: *"the formula φ holds/is true at state w in M"*. We may write $w \models \phi$ instead of $(M, w) \models \phi$ if the meaning is clear from the context.

2.1 Compositions of Models

Our paper is mainly concerned with the study of the interplay of the logic $\mathcal{L}(\mathsf{P})$ and the composition operator introduced in [22], which we will denote $\oslash$ and formally define in Definition 2.5. Essentially this composition makes a *synchronous* composition of the relations of the two models, but the new set of states is only *a subset of the Cartesian product* of the two initial sets of states. For later use, we redefine the restriction on states from [22] in terms of the notion of *(in)consistent states*. Though in [22] the operation $\oslash$ is defined over $S5^n$ models, it can actually be applied to arbitrary multi-modal Kripke models. Since our decomposition technique does not use the restrictions of the $S5^n$ models, it can be readily used over any class of multi-modal Kripke models that is closed under the operation of Definition 2.5; $S5^n$ models form one such class.

Definition 2.4 (Consistent States). *For two models $M = (W_M, \to_M, \mathsf{P}_M, V_M)$ and $N = (W_N, \to_N, \mathsf{P}_N, V_N)$, where P_M and P_N may overlap, we say that two states $w \in W_M$ and $v \in W_N$ are* inconsistent, *written $(M, w) \sharp (N, v)$, iff*

$$\exists p \in \mathsf{P}_M \cap \mathsf{P}_N : (p \in V_M(w) \text{ and } p \notin V_N(v)) \text{ or } (p \notin V_M(w) \text{ and } p \in V_N(v)).$$

We say that w and v are consistent, *written $(M, w) \diamond (N, v)$, iff the two states are not inconsistent. We often write $w \sharp v$ for $(M, w) \sharp (N, v)$ and $w \diamond v$ for $(M, w) \diamond (N, v)$ when the models are clear from the context.*

Definition 2.5 (Composition of Models [22]). *Let $M = (W_M, \to_M, \mathsf{P}_M, V_M)$ and $N = (W_N, \to_N, \mathsf{P}_N, V_N)$ be two finite models, with possibly overlapping vocabularies P_M and P_N. The composition of M and N is the finite model defined as $M \oslash N = (W, \to, \mathsf{P}_M \cup \mathsf{P}_N, V)$ with:*

- $W = \{(w, v) \mid w \in W_M, v \in W_N,$ *and* $w \diamond v\}$,
- $(w, v) \xrightarrow{i} (w', v')$ *iff* $w \xrightarrow{i}_M w'$ *and* $v \xrightarrow{i}_N v'$, *for* $(w, v), (w', v') \in W$ *and* $i \in I$, *and*
- $V((w, v)) = V_M(w) \cup V_N(v)$, *for* $(w, v) \in W$.

Note that, when the vocabularies are disjoint, the definition of $\diamond$ becomes vacuously true, whereas that of $\sharp$ is vacuously false. In this case, the above definition becomes the standard synchronous composition, where new states are from the full Cartesian product (as the requirement $w \diamond v$ can be ignored).

It was shown in [22, Th. 3] that the composition $\oslash$ endows the collection of epistemic S5^n with a commutative monoid structure, that is, up to total bisimilarity, the composition $\oslash$ is commutative, associative, and if E is the (epistemic $S5^n$) model with one point that is reflexive for every agent and has an empty set of atomic propositions, then E is a left and right unit for $\oslash$.

Remark 2.6. It is folklore from model theory that a sentence of first order logic is preserved under restriction and product if and only if the sentence is universal Horn. A universal Horn sentence of first-order logic is the universal closure of a disjunction with at most one atom disjunct, and where the remaining disjuncts are negations of atoms (see, e.g., [17]). The classes of S5 models and S5^n models are universal Horn: the formulas for reflexivity, symmetry and transitivity can be written as Horn formulas. Hence the collection of epistemic models must be closed under the composition $\oslash$. However, the class of KD45 models, often used to model belief, is not universal Horn, for the seriality requirement cannot be expressed as a universal Horn sentence. Although a property that is not expressible by a universal Horn might be preserved under *some* products and restrictions, one can easily check that KD45 is indeed not preserved under $\oslash$ (see e.g. [1]).

3 Compositional Reasoning Wrt. the $\oslash$ Composition

This section presents our main result, a general decomposition for $\mathcal{L}(\mathsf{P})$ with respect to $\oslash$, and which we describe as follows. We consider two finite models $M = (W_M, \rightarrow_M, \mathsf{P}_M, V_M)$ and $N = (W_N, \rightarrow_N, \mathsf{P}_N, V_N)$ and a formula $\phi \in \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N)$. Our aim is to find two formulas $\psi_1 \in \mathcal{L}(\mathsf{P}_M)$ and $\psi_2 \in \mathcal{L}(\mathsf{P}_N)$ such that

$$(M \oslash N, (w, v)) \models \phi \;\text{ iff }\; (M, w) \models \psi_1 \text{ and } (N, v) \models \psi_2.$$

We want ψ_1 and ψ_2 to depend only on φ, but for each φ there can actually be multiple candidate pairs of formulas (ψ_1, ψ_2). We thus follow the works on compositional reasoning for Hennessy-Milner logic [9], and reformulate the problem into finding a function $\chi : \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N) \rightarrow \mathcal{P}(\mathcal{L}(\mathsf{P}_M) \times \mathcal{L}(\mathsf{P}_N))$ such that

$$(M \oslash N, (w, v)) \models \phi \quad \text{iff} \quad \exists(\psi_1, \psi_2) \in \chi(\phi) : (M, w) \models \psi_1 \text{ and } (N, v) \models \psi_2.$$

Note that this function χ returns a subset of $\mathcal{L}(\mathsf{P}_M) \times \mathcal{L}(\mathsf{P}_N)$. This motivates the following definition, an auxiliary composition that we use to prove the main decomposition result of this section (Th. 3.9).

Definition 3.1 (Auxiliary Composition). *Let* $M = (W_M, \to_M, \mathsf{P}_M, V_M)$ *and* $N = (W_N, \to_N, \mathsf{P}_N, V_N)$ *be two finite models. The* auxiliary composition *of* M *and* N *is defined as the model* $M \odot N = (W, \to, \mathsf{P}, V)$ *(also written* $\begin{pmatrix} M \\ N \end{pmatrix}$*) with:*

- $W = W_M \times W_N$*, whose elements are also written* $\begin{pmatrix} w \\ v \end{pmatrix}$ *for* $(w, v) \in W_M \times W_N$*,*
- $\begin{pmatrix} w \\ v \end{pmatrix} \xrightarrow{i} \begin{pmatrix} w' \\ v' \end{pmatrix}$ *iff* $w \xrightarrow{i}_M w'$ *and* $v \xrightarrow{i}_N v'$*, for* $\begin{pmatrix} w \\ v \end{pmatrix}, \begin{pmatrix} w' \\ v' \end{pmatrix} \in W$ *and* $i \in I$*,*
- $\mathsf{P} = \mathcal{L}(\mathsf{P}_M) \times \mathcal{L}(\mathsf{P}_N)$*, whose elements are also written* $\begin{pmatrix} \psi_1 \\ \psi_2 \end{pmatrix}$ *for* (ψ_1, ψ_2)*,*
- $V((w, v)) = \{(\varphi, \psi) \in \mathsf{P} \mid (M, w) \models \varphi$ *and* $(N, v) \models \psi\}$*.*

As before, we may subscript the components with the model (such as by writing $\mathsf{P}_{M\odot N}$ for the set atomic propositions in $M \odot N$). The usual laws of multi-modal logic apply when determining the truth of a formula $\Phi \in \mathcal{L}(\mathsf{P}_{M\odot N})$ in a pointed model. For example, from the definition of $V_{M\odot N}$, we have, for $(\phi, \psi) \in \mathsf{P}_{M\odot N}$, that

$$\begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w \\ v \end{pmatrix} \models \begin{pmatrix} \phi \\ \psi \end{pmatrix} \quad \text{iff} \quad (M, w) \models \phi \text{ and } (N, v) \models \psi,$$

and, given $\Phi \in \mathcal{L}(\mathsf{P}_{M\odot N})$,

$$\begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w \\ v \end{pmatrix} \models \langle i \rangle \Phi \text{ iff } \begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w' \\ v' \end{pmatrix} \models \Phi \text{ for some } \begin{pmatrix} w' \\ v' \end{pmatrix} \text{ with } \begin{pmatrix} w \\ v \end{pmatrix} \xrightarrow{i} \begin{pmatrix} w' \\ v' \end{pmatrix}.$$

We may write $\begin{pmatrix} w \\ v \end{pmatrix} \models \Phi$ for $\begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w \\ v \end{pmatrix} \models \Phi$ if the model is clear from context.

3.1 Relationship between $\odot$ and $\oslash$

Our first step is to compare the compositions $\odot$ and $\oslash$. A primary difference between these two is that $\odot$ does not remove states that are considered inconsistent, while $\oslash$ does. We thus provide the following formulas in the language $\mathcal{L}(\mathsf{P}_{M\odot N})$ that characterize inconsistency and consistency:

$$\sharp_{M\odot N} = \bigvee_{p \in \mathsf{P}_M \cap \mathsf{P}_N} \left(\begin{pmatrix} p \\ \neg p \end{pmatrix} \vee \begin{pmatrix} \neg p \\ p \end{pmatrix} \right) \text{ and } \diamond_{M\odot N} = \neg \sharp_{M\odot N}. \tag{1}$$

Lemma 3.2. *For two finite pointed models* (M, w) *and* (N, v)*, we have*

$$\begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w \\ v \end{pmatrix} \models \sharp_{M\odot N} \quad \textit{iff} \quad (M, w) \sharp (N, v),$$

with the notation on the right taken from Definition 2.4.

We now define a "meaning preserving" translation of formulas to be evaluated on models composed using $\oslash$ to those evaluated on models composed using $\odot$. The correctness of this translation is given in Theorem 3.4.

Definition 3.3 (Translation Function). *We define* $Z : \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N) \to \mathcal{L}(\mathsf{P}_{M\odot N})$ *as follows:*

– $Z(p) = \begin{cases} \begin{pmatrix} p \\ p \end{pmatrix} \text{ if } p \in \mathsf{P}_M \cap \mathsf{P}_N, \\ \begin{pmatrix} p \\ \top \end{pmatrix} \text{ if } p \in \mathsf{P}_M \setminus \mathsf{P}_N \quad, \\ \begin{pmatrix} \top \\ p \end{pmatrix} \text{ if } p \in \mathsf{P}_N \setminus \mathsf{P}_M. \end{cases}$
– $Z(\phi_1 \vee \phi_2) = Z(\phi_1) \vee Z(\phi_2)$,
– $Z(\neg\phi) = \neg Z(\phi)$,
– $Z(\langle i \rangle \phi) = \langle i \rangle (Z(\phi) \wedge \diamond)$.

Theorem 3.4. *Let* $M = (W_M, \to_M, \mathsf{P}_M, V_M)$ *and* $N = (W_N, \to_N, \mathsf{P}_N, V_N)$ *be finite models and* $\phi \in \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N)$*. Then for all* $(w, v) \in W_{M\oslash N}$ *(i.e. such that* $w \diamond v$*)*

$$M \oslash N, (w, v) \models \phi \quad \textit{iff} \quad \begin{pmatrix} M \\ N \end{pmatrix}, \begin{pmatrix} w \\ v \end{pmatrix} \models Z(\phi).$$

Proof. We prove the statement by structural induction on ϕ. We only detail the case when $\phi = \langle i \rangle \phi_1$, for which $Z(\langle i \rangle \phi_1) = \langle i \rangle (Z(\phi_1) \wedge \diamond)$. We proceed as follows:

$(w, v) \models \langle i \rangle \phi_1$ iff (by the definition of $\models$)
$\exists (w', v') \in W_{M\oslash N} : (w, v) \xrightarrow{i} (w', v')$ and $(w', v') \models \phi_1$ iff (by induction)
$\exists w' \in W_M, \exists v' \in W_N : \begin{pmatrix} w \\ v \end{pmatrix} \xrightarrow{i} \begin{pmatrix} w' \\ v' \end{pmatrix}, w' \diamond v'$ and $\begin{pmatrix} w' \\ v' \end{pmatrix} \models Z(\phi_1)$ iff
(by Lemma 3.2)
$\exists w' \in W_M, \exists v' \in W_N : \begin{pmatrix} w \\ v \end{pmatrix} \xrightarrow{i} \begin{pmatrix} w' \\ v' \end{pmatrix}$ and $\begin{pmatrix} w' \\ v' \end{pmatrix} \models Z(\phi_1) \wedge \diamond$ iff
$\begin{pmatrix} w \\ v \end{pmatrix} \models \langle i \rangle (Z(\phi_1) \wedge \diamond)$. □

3.2 Decomposing Formulas

Recall from Theorem 3.4 that we relate the formula ϕ with a formula $Z(\phi)$ from $\mathcal{L}(\mathsf{P}_{M\odot N})$. We now proceed to show that any formula in $\mathcal{L}(\mathsf{P}_{M\odot N})$ is equivalent on $M \odot N$ to a disjunction of atomic propositions in $\mathsf{P}_{M\odot N}$.

Definition 3.5 (Disjunctive Normal Form in $\mathcal{L}(\mathsf{P}_{M\odot N})$). *The set of Disjunctive Normal Forms in* $\mathcal{L}(\mathsf{P}_{M\odot N})$*, written* $\mathbf{D}(\mathsf{P}_{M\odot N})$*, is defined as the smallest set such that:*

– $\mathcal{L}(\mathsf{P}_M) \times \mathcal{L}(\mathsf{P}_N) \subseteq \mathbf{D}(\mathsf{P}_{M\odot N})$;
– *if* $\Phi_1, \Phi_2 \in \mathbf{D}(\mathsf{P}_{M\odot N})$ *then* $\Phi_1 \vee \Phi_2 \in \mathbf{D}(\mathsf{P}_{M\odot N})$.

Note the difference between this definition and the standard notion of disjunctive normal form (DNF). The conjuncts that normally appear in a DNF are, in our case, part of the pairs (elements of $\mathsf{P}_{M\odot N}$), and similarly for the negation. Moreover, this is a DNF for modal formulas, and similarly the modality is part of the atomic pairs.. These are possible because of the following result.

Lemma 3.6 (Equivalences). *The following are valid on $M \odot N$.*

$$\neg\begin{pmatrix}\phi\\ \psi\end{pmatrix} \leftrightarrow \begin{pmatrix}\neg\phi\\ \top\end{pmatrix} \vee \begin{pmatrix}\top\\ \neg\psi\end{pmatrix}$$

$$\begin{pmatrix}\phi_1\\ \psi_1\end{pmatrix} \wedge \begin{pmatrix}\phi_2\\ \psi_2\end{pmatrix} \leftrightarrow \begin{pmatrix}\phi_1 \wedge \phi_2\\ \psi_1 \wedge \psi_2\end{pmatrix},$$

$$\langle i\rangle\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \leftrightarrow \begin{pmatrix}\langle i\rangle\psi_1\\ \langle i\rangle\psi_2\end{pmatrix}.$$

Definition 3.7. *We define a function $\mathbf{d} : \mathcal{L}(\mathsf{P}_{M\odot N}) \to \mathbf{D}(\mathsf{P}_{M\odot N})$ inductively as follows:*

– *If $\Phi \in \mathsf{P}_{M\odot N}$, then $\mathbf{d}(\Phi) = \Phi$.*
– *If $\Phi_1, \Phi_2 \in \mathcal{L}(\mathsf{P}_{M\odot N})$, then $\mathbf{d}(\Phi_1 \vee \Phi_2) = \mathbf{d}(\Phi_1) \vee \mathbf{d}(\Phi_2)$.*
– *If $\Phi \in \mathcal{L}(\mathsf{P}_{M\odot N})$ and $\mathbf{d}(\Phi) = \bigvee_{k\in K}\begin{pmatrix}\phi_k\\ \psi_k\end{pmatrix}$ then*
  - $\mathbf{d}(\langle i\rangle\Phi) = \bigvee_{k\in K}\begin{pmatrix}\langle i\rangle\phi_k\\ \langle i\rangle\psi_k\end{pmatrix}$,
  - $\mathbf{d}(\neg\Phi) = \bigvee\left\{\begin{pmatrix}\neg\bigvee_{i\in I}\phi_i\\ \neg\bigvee_{j\in K\setminus I}\psi_j\end{pmatrix} \mid I \subseteq K\right\}$.

The following result states that $\mathbf{d}$ preserves the semantics of the formulas.

Theorem 3.8. *For all $\Phi \in \mathcal{L}(\mathsf{P}_{M\odot N})$, $w \in W_M$ and $v \in W_N$,*

$$\begin{pmatrix}w\\ v\end{pmatrix} \models \Phi \quad \textit{iff} \quad \begin{pmatrix}w\\ v\end{pmatrix} \models \mathbf{d}(\Phi).$$

We are now ready for our main decomposition theorem.

Theorem 3.9. *Let $\chi : \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N) \to \mathcal{P}(\mathcal{L}(\mathsf{P}_M) \times \mathcal{L}(\mathsf{P}_N))$ be defined by mapping ϕ to the set of disjuncts in $\mathbf{d}(Z(\phi))$. Then*

$$(M \oslash N, (w,v)) \models \phi \quad \textit{iff} \quad \exists(\psi_1,\psi_2) \in \chi(\phi) : (M,w) \models \psi_1 \textit{ and } (N,v) \models \psi_2.$$

Proof. This result immediately follows from Theorems 3.4 and 3.8, and the definition of the semantics of disjunction. □

4 Quotienting

In this section, we present our quotienting result, which we describe as follows. Having a composed pointed model $(M \oslash N, (w,v))$ and a formula $\varphi \in \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N)$, we build a new formula, denoted $Q_{(N,v)}(\varphi)$, that depends explicitly only on one of the components, so that

$$(M \oslash N, (w,v)) \models \varphi \text{ iff } M, w \models Q_{(N,v)}(\varphi).$$

If for our logic and our composition operation $\oslash$, the resulting quotient formula is not significantly larger than the original formula and the component, then the model checking task can be simplified [2].

We show how $Q_{(N,v)}(\varphi)$ can be derived, by beginning with the following formula for consistency.

Definition 4.1 (Consistent with v). *Given a finite model $M = (W_M, \rightarrow_M, \mathsf{P}_M, V_M)$ and a finite pointed model $(N, v) = (W_N, \rightarrow_N, \mathsf{P}_N, V_N)$ with $v \in W_N$, we define $\diamond_v \in \mathcal{L}(\mathsf{P}_M \cap \mathsf{P}_N)$ as:*

$$\begin{aligned}\diamond_v = &\bigwedge\{p \mid p \in \mathsf{P}_M \cap \mathsf{P}_N,\ (N, v) \models p\}\\ &\wedge \bigwedge\{\neg p \mid p \in \mathsf{P}_M \cap \mathsf{P}_N,\ (N, v) \models \neg p\}.\end{aligned}$$

This definition essentially encodes the valuation of (N, v) over the common part of the vocabularies. Before, e.g. in Definition 5.2, $\diamond$ was encoding all possible valuations, because we did not know in advance the state v. The intuition now is that if $M, w \models \diamond_v$ then w and v are consistent in the same sense as before. Again, we can observe that $\diamond_v$ is a tautology when P_M and P_N are disjoint.

One can already see how for quotienting, the knowledge of one component (N, v) is used to build the quotient formula $Q_{(N,v)}(\varphi)$; whereas before we were taking all possibilities into account in the pairs of formulas.

Definition 4.2 (Modal Quotient Function). *For some set of propositional constants P_M and a finite pointed model (N, v), we define the function $Q_{(N,v)} : \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N) \rightarrow \mathcal{L}(\mathsf{P}_M)$ by*

- $Q_{(N,v)}(p) = \begin{cases} p \textit{ iff } p \in \mathsf{P}_M \setminus \mathsf{P}_N, \textit{ or both } p \in \mathsf{P}_M \cap \mathsf{P}_N \textit{ and } N, v \models p \\ \top \textit{ iff } p \in \mathsf{P}_N \setminus \mathsf{P}_M \textit{ and } N, v \models p \\ \bot \textit{ otherwise.} \end{cases}$,
- $Q_{(N,v)}(\phi_1 \vee \phi_2) = Q_{(N,v)}(\phi_1) \vee Q_{(N,v)}(\phi_2)$,
- $Q_{(N,v)}(\neg\phi) = \neg Q_{(N,v)}(\phi)$,
- $Q_{(N,v)}(\langle i\rangle\phi) = \langle i\rangle \bigvee_{v \xrightarrow{i} v'} (Q_{(N,v')}(\phi) \wedge \diamond_{v'})$.

Theorem 4.3. *For finite models $M = (W_M, \rightarrow_M, \mathsf{P}_M, V_M)$ and $N = (W_N, \rightarrow_N, \mathsf{P}_N, V_N)$, a formula $\varphi \in \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N)$, and two consistent states $w \diamond v$, we have*

$$M \oslash N, (w, v) \models \varphi \quad \textit{iff} \quad M, w \models Q_{(N,v)}(\varphi).$$

Proof (sketch). We prove the theorem by structural induction on φ where the base case for $\varphi = p$ follows directly from the definition and the inductive cases for $\varphi = \phi_1 \vee \phi_2$ and $\varphi = \neg\phi_1$ use simple induction arguments.

For the case of $\varphi = \langle i\rangle\phi_1$ the following are equivalent:

1. $M \oslash N, (w, v) \models \langle i\rangle\phi_1$
2. $\exists (w', v') \in W : (w, v) \xrightarrow{i} (w', v')$ and $M \oslash N, (w', v') \models \phi_1$
3. $\exists (w', v') \in W : (w, v) \xrightarrow{i} (w', v')$ and $M, w' \models Q_{(N,v')}(\phi_1)$

4. there exists w', such that $w \xrightarrow{i} w'$ and there exists v', such that $v \xrightarrow{i} v'$ and both $M, w' \models \diamond_{v'}$ and $M, w' \models Q_{(N,v')}(\phi_1)$
5. there exists w', such that $w \xrightarrow{i} w'$ and $M, w' \models \bigvee_{v \xrightarrow{i} v'}(Q_{(N,v')} \wedge \diamond_{v'})$
6. $M, w \models \langle i \rangle (\bigvee_{v \xrightarrow{i} v'}(Q_{(N,v')} \wedge \diamond_{v'}))$. □

An interesting corollary of Theorem 4.3 is that checking whether a pointed model (M, w) satisfies a formula φ can always be reduced to an equivalent model-checking question over the pointed model (E, v), where E is the left and right unit for the composition operator $\oslash$ and v is the only state of E.

Corollary 4.4. *For each finite model $M = (W_M, \rightarrow_M, \mathsf{P}_M, V_M)$, state $w \in W_M$ and formula $\varphi \in \mathcal{L}(\mathsf{P}_M)$, there is some formula $\psi \in \mathcal{L}(\emptyset)$ such that*

$$M, w \models \varphi \quad \textit{iff} \quad E, v \models \psi \ .$$

Proof. Recall that, by Theorem 3 in [22], E is a left unit for $\oslash$ modulo total bisimilarity. In fact, each state (v, w) in $E \oslash M$ is bisimilar to the state w in M. This means that the pointed models $(E \oslash M, (v, w))$ and (M, w) satisfy the same formulas in $\mathcal{L}(\mathsf{P}_M)$. By Theorem 4.3, we now have that, for each formula $\varphi \in \mathcal{L}(\mathsf{P}_M)$,

$$M, w \models \varphi \ \text{ iff } \ E \oslash M, (v, w) \models \varphi \ \text{ iff } \ E, v \models Q_{(M,w)}(\varphi).$$

By the definition of quotienting, it is easy to see that $Q_{(M,w)}(\varphi) \in \mathcal{L}(\emptyset)$. We may therefore take that formula as the ψ mentioned in the statement of the theorem. □

5 Related Results and Relationships

5.1 Composing with Disjoint Vocabularies

The results of this section show that the problem of determining the truth value of a formula in the composition of models with arbitrary (overlapping) vocabularies can be equivalently formulated in terms of composition of models with *disjoint* vocabularies.

We first provide functions that transform the models.

Definition 5.1. *For some model $M = (W, \rightarrow, \mathsf{P}_M, V_M)$ and $i \in \{1, 2\}$, we define $g_i(M) = (W, \rightarrow, \mathsf{P}_M \times \{i\}, V)$, where $V(w) = V_M(w) \times \{i\}$.*

Given any two sets A and B, we define their disjoint union $A + B$ to be $(A \times \{1\}) \cup (B \times \{2\})$. We now define formulas in $\mathcal{L}(\mathsf{P}_M + \mathsf{P}_N)$ that characterize when two states are consistent or inconsistent.

Definition 5.2. *Let P_M and P_N be finite vocabularies. We define the Boolean formulas:*

- $\sharp(\mathsf{P}_M + \mathsf{P}_N) = \bigvee_{p \in \mathsf{P}_M \cap \mathsf{P}_N}(((p, 1) \wedge \neg(p, 2)) \vee (\neg(p, 1) \wedge (p, 2)))$.
- $\diamond(\mathsf{P}_M + \mathsf{P}_N) = \neg \sharp(\mathsf{P}_M + \mathsf{P}_N)$.

When M and N are understood from context, we simply write $\sharp$ and $\diamond$ for $\sharp(\mathsf{P}_M + \mathsf{P}_N)$ and $\diamond(\mathsf{P}_M + \mathsf{P}_N)$ respectively.

Note the similarity of the definition for $\sharp_{M \odot N}$ and $\sharp(P_M + P_N)$. Because of the pairing of models and of formulas in the valuation $V_{M \odot N}$, we did not need the change of the common propositions, as we are doing here for $\sharp(P_M + P_N)$. Otherwise the definitions are the same.

Proposition 5.3. *Let $M = (W_M, \rightarrow_M, \mathsf{P}_M, V_M)$ and $N = (W_N, \rightarrow_N, \mathsf{P}_N, V_N)$ be two finite models. For all $w \in W_M$ and $v \in W_N$, we have*

$$g_1(M) \oslash g_2(N),\ (w,v) \models \sharp(\mathsf{P}_M + \mathsf{P}_N) \quad \textit{iff} \quad (M,w) \sharp (N,v).$$

Note that, by negating both sides of the above "iff", we have an equivalent formulation of the proposition with $\diamond$ in place of $\sharp$. We use the consistency Boolean formula $\diamond$ to rewrite a multi-modal formula that is defined over two possibly overlapping vocabularies, into a multi-modal formula over the two disjoint vocabularies of the corresponding models changed by the functions g_i from above.

Definition 5.4 (Function $f_{(\mathsf{P}_M,\mathsf{P}_N)}$). *For two sets of propositional constants $\mathsf{P}_M, \mathsf{P}_N$, we define a function $f_{(\mathsf{P}_M,\mathsf{P}_N)} : \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N) \rightarrow \mathcal{L}(\mathsf{P}_M + \mathsf{P}_N)$ as follows:*

- $f_{(\mathsf{P}_M,\mathsf{P}_N)}(p) = \begin{cases} (p,1) \wedge (p,2) & p \in \mathsf{P}_M \cap \mathsf{P}_N, \\ (p,1) & p \in \mathsf{P}_M \setminus \mathsf{P}_N \\ (p,2) & p \in \mathsf{P}_N \setminus \mathsf{P}_M. \end{cases}$
- $f_{(\mathsf{P}_M,\mathsf{P}_N)}(\neg\phi) = \neg f_{(\mathsf{P}_M,\mathsf{P}_N)}(\phi)$.
- $f_{(\mathsf{P}_M,\mathsf{P}_N)}(\phi_1 \vee \phi_2) = f_{(\mathsf{P}_M,\mathsf{P}_N)}(\phi_1) \vee f_{(\mathsf{P}_M,\mathsf{P}_N)}(\phi_2)$.
- $f_{(\mathsf{P}_M,\mathsf{P}_N)}(\langle i \rangle \phi) = \langle i \rangle (f_{(\mathsf{P}_M,\mathsf{P}_N)}(\phi) \wedge \diamond)$.

The functions $g_1(M)$ and $g_2(N)$ produce models with the same structure but with disjoint vocabularies, thus the following is the result we are looking for.

Theorem 5.5. *Given any finite pointed models (M,w) and (N,v), such that $w \diamond v$, and any formula $\varphi \in \mathcal{L}(\mathsf{P}_M \cup \mathsf{P}_N)$,*

$$M \oslash N, (w,v) \models \varphi \quad \textit{iff} \quad g_1(M) \oslash g_2(N), (w,v) \models f_{(\mathsf{P}_M,\mathsf{P}_N)}(\varphi).$$

Proof (sketch). We use induction on the structure of the formula φ. The base case for $\varphi = p$ follows from the definition of the satisfiability relation $\models$, the definition of f, and the fact that $w \diamond v$. The inductive cases for $\varphi = \phi_1 \vee \phi_2$ and $\varphi = \neg\phi_1$ use simple inductive arguments. The crux of the case of $\varphi = \langle i \rangle \phi$ is the fact that $f(\langle i \rangle \phi) = \langle i \rangle (f(\phi) \wedge \diamond)$ is defined using the $\diamond$ formula inside the $\langle i \rangle$ modality. This ensures that the induction goes through. Essentially it guarantees that in the composition of the transformed models $g_1(M) \oslash g_2(N)$, we focus on the consistent states (w', v') that are reached from (w,v) in $M \oslash N$, thus looking only at states that correspond to those in $M \oslash N$. $\square$

5.2 Special Instances and Extensions

In this section, we show that our quotienting result generalizes Theorem 16 from [22], and then we discuss how to extend our decomposition result (Th. 3.9) to one involving a more general composition operator described in [22, Remark 2].

Corollary 5.6 (for Th.16 of [22]). *Let (M_i, w_i), for $i \in \{1, \ldots, n\}$, be pointed models such that the P_{M_i} are pairwise disjoint. Then for any $\varphi \in \mathcal{L}(\mathsf{P}_{M_i})$, $i \in \{1, \ldots, n\}$, we have that*

$$(M_1 \oslash \cdots \oslash M_n), (w_1, \ldots, w_n) \models \varphi \quad \textit{iff} \quad M_i, w_i \models \varphi \ .$$

Proof (sketch). This is an easy corollary of Theorem 4.3. Because $\oslash$ is commutative and associative, we can assume without loss of generality that $i = 1$. Let (N, v) be the pointed model $((M_2 \oslash \cdots \oslash M_n), (w_2, \ldots, w_n)$. Note that $\mathsf{P}_{M_1} \cap \mathsf{P}_N = \emptyset$. Now, $\varphi \in \mathcal{L}(\mathsf{P}_1)$, and hence $Q_{(N,v)}(p) = p$. The disjointness of the vocabularies ensures that $\diamond_v$ is always equivalent to $\top$. A simple induction on the structure of the input formula shows that $Q_{(N,v)}(\varphi)$ is equivalent to φ itself. The desired theorem then immediately results from Theorem 4.3. □

Compositional Reasoning wrt. a Generalized $\oslash$ Composition: Our decomposition method (and the proofs) can be easily adapted to other settings. One is the application to compositional reasoning with respect to a generalization of the $\oslash$ operator, remarked in [22, Remark 2].

Definition 5.7 (Generalized $\oslash$ Composition). *The* modal depth *of a formula φ is the maximum nesting of $\langle i \rangle$, $i \in I$, occurring in it. For each $n \geq 0$, and set of propositional constants P, we write $\mathcal{L}_n(\mathsf{P})$ for the collection of formulas in $\mathcal{L}(\mathsf{P})$ whose modal depth is at most n.*

Take the definition of $\diamond_0$ to be that of $\diamond$ from Definition 2.4. Define $\diamond_n$ to be the same as $\diamond$ only that instead of requiring agreement on the set of propositional constants $\mathsf{P}_M \cap \mathsf{P}_N$, we ask consistent states to satisfy the same formulas in $\mathcal{L}_n(\mathsf{P}_M \cap \mathsf{P}_N)$. Define the general composition operator $\oslash_n$ over finite models to be the same as $\oslash$ in Definition 2.5 but with $\diamond$ replaced by $\diamond_n$.

Note that $\oslash_0$ is the same as $\oslash$. All the proofs from Section 3 work for any of the generalized compositions $\oslash_n$. We only need to adapt the definitions of the $\diamond$ formulas to be in terms of the languages $\mathcal{L}_n(\mathsf{P}_M \cap \mathsf{P}_N)$. These languages are infinite. However, since $\mathsf{P}_M \cap \mathsf{P}_N$ is finite, if we quotient $\mathcal{L}_n(\mathsf{P}_M \cap \mathsf{P}_N)$ by the equivalence relation identifying every two formulas φ and ψ whenever $\varphi \leftrightarrow \psi$ is valid, then we are left with a finite language. Therefore the formula $\diamond_n$ can be expressed in $\mathcal{L}_n(\mathsf{P}_M \cap \mathsf{P}_N)$.

Another application of the decomposition method is to dynamic epistemic logic (DEL) [21]. One approach is to use reductions of DEL to the epistemic logic that we treated (see e.g. [21]) and then our Theorem 3.9. Another interesting way is to use the logical language of DEL directly in our decomposition technique and use a result from [22, Th.18] (only for *propositionally differentiated* action models). We leave for future work the development of decomposition and quotienting results that apply directly to DEL.

6 Complexity Issues

In this section, we investigate how the decomposition operator $\oslash$ affects size (which we call dimension) of the models being composed, and how the transformations Z and $\mathbf{d}$ affect the size (dimension) of the formulas. We also point out some techniques for optimizing these, though we leave the pursuit of these techniques for future work.

In what follows, for any finite set S, we denote the number of elements of S by $|S|$. Let the *dimension* of a finite model $M = (W, \rightarrow, \mathsf{P}, V)$ be $|W| + |P| + \sum_{i \in I} | \xrightarrow{i} | + \sum_{p \in \mathsf{P}} |V(p)|$. Given two models M and N, if P_M and P_N are disjoint, then the dimension of $M \oslash N$ is much larger than the sum of the dimensions of the components. In this case, the sizes of the components of the composed model $M \oslash N$ are as follows:

- $|W_{M \oslash N}| = |W_M| \times |W_N|$,
- $| \xrightarrow{i}_{M \oslash N} | = | \xrightarrow{i}_M | \times | \xrightarrow{i}_N |$,
- $|\mathsf{P}_{M \oslash N}| = |\mathsf{P}_M| + |\mathsf{P}_N|$,
- $|V_{M \oslash N}(p)| = \begin{cases} |V_M(p)| \times |W_N| \text{ if } p \in \mathsf{P}_M \\ |M_N| \times |V_N(p)| \text{ if } p \in \mathsf{P}_N \end{cases}$.

The first two equalities hold also with respect to the synchronous parallel composition between M and N. The other two are perhaps less familiar, and a bit more complicated. But clearly, the dimension of the composition $M \oslash N$ is much larger than the sum of the dimensions of the M and N when the vocabularies are disjoint. If the vocabularies of M and N are not disjoint or even coincide, the situation is more complicated. It is possible that the formulas are the same as above if the valuations of both models are uniform, providing each state with the same valuation. But it is also possible that some or even all the states be removed when eliminating the "inconsistent" states from the composition (such as when M and N have uniform valuations, but disagree on each atomic proposition), in which case the dimension of the composition can be much smaller. The techniques of this paper are most useful when the dimension of the composition is much larger than the dimension of the parts, and where the formula translations do not increase the complexity of the formula too much.

As usual, we consider the complexity of a formula φ to be the number of occurrences of symbols in it, and call this its *dimension.* The formulas in the decomposition result are built in two stages, first using the function Z in Definition 3.3 and then generating the DNF of the resulting formula using the function $\mathbf{d}$ in Definition 3.7. For Z we use the Boolean formula $\diamond_{M \odot N}$ from (1).

Proposition 6.1 (Dimension of $\diamond$). *The dimensions of $\sharp_{M \odot N}$ and $\diamond_{M \odot N}$ from (1) are linear in the size of $\mathsf{P}_M \cap \mathsf{P}_N$. The dimension of the DNF of $\diamond_{M \odot N}$ is exponential in the size of $\mathsf{P}_M \cap \mathsf{P}_N$.*

Since the dimension of $\diamond$ depends only on the (propositional vocabularies of the) models that are composed, we view it as a constant when calculating the dimension of the formula generated by Z with respect to the input formula.

Proposition 6.2 (Dimension of Z). *The dimension of the formula $Z(\varphi)$ from Definition 3.3 is linear in the size of the input formula φ.*

To calculate the dimension of the formulas in disjunctive normal form, resulting from the function $\mathbf{d}$ in Definition 3.7, applied to formulas $Z(\varphi)$, we involve a notion of *disjunctive dimension*; this is the number of disjuncts in a DNF.

Definition 6.3 (Disjunctive Dimension). *For a formula Φ in $\mathbf{D}(\mathsf{P}_{M\odot N})$, the* disjunctive dimension*, denoted $\delta(\Phi)$, is defined to be the number of occurrences in Φ of elements from $\mathsf{P}_{M\odot N}$.*

Note that the dimension of a formula in $\mathbf{D}(\mathsf{P}_{M\odot N})$ is at least as large as its disjunctive dimension.

Proposition 6.4. *Let $\Phi \in \mathcal{L}(\mathsf{P}_{M\odot N})$ be a formula with a nesting of $k+1$ ($k \geq 0$) negation symbols. Then*

$$\delta(\mathbf{d}(\Phi)) \geq \left. 2^{\cdot^{\cdot^{\cdot^{2}}}} \right\} k \text{ occurrences of } 2.$$

For calculating the disjunctive dimension of $\mathbf{d}$ applied to $Z(\varphi)$ in terms of the dimension of φ, observe that Z introduces, for every occurrence of a modal operator $\langle i \rangle$ in φ, a conjunction symbol, which is an abbreviation for an expression with negation symbols. Furthermore, for a nesting of $k > 0$ modal operators $\langle i \rangle$, Z introduces a nesting of $2k$ negation operators, and hence by Proposition 6.4, the disjunctive dimension of $\mathbf{d}(Z(\varphi))$ is at least a tower of $2k-1$ exponents. As the disjunctive dimension is a lower bound to the actual dimension, this means that the dimension of $\mathbf{d}(Z(\varphi))$ is at least a tower of $2k-1$ exponents.

To reduce these dimensions, one may investigate the use of *term graphs* (see [12] or [3, Sec. 4.4]) to identify repeated subformulas. One may also consider representing formulas as *binary decision diagrams* (see [7]). A direct method could be to process φ or $Z(\varphi)$, so as to remove double negations, or to identify patterns of negation and disjunction that allow us to apply the conjunctive item of Lemma 3.6. Furthermore, each step of the translation reduction methods in [2] could be applied to eliminate redundant formulas by simple Boolean evaluations.

References

1. Aceto, L., Ingolfsdottir, A., Prisacariu, C., Sack, J.: Compositional reasoning for multi-modal logics. Technical Report 419, Department of Informatics, University of Oslo, Oslo, Norway (September 2012)
2. Andersen, H.R.: Partial model checking (extended abstract). In: 10th IEEE Symposium on Logic in Computer Science (LICS 1995), pp. 398–407. IEEE Computer Society (1995)
3. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
4. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process Algebra: Equational Theories of Communicating Processes. Cambridge Tracts in Theoretical Computer Science 50. Cambridge University Press (2009)

5. Baltag, A., Moss, L.S.: Logics for epistemic programs. Synthese 139(2), 165–224 (2004)
6. Bloom, B., Fokkink, W., van Glabbeek, R.J.: Precongruence formats for decorated trace semantics. ACM Transactions on Computational Logic 5(1), 26–78 (2004)
7. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers 35(8), 677–691 (1986)
8. Castellani, I., Mukund, M., Thiagarajan, P.S.: Synthesizing Distributed Transition Systems from Global Specifications. In: Pandu Rangan, C., Raman, V., Ramanujam, R. (eds.) FST TCS 1999. LNCS, vol. 1738, pp. 219–231. Springer, Heidelberg (1999)
9. Fokkink, W., van Glabbeek, R.J., de Wind, P.: Compositionality of Hennessy-Milner logic by structural operational semantics. Theoretical Computer Science 354(3), 421–440 (2006)
10. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of ACM 32(1), 137–161 (1985)
11. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall (1985)
12. Kozen, D.: Complexity of finitely presented algebras. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, pp. 164–177. ACM (1977)
13. Kozen, D.: Results on the propositional mu-calculus. Theoretical Computer Science 27, 333–354 (1983)
14. Laroussinie, F., Larsen, K.G.: Compositional Model Checking of Real Time Systems. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 27–41. Springer, Heidelberg (1995)
15. Larsen, K.G., Xinxin, L.: Compositionality through an operational semantics of contexts. Journal of Logic and Computation 1(6), 761–795 (1991)
16. Lomuscio, A., Raimondi, F.: The complexity of model checking concurrent programs against CTLK specifications. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 548–550. ACM (2006)
17. McNulty, G.F.: Fragments of first order logic, I: Universal Horn logic. The Journal of Symbolic Logic 42(2), 221–237 (1977)
18. Milner, R.: Calculi for synchrony and asynchrony. Theorethical Computer Science 25, 267–310 (1983)
19. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
20. Mohalik, S., Ramanujam, R.: A presentation of regular languages in the assumption - commitment framework. In: Proceedings of the 1998 International Conference on Application of Concurrency to System Design, ACSD 1998, pp. 250–260. IEEE Computer Society, Washington, DC (1998)
21. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer (2007)
22. van Eijck, J., Sietsma, F., Wang, Y.: Composing models. Journal of Applied Non-Classical Logics 21(3-4), 397–425 (2011)
23. Winskel, G.: A Complete System for SCCS with Modal Assertions. In: Maheshwari, S.N. (ed.) FSTTCS 1985. LNCS, vol. 206, pp. 392–410. Springer, Heidelberg (1985)

Explicit Generic Common Knowledge

Evangelia Antonakos

Graduate Center CUNY

Abstract. The name Generic Common Knowledge (*GCK*) was suggested by Artemov to capture a state of a multi-agent epistemic system that yields iterated knowledge $I(\varphi)$: 'any agent knows that any agent knows that any agent knows... φ' for any number of iterations. The generic common knowledge of φ, $GCK(\varphi)$, yields $I(\varphi)$,

$$GCK(\varphi) \to I(\varphi)$$

but is not necessarily logically equivalent to $I(\varphi)$. Modal logics with *GCK* were suggested by McCarthy and Artemov. It has been shown that in the usual epistemic scenarios, *GCK* can replace the conventional common knowledge. Artemov noticed that such epistemic actions as public announcements of atomic sentences, generally speaking, yield *GCK* rather than the conventional common knowledge. In this paper we introduce logics with explicit *GCK* and show that they realize corresponding modal systems, i.e., *GCK*, along with the individual knowledge modalities, can be always made explicit.

Keywords: generic common knowledge, common knowledge, justification logic, epistemic modal logic, realization.

1 Introduction

Common knowledge C is perhaps the most studied form of shared knowledge. It is often cast as equivalent to iterated knowledge I, "everyone knows that everyone knows that..." [9,12]. However there is an alternate view of common knowledge, generic common knowledge (*GCK*), which has advantages. The characteristic feature of *GCK* is that it implies, but not equivalent to, iterated knowledge I. Logics with this type of common knowledge have already been seen ([7,14,15]) but this new term "*GCK*" clarifies this distinction ([3]). Generic Common Knowledge can be used in many situations where C has traditionally been used ([2,5,3]) and has a technical asset in that the cut rule can be eliminated.[1]

Moreover, Artemov pointed out in [3] that public announcements of atomic sentence – a prominent vehicle for attaining common knowledge – generally speaking, leads to *GCK* rather than to the conventional common knowledge. Artemov also argues in [5] that in the analysis of perfect information games in the belief revision setting, Aumann's "no irrationality in the system" condition

[1] See details in [1] as to why the finitistic cut-elimination in traditional common knowledge systems may be seen as unsatisfactory.

is fairly represented by some kind of generic common knowledge rather than conventional common knowledge, and that this distinction lies in the heart of the well-known Aumann–Stalnaker controversy.

We assume that the aforementioned arguments provide sufficient motivation for mathematical logical studies of the generic common knowledge and its different forms.

Another research thread we consider is Justification Logic. In the generative justification logic LP, logic of proofs, knowledge and reasoning are made explicit with proof terms representing evidence for facts and new logic atoms $t{:}F$ are introduced with the reading "t is (sufficient) evidence for knowing F" or simply "t is a proof of F."

In this paper we consider justification logic systems with multiple knowers and generic common knowledge. As the standard example, we assume that all knowers as well as their *GCK* system are confined to LP. We call the resulting system $\mathsf{LP}_n(\mathsf{LP})$ which symbolically indicates n LP-type agents with an LP-type common knowledge evidence system.

Multi-agent justification logic systems were first considered in [18], but without any common knowledge component. Systems with the explicit equivalent of the traditional common knowledge were considered in [11,10]; capturing common knowledge explicitly proved to be a serious technical challenge and the desirable realization theorem has not yet been obtained.

Generic common knowledge in the context of modal epistemic logic, in which individual agents' knowledge is represented 'implicitly' by the standard epistemic modalities was considered by Artemov in [7]. In the resulting modal epistemic logic $\mathsf{S4}_n^J$, sentences may be known, but specific reasons are not. This is a multi-agent logic augmented with a *GCK* operator J (previously termed justified common knowledge in [7] and elsewhere). Artemov reconstructed $\mathsf{S4}_n^J$-derivations in $\mathsf{S4}_n\mathsf{LP}$ via a Realization algorithm which makes the generic common knowledge operator J explicit, but does not realize individual knowledge modalities.

The current paper takes a natural next step by offering a realization of the entire *GCK* system $\mathsf{S4}_n^J$ in the corresponding explicit knowledge system $\mathsf{LP}_n(\mathsf{LP})$, In particular, all epistemic operators in $\mathsf{S4}_n^J$, not only J, become explicit in such a realization.

2 Explicit Epistemic Systems with *GCK*

Here we introduce an explicit generic common knowledge operator into justification logics in the context of a multi-agent logic of explicit justifications to form a logic $\mathsf{LP}_n(\mathsf{LP})$. The "$(\mathsf{LP})$" corresponds to *GCK*.

Definition 1. $\mathcal{L}_{\mathsf{LP}_n(\mathsf{LP})}$*, the* language of $\mathsf{LP}_n(\mathsf{LP})$*, is an extension of the propositional language:*

$$\mathcal{L}_{\mathsf{LP}_n(\mathsf{LP})} := \{\, \mathit{Var},\ \mathit{pfVar},\ \mathit{pfConst}, \vee, \wedge, \rightarrow, \neg, +, \cdot, !, \mathit{Tm}\} \ .$$

Var is propositional variables $(p, q, \dots)$. Justification terms *Tm are built from pfVar and pfConst, proof variables $(x, y, z, \dots)$ and constants $(c, d, \dots)$, by the grammar*

$$t := x \mid c \mid t + t \mid t{\cdot}t \mid !t \ .$$

Formulas *(Fm) are defined by the grammar, for* $i \in \{0, 1, 2, \dots, n\}$,

$$\varphi := p \mid e \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid \neg\varphi \mid t{:}_i\,\varphi \ .$$

The formulas $t{:}_i\,\varphi$ have the intended reading of "t is a justification of φ for agent i." Index $i = 0$ is reserved for explicit generic common knowledge, for which we will also use the alternative notation $[t]\varphi$ for better readability.

Definition 2. *The* axioms and rules of $\mathsf{LP}_n(\mathsf{LP})$*:*

CLASSICAL PROPOSITIONAL LOGIC:
A. axioms of classical propositional logic
R. modus ponens
LP AXIOMS FOR ALL $n+1$ AGENTS, $i \in \{0, 1, 2, \dots, n\}$:
L1. $t{:}_i\,(\varphi \rightarrow \psi) \rightarrow (s{:}_i\,\varphi \rightarrow (t{\cdot}s){:}_i\,\psi)$
L2. $t{:}_i\,\varphi \rightarrow (t+s){:}_i\,\varphi$ *and* $t{:}_i\,\varphi \rightarrow (s+t){:}_i\,\varphi$
L3. $t{:}_i\,\varphi \rightarrow \varphi$
L4. $t{:}_i\,\varphi \rightarrow !t{:}_i\,(t{:}_i\,\varphi)$
CONNECTION PRINCIPLE:
C. $[t]\varphi \rightarrow t{:}_i\,\varphi$.

Term operators mirror properties of justifications: "$\cdot$" is *application* for deduction; "+", *sum*, maintains that justifications are not spoiled by adding (possibly irrelevant) evidence; and "!" is *inspection* and stipulates that justifications themselves are justified. This last operator appears only in justification logics with L4, whose corresponding modal logic contains the modal axiom $\mathsf{4}$ ($\Box\varphi \rightarrow \Box\Box\varphi$), as shown in [6]. A multitude of justification logics of a single agent corresponding to standard modal logics have been developed ([6]). Yavorskaya has investigated versions of LP with two agents in which agents can check each other's proofs ([18]).

Definition 3. *A* constant specification *for each agent,* $i \in \{0, 1, \dots, n\}$, $\mathcal{CS}_i$ *is a set of sentences of sort* $c{:}_i\,A$ *where c is a constant and A an axiom of* $\mathsf{LP}_n(\mathsf{LP})$. *The intuitive reading of these sentences is 'c is a proof of A for agent i.' Let*

$$\mathcal{CS} = \{\mathcal{CS}_1, \dots, \mathcal{CS}_n\}$$

and $\mathcal{CS}_0 \subseteq \mathcal{CS}_i$ *for all* $i \in \{1, 2, \dots, n\}$. *By* $\mathsf{LP}_{n,\mathcal{CS}}(\mathsf{LP}_{\mathcal{CS}_0})$ *we mean the system with the postulates A, R, L1–L4, C above, plus* $\mathcal{CS}_0$ *and* $\mathcal{CS}$ *as additional axioms. As formulas in a constant specification are taken as axioms, they themselves may be used to form other formulas in a* $\mathcal{CS}$ *so that it's possible to have* $c{:}_1\,(d{:}_2\,A) \in \mathcal{CS}_1$ *if* $d{:}_2\,A \in \mathcal{CS}_2$.

The constant specification represents assumptions about proofs of basic postulates that are not further analyzed. If $\mathcal{CS}_i = \emptyset$, agent i is totally skeptical; no formulas are justified. If this is so for all agents, it would be denoted $\mathsf{LP}_{n,\{\emptyset\}}(\mathsf{LP}_\emptyset)$. Constant Specifications of different types have been studied: schematic, injective, full, etc. and have been defined with various closure properties. See [6] for a fuller discussion of constant specifications. The *total constant specification* for any agent, $\mathcal{TCS}_i$, is the union of all possible $\mathcal{CS}_i$. Henceforth we will assume each agent's constant specification is total and will abbreviate this to $\mathsf{LP}_n(\mathsf{LP})$.

Definition 4. *A* modular model of $\mathsf{LP}_n(\mathsf{LP})$ *is* $\mathcal{M} = (W, R_0, R_1, R_2, \ldots, R_n, *, \Vdash)$ *where*

1. - *W is a nonempty set,*
   - *$R_i \subseteq W \times W$ are reflexive for $i \in \{0, 1, 2, \ldots, n\}$. R_0 is the designated accessibility relation for GCK.*
   - *$* : W \times Var \to \{0, 1\}$ and $* : W \times \{0, 1, 2, \ldots, n\} \times Tm \to 2^{Fm}$*

   i.e., for each agent i at node u, $*(u, i, t)$ is a set of formulas t justifies. We write $t_u^{*,i}$ for $*(u, i, t)$. We assume that GCK evidence is everybody's evidence:
   $$t_u^{*,0} \subseteq t_u^{*,i}, \quad \textit{for} \ \ i \in \{0, 1, 2, \ldots, n\} \ .$$
2. *For each agent i and node u, $*$ is closed under the following conditions:*
   $$\begin{aligned} \text{Application: } & s_u^{*,i} \cdot t_u^{*,i} \subseteq (s \cdot t)_u^{*,i} \\ \text{Sum: } & s_u^{*,i} \cup t_u^{*,i} \subseteq (s + t)_u^{*,i} \\ \text{Inspection: } & \{t :_i \varphi \mid \varphi \in (t_u^{*,i})\} \subseteq (!t)_u^{*,i} \end{aligned}$$
   where $s^* \cdot t^* = \{\psi \mid \varphi \to \psi \in s^*$ and $\varphi \in t^*$ for some $\varphi\}$, the set of formulas resulting from applying modus ponens to implications in s^* whose antecedents are in t^*.
3. *For $p \in Var$, we define forcing $\Vdash$ for atomic formulas at node u as $u \Vdash p$ if and only if $*(u, p) = 1$. To define the truth value of all formulas, extend forcing $\Vdash$ to compound formulas by Boolean laws, and define*
   $$u \Vdash t :_i \varphi \quad \Leftrightarrow \quad \varphi \in t_u^{*,i} \ .$$
4. *'justification yields belief' (JYB), i.e., for $i \in \{0, 1, 2, \ldots, n\}$, $u \Vdash t :_i \varphi$ yields $v \Vdash \varphi$ for all v such that uR_iv.*

Modular models, first introduced for the most basic justification logic in [4], are useful for their clear semantical interpretation of justifications as sets of formulas. For modular models of some other justification logics refer to [13]. For a detailed discussion of the relationship between modular models and Mkrtychev–Fitting models for justification logics, see [4].

A model *respects* $\mathcal{CS}_0, \ldots, \mathcal{CS}_n$, if each $c :_i \varphi$ in these constant specifications holds (at each world u) in the model.

Theorem 1 (soundness and completeness). $\mathsf{LP}_{n,\mathcal{CS}}(\mathsf{LP}_{\mathcal{CS}_0}) \vdash F$ *iff F holds in any basic modular model respecting $\mathcal{CS}_i$, $i \in \{0, 1, 2, \ldots, n\}$.*

Proof. Soundness – by induction on the derivation of F, for $i \in \{0, 1, 2, \ldots, n\}$.

- Constant Specifications: If $c{:}_i\,\varphi \in \mathcal{CS}_i$, then $u \Vdash c{:}_i\,\varphi$ as the model respects $\mathcal{CS}_i$.
- Boolean connectives: hold by definition of the truth of formulas.
- Application: Suppose $u \Vdash s{:}_i\,(F \to G)$ and $u \Vdash t{:}_i\,F$. Then by assumption, $(F \to G) \in s_u^{*,i}$ and $F \in t_u^{*,i}$. Then $G \in s_u^{*,i}{\cdot}t_u^{*,i} \subseteq (s{\cdot}t)_u^{*,i}$; thus $u \Vdash (s{\cdot}t){:}_i\,G$.
- Sum: Suppose $u \Vdash t{:}_i\,F$. Then $F \in t_u^{*,i}$ and so $F \in s_u^{*,i} \cup t_u^{*,i} \subseteq (s+t)_u^{*,i}$. Thus $u \Vdash (s+t){:}_i\,F$. Likewise, $u \Vdash (t+s){:}_i\,F$.
- Modus Ponens: Suppose $u \Vdash F \to G$. Then by the definition of the connectives either $u \nVdash F$ or $u \Vdash G$. So if also $u \Vdash F$, then $u \Vdash G$.
- Factivity: Suppose $u \Vdash t{:}_i\,F$. By the 'justification yields belief' condition, $v \Vdash F$ for all v such that uR_iv. As each R_i is reflexive, uR_iu, so also $u \Vdash F$. Inspection: Suppose $u \Vdash t{:}_i\,F$. Then $F \in t_u^{*,i}$ so $t{:}_i\,F \in (!t)_u^{*,i}$. Thus $u \Vdash !t{:}_i\,(t{:}_i\,F)$.
- Connection Principle: Suppose $u \Vdash t{:}_0\,F$. Then $F \in t_u^{*,0} \subseteq t_u^{*,i}$ so $u \Vdash t{:}_i\,F$.

Completeness – by the maximal consistent set construction. For $i \in \{0, 1, 2, \ldots, n\}$, let

- W the set of all maximal consistent sets,
- $\Gamma R_i \Delta$ iff $\Gamma^{i,\#} \subseteq \Delta$ where $\Gamma^{i,\#} = \{F \mid t{:}_i\,F \in \Gamma\}$,
- For $p \in \mathit{Var}$, $*(\Gamma, p) = 1$ iff $p \in \Gamma$,
- $t_\Gamma^{*,i} = \{F \mid t{:}_i\,F \in \Gamma\}$ (i.e., for $X = p, t{:}_i\,F$, $\Gamma \Vdash X$ iff $X \in \Gamma$) .

To confirm that these comprise a modular model, the R_i need to be reflexive, the *GCK* and closure conditions must be checked, and the model must satisfy 'justification yields belief'. As each world is maximally consistent $\Gamma^{i,\#} \subseteq \Gamma$, hence $\Gamma R_i \Gamma$ by L3, so each R_i is reflexive. The *GCK* conditions $t_\Gamma^{*,0} \subseteq t_\Gamma^{*,i}$ for $i \in \{0, 1, 2, \ldots, n\}$ follow from the C axiom $t{:}_0\,F \to t{:}_i\,F$ for $i \in \{1, 2, \ldots, n\}$. Closure conditions for $\cdot$, $+$, and ! follow straightforwardly from the axioms L1, L2, and L4. It remains to check the JYB condition, following the Truth Lemma.

Lemma 1 (Truth Lemma). *$\Gamma \Vdash X$ iff $X \in \Gamma$, for each Γ and X.*

Proof. Induction on X. The atomic and Boolean cases are standard. The only interesting cases are $X = t{:}_i\,F$. Note that $\Gamma \Vdash t{:}_i\,F$ iff $F \in t_\Gamma^{*,i}$ by the definition of modular models. Moreover, under the evaluation particular to this model, $F \in t_\Gamma^{*,i}$ iff $t{:}_i\,F \in \Gamma$. Thus $\Gamma \Vdash t{:}_i\,F$ iff $t{:}_i\,F \in \Gamma$. □

Now to see the JYB condition, suppose $\Gamma \Vdash t{:}_i\,F$ and consider an arbitrary Δ such that $\Gamma R_i \Delta$. By the definition of this model, $t{:}_i\,F \in \Gamma$, hence $F \in \Gamma^{i,\#}$, hence $F \in \Delta$. By the Truth Lemma, $\Delta \Vdash F$.

To finish the proof of completeness, let $\mathsf{LP}_{n,\mathcal{CS}}(\mathsf{LP}_{\mathcal{CS}_0}) \nvdash G$, hence $\{\neg G\}$ is consistent and has a maximal consistent extension, Φ. Since $G \notin \Phi$, by the Truth Lemma, $\Phi \nVdash G$. □

Corollary 1. *Modular models for* LP *(i.e.,* $\mathsf{LP}_0(\mathsf{LP})$*) are* $M = (W, R, *, \Vdash)$ *where*

1. • *W is nonempty*
   • *R is reflexive*
   • *$* : W \times \mathrm{Var} \to \{0,1\}$, $* : W \times \mathrm{Tm} \to 2^{\mathrm{Fm}}$;*
2. *$*$ closure conditions for $\cdot$, $+$, and $!$;*
3. *$u \Vdash p \Leftrightarrow *(u,p) = 1$ and forcing $\Vdash$ extends a truth value to all formulas by Boolean laws and $u \Vdash t{:}F \Leftrightarrow F \in t^*_u$.*
4. *justification yields belief (JYB): $u \Vdash t:F$ yields $v \Vdash F$ for all v such that uRv.*

These modular models for LP differ from those by Kuznets and Studer in [13] as no transitivity is required of R, which enlarges the class of modular models for LP. Artemov suggests (personal communication) this modular model for $LP_\emptyset$ which satisfies Definition 4 and is not transitive and hence ruled out by the formulation offered in [13]:

- $W = \{a, b, c\}$
- $R = \{(aa), (bb), (cc), (ab), (bc)\}$
- $*$ is arbitrary on propositional variables, t^*_a, t^*_b, t^*_c are all empty.

Of course, one could produce more elaborate examples as well, e.g., on the same non-transitive frame, fix a propositional variable p and have $t_1{:}t_2{:}\ldots{:}t_n{:}p$ hold for all proof terms $t_1, \ldots, t_n$, for all n, at any node (in particular, make p true at a, b, c).

While it does not appear to be justified to confine consideration a priori to transitive modular models, the exact role of transitivity of accessibility relations in modular models is still awaiting a careful analysis.

3 Realizing Generic Common Knowledge

We show that $\mathsf{LP}_n\mathsf{LP}$, a logic of explicit knowledge using proof terms, has a precise modal analog in the epistemic logic with GCK, $\mathsf{S4}^J_n$.

Definition 5. *The* axioms and rules of $\mathsf{S4}^J_n$*:*

CLASSICAL PROPOSITIONAL LOGIC:
A. *axioms of classical proposition logic*
R1. *modus ponens*
S4-KNOWLEDGE PRINCIPLES FOR EACH K_i, $i \in \{0, 1, \ldots, n\}$,
(J may be used in place of K_0):
K. $K_i(\varphi \to \psi) \to (K_i\varphi \to K_i\psi)$
T. $K_i\varphi \to \varphi$
4. $K_i\varphi \to K_iK_i\varphi$
R2. $\vdash \varphi \Rightarrow \vdash K_i\varphi$
CONNECTION PRINCIPLE:
C1. $J\varphi \to K_i\varphi$.

In $\mathsf{S4}_n^J$, the common knowledge operator J is indeed generic as $J(\varphi) \to C(\varphi)$ while $C(\varphi) \not\to J(\varphi)$, as illustrated in [2]. McCarthy et al. provide Kripke models for one of their logics in [15], see also [7]. In Kripke models, a distinction between generic and conventional common knowledge is clear. The accessibility relation for C, R_C, is the exact transitive closure of the union of all other agents' accessibility relations R_i. R_J, the accessibility relation for J is any transitive and reflexive relation which *contains* the union of all other agents' relations, thus

$$R_{GCK} = R_J \supseteq R_C.$$

This means that generally speaking, there is flexibility in choosing R_J while R_C is unique in each given model. Note that in the case where we have explicit proof terms and not just modalities of implicit knowledge, we also have this multiplicity of options for generic common knowledge: there may be many evaluations $*$ such that $t_u^{*,0}$ that satisfiest $t_u^{*,0} \subseteq t_u^{*,i}$ for all i.

We now have $\mathsf{LP}_n(\mathsf{LP})$ and $\mathsf{S4}_n^J$, each is a multi-agent epistemic logic with generic common knowledge, where all justifications are explicit in the former and implicit in the latter. By proving the Realization Theorem, we will establish that $\mathsf{LP}_n(\mathsf{LP})$ is the exact explicit version of $\mathsf{S4}_n^J$.

Definition 6. *The* forgetful projection *is a translation*

$$\circ : \mathcal{L}_{\mathsf{LP}_n(\mathsf{LP})} \to \mathcal{L}_{\mathsf{S4}_n^J}$$

defined inductively as follows:

- $p^\circ = p$, *for* $p \in Var$
- $(\neg\psi)^\circ = \neg(\psi^\circ)$
- $\circ$ *commutes with binary Boolean connectives:* $(\psi \wedge \varphi)^\circ = \psi^\circ \wedge \varphi^\circ$ *and* $(\psi \vee \varphi)^\circ = \psi^\circ \vee \varphi^\circ$
- $(t{:}_i\,\psi)^\circ = K_i(\psi^\circ)$ *for* $i \in \{0, 1, \ldots, n\}$.

Proposition 1. $[\mathsf{LP}_n(\mathsf{LP})]^\circ \subseteq \mathsf{S4}_n^J$.

Proof. The $\circ$ translations of all the $\mathsf{LP}_n(\mathsf{LP})$ axioms and rules are easily seen to be theorems of $\mathsf{S4}_n^J$. $\square$

We want to show that these two logics are really correspondences and that

$$\mathsf{S4}_n^J \subseteq [\mathsf{LP}_n(\mathsf{LP})]^\circ$$

also holds. This is much more involved. Theorem 3 shows that a derivation of any $\mathsf{S4}_2^J$ theorem σ can yield an $\mathsf{LP}_2(\mathsf{LP})$ theorem τ such that $\tau^\circ = \sigma$. This process, the converse of the $\circ$-translation, is a *Realization* r.

Definition 7. *A realization* r *is* normal *if all negative occurrences of modalities (whether a* K_i *or* J*) are realized by distinct proof variables.*

To provide an algorithm r for such a process, we first give the Gentzen system for $\mathsf{S4}_n^J$ and the Lifting Lemma (Proposition 2).

Definition 8. $\mathsf{S4}_n^J\mathsf{G}$, *the* Gentzen *version of* $\mathsf{S4}_n^J$, *is the usual propositional Gentzen rules (i.e., system* $\mathsf{G1c}$ *in [16]) with addition of* $n+1$ *pairs of rules:*

$$\frac{\varphi, \Gamma \ \Rightarrow\ \Delta}{\Box\varphi, \Gamma \ \Rightarrow\ \Delta}\,(\Box,\Rightarrow) \qquad \textit{and} \qquad \frac{J\Gamma, \Box\Delta \ \Rightarrow\ \varphi}{J\Gamma, \Box\Delta \ \Rightarrow\ \Box\varphi}\,(\Rightarrow,\Box)$$

where $\Box$ *is* J *or some* K_i. *As usual, capital letters are multisets and* $\Box\{\varphi_1, \ldots, \varphi_n\} = \{\Box\varphi_1, \ldots, \Box\varphi_n\}$.

Theorem 2. $\mathsf{S4}_n^J\mathsf{G}$ *is equivalent to* $\mathsf{S4}_n^J$ *and admits cut-elimination.*

Proof. See Artemov's proof in Section 6 of [7]. □

Let $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be finite lists of formulas, $\vec{y}, \vec{z}$ finite lists of proof variables of matching length, respectively. Then $[\vec{y}\,]\Gamma = [y_1]\gamma_1, \cdots, [y_m]\gamma_m$ and $\vec{z}{:_i}\Sigma = z_1{:_i}\sigma_1, \cdots, z_n{:_i}\sigma_n,\ i \in \{0, 1, 2, \ldots, n\}$.

Proposition 2 (Lifting Lemma). *In* $\mathsf{LP}_n(\mathsf{LP})$, *for* $i \in \{0, 1, 2, \ldots, n\}$ *and each* $\Gamma, \Sigma, \vec{y}, \vec{z}$,

$$\frac{[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash \varphi}{[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash f(\vec{y}, \vec{z}){:_i}\varphi}$$

for the corresponding proof term $f(\vec{y}, \vec{z})$.

Proof. By induction on the complexity of φ.

- φ is an axiom of $\mathsf{LP}_n(\mathsf{LP})$, then as $\mathsf{LP}_n(\mathsf{LP})$ has $\mathcal{TCS}$, for any constant c, $c{:_i}\varphi$ so let $f(\vec{y}, \vec{z}) = c$. As $\vdash_{\mathsf{LP}_n(\mathsf{LP})} c{:_i}\varphi$, also $[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} c{:_i}\varphi$.
- φ is $[y_j]\gamma_j$ for some $[y_j]\gamma_j \in [\vec{y}]\Gamma$, then
$$[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} [y_j]\gamma_j\ ,$$
hence
$$[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} [!y_j]([y_j]\gamma_j)\ ,$$
and
$$[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} !y_j{:_i}([y_j]\gamma_j)\ .$$
So,
$$[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} !y_j{:_i}\varphi\ ,$$
and we can put $f(\vec{y}, \vec{z}) = !y_j$.
- φ is $z_j{:_i}\sigma_j$ for some $z_j{:_i}\sigma_j \in \vec{z}{:_i}\Sigma$, then as $!z_j{:_i}(z_j{:_i}\sigma_j)$ is given,
$$[\vec{y}\,]\Gamma, \vec{z}{:_i}\Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} !z_j{:_i}\varphi\ .$$
So let $f(\vec{y}, \vec{z}) = !z_j$.
- φ is derived by modus ponens from ψ and $\psi \to \varphi$. By the Induction Hypothesis, there exists $t{:_i}\psi$ and $u{:_i}(\psi \to \varphi)$ (where $t = f_t(\vec{y}, \vec{z})$ and $u = f_u(\vec{y}, \vec{z})$). Since $u{:_i}(\psi \to \varphi) \to (t{:_i}\psi \to (u{\cdot}t){:_i}\varphi)$, by modus ponens $(u{\cdot}t){:_i}\varphi$. So let $f(\vec{y}, \vec{z}) = (u{\cdot}t)$.

– φ is $c :_i A \in \mathcal{TCS}$. Since $c :_i A \to !c :_i (c :_i A)$ and $\vdash_{\mathsf{LP}_n(\mathsf{LP})} c :_i A$, also $\vdash_{\mathsf{LP}_n(\mathsf{LP})} !c :_i (c :_i A)$ thus

$$[\vec{y}\,]\Gamma, \vec{z} :_i \Sigma \vdash_{\mathsf{LP}_n(\mathsf{LP})} !c :_i \varphi.$$

So let $f(\vec{y}, \vec{z}) = !c$. □

Theorem 3 (Realization Theorem). *If* $\mathsf{S4}_n^J \vdash \varphi$*, then* $\mathsf{LP}_n(\mathsf{LP}) \vdash \varphi^r$ *for some normal realization* r*.*

Proof. The proof follows closely the realization proof from [8] with adjustments to account for the Lifting Lemma.

If $\mathsf{S4}_n^J \vdash \varphi$, then by Theorem 2 there is a cut-free derivation $\mathcal{D}$ of the sequent $\Rightarrow \varphi$ in $\mathsf{S4}_n^J\mathsf{G}$. We now construct a normal realization algorithm r that runs on $\mathcal{D}$ and returns an $\mathsf{LP}_n(\mathsf{LP})$ theorem $\varphi^r = \psi$ such that $\psi^\circ = \varphi$.

In φ, positive and negative modalities are defined as usual. The rules of $\mathsf{S4}_n^J\mathsf{G}$ respect these polarities so that $(\Rightarrow, \Box)$ introduces positive occurrences and $(\Box, \Rightarrow)$ introduces negative occurrences of $\Box$, where $\Box$ is J or some K_i. Call the occurrences of $\Box$ *related* if they occur in related formulas in the premise and conclusion of some rule: the same formula, that formula boxed or unboxed, enlarged or shrunk by $\wedge$ or $\vee$, or contracted. Extend this notion of related modalities by transitivity. Classes of related $\Box$ occurrences in $\mathcal{D}$ naturally form disjoint *families* of related occurrences. An *essential* family is one which at least one of its members arises from the $(\Rightarrow, \Box)$ rule, these are clearly positive families.

Now the desired r is constructed by the following three steps so that negative and non-essential positive families are realized by proof variables while essential families will be realized by sums of functions of those proof variables.

Step 1. For each negative family and each non-essential positive family, replace all $\Box$ occurrence so that $J\alpha$ becomes $[x]\alpha$ and $K_i\alpha$ becomes $y_i :_i \alpha$. Choose new and distinct proof variables x and y_i for each of these families.

Step 2. Choose an essential family f. Count the number n_f of times the $(\Rightarrow, \Box)$ rule introduces a box to this family. Replace each $\Box$ with a sum of proof terms so that for $i \in \{0, 1, 2, \ldots, n\}$, $K_i\alpha$ becomes

$$(w_{i,1} + w_{i,2} + \cdots + w_{i,n_f}) :_i \alpha,$$

with each $w_{i,j}$ a fresh provisional variable. Do this for each essential family. The resulting tree $\mathcal{D}'$ is now labeled by $\mathsf{LP}_n(\mathsf{LP})$-formulas.

Step 3. Now the provisional variables need to be replaced, starting with the leaves and working toward the root. By induction on the depth of a node in $\mathcal{D}'$ we will show that after the process passes a node, the sequent at that level becomes derivable in $\mathsf{LP}_n(\mathsf{LP})$ where

$$\Gamma \Rightarrow \Delta$$

is read as provability of

$$\Gamma \vdash_{\mathsf{LP}_n(\mathsf{LP})} \bigvee \Delta.$$

Note that axioms $p \Rightarrow p$ and $\bot \Rightarrow$ are derivable in $\mathsf{LP}_n(\mathsf{LP})$. For each move down the tree other than by the rule $(\Rightarrow, \Box)$, the concluding sequent is $\mathsf{LP}_n(\mathsf{LP})$-derivable if its premises are; for rules other that this one, do not change the realization of formulas. For a given essential family f, for the occurrence numbered j of the $(\Rightarrow, \Box)$ rule, the corresponding node in $\mathcal{D}'$ is labeled

$$\frac{[\vec{z}]\Gamma,\ \vec{q}{:}_i\,\Sigma \Rightarrow \alpha}{[\vec{z}]\Gamma,\ \vec{q}{:}_i\,\Sigma \Rightarrow (u_1 + \cdots + u_{n_f}){:}_i\,\alpha}\ , \quad \text{for } \Box \text{ is } K_i, \text{ for } i \in \{0, 1, 2, \ldots, n\}$$

where the z's and q's are proof variables and the u's are evidence terms, with u_j a provisional variable. By the Induction Hypothesis, the premise is derivable in $\mathsf{LP}_n(\mathsf{LP})$. By the Lifting Lemma (Proposition 2), construct a justification term $f(\vec{z}, \vec{q})$ for α where

$$[\vec{z}]\Gamma,\ \vec{q}{:}_i\,\Sigma \vdash f(\vec{z}, \vec{q}){:}_i\,\alpha\ .$$

Now we will replace the provisional variable u_j as follows

$$[\vec{z}]\Gamma,\ \vec{q}{:}_i\,\Sigma \vdash (u_1 + \cdots + u_{j-1} + f(\vec{z}, \vec{q}) + u_{j+1} + \cdots + u_{n_f}){:}_i\,\alpha\ .$$

Substitute each u_j with $f(\vec{z}, \vec{q})$ everywhere in $\mathcal{D}'$. There is now one fewer provisional variable in the tree as $f(\vec{z}, \vec{q})$ has none. The conclusion to this j^{th} instance of the rule $(\Rightarrow, \Box)$ becomes derivable in $\mathsf{LP}_n(\mathsf{LP})$, completing the induction step.

Eventually all provisional variables are replaced by terms of non-provisional variables, establishing that the root sequent of $\mathcal{D}$, φ^r, is derivable in $\mathsf{LP}_n(\mathsf{LP})$. The realization constructed in this manner is normal. $\Box$

Corollary 2. $\mathsf{S4}_n^J$ *is the forgetful projection of* $\mathsf{LP}_n(\mathsf{LP})$.

Proof. A straightforward consequence of Proposition 1 and Theorem 3. $\Box$

We see that the common knowledge component of $\mathsf{LP}_n(\mathsf{LP})$ indeed corresponds to the generic common knowledge J and hence can be regarded as the *explicit GCK*.

4 Realization Example

Here we demonstrate a realization of an $\mathsf{S4}_2^J$ theorem in $\mathsf{LP}_2(\mathsf{LP})$.

Proposition 3. $\mathsf{S4}_2^J \vdash J\neg\phi \to K_2\neg K_1\phi$.

Proof. Here is an $\mathsf{S4}_2^J\mathsf{G}$ derivation of the corresponding sequent.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\phi \Rightarrow \phi}{K_1\phi \Rightarrow \phi}\,(\Box,\Rightarrow)}{\neg\phi, K_1\phi \Rightarrow}\,(\neg,\Rightarrow)}{\neg\phi \Rightarrow \neg K_1\phi}\,(\Rightarrow,\neg)}{J\neg\phi \Rightarrow \neg K_1\phi}\,(\Box,\Rightarrow)}{J\neg\phi \Rightarrow K_2\neg K_1\phi}\,(\Rightarrow,\Box)}{\Rightarrow J\neg\phi \to K_2\neg K_1\phi}\,(\Rightarrow,\to)$$

$\Box$

Now we follow the realization algorithm to end up with an $\mathsf{LP}_2(\mathsf{LP})$ theorem. In the sequent proof, the J in the conclusion is in negative position and all the Js in this derivation are related and form a negative family. The occurrences of the K_1 modality are all related and they too form a negative family. The two occurrences of K_2 form an essential positive family with $n_f = 1$ as there is one use of the $(\Rightarrow, \Box)$ rule.

Step 1. Replace all J occurrences with '$[x]$' and K_1 occurrences with '$y{:}_1$'.

Step 2. Replace all K_2 occurrences with a '$w{:}_2$' with w a provisional variable. Since here $n_f = 1$, a sum is not required. At this stage the derivation tree looks like this, where '$\Rightarrow$' is read as '$\vdash$' in $\mathsf{LP}_2(\mathsf{LP})$:

$$\begin{array}{rcll}
\phi & \Rightarrow & \phi & \\ \hline
y{:}_1\phi & \Rightarrow & \phi & (\Box,\Rightarrow) \\ \hline
\neg\phi, y{:}_1\phi & \Rightarrow & & (\neg,\Rightarrow) \\ \hline
\neg\phi & \Rightarrow & \neg y{:}_1\phi & (\Rightarrow,\neg) \\ \hline
[x]\neg\phi & \Rightarrow & \neg y{:}_1\phi & (\Box,\Rightarrow) \\ \hline
[x]\neg\phi & \Rightarrow & w{:}_2(\neg y{:}_1\phi) & (\Rightarrow,\Box) \\ \hline
 & \Rightarrow & [x]\neg\phi \to w{:}_2(\neg y{:}_1\phi) & (\Rightarrow,\to)
\end{array}$$

Step 3. The one instance of the $(\Rightarrow, \Box)$ rule calls for the Lifting Lemma to replace w with $f(x)$ so that

$$[x]\neg\phi \vdash f(x){:}_2(\neg y{:}_1\phi)$$

in $\mathsf{LP}_2(\mathsf{LP})$. The proof of the Lifting Lemma is constructive and provides a general algorithm of finding such f. To skip some routine computations we will use the trivial special case of Lifting Lemma: if F is proven from the axioms of $\mathsf{LP}_2(\mathsf{LP})$ by classical propositional reasoning, then there is a ground[2] term g such that $g :_i F$ is also derivable in $\mathsf{LP}_2(\mathsf{LP})$ for each $i \in \{0, 1, 2\}$, without specifying g.

Consider the following Hilbert-style derivation in $\mathsf{LP}_2(\mathsf{LP})$, line 7 in particular.

1.	$y{:}_1\phi \to \phi$	L3 axiom for agent 1
2.	$\neg\phi \to \neg y{:}_1\phi$	from 1. by contraposition
3.	$[g](\neg\phi \to \neg y{:}_1\phi)$	for some ground term g
4.	$[g](\neg\phi \to \neg y{:}_1\phi) \to ([x]\neg\phi \to [g\cdot x]\neg y{:}_1\phi)$	L1 axiom for *GCK*
5.	$[x]\neg\phi \to [g\cdot x]\neg y{:}_1\phi$	from 3. and 4. by modus ponens
6.	$[g\cdot x]\neg y{:}_1\phi \to (g\cdot x){:}_2\,\neg y{:}_1\phi$	connection principle
7.	$[x]\neg\phi \to (g\cdot x){:}_2\,\neg y{:}_1\phi$	from 5. and 6. by propositional reasoning

[2] Ground proof terms are built from constants only and do not contain proof variables.

So, it suffices to put $f(x) = g\cdot x$ where g is a ground proof term from line 3.[3] Note the forgetful projection of the $\mathsf{LP}_2(\mathsf{LP})$ theorem line 7.,

$$[[x]\neg\phi \rightarrow (g\cdot x){:}_2\, \neg y{:}_1\, \phi]^\circ = J\phi \rightarrow K_2\neg K_1\phi\ ,$$

is the original $\mathsf{S4}_n^J$ theorem which was Realized.

5 Conclusion

The family of Justification Logics offers a robust and flexible setting in which to investigate explicit reasons for knowing: $t{:}F$, "F is know for reason t", in contrast to a modal approach in which $\Box F$ or $\mathbf{K}F$ represent implicit knowledge of F, where reasons are not specified. The addition of generic common knowledge opens these systems to numerous epistemic applications ([2,5,3]). The Realization Theorem for $\mathsf{S4}_n^J$ allows for all modalities, including GCK (J), to be made explicit in $\mathsf{LP}_n(\mathsf{LP})$, allowing reasoning to be tracked.

In the $\mathsf{LP}_n(\mathsf{LP})$ case presented here all agent reasoning represents knowledge. While it is useful to track the justifications, in the knowledge domain, each justification is a proof and so yields truth. However, in a belief setting, justifications are not necessarily sufficient to yield truth. In these situations it may become even more crucial, essential, to track specific evidence in order to analyze their reliability and compare justifications arriving from different sources. Logics of belief with GCK can be constructed: without factivity (L3) belief rather than knowledge is modeled. Investigating multi-agent logics of belief with GCK will likely also yield a rich source of models in which to analyze several traditional epistemic scenarios and may also offer an entry to considering an explicit version of common belief.

Generic common knowledge is a useful choice for modeling many epistemic situations and here we have presented what has yet to be shown for conventional common knowledge: that a modal epistemic logic with generic common knowledge can be made fully explicit. This is done through the introduction of the justification logic $\mathsf{LP}_n(\mathsf{LP})$ with explicit GCK and the Realization algorithm.

References

1. Alberucci, L., Jaeger, G.: About cut elimination for logics of common knowledge. Annals of Pure and Applied Logic 133(1-3), 73–99 (2005)
2. Antonakos, E.: Justified and Common Knowledge: Limited Conservativity. In: Artemov, S., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 1–11. Springer, Heidelberg (2007)
3. Artemov, S.: Multiplicity of Common Knowledge. In: Prague Workshop on Nonclassical Epistemic Logics, June 14-15, Abstracts (2012), `http://logicka.flu.cas.cz/redaction.php?action=showRedaction&id_categoryNode=1996`

[3] Here g is built from the constant that proves this instance of L3 and those used in the derivation of the contrapositive from an implication.

4. Artemov, S.: The ontology of justifications in the logical setting. Studia Logica 100(1-2), 17–30 (2012)
5. Artemov, S.: Robust Knowledge and Rationality. Technical Report TR-2010010, CUNY Ph.D. Program in Computer Science (2010)
6. Artemov, S.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
7. Artemov, S.: Justified Common Knowledge. Theoretical Computer Science 357 (1-3), 4–22 (2006)
8. Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
9. Aumann, R.: Agreeing to Disagree. Annals of Statistics 4(6), 1236–1239 (1976)
10. Bucheli, S.: Justification Logics with Common Knowledge. PhD thesis, Universität Bern (2012)
11. Bucheli, S., Kuznets, R., Studer, T.: Justifications for common knowledge. Journal of Applied Non-Classical Logics 21(1), 35–60 (2011)
12. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning About Knowledge. MIT Press (1995)
13. Kuznets, R., Studer, T.: Justifications, Ontology, and Conservativity. In: Bolander, T., Braüner, T., Ghilardi, S., Moss, L. (eds.) Advances in Modal Logic, vol. 9, pp. 437–458 (2012)
14. Lewis, D.: Convention, A Philosophical Study. Harvard University Press (1969)
15. McCarthy, J., Sato, M., Hayashi, T., Igarishi, S.: On the Model Theory of Knowledge. Technical Report STAN-CS-78-657, Stanford University (1978)
16. Troelstra, A.S., Schwittenberg, H.: Basic Proof Theory. Cambridge Tracts in Theoretical Computer Science, vol. 43. Cambridge University Press (1996)
17. Yavorskaya(Sidon), T.: Interacting explicit evidence systems. Theory of Computing Systems 43(2), 272–293 (2008), Published online (October 2007)
18. Yavorskaya(Sidon), T.: Multi-agent Explicit Knowledge. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, pp. 369–380. Springer, Heidelberg (2006)

Beyond 2-Safety: Asymmetric Product Programs for Relational Program Verification

Gilles Barthe[1], Juan Manuel Crespo[1], and César Kunz[1,2]

[1] IMDEA Software Institute, Spain
[2] Universidad Politécnica de Madrid, Spain

Abstract. Relational Hoare Logic is a generalization of Hoare logic that allows reasoning about executions of two programs, or two executions of the same program. It can be used to verify that a program is robust or (information flow) secure, and that two programs are observationally equivalent. Product programs provide a means to reduce verification of relational judgments to the verification of a (standard) Hoare judgment, and open the possibility of applying standard verification tools to relational properties. However, previous notions of product programs are defined for deterministic and structured programs. Moreover, these notions are symmetric, and cannot be applied to properties such as refinement, which are asymmetric and involve universal quantification on the traces of the first program and existential quantification on the traces of the second program.

Asymmetric products generalize previous notions of products in three directions: they are based on a control-flow graph representation of programs, they are applicable to non-deterministic languages, and they are by construction asymmetric. Thanks to these characteristics, asymmetric products allow to validate abstraction/refinement relations between two programs, and to prove the correctness of advanced loop optimizations that could not be handled by our previous work. We validate their effectiveness by applying a prototype implementation to verify representative examples from translation validation and predicate abstraction.

1 Introduction

Program verification tools provide an effective means to verify trace properties of programs. However, many properties of interest are 2-properties, i.e. consider pairs of traces, rather than traces; examples include non-interference and robustness, which consider two executions of the same program, and abstraction/equivalence/refinement properties, which relate executions of two programs. Relational Hoare logic [8] generalizes Hoare logic by allowing to reason about two programs, and provides an elegant theoretical framework to reason about 2-properties. However, relational Hoare logic is confined to reason about universally quantified statements over traces, and only relates programs with the same termination behavior. Thus, relational Hoare logic cannot capture notions of refinement, and more generally properties that involve an alternation of existential and universal quantification. Moreover, relational Hoare logic is not tool supported.

Product programs [20,4] provide a means to reduce verification of relational Hoare logic quadruples to verification of standard Hoare triples. Informally, the product program construction transforms two programs P_1 and P_2 into a single program P that soundly abstracts the behavior of P_1 and P_2, so that relational verification over P_1 and P_2 can be reduced to verification of P. Product programs are attractive, because they allow reusing existing verification tools for relational properties. However, like relational Hoare logic, the current definition of product program is only applicable to universally quantified statements over traces. Moreover, the construction of product programs has been confined to structured and deterministic programs written in a single language. This article introduces asymmetric (left or right) product programs, which generalize symmetric products from [20,4], and allow showing abstraction/refinement properties, which are typically of the form: for all execution of the first program, there is a related execution of the second program. Furthermore, asymmetric product are based on a flow-graph representation of programs, which provides significant advantages over previous works. In particular, asymmetric products can relate programs: 1. with different termination behaviors; 2. including non-deterministic statements; 3. written in two different languages (provided they support a control flow graph representation). Finally, asymmetric products allow justifying some loop transformations that where out of reach of our previous work on translation validation. We evaluate our method on representative examples, using a prototype implementation that builds product programs and sends the verification task to the Why platform.

Section 2 motivates left products with examples of predicate abstraction and translation validation. Sections 3 and 4 introduce the notion of left product and show how they can be used to reduce relational verification to functional verification. Section 5 introduces full products, a symmetric variant of left products that is used to validate examples of translation validation that were not covered by [4]. Section 6 presents an overview of our implementation.

2 Motivating Examples

In this section we illustrate our technique through some examples. The first two are abstraction validation examples and for their verification we use the asymmetric framework, while for the verification of the loop optimization, we use a stronger version of the method, introduced in Section 5.

For both domains of application, we first provide an informal overview of the verification technique. Throughout the rest of the paper, we refer back to these examples in order to illustrate the technical concepts and results.

2.1 Abstraction Validation

The correctness of the verification methods based on program abstraction relies on the soundness of its abstraction mechanism. Since such abstraction mechanisms are increasingly complex it becomes desirable to perform a posteriori, independent validation of their results.

In general, abstractions induce some loss of information, represented in the abstract programs as non-deterministic statements. The extensions presented in this paper enable our framework to cope with non-determinism.

Predicate Abstraction. Predicate abstraction [1,12] reduces complexity of a program verification to the analysis of a bounded-state program, by representing infinite-state systems in terms of a finite set of user-provided predicates. The program on the left of Figure 1, drawn from [1], partitions a singly linked list of integers into two lists: one containing the elements with value greater than the parameter v and the other one containing the cells with value less than or equal to v. The program on the right represents the predicate abstraction of the program on the left, w.r.t. a set of user-provided boolean predicates: $\{curr = \mathsf{null}, prev = \mathsf{null}, curr{\mapsto}val > v, prev{\mapsto}val < v\}$. The abstraction is performed by representing each boolean predicate with a boolean variable: e.g., $\overline{curr}$ represents the condition $curr = \mathsf{null}$. The effect of the instructions of the original program is captured by assignments and assert statements involving the boolean variables of the abstraction: e.g. the effect of the assignment $prev := \mathsf{null}$ on the predicate $prev = \mathsf{null}$ is reflected by the assignment $\overline{prev} := \mathsf{true}$ on the right program. Note that some of the abstract predicates will have an unknown value after some of the concrete instructions, as is the case with the predicate $curr = \mathsf{null}$ after the assignment $curr := *\,l$, reflected by the non-deterministic assignment $\overline{curr} := ?$.

We consider the problem of automatically validating abstractions that are expressed as non-deterministic programs in some variant of the original programming language. Our goal is to verify that the program on the right soundly abstracts the original one, i.e. any execution path of the original program can be simulated by an execution path of the abstracted program. In order to establish the correctness of the program abstraction, we must verify a simulation relation between the execution traces of both programs. This simulation is captured by a new program constructed from the original and abstract programs, shown in Figure 4, providing a fixed control flow for the simulation relation.

The validation of the abstraction is carried over the product program in Fig. 4 by two independent verification steps. One must first verify that the product program captures correctly the synchronous executions of the original and abstract programs, i.e., that for any trace on the left program there exists a trace on the right program. We say then that the graph is a *left product* and it satisfies the properties stated in Lemma 2. In a second step, one must check that the product program satisfies the given refinement relation, stated as a relational invariant specification: $(curr = \mathsf{null} \Leftrightarrow \overline{curr}) \wedge (prev = \mathsf{null} \Leftrightarrow \overline{prev}) \wedge (curr{\mapsto}val < v \Leftrightarrow currV) \wedge (prev{\mapsto}val < v \Leftrightarrow prevV)$

Numeric Abstraction. Numeric abstraction [14] is a similar program abstraction strategy based on a shape analysis defined from user-provided size abstractions. The output of this transformation is not necessarily a bounded-state program, but it can be used to establish some properties of the original program, e.g., termination behavior, or resource consumption.

Figure 2 shows an example of a source and an abstract programs, drawn from [14]. The program on the left performs left to right, depth first traversal of the binary tree pointed by its argument. It maintains a stack of nodes to be processed. On each iteration, the top of the stack is removed and its children (if any) are added. The program on the right of the figure is a numeric abstraction of the source program that explicitly keeps track of the changes in data structure sizes. In the abstract program, $tsizeroot$ represents the number of nodes in the tree, $slen$ the length of the list representing the stack and $ssize$ the number of nodes contained in the trees held within the stack. More

```
$curr := *l$; $prev := \mathsf{null}$;
$newl := \mathsf{null}$;
while ($curr \neq \mathsf{null}$) do
  $nextCurr := curr \mapsto next$;
  if ($curr \mapsto val > v$) then
    if ($prev \neq \mathsf{null}$) then
      $prev \mapsto next := nextCurr$;
    if ($curr = *l$) then $*l := nextCurr$;
    $curr \mapsto next := newl$; $newl := curr$;
  else
    $prev := curr$;
  $curr := nextCurr$;
```

```
$\overline{\mathbf{curr}} := ?$; $\overline{\mathbf{prev}} := \mathsf{true}$;
$\mathbf{currV} := ?$; $\mathbf{prevV} := ?$;
while ($*$) do
  assert($\neg\overline{\mathbf{curr}}$);
  if ($*$) then
    assert($\mathbf{currV}$);
    if ($*$) then
      assert($\neg\overline{\mathbf{prev}}$);
  else
    assert($\mathbf{currV} = \mathbf{false}$);
    $\overline{\mathbf{prev}} := \overline{\mathbf{curr}}$;
    $\mathbf{prevV} := \mathbf{currV}$;
  $\overline{\mathbf{curr}} := ?$; $\mathbf{currV} := ?$;
assert($\overline{\mathbf{curr}}$);
```

Fig. 1. Predicate abstraction

precisely, the user-provided abstractions are defined as inductive predicates over acyclic heap structures, e.g.:

$$\frac{}{\mathsf{ListLength}(\mathsf{null}, 0)} \qquad \frac{\mathsf{ListLength}(ls\mapsto tail, n)}{\mathsf{ListLength}(ls, n+1)}\ ls \neq \mathsf{null}$$

$$\frac{}{\mathsf{TreeSize}(\mathsf{null}, 0)} \qquad \frac{\mathsf{TreeSize}(t\mapsto left, n_l) \qquad \mathsf{TreeSize}(t\mapsto right, n_r)}{\mathsf{TreeSize}(t, n_l+n_r+1)}\ t \neq \mathsf{null}$$

Note that upon entering the loop, we do not have information on the size of the first tree contained in the stack, nor of the size of the trees in the rest of the stack. This is represented in the abstraction by a non-deterministic assignment.

As in the previous example, we can verify a posteriori that the numeric program soundly abstracts a heap manipulating program by constructing a product program that fixes the control flow of the simulation to be verified. The product program shown in Figure 5 is totally synchronized, in the sense that every program edge represents a simultaneous execution of the program components.

The simulation relation is defined in terms of the user-provided size abstractions. This relational specification makes explicit the correspondence between the abstract numeric variables and the size predicates over the original data structures; these size relations include, e.g., $\mathsf{ListLength}(st, slen)$ and $\mathsf{TreeSize}(root, tsizeroot)$, which must hold whenever the variables are in scope.

We develop the notion of left product used for abstraction validation in Section 3.

2.2 Translation Validation

Translation validation [3,16] is a general method for ensuring the correctness of optimizing compilation by means of a validator which checks, after each run of the compiler, that the source and target programs are semantically equivalent. In previous work, we have used a notion of program products to formally verify the correctness of several program optimizations [4]. An important limitation of our previous notion of program

```
$st := push(root, 0)$;                        assert($0 \leq$ tsizeroot);
                                              slen := 1; ssize := tsizeroot;
while ($st \neq 0$) do                        while (slen > 0) do
   $tail := st \rightarrow next$;                 tsize := ?; ssizetail := ?;
                                                 assert($0 \leq$ tsize $\wedge$ $0 \leq$ ssizetail);
                                                 assert(ssize = tsize + ssizetail);
   if ($st \rightarrow tree = 0$) then               if (tsize = 0) then
      free($st$); $st := tail$;                       slen--;
   else                                          else
      $tail := push(st \rightarrow tree \rightarrow right, tail)$;   tsizel := ?; tsizer := ?;
      $tail := push(st \rightarrow tree \rightarrow left, tail)$;    assert($0 \leq$ tsizel $\wedge$ $0 \leq$ tsizer);
      free($st$);                                   assert(tsize = tsizel + tsizer + 1);
      $st := tail$;                                 ssize := tsizel + tsizer + ssizetail;
                                                    slen++;
```

Fig. 2. Numeric abstraction

```
a:  $x := 0$;                  0:  i := 0;
b:  while ($x < NM$) do        1:  while (i < N) do
       $a[x] := f(x)$;                j := 0;
       $x$++                   2:     while (j < M) do
                                         A[i, j] := f(iM + j); j++;
                                      i++
```

Fig. 3. Loop tiling example

products is that they are required to be representable syntactically as structured code. The extension provided in this work enables the verification of more complex loop optimizations that were not considered in previous work.

Loop tiling is an optimization that splits the execution of a loop into smaller blocks, improving the cache performance. If the loop accesses a block of contiguous data during its execution, splitting the block in fragments that fits the cache size can help avoiding cache misses, depending on the target architecture. The program at the right of Fig. 3 shows the result of applying a loop tiling transformation to the code at the left. The traversal of a block of size NM is split into N iterations accessing smaller blocks of size M, by the introduction of an inner loop and new iteration variables i and j. It is not hard to see that the iteration space of the outermost loops are equal and that the relational invariant $x = iM+j$ holds.

The structural dissimilarity of the original and transformed loop is a main obstacle for the application of our previous relational verification method. However, the relaxed notion of program product presented in this article can be used to validate this transformation. Figure 6 shows a possible product of the two programs in Fig. 3. The loop bodies (i.e., edges $\langle 2, 2\rangle$ and $\langle b, b\rangle$) are executed synchronously, represented by edge $\langle (b, 2), (b, 2)\rangle$. Notice that asynchronous edges represents the transitions of the right program that cannot be matched with transitions on the left program. We develop the notion of full products for the validation of compiler optimizations in Section 5.

3 Simulation by Left Products

We define a general notion of product program and prove that under mild conditions they mimic the behavior of their constituents. We adopt a representation of programs based on labeled directed graphs. Nodes correspond to program points, and include an initial and a final node; for simplicity, we assume their unicity. Edges are labeled with statements from the set Stmt.

Definition 1 (Program). *A program P is a tuple $\langle \mathcal{N}, \mathcal{E}, G\rangle$, where $\langle \mathcal{N}, \mathcal{E}\rangle$ is a directed graph with unique source* $\mathsf{in} \in \mathcal{N}$ *and sink* $\mathsf{out} \in \mathcal{N}$*, and $G : \mathcal{E} \to \mathsf{Stmt}$ maps edges to statements.*

The semantics of statements is given by a mapping $[\![.]\!] : \mathsf{Stmt} \to \mathcal{P}(\mathcal{S} \times \mathcal{S})$, where $\mathcal{S}$ is a set of states. A configuration is a pair $\langle l, \sigma\rangle$, where $l \in \mathcal{N}$ and $\sigma \in \mathcal{S}$; we let $\langle l, \sigma\rangle \rightsquigarrow \langle l', \sigma'\rangle$ stand for $(\sigma, \sigma') \in [\![G\langle l, l'\rangle]\!]$. A trace is a sequence of configurations s.t. the first configuration is of the form $\langle \mathsf{in}, \sigma\rangle$, and $(\sigma, \sigma') \in [\![G\langle l, l'\rangle]\!]$ for any two consecutive elements $\langle l, \sigma\rangle$ and $\langle l', \sigma'\rangle$ of the sequence; we let $\mathsf{Tr}(P)$ denote the set of traces of P. Moreover, an execution is a trace whose last configuration is of the form $\langle \mathsf{out}, \sigma\rangle$; we let $\mathsf{Ex}(P) \subseteq \mathsf{Tr}(P)$ denote the set of executions of P. Finally, we write $(\sigma, \sigma') \in [\![P]\!]$ if there exists an execution of P with initial state σ and final state σ'; and we say that P is strongly terminating, written $P \Downarrow^\star$, iff for every $t \in \mathsf{Tr}(P)$ there exists $t' \in \mathsf{Ex}(P)$ such that t is a prefix of t'. For example, the abstract program in the right of Fig. 1 is strongly terminating, since every execution trace can be extended to a terminating trace by suitable choices when evaluating the non-deterministic guards.

3.1 Synchronized Products

Informally, a product of two programs is a program that combines their effects. We begin with a weaker definition (Def. 3) which only guarantees that the behavior of products is included in the behavior of their constituents. Then, we provide a sufficient condition (Def. 4) for the behavior of products to coincide with the behavior of its constituents.

One practical goal of this article is to be able to perform relational reasoning about programs that are written in the same language, by using off-the-shelf verification tools for this language. The embedding relies on separability; our conditions are inspired from self-composition [5], and are reminiscent of the monotonicity and frame properties of separation logic [19].

Assume given two functions $\pi_1, \pi_2 : \mathcal{S} \to \mathcal{S}$ s.t. for all $\sigma, \sigma' \in \mathcal{S}$, $\sigma = \sigma'$ iff $\pi_1(\sigma) = \pi_1(\sigma')$ and $\pi_2(\sigma) = \pi_2(\sigma')$. Given two states $\sigma_1, \sigma_2 \in \mathcal{S}$, we define $\sigma_1 \uplus \sigma_2 \in \mathcal{S}$ to be the unique, if it exists, state σ s.t. $\pi_1(\sigma) = \sigma_1$ and $\pi_2(\sigma) = \sigma_2$.

Definition 2 (Separable statements). *A statement c is a left statement iff for all σ_1, σ_2 in $\mathcal{S}$ s.t. $\sigma_1 \uplus \sigma_2$ is defined:*

1. *for all $\sigma_1' \in \mathcal{S}$, if $(\sigma_1, \sigma_1') \in [\![c]\!]$, then $\sigma_1' \uplus \sigma_2$ is defined and $(\sigma_1 \uplus \sigma_2, \sigma_1' \uplus \sigma_2) \in [\![c]\!]$;*
2. *for all $\sigma' \in \mathcal{S}$, if $(\sigma_1 \uplus \sigma_2, \sigma') \in [\![c]\!]$, then there exists $\sigma_1' \in \mathcal{S}$ s.t. $(\sigma_1, \sigma_1') \in [\![c]\!]$ and $\sigma_1' \uplus \sigma_2 = \sigma'$.*

Right statements are defined symmetrically. Two statements c_1 and c_2 are separable iff c_1 is a left statement and c_2 is a right statement. Finally, two programs P_1 and P_2 are separable iff P_1 is a left program, i.e. it only contains left statements, and P_2 is a right program, i.e. it only contains right statements. In this section, we let $P_1{=}\langle\mathcal{N}_1,\mathcal{E}_1,G_1\rangle$ and $P_2{=}\langle\mathcal{N}_2,\mathcal{E}_2,G_2\rangle$ be separable programs.

Example 1. The programs in Fig. 1 manipulate disjoint fragments of scalar state, thus they are clearly separable. Dynamic memory manipulation may break separability if both the left and right programs invoke a non-deterministic allocator. However, in this particular example one of the product components does not manipulate the heap.

Definition 3 (Product). *Let $P = \langle\mathcal{N},\mathcal{E},G\rangle$ be a program with statements in* Stmt. *P is a product of P_1 and P_2, written $P{\in}P_1{\times}P_2$, iff $\mathcal{N}\subseteq\mathcal{N}_1{\times}\mathcal{N}_2$, and $(\mathsf{in}_1,\mathsf{in}_2){\in}\mathcal{N}$ and for all $(l_1,l_2){\in}\mathcal{N}$ $l_1{=}\mathsf{out}_1$ iff $l_2{=}\mathsf{out}_2$, and every edge $e\in\mathcal{E}$ is of one of the forms:*

- left edge: $(l_1,l_2)\overset{\mathsf{l}}{\mapsto}(l_1',l_2)$, *with $\langle l_1,l_1'\rangle$ in $\mathcal{E}_1$, and $[\![G\,e]\!]=[\![G_1\,\langle l_1,l_1'\rangle]\!]$;*
- synchronous edge: $(l_1,l_2)\Mapsto(l_1',l_2')$, *with edges $\langle l_1,l_1'\rangle$ in $\mathcal{E}_1$ and $\langle l_2,l_2'\rangle$ in $\mathcal{E}_2$, and $[\![G\,e]\!]{=}[\![G_1\,\langle l_1,l_1'\rangle]\!]\circ[\![G_2\,\langle l_2,l_2'\rangle]\!]$; or*
- right edge: $(l_1,l_2)\overset{\mathsf{r}}{\mapsto}(l_1,l_2')$, *with $\langle l_2,l_2'\rangle$ in $\mathcal{E}_2$, and $[\![G\,e]\!]=[\![G_2\,\langle l_2,l_2'\rangle]\!]$.*

For simplicity, the notion of product program is defined for two programs of the same language. However, the definition readily extends to 2-languages products, i.e. products of programs written in two distinct languages. Alternatively, 2-languages products can be encoded in our setting: given two programming languages with statements in Stmt_1 and Stmt_2 respectively, and with state spaces $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively and semantics $[\![.]\!]_1:\mathsf{Stmt}_1\to\mathcal{P}(\mathcal{S}_1\times\mathcal{S}_1)$ and $[\![.]\!]_2:\mathsf{Stmt}_2\to\mathcal{P}(\mathcal{S}_2\times\mathcal{S}_2)$, one can define $\mathsf{Stmt}=\mathsf{Stmt}_1+\mathsf{Stmt}_2$, and $\mathcal{S}=\mathcal{S}_1+\mathcal{S}_2$, and $[\![.]\!]=[\![.]\!]_1+[\![.]\!]_2$. Then, programs of the first and second languages can be embedded in a semantic-preserving manner into the "sum" language, and one can use the notion of product program the usual way.

Example 2. The definition of products ensures that every edge in $\mathcal{E}$ represents either an execution step of program P_1, an execution step of program P_2, or a pair of simultaneous steps of both programs. The program product in Fig. 4 contains both synchronous and left edges. In this particular example, the left edges represent portions of the original program that are sliced out in the abstract program, since they do not have an effect on the validity of the boolean predicates.

Products underapproximate the behavior of their constituents, i.e, every trace of $P\in P_1\times P_2$ is a combination of a trace of P_1 and a trace of P_2. We formalize this fact using left and right projections of traces. The left projection of an execution step in P is defined by case analysis: 1. if $\langle(l_1,l_2),\sigma\rangle\rightsquigarrow\langle(l_1',l_2'),\sigma'\rangle$ and either $(l_1,l_2)\overset{\mathsf{l}}{\mapsto}(l_1',l_2')$ or $(l_1,l_2)\Mapsto(l_1',l_2')$, then the left projection is defined as $\langle l_1,\pi_1(\sigma)\rangle\rightsquigarrow\langle l_1',\pi_1(\sigma')\rangle$; 2. otherwise, the left projection is undefined. The left projection $\pi_1(t)$ of a trace t is then defined as the concatenation of the left projections of its steps (steps with undefined projections are omitted). The right projection $\pi_2(t)$ of a trace t is defined in a similar way.

Lemma 1. *Let $P{\in}P_1{\times}P_2$. For all $t{\in}\mathsf{Tr}(P)$, $\pi_1(t){\in}\mathsf{Tr}(P_1)$ and $\pi_2(t){\in}\mathsf{Tr}(P_2)$.*

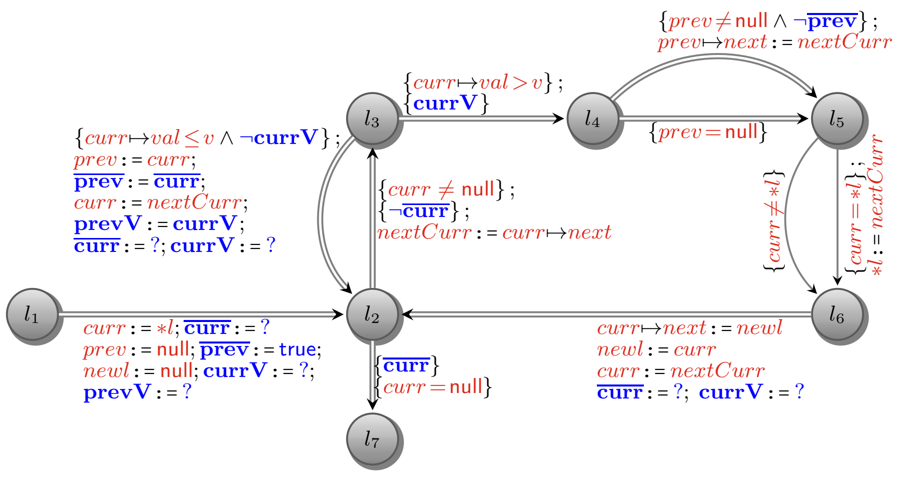

$\{prev \neq \mathsf{null} \wedge \neg\overline{\mathbf{prev}}\}$;
$prev \mapsto next := nextCurr$
$\{curr \mapsto val > v\}$; $\{\mathbf{currV}\}$
l_3
l_4
l_5
$\{curr \mapsto val \leq v \wedge \neg\mathbf{currV}\}$; $prev := curr$; $\overline{\mathbf{prev}} := \overline{\mathbf{curr}}$; $curr := nextCurr$; $\mathbf{prevV} := \mathbf{currV}$; $\overline{\mathbf{curr}} := ?$; $\mathbf{currV} := ?$
$\{prev = \mathsf{null}\}$
$\{curr \neq \mathsf{null}\}$; $\{\neg\overline{\mathbf{curr}}\}$; $nextCurr := curr \mapsto next$
$\{curr \neq *l\}$
$\{curr = *l\}$; $*l := nextCurr$
l_1
l_2
l_6
$curr := *l$; $\overline{\mathbf{curr}} := ?$ $prev := \mathsf{null}$; $\overline{\mathbf{prev}} := \mathsf{true}$; $newl := \mathsf{null}$; $\mathbf{currV} := ?$; $\mathbf{prevV} := ?$
$curr \mapsto next := newl$ $newl := curr$ $curr := nextCurr$ $\overline{\mathbf{curr}} := ?$; $\mathbf{currV} := ?$
$\{\overline{\mathbf{curr}}\}$ $\{curr = \mathsf{null}\}$
l_7

Fig. 4. Predicate abstraction example — Product program

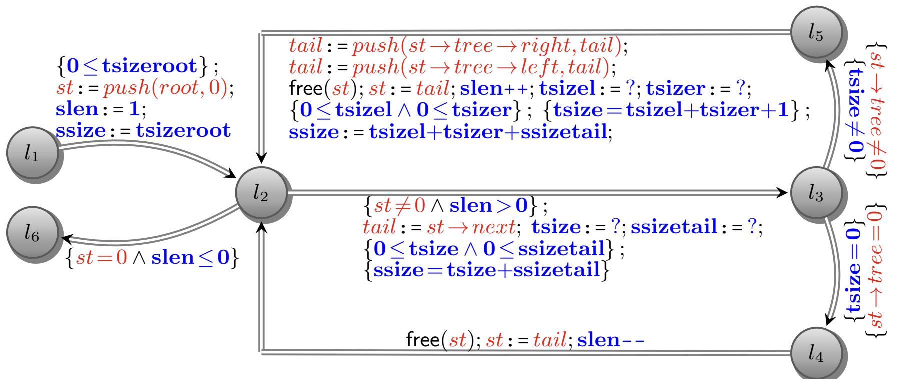

l_5
$tail := push(st \to tree \to right, tail)$; $tail := push(st \to tree \to left, tail)$; $\mathsf{free}(st)$; $st := tail$; $\mathbf{slen}{+}{+}$; $\mathbf{tsizel} := ?$; $\mathbf{tsizer} := ?$; $\{\mathbf{0} \leq \mathbf{tsizel} \wedge \mathbf{0} \leq \mathbf{tsizer}\}$; $\{\mathbf{tsize} = \mathbf{tsizel} + \mathbf{tsizer} + \mathbf{1}\}$; $\mathbf{ssize} := \mathbf{tsizel} + \mathbf{tsizer} + \mathbf{ssizetail}$;
$\{\mathbf{0} \leq \mathbf{tsizeroot}\}$; $st := push(root, 0)$; $\mathbf{slen} := \mathbf{1}$; $\mathbf{ssize} := \mathbf{tsizeroot}$
l_1
l_2
l_3
l_6
$\{st = 0 \wedge \mathbf{slen} \leq \mathbf{0}\}$
$\{st \neq 0 \wedge \mathbf{slen} > \mathbf{0}\}$; $tail := st \to next$; $\mathbf{tsize} := ?$; $\mathbf{ssizetail} := ?$; $\{\mathbf{0} \leq \mathbf{tsize} \wedge \mathbf{0} \leq \mathbf{ssizetail}\}$; $\{\mathbf{ssize} = \mathbf{tsize} + \mathbf{ssizetail}\}$
$\{\mathbf{tsize} \neq \mathbf{0}\}$ $\{st \to tree \neq 0\}$
$\{\mathbf{tsize} = \mathbf{0}\}$ $\{st \to tree = 0\}$
$\mathsf{free}(st)$; $st := tail$; $\mathbf{slen}{-}{-}$
l_4

Fig. 5. Numeric abstraction — Product Program

The notion of left product guarantees that some converse of Lemma 1 holds. Informally, a program P is a left product of P_1 and P_2 if P can progress at any program point where P_1 can progress. More precisely, we informally want that for every node (l_1, l_2) such that P_1 can progress from l_1 to l'_1 and P_2 is not stuck, there exists a left or synchronous edge from (l_1, l_2) to (l'_1, l'_2). Since it would be clearly too strong to require this progress property for arbitrary states, the definition is parametrized by a precondition.

Definition 4 (Left product). *$P \in P_1 \times P_2$ is a left product w.r.t. a precondition φ, written $P \in P_1 \ltimes_\varphi P_2$, iff for every trace $t :: \langle (l_1, l_2), \sigma_1 \uplus \sigma_2 \rangle \in \mathsf{Tr}(P)$ with initial state σ such that $\varphi(\sigma)$, and for every nodes $l'_1 \in \mathcal{N}_1$ and $l'_2 \in \mathcal{N}_2$ such that $\sigma_1 \in \mathsf{dom}([\![G_1\langle l_1, l'_1\rangle]\!])$ and $\sigma_2 \in \mathsf{dom}([\![G_2\langle l_2, l'_2\rangle]\!])$, one of the following holds:*

1. *$(l_1, l_2) \overset{\mathsf{l}}{\mapsto} (l'_1, l_2)$ or $(l_1, l_2) \overset{\mathsf{r}}{\mapsto} (l_1, l'_2)$ belongs to P;*
2. *there exists an edge $(l_1, l_2) \mapsto (l'_1, l''_2)$ in P s.t. $\sigma_2 \in \mathsf{dom}([\![G_2\langle l_2, l''_2\rangle]\!])$;*

Example 3. One can verify that the product examples shown in Section 2 are left products. For the product in Fig. 4, one can deduce at node l_2 the validity of the invariant $curr = \mathsf{null} \Leftrightarrow \overline{curr}$. In order to verify the leftness condition at node l_2 one must check that every feasible transition on the left program is eventually feasible in the product program. In this particular case, the equivalent of the boolean guards holds by the invariant above.

Lemma 2 (Lifting left products). *Assume $P \in P_1 \ltimes_{\varphi} P_2$ with $P_2 \Downarrow^{\star}$. Let $t_1 \in \mathsf{Tr}(P_1)$ with initial state σ_1, and let $\sigma_2 \in \mathcal{S}$ s.t. $\varphi\,(\sigma_1 \uplus \sigma_2)$. Then there exists a trace $t \in \mathsf{Tr}(P)$ with initial state $\sigma_1 \uplus \sigma_2$ s.t. $\pi_1(t) = t_1$.*

The result above requires in general proving strong-termination of the right component P_2. However, it is often sufficient to perform a syntactic check over a program product $P \in P_1 \ltimes_{\varphi} P_2$ as suggested by the following result.

Lemma 3. *Assume $P \in P_1 \ltimes_{\varphi} P_2$ has no asynchronous right loops, i.e., that for all sequences of edges $l_1 \overset{\mathsf{r}}{\mapsto} l_2, \ldots, l_{n-1} \overset{\mathsf{r}}{\mapsto} l_n$ we have $l_1 \neq l_n$. Then $P_1 \Downarrow^{\star}$ implies $P_2 \Downarrow^{\star}$.*

It follows from the lemma above, and the fact that we are interesting in terminating executions of P_1, that it is enough to check for the absence of asynchronous right loops in the product P.

4 Logical Validation

We now show how to check the correctness of product constructions and relational specifications using standard logical frameworks. Assuming that P_1 and P_2 are separable, we cast the correctness of two programs P_1 and P_2 w.r.t. a relational specification Φ, in terms of the functional correctness of a left product $P \in P_1 \ltimes_{\Phi(\mathsf{in})} P_2$. If the statement languages of P_1 and P_2 are amenable to verification condition generation, one can generate from a product program P a set of verification conditions that ensure that P is a left product of P_1 and P_2, and that P_1 and P_2 are correct w.r.t. a relational specification Φ. For clarity, we instantiate this section to the programming model used for the examples in Section 2, and a weakest precondition calculus over first-order formulae.

Program correctness is usually expressed by a judgment of the form $\{\varphi\}\,P\,\{\psi\}$, where $P = \langle \mathcal{N}, \mathcal{E}, G \rangle$ is a program, and φ, ψ are assertions. A judgment is valid, written $\models \{\varphi\}\,P\,\{\psi\}$, iff for all states $\sigma, \sigma' \in \mathcal{S}$ s.t. $(\sigma, \sigma') \in [\![P]\!]$, if $\varphi\,\sigma$ then $\psi\,\sigma'$. One can prove the validity of triples using a variant of Hoare logic [2], or working with a compositional flow logic [17]. However, the prominent means to prove that $\{\varphi\}\,P\,\{\psi\}$ is valid is to exhibit a partial specification $\Phi : \mathcal{N} \rightharpoonup \phi$ s.t. all cycles in the graph of P go through an annotated node, i.e. a node in $\mathsf{dom}(\Phi)$; and $\mathsf{in}, \mathsf{out} \in \mathsf{dom}(\Phi)$ with $\varphi = \Phi(\mathsf{in})$ and $\psi = \Phi(\mathsf{out})$.

We adopt a simplified version of the memory model of Leroy and Blazy [13]—locations are interpreted as integer values and field accesses as pointer offsets. We introduce to the assertion language a variable h, and the non-interpreted functions load, store, alloc, and free, and the predicate Valid. We also introduce a suitable set of axioms, including for instance:

$$\begin{array}{rcl}
\mathsf{Valid}(\mathsf{h},l) & \Longrightarrow & \mathsf{load}(\mathsf{store}(\mathsf{h},l,v),l)=v \\
\mathsf{Valid}(\mathsf{h},l)\wedge\mathsf{Valid}(\mathsf{h},l')\wedge l\neq l' & \Longrightarrow & \mathsf{load}(\mathsf{store}(\mathsf{h},l',v),l)=\mathsf{load}(\mathsf{h},l) \\
\mathsf{alloc}(\mathsf{h})=(\mathsf{h}',l) & \Longrightarrow & \mathsf{Valid}(\mathsf{h}',l) \\
\mathsf{Valid}(\mathsf{h},l)\wedge l\neq l'\wedge\mathsf{free}(\mathsf{h},l)=\mathsf{h}' & \Longrightarrow & \mathsf{Valid}(\mathsf{h}',l')
\end{array}$$

The weakest precondition calculus is standard, with the exception perhaps of heap operations:

$$\begin{array}{rclrcl}
\mathsf{wp}(x:=[l],\phi) & \doteq & \phi[\mathsf{load}(\mathsf{h},l)/x] & \mathsf{wp}([l]:=x,\phi) & \doteq & \phi[\mathsf{store}(\mathsf{h},l,x)/\mathsf{h}] \\
\mathsf{wp}(\mathsf{free}(l),\phi) & \doteq & \phi[\mathsf{free}(\mathsf{h},l)/\mathsf{h}] & \mathsf{wp}(l:=\mathsf{alloc},\phi) & \doteq & \phi[\mathsf{h}^\star/\mathsf{h}]\wedge(\mathsf{h}^\star,l)=\mathsf{alloc}(\mathsf{h})
\end{array}$$

where $\mathsf{h}^\star$ stands for a fresh variable. One can use the weakest preconditions to generate a specification $\Phi^\natural$ that extends Φ to all nodes. Using the well-founded induction principle attached to partial specifications, see e.g. [7], we set

$$\Phi^\natural(l)\doteq\bigwedge_{\langle l,l'\rangle\in\mathcal{E}}\mathsf{wp}(G\langle l,l'\rangle,\Phi^\natural(l'))\qquad\text{for all } l\notin\mathsf{dom}(\Phi)$$

The logical judgement $\vdash\{\varphi\}\,P\,\{\psi\}$ is verifiable if there is a specification $\Phi^\natural$ with $\varphi\doteq\gamma(\Gamma(\mathsf{in}))$ and $\psi\doteq\gamma(\Gamma(\mathsf{out}))$ such that the verification conditions $\Phi(l)\Rightarrow\mathsf{wp}(G\langle l,l'\rangle,\Phi^\natural(l'))$ are valid for all $\langle l,l'\rangle\in\mathcal{E}$ and $l\in\mathsf{dom}(\Phi)$.

The leftness of a product can also be checked by logical means. We use a simple form of path condition, which we call edge condition, to express leftness. Formally, the edge condition $\mathsf{ec}(c)$ for a statement c is, if it exists, the unique (up to logical equivalence) formula ϕ s.t. for all states $\sigma\in\mathcal{S}$, $\sigma\in[\![\phi]\!]$ iff $\sigma\in\mathsf{dom}([\![c]\!])$. We define for every node $(l_1,l_2)\in\mathcal{N}$ and edges $\langle l_1,l_1'\rangle\in\mathcal{E}_1$ and $\langle l_2,l_2'\rangle\in\mathcal{E}_2$ s.t. $(l_1,l_2)\overset{\llcorner}{\not\mapsto}(l_1',l_2)$ and $(l_1,l_2)\overset{\lrcorner}{\not\mapsto}(l_1,l_2')$ the Φ-leftness condition as

$$\Phi(l_1,l_2)\wedge\mathsf{ec}(G_1\langle l_1,l_1'\rangle)\wedge\mathsf{ec}(G_2\langle l_2,l_2'\rangle)\Rightarrow\bigvee_{l_2'':(l_1,l_2)\Mapsto(l_1',l_2'')}\mathsf{ec}(G_2\langle l_2,l_2''\rangle)$$

and say that P is Φ-left iff all its Φ-leftness conditions are valid.

Weakest preconditions can be used to compute edge conditions. For instance one can define $\mathsf{ec}(c)$ by the clauses:

$$\begin{array}{rclrcl}
\mathsf{ec}(\mathsf{skip}) & \doteq & \mathsf{true} & \mathsf{ec}(x:=e) & \doteq & \mathsf{true} \\
\mathsf{ec}(\{b\}) & \doteq & b & \mathsf{ec}(c_1;c_2) & \doteq & \mathsf{ec}(c_1)\wedge\mathsf{wp}(c_1,\mathsf{ec}(c_2)) \\
\mathsf{ec}([l]:=x) & \doteq & \mathsf{Valid}(\mathsf{h},l) & \mathsf{ec}(x:=[l]) & \doteq & \mathsf{Valid}(\mathsf{h},l) \\
\mathsf{ec}(\mathsf{free}(l)) & \doteq & \mathsf{Valid}(\mathsf{h},l) & \mathsf{ec}(l:=\mathsf{alloc}) & \doteq & \mathsf{true}
\end{array}$$

Example 4. In order to verify the leftness of the product program in Figure 4 it is sufficient to check for every synchronous edge $(l_1,l_2)\Mapsto(l_1',l_2')$ that $\mathsf{ec}(G_1\langle l_1,l_1'\rangle)$ implies $\mathsf{ec}(G_2\langle l_2,l_2'\rangle)$. Consider for instance the product edge $\langle l_2,l_3\rangle$. The edge condition of the corresponding left edge is $curr\neq\mathsf{null}$ whereas the edge condition of the corresponding right edge is $\neg\overline{curr}$. The validity of $curr\neq\mathsf{null}\Rightarrow\neg\overline{curr}$ follows trivially from the strong invariant $curr=\mathsf{null}\Leftrightarrow\overline{curr}$.

Let $\mathcal{S}_1,\mathcal{S}_2\subseteq\mathcal{S}$ be sets of pairwise separable[1] states. From the separability hypothesis, one can embed relational assertions on $\mathcal{P}(\mathcal{S}_1\times\mathcal{S}_2)$ as assertions on the set $\{\sigma_1\uplus\sigma_2\mid$

[1] Two states σ_1 and σ_2 are separable if $\sigma_1\uplus\sigma_2$ is defined.

$\sigma_1 \in \mathcal{S}_1, \sigma_2 \in \mathcal{S}_2\}$. Relational program correctness is formalized by refinement quadruples of the form $\models \{\varphi\}P_1 \mapsto P_2\{\psi\}$, where P_1, P_2 are programs, and φ, ψ are assertions. Such refinement judgment is valid iff for all $t_1 \in \mathsf{Ex}(P_1)$ with initial state σ_1 and final state σ_1', and σ_2 s.t. $\varphi\,(\sigma_1 \uplus \sigma_2)$ and $P_2 \Downarrow^\star$, there exists $t_2 \in \mathsf{Ex}(P_2)$ with initial state σ_2 and final state σ_2' s.t. $\psi\,(\sigma_1' \uplus \sigma_2')$.

Theorem 1. *Let P_1, P_2 be separable programs and let φ, ψ be assertions. Then the judgement $\models \{\varphi\}P_1 \mapsto P_2\{\psi\}$ holds, provided there is a partial specification Φ s.t. $\varphi = \Phi(\mathsf{in}_1, \mathsf{in}_2)$ and $\psi = \Phi(\mathsf{out}_1, \mathsf{out}_2)$, and a product program $P \in P_1 \times P_2$ that is Φ-left and correct w.r.t. Φ.*

Theorem 1 provides direct proofs of correctness for many common refinement steps, e.g. replacing a non-deterministic assignment by an assignment (or a non-deterministic choice by one of its substatements). Observe that by Lemma 3 it is enough to check alternatively for the absence of right loops in the product program instead of requiring the strong-termination of P_2.

4.1 Completeness of Abstraction Validation

We briefly show that abstraction validation is relatively complete under a soundness assumption of the program abstraction procedure. To this end, we use the framework of abstract interpretation [11] to characterize sound program abstractions. Then we show that the correctness of the abstract semantics w.r.t. the verification calculus implies the verifiability of the resulting program abstraction using left products. For brevity, we only consider forward abstract semantics; the adaptation to backward semantics is straightforward.

In the rest of this section we let $I = \langle A, [\![.]\!]^\sharp \rangle$ be an abstract semantics composed of

– an abstract domain A, that can be interpreted as assertions over states;
– an abstract interpretation function $[\![.]\!]^\sharp : \mathsf{Stmt} \to A \to A$ for statements: $[\![c]\!]^\sharp$ approximates the execution of statement c in the abstract domain;

We assume the existence of a concretization function γ from abstract values in A to first order formulae. We need to assume also the soundness of the abstract semantics $I = \langle A, [\![.]\!]^\sharp \rangle$ w.r.t. the wp calculus, i.e. that for all $c \in \mathsf{Stmt}$ and $a \in A$, $\Phi \doteq \gamma(a) \Rightarrow \mathsf{wp}(c, \gamma([\![c]\!]^\sharp a))$ is a verifiable formula. We also assume a standard characterization of valid post-fixpoints: a labeling $\Gamma : \mathcal{N} \to A$ is a post-fixpoint of the abstract semantics I if for all $\langle l, l' \rangle \in \mathcal{E}$ $[\![G\langle l, l' \rangle]\!]\Gamma(l) \sqsubseteq \Gamma(l')$.

Let $P = \langle \mathcal{N}, \mathcal{E}, G \rangle$ and $\hat{P} = \langle \mathcal{N}, \mathcal{E}, \hat{G} \rangle$ be separable programs, and assume that the abstract domain A represents relations between the disjoint memories of P and $\hat{P}$. We say that a $\hat{P}$ is a sound abstraction of P w.r.t. a labeling $\Gamma : \mathcal{N} \to A$ if for all $e = \langle l, l' \rangle \in \mathcal{E}$ we have

$$[\![G\,e; \hat{G}\,e]\!]^\sharp \Gamma(l) \sqsubseteq \Gamma(l')$$

Lemma 4. *Let P be a program, $I = \langle A, [\![.]\!]^\sharp \rangle$ an abstract semantics, and P' a sound abstraction of P w.r.t. a post-fixpoint $\Gamma : \mathcal{N} \to A$. Assume that $\gamma\, a \Rightarrow \gamma\, a'$ is verifiable* *for all $a, a' \in A$ s.t. $a \sqsubseteq a'$. If I is sound w.r.t the* wp *calculus then there exists $Q \in P \ltimes_\varphi P'$ s.t. $\vdash \{\varphi\}\, Q\, \{\psi\}$ is a verifiable judgement, where $\varphi \doteq \gamma(\Gamma(\mathsf{in}))$ and $\psi \doteq \gamma(\Gamma(\mathsf{out}))$.*

It follows from the lemma above that, under mild conditions, if P' is an abstract program computed from P using a sound abstract semantics, then one can verify that P is correctly abstracted by P'. Besides, in settings in which the abstract semantics is defined as a strongest postcondition calculus, as in e.g. predicate abstraction, abstraction validation is decidable. Indeed, it is sufficient that the decision procedure used for program verification is as complete as the one used by the program abstraction algorithm.

5 Full Products

We introduce a symmetric variant of the notion of left product of Section 3, which allows verifying one-to-one correspondences between traces of a source and transformed program, as required by translation validation.

Definition 5 (Full product). *$P \in P_1 \times P_2$ is a full product w.r.t. a precondition φ, written $P \in P_1 \bowtie_\varphi P_2$, iff for every trace $t :: \langle (l_1, l_2), \sigma_1 \uplus \sigma_2 \rangle \in \mathsf{Tr}(P)$ with initial state σ such that $\varphi(\sigma)$, and for every nodes $l'_1 \in \mathcal{N}_1$ and $l'_2 \in \mathcal{N}_2$ such that $\sigma_1 \in \mathsf{dom}([\![G_1\langle l_1, l'_1\rangle]\!])$ and $\sigma_2 \in \mathsf{dom}([\![G_2\langle l_2, l'_2\rangle]\!])$, one of the edges $(l_1, l_2) \overset{\mathsf{l}}{\mapsto} (l'_1, l_2)$, $(l_1, l_2) \overset{\mathsf{r}}{\mapsto} (l_1, l'_2)$, or $(l_1, l_2) \Mapsto (l'_1, l'_2)$ belongs to P;*

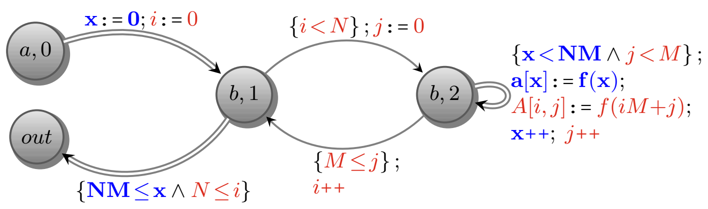
Product fullness is a stronger property than being both left and right. Indeed, requiring the existence of the edge $(l_1, l_2) \Mapsto (l'_1, l'_2)$ is stronger that requiring the existence of $(l_1, l_2) \Mapsto (l'_1, l''_2)$ or $(l_1, l_2) \Mapsto (l''_1, l'_2)$ for some l''_1 or l''_2. In a deterministic setting, however, a product program P is full iff P is left and right. Moreover, for deterministic programs, left products and full products coincide: assume that P_2 is deterministic, i.e. if $\sigma \in \mathsf{dom}([\![G\langle l, l'\rangle]\!])$ and $\sigma \in \mathsf{dom}([\![G\langle l, l''\rangle]\!])$ then $l' = l''$. Then $P \in P_1 \ltimes_\varphi P_2$ iff $P \in P_1 \bowtie_\varphi P_2$. This has practical advantages when verifying deterministic programs, since it is sufficient to discharge verification conditions for leftness to formally verify the fullness of a program product.

Relational correctness is formalized by judgments of the form $\{\varphi\}\, P_1 \sim P_2\, \{\psi\}$, where P_1, P_2 are separable programs, and φ, ψ are assertions. A relational judgment is valid, written $\models \{\varphi\}\, P_1 \sim P_2\, \{\psi\}$, iff for all $t_1 \in \mathsf{Ex}(P_1)$ and $t_2 \in \mathsf{Ex}(P_2)$ with initial states σ_1 and σ_2, and final states σ'_1 and σ'_2, $\varphi\,(\sigma_1 \uplus \sigma_2)$ imply $\psi\,(\sigma'_1 \uplus \sigma'_2)$. Full products yield a symmetric variant of Theorem 1.

Theorem 2. *Let P_1, P_2 be deterministic separable programs and let φ, ψ be assertions. Then $\models \{\varphi\}\, P_1 \sim P_2\, \{\psi\}$, provided there is a partial specification Φ and a product program $P \in P_1 \times P_2$ s.t. $\varphi = \Phi(\mathsf{in}_1, \mathsf{in}_2)$, $\psi = \Phi(\mathsf{out}_1, \mathsf{out}_2)$, and P is Φ-left and correct w.r.t. Φ.*

Example 5. Figure 6 shows a full product for the validation of the loop tiling example in Fig. 3. From Theorem 2, one can show that the product is full by proving that is Φ-left, where Φ is shown in the figure. E.g. Φ-leftness at the node $(b, 2)$ and for the edges $\langle b, b\rangle$ and $\langle 2, 2\rangle$ reduces to showing that $\Phi(b, 2) \wedge \mathsf{ec}(2, 2) \wedge \mathsf{ec}(b, b)$ implies $\mathsf{ec}((b, 2) \Mapsto (b, 2))$.

Product


$a, 0$
$b, 1$
$b, 2$
out
$\mathbf{x} := \mathbf{0}; i := 0$
$\{i < N\}; j := 0$
$\{\mathbf{x} < \mathbf{NM} \wedge j < M\};$
$\mathbf{a}[\mathbf{x}] := \mathbf{f}(\mathbf{x});$
$A[i, j] := f(iM+j);$
$\mathbf{x}{+}{+};\ j{+}{+}$
$\{M \le j\};$
$i{+}{+}$
$\{\mathbf{NM} \le \mathbf{x} \wedge N \le i\}$

Specification

$\Phi(a, 0) \doteq \mathsf{true}$
$\Phi(b, 2) \doteq x = iM+j \wedge i < N \wedge j \le M \wedge \varphi(i) \wedge \forall r.\, 0 \le r < j \Rightarrow A[i, r] = a[iM+r]$
$\Phi(out) \doteq \varphi(N)$
where $\varphi(i) \doteq \forall l, r.\, 0 \le l < i \wedge 0 \le r < M \Rightarrow A[l, r] = a[lM+r]$

Fig. 6. Loop tiling example — Product program

6 Implementation

We have implemented a proof of concept verification plugin in the Frama-C environment. We have used our this plugin to validate abstraction examples for list traversing algorithms.

The plugin receives as input a file with a program, its abstraction, and a predicate that describes the relation between the abstract and concrete states, using the ANSI C Specification Language (ACSL). A product of the supplied programs is constructed by following the program graphs and deciding at each branch statement whether to introduce a right, left or synchronized edge, and generating additional program annotations. Non-deterministic assignments are modeled in abstract programs with the use of undefined functions, and assert statements were added to introduce hypotheses regarding the non-deterministic output values. In order to deal with the weakness of the alias analysis, we added some memory disjointness annotations manually.

The final annotated product program is fed into the Frama-C Jessie plugin, which translates the C product program into Why's intermediate language and discharges the verification conditions using the available SMT solvers (AltErgo, Simplify, Z3, etc.). Figure 7 depicts the interaction of the plugin with other components of the framework.

7 Related Work

Our technique builds upon earlier work on relational verification using product programs [4,20], and is closely related to relational logics [8,18] used to reason about compiler correctness and program equivalence. Furthermore, there exist strong connections between abstraction validation and refinement proofs—refinement can be viewed as a form of contextual approximation. In particular, developing connections between our method and proof methods for program refinement, such as the refinement calculus [15], or refinement with angelic non-determinism [10] is left for future work.

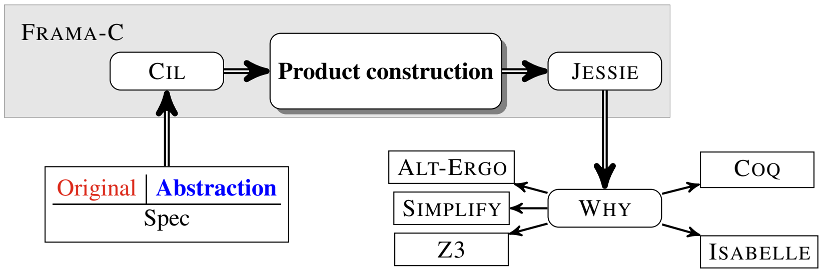


Fig. 7. Tool architecture

Abstraction validation may be seen as an instance of result checking, i.e. of the a posteriori validation of a computed result, in the context of program analysis and program transformations algorithms. In this sense, it is closely related to translation validation [21] and abstraction checking for embedded systems [9].

8 Conclusion

Asymmetric products provide a convenient means to validate relational properties using standard verification technology. They provide an automated method for reducing a refinement task to a functional verification task, and allow the validation of a broad set of program optimizations. Their applicability has been illustrated with the implementation a product construction prototype. In the future, we intend to used asymmetric products for performing a certified complexity analysis of cryptographic games [6]. Another target for future work is to broaden the scope of relational validation to object-oriented and concurrent programs.

Acknowledgments. Partially funded by European Projects FP7-231620 HATS and FP7-256980 NESSoS, Spanish project TIN2009-14599 DESAFIOS 10, Madrid Regional project S2009TIC-1465 PROMETIDOS. C. Kunz is funded by the Spanish Juan de la Cierva programme (JCI-2010-08550). Juan Manuel Crespo is funded by FPI Spanish programme (BES-2010-031271).

References

1. Ball, T., Majumdar, R., Millstein, T.D., Rajamani, S.K.: Automatic predicate abstraction of C programs. In: Programming Languages Design and Implementation, pp. 203–213 (2001)
2. Bannwart, F.Y., Müller, P.: A program logic for bytecode. Electronic Notes in Theoretical Computer Science 141, 255–273 (2005)
3. Barrett, C.W., Fang, Y., Goldberg, B., Hu, Y., Pnueli, A., Zuck, L.D.: TVOC: A Translation Validator for Optimizing Compilers. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 291–295. Springer, Heidelberg (2005)
4. Barthe, G., Crespo, J.M., Kunz, C.: Relational Verification Using Product Programs. In: Butler, M., Schulte, W. (eds.) FM 2011. LNCS, vol. 6664, pp. 200–214. Springer, Heidelberg (2011)

5. Barthe, G., D'Argenio, P., Rezk, T.: Secure Information Flow by Self-Composition. In: Foccardi, R. (ed.) Computer Security Foundations Workshop, pp. 100–114. IEEE Press (2004)
6. Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z.: Computer-Aided Security Proofs for the Working Cryptographer. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 71–90. Springer, Heidelberg (2011)
7. Barthe, G., Kunz, C.: Certificate Translation in Abstract Interpretation. In: Drossopoulou, S. (ed.) ESOP 2008. LNCS, vol. 4960, pp. 368–382. Springer, Heidelberg (2008)
8. Benton, N.: Simple relational correctness proofs for static analyses and program transformations. In: Jones, N.D., Leroy, X. (eds.) Principles of Programming Languages, pp. 14–25. ACM Press (2004)
9. Blech, J.O., Schaefer, I., Poetzsch-Heffter, A.: Translation Validation of System Abstractions. In: Sokolsky, O., Taşıran, S. (eds.) RV 2007. LNCS, vol. 4839, pp. 139–150. Springer, Heidelberg (2007)
10. Bodík, R., Chandra, S., Galenson, J., Kimelman, D., Tung, N., Barman, S., Rodarmor, C.: Programming with angelic nondeterminism. In: Principles of Programming Languages, pp. 339–352 (2010)
11. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Principles of Programming Languages, pp. 238–252 (1977)
12. Graf, S., Saïdi, H.: Construction of Abstract State Graphs with PVS. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)
13. Leroy, X., Blazy, S.: Formal verification of a C-like memory model and its uses for verifying program transformations. J. Autom. Reasoning 41(1), 1–31 (2008)
14. Magill, S., Tsai, M.-H., Lee, P., Tsay, Y.-K.: Automatic numeric abstractions for heap-manipulating programs. In: Hermenegildo, M., Palsberg, J. (eds.) Principles of Programming Languages, pp. 211–222. ACM (2010)
15. Morgan, C.: Programming from specifications. Prentice-Hall International Series in Computer Science. Prentice-Hall, Inc. (June 1990)
16. Pnueli, A., Siegel, M., Singerman, E.: Translation Validation. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 151–166. Springer, Heidelberg (1998)
17. Tan, G., Appel, A.W.: A Compositional Logic for Control Flow. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 80–94. Springer, Heidelberg (2006)
18. Yang, H.: Relational separation logic. Theoretical Computer Science 375(1-3), 308–334 (2007)
19. Yang, H., O'Hearn, P.W.: A Semantic Basis for Local Reasoning. In: Nielsen, M., Engberg, U. (eds.) FOSSACS 2002. LNCS, vol. 2303, pp. 402–416. Springer, Heidelberg (2002)
20. Zaks, A., Pnueli, A.: CoVaC: Compiler Validation by Program Analysis of the Cross-Product. In: Cuellar, J., Sere, K. (eds.) FM 2008. LNCS, vol. 5014, pp. 35–51. Springer, Heidelberg (2008)
21. Zuck, L.D., Pnueli, A., Goldberg, B.: Voc: A methodology for the translation validation of optimizing compilers. J. UCS 9(3), 223–247 (2003)

Assignment Calculus: A Pure Imperative Language

Marc Bender and Jeffery Zucker[⋆]

Department of Computing and Software, McMaster University,
Hamilton, Ont. L8S 4L8, Canada
{bendermm,zucker}@mcmaster.ca

Abstract. We undertake a study of *imperative computation.* Beginning with a philosophical analysis of the distinction between imperative and functional language features, we define a (pure) *imperative language* as one whose constructs are (inherently) *referentially opaque.* We then give a definition of a *computation language* by identifying desirable properties for such a language.

We present a new pure imperative computation language, *Assignment Calculus* $\boldsymbol{AC}$. The main idea behind $\boldsymbol{AC}$ is based on the insight of T. Janssen that Montague's modal operators of *intension* and *extension*, developed for the study of natural language semantics, are also useful for the semantics of programming language features such as assignments and pointers. $\boldsymbol{AC}$ consists of only four basic constructs, *assignment* '$\boldsymbol{X} := \mathbf{t}$', *sequence* '$\mathbf{t}\,;\mathbf{u}$', *procedure formation* '$¡\mathbf{t}$' and *procedure invocation* '$!\mathbf{t}$'. Two interpretations are given for $\boldsymbol{AC}$: an *operational semantics* and a *term-rewriting* system; these interpretations turn out to be equivalent.

1 Introduction

What is a pure imperative language? This paper attempts to answer this question by pursuing one possible definition: a pure imperative language is one whose operators are fundamentally *referentially opaque.*

In Section 2 we give a discussion of this approach, involving *referential transparency* and *opacity*, the *substitutivity* principle and *intensionality*, with natural-language examples. We show that these problems are also present in imperative programming languages, and introduce our imperative computation language Assignment Calculus, $\boldsymbol{AC}$.

Section 3 presents the syntax and operational semantics of $\boldsymbol{AC}$, with a running example which will be used throughout the paper. We examine an important aspect of $\boldsymbol{AC}$: *state backtracking.*

In Section 4 a *term rewriting system* for $\boldsymbol{AC}$ is presented, and used with our running example. We state the equivalence of the operational semantics and rewriting system, and (without proof) a confluence theorem for this system. We

⋆ Research for this paper was supported by the Natural Sciences and Engineering Research Council of Canada and by the McMaster Center for Software Certification.

conclude in Section 5 with a summary of the significance of the ideas presented here, a brief examination of similar work, and possible lines of research.

A comment on notation: syntax is represented by bold text; the equivalence of two syntactic entities is indicated by '≡'.

This paper is based on the thesis [Ben10], which contains full proofs of most of the results stated here. The thesis also provides, among other things, a full denotational semantics for ***AC***, along with related results including the equivalence of operational, denotational, and rewriting semantics. This paper extends our work by presenting a stronger syntax-directed proof of equivalence between the operational semantics and term rewriting system.

2 Imperative Computation

To program a computer, we must provide it with instructions to arrive at the desired result. There are two types of instructions: *commands* (or statements) and *goals*. The latter provide a "specification," and the computer (or compiler) must figure out how to arrive at a solution. Goals are generally written as *declarations*, often presented in a "functional" style [Bac78].

This distinction between types of instructions forms the basis of the two types of programming language: a language that is based on commands is called *imperative*, and one that is based on goals is called *declarative*.

This distinction can be traced back to two pioneers in computation theory, Alan Turing and Alonzo Church. Turing's analysis of computation [Tur36] is via a machine that executes tasks sequentially, reading from and writing to a storage device (the "tape"). Church's system [Chu41] is more abstract, taking the mathematical notion of *function* as basic, and employing only the operations of (functional) *abstraction* and *application* to express computational goals. Turing proved that their systems have the same computational power, and argued convincingly that they are *universal* (the *Church-Turing Thesis*).

The benefit of Turing's approach is its suggestion of a real computer; a variant, due to von Neumann, lies at the core of virtually every computing device in use today. Since computers are basically built on Turing's idea, so are imperative languages. Thus any imperative language contains operations for sequencing ('**;**') and reading from and writing to memory (assignments). Imperative languages are thus closely connected to practice. However, dealing with imperative languages' semantics can be tricky, and the Turing machine can be a clumsy theoretical tool.

Church's λ-calculus stems from research into the foundations of mathematics. Its language of functional abstraction and application is small, elegant, and powerful, and makes it immediately amenable to theoretical study [Bar84]. Many "functional languages" have been designed around the λ-calculus. However, a major drawback to functional languages is their lack of "machine intuition", which can make them difficult to implement.

Real computers are based on the imperative model, so compilers for functional languages are needed to translate Church-style computation into Turing's model.

Conversely, for those working in denotational semantics, giving a mathematical meaning to imperative programming languages means interpreting Turing-style computation in a Church-style language.

Can we "short-circuit" these interpretations in either direction? In other words, can we either (a) build a computer on Church's notion, or (b) design a formal language that embodies Turing's conception? In this paper we focus on question (b). In this regard, it is somewhat surprising that there is no accepted canonical *theoretical computation language* that is fundamentally imperative in character. Our goal is to present such a language. It is not presented as "the" basic language for imperative computation, but simply as a potential candidate. First, however, we must answer a pressing question: what, exactly, is "imperative computation"?

2.1 What Is Imperative Computation?

A first attempt to define imperative computation could be made from the point of view of *machine behaviour.* If functional features are seen as "high-level," then we should remain close to a machine-based intuition.

The problem is that this does not sufficiently *restrict* our definition: there are many different machine architectures, instruction sets, abstractions from machine to assembly languages, and implementations of control structures; in short, there is too much freedom when working with machine intuition alone.

So we want an imperative language which is (a) as simple as possible, (b) Turing complete, and (c) "pure". How do we understand (c)? A good approach is to take the concepts of *pure imperative* and *pure functional* as "orthogonal" in some sense.

We begin by asking: what is *functional purity*?

Purely functional languages are *referentially transparent* [FH88]. This allows for *call-by-name* or *lazy* evaluation, as in Haskell [HHJW07]. Referential transparency as a concept goes back to Quine [Qui60, §30] and essentially embodies Leibniz's *principle of substitutivity of equals*:

$$\boldsymbol{e}_1 = \boldsymbol{e}_2 \implies \boldsymbol{e}(\cdots \boldsymbol{e}_1 \cdots) = \boldsymbol{e}(\cdots \boldsymbol{e}_2 \cdots).$$

The benefit of this is clear: "computation" proceeds by substituting expressions for variables. This idea is taken to an extreme in the pure λ-calculus, which reduces functional computation to its smallest possible core. Its (only!) operators are *λ-abstraction* and *application*; its only atoms are *variables.* Syntactically, application of a function to an argument is accomplished by substitution, taking advantage of referential transparency.

In fact we take the λ-calculus as the paragon of theoretical computation languages: (a) it is small, elegant and intuitive; (b) its operators represent well-understood, fundamental concepts; (c) we can rewrite its terms using simple rules; and (d) it has multiple equivalent forms of semantics. Our aim is to develop a formal language for *imperative* computation that has as many of the above virtues as possible.

Going back to our question: what is a pure imperative language? We now propose: it is a language whose features are fundamentally non-transparent, or *opaque*, i.e., substitutivity is the exception rather than the rule.

2.2 Referential Opacity

We begin with an example in natural language. Consider the sentence:

The temperature is twenty degrees and rising.

Formalizing this gives us

$$\boldsymbol{twenty}(\boldsymbol{temp}) \wedge \boldsymbol{rising}(\boldsymbol{temp}) \tag{1}$$

Suppose the temperature is $\mathbf{20°}$. Substituting this for $\boldsymbol{temp}$ in (1) (using substitutivity of equals) yields '$\boldsymbol{twenty}(\mathbf{20°})$' for the first conjunct, which is fine, but '$\boldsymbol{rising}(\mathbf{20°})$' for the second, which is not even false, but nonsense: "temperature" here means not its current value, but its value over time.

The problem is that although the predicate '$\boldsymbol{twenty}$' introduces a *transparent context*, the predicate '$\boldsymbol{rising}$' creates an *opaque context*.

Such intensional phenomena abound in natural language, and have been studied by philosophers and linguists for some time [Fre92]. They can be recognized by apparent violations of substitutivity as above. This is also the case for imperative programming languages, to which we now turn.

Consider the assignment statement $\boldsymbol{X} := \boldsymbol{X} + \mathbf{1}$. Clearly, we can substitute the current value of $\boldsymbol{X}$ for '$\boldsymbol{X}$' on the right-hand side, but not on the left-hand side; attempting the latter gives a meaningless expression. Janssen [JvEB77, Jan86] noticed that these are simple examples of *transparent* and *opaque* contexts in programming languages; and he was able to develop a compositional semantics for programming languages, dealing with assignments and pointers, comparable to what Montague had done for natural languages. This penetrating insight of Janssen's was the starting point for a line of investigation continued in [Hun90, HZ91], and further in [Ben10] and the present paper.

In fact, it turns out that opaque contexts are inherent in *all* the fundamental imperative operations.

2.3 Intensions

Frege [Fre92] analyzed the substitutivity problem. He distinguished between two kinds of meaning: *sense* (*Sinn*) and *denotation* (*Bedeutung*). He showed that, in cases where substitutivity does not hold in terms of the *denotations* of expressions, it can be *restored* if we consider, not the *denotation* of the expression, but its *sense*, which can be understood as a function from "contexts," "states," "settings," or (as in example (1)) "time instants" to *values*.

A formal system implementing Frege's ideas was developed by Church [Chu51], and a semantic interpretation by Carnap [Car47], who introduced the terms *intension* and *extension* for sense and denotation. Kripke [Kri59] rounded out the semantic treatment by providing the setting of *possible worlds* for modal logic. By combining his work with Carnap's, we can treat intension as a function from possible worlds to values.

The next major step was accomplished by Montague [Mon70, Mon73], who developed a system ***IL*** of *intensional logic* for the mathematical treatment of natural language. Next, Janssen and van Emde Boas [JvEB77, Jan86] applied Montague's techniques to imperative programming languages. By identifying possible worlds with *machine states*, they provide a strikingly elegant treatment of assignment statements and pointers.

A significant extension was accomplished by Hung and Zucker [Hun90, HZ91], who provided compositional denotational semantics for quite intricate language features such as *blocks* with local identifiers; *procedure parameters* passed by *value*, by *reference* and by *name*; and *pointers* which can be dereferenced anywhere, including the left-hand side of assignment statements. Janssen's system — specifically, Hung's version — is the genesis of Assignment Calculus ***AC***, the language to be presented in this paper.

2.4 Intentions

We begin with the observation that

the intension operator generalizes *the (parameterless) procedure.* (2)

A procedure is a function from states to *states*, and an intension is a function from states to *values*. If we include states in the set of values, then the generalization is clear. The reason that it was not noticed by Janssen or Hung is that, in Montague's systems, *states cannot be treated as values.*

Another observation is that we can allow "storage" of intensions in the state. Stored procedures are a well-known but difficult feature of imperative languages [Sco70, SS71]. They allow general recursion, and also a generalization of the treatment of pointers given by Janssen and Hung.

The resulting system is sufficiently interesting to be studied in "isolation", removing all the functional and logical components in ***DIL***, such as variables, abstraction and application. The resulting language ***AC*** is a *case study in pure imperative computation.*

3 Assignment Calculus *AC*

In attempting to create a pure imperative computation language, it is important to remain as close as possible to existing imperative languages.

An imperative programming language can perhaps be defined as a language ***L*** with the following characteristics:

1. The interpretation of ***L*** depends on some form of computer memory (state) through which data can be stored and retrieved.

2. The state consists of contents of discrete memory locations that can be read from or written to independently.
3. An $\boldsymbol{L}$-program consists of a *sequence* of explicit *commands* or instructions that fundamentally depend on (and often change) the state.
4. $\boldsymbol{L}$ contains some form of looping, branching or recursion mechanism to allow for repeated execution of program parts.

The first three characteristics are common to all imperative languages. Therefore $\boldsymbol{AC}$ includes them directly.

Characteristic 4 can be embodied in many ways: conditional branching, "goto" statements; looping constructs ("while" etc.); and recursion.

$\boldsymbol{AC}$'s approach to 4 is to employ Montague's intension operator as a *generalization of parameterless procedure*, in accordance with our observation above. In $\boldsymbol{AC}$, intensions are treated as *first-class values*: they can be defined anonymously, assigned to locations, and invoked freely.

The goals of $\boldsymbol{AC}$ are twofold: first to continue the line of research initiated by Janssen and continued by Hung and Zucker in applying Montague's work to programming languages, and secondly to attempt to provide a small, elegant, useful *core language* for imperative-style computation—a *pure imperative computation language* as defined in Section 2.

3.1 Introduction to $\boldsymbol{AC}$

$\boldsymbol{Term}$ is the set of $\boldsymbol{AC}$ terms **t**, **u**, ... Before defining $\boldsymbol{Term}$ formally, we go over the basic constructs of $\boldsymbol{AC}$ and their intuitive meanings.

1. *Locations*: $\boldsymbol{X}$, $\boldsymbol{Y}$, ... correspond to *memory locations*. The collection of all locations is the *store*.
2. *Assignment*: $\boldsymbol{X} := \mathbf{t}$. Overwrites the contents of location $\boldsymbol{X}$ with whatever **t** computes. This operation computes the *store* that results from such an update.
3. *Sequence*: **t ; u**. Interpreted as follows: first compute **t**, which returns a store, then compute **u** in this new context.
4. *Intension*: **¡t**. This is *procedure formation.* It "stops" evaluation of **t** so that **¡t** is the procedure that, when invoked, returns whatever **t** computes.
5. *Extension*: **!t**. This is *procedure invocation*, that is, it "runs" the procedure computed by **t**.

An easy way to remember our notation (different from Montague's) for intension and extension is that '**¡**' looks like a lowercase 'i', for intension, and that '**!**', an *ex*clamation mark, is used for extension.

This constitutes *pure* $\boldsymbol{AC}$. For convenience, we can add *supplementary operators*: *numerals* **n**, ..., *booleans* **b**, ... and their standard operations.

We will use the following as a running example of an $\boldsymbol{AC}$ term:

$$\boldsymbol{P} := \text{¡}\boldsymbol{X};\ \boldsymbol{X} := \mathbf{1};\ !\boldsymbol{P}. \tag{3}$$

This term sets $\boldsymbol{P}$ to the procedure that returns $\boldsymbol{X}$, then sets $\boldsymbol{X}$ to $\mathbf{1}$. Finally, $\boldsymbol{P}$ is invoked thus returning the current value of $\boldsymbol{X}$ which is $\mathbf{1}$.

3.2 Types and Syntax of $\boldsymbol{AC}$

$\boldsymbol{AC}$ is a simply typed language in the sense of Church [Chu40].

Definition 3.1 (Types). *The set* $\boldsymbol{Type}$ *of types* $\boldsymbol{\tau}, \ldots,$ *is generated by:*

$$\boldsymbol{\tau} \ ::= \ \mathsf{B} \mid \mathsf{N} \mid \mathsf{S} \mid \mathsf{S}\!\rightarrow\!\boldsymbol{\tau},$$

where B, N *and* S *are the types of booleans, naturals and stores respectively, and* $\mathsf{S}\!\rightarrow\!\boldsymbol{\tau}$ *is that of* intensions *(procedures) which return values of type* $\boldsymbol{\tau}$*. (The base types* B *and* N *are added only to handle the supplementary operators described above; for pure* $\boldsymbol{AC}$*, they can be removed.)*

Now we define the set of terms $\boldsymbol{Term}$ of $\boldsymbol{AC}$. For each type $\boldsymbol{\tau}$, the sets $\boldsymbol{Term}^{\boldsymbol{\tau}}$ of terms of type $\boldsymbol{\tau}$ are defined by mutual recursion:

Definition 3.2 (Syntax of $\boldsymbol{AC}$)

1. $\boldsymbol{X} \in \boldsymbol{Loc}^{\boldsymbol{\tau}} \quad \Rightarrow \quad \boldsymbol{X} \in \boldsymbol{Term}^{\boldsymbol{\tau}}$
2. $\mathbf{t} \in \boldsymbol{Term}^{\boldsymbol{\tau}} \quad \Rightarrow \quad ¡\mathbf{t} \in \boldsymbol{Term}^{\mathsf{S}\rightarrow\boldsymbol{\tau}}$
3. $\mathbf{t} \in \boldsymbol{Term}^{\mathsf{S}\rightarrow\boldsymbol{\tau}} \quad \Rightarrow \quad !\mathbf{t} \in \boldsymbol{Term}^{\boldsymbol{\tau}}$
4. $\boldsymbol{X} \in \boldsymbol{Loc}^{\boldsymbol{\tau}},\ \mathbf{t} \in \boldsymbol{Term}^{\boldsymbol{\tau}} \quad \Rightarrow \quad \boldsymbol{X}^{\boldsymbol{\tau}} := \mathbf{t}^{\boldsymbol{\tau}} \in \boldsymbol{Term}^{\mathsf{S}}$
5. $\mathbf{t} \in \boldsymbol{Term}^{\mathsf{S}},\ \mathbf{u} \in \boldsymbol{Term}^{\boldsymbol{\tau}} \quad \Rightarrow \quad \mathbf{t}\,;\mathbf{u} \in \boldsymbol{Term}^{\boldsymbol{\tau}}$

Supplementary operators are defined in a standard way [Ben10].

Notice that the intension of a term $\mathbf{t}$ of type $\boldsymbol{\tau}$ is of type $\mathsf{S}\!\rightarrow\!\boldsymbol{\tau}$, and the extension of an (intensional) term $\mathbf{t}$ of type $\mathsf{S}\!\rightarrow\!\boldsymbol{\tau}$ is of type $\boldsymbol{\tau}$. The assignment construct is (always) of type S. Most interestingly, the sequence operator allows terms of type other than S to the right of assignments; for example we have the term $\boldsymbol{X}^{\boldsymbol{\tau}} := \boldsymbol{X} + \mathbf{1}\,; \boldsymbol{X}$ of type N. This is discussed further in §3.5.

Assignments are of type S due to the fact that they return stores, and the type of the location (on the left-hand side) and the assigned term must be in agreement. We do not allow locations of type S, but we *do* allow locations of type $\mathsf{S}\!\rightarrow\!\boldsymbol{\tau}$ for any $\boldsymbol{\tau}$. This amounts to storage of intensions in the store, which accounts for much of $\boldsymbol{AC}$'s expressive power.

As an example, think of $¡\mathbf{t}$ as representing the "text" of $\mathbf{t}$, which, in an actual computer, is the way that procedures and programs are stored. Now consider the term

$$\boldsymbol{X} := ¡!\boldsymbol{X},$$

which is read "store in $\boldsymbol{X}$ the procedure that invokes $\boldsymbol{X}$". Once this action is performed, an invocation of $\boldsymbol{X}$

$$\boldsymbol{X} := ¡!\boldsymbol{X};\ \ !\boldsymbol{X}$$

will proceed to invoke $\boldsymbol{X}$ again and again, leading to divergence.

3.3 Operational Semantics of *AC*

We access the contents of $\boldsymbol{X}$ in a store ς by function application $\varsigma(\boldsymbol{X})$.

We will use the standard notion of *function variant* to update the contents of a location, where the variant $f[x/d]$ of a function f at x for d is defined by: $(f[x/d])(x) = d$ and $(f[x/d])(y) = f(y)$ for $y \neq x$.

We employ "big-step" operational semantics [Win93]. The rules define the *computation relation* $\Downarrow \subset ((\boldsymbol{Term} \times \boldsymbol{Store}) \times (\boldsymbol{Term} \cup \boldsymbol{Store}))$. Really, since a term can compute *either* a store (if the term is of type $\mathbf{S}$) *or* another term (if it is of any type other than $\mathbf{S}$), the computation relation can be broken into two disjoint relations $\Downarrow_c \subset ((\boldsymbol{Term}^{\mathbf{S}} \times \boldsymbol{Store}) \times \boldsymbol{Store})$ and $\Downarrow_v \subset ((\boldsymbol{Term}^{\boldsymbol{\tau}} \times \boldsymbol{Store}) \times \boldsymbol{Term})$. However, since the rules are so simple, we choose instead to use the metavariable $\mathbf{d}$ to range over *both* terms and stores, and give the rules as follows:

Definition 3.3 (Operational semantics of *AC*). *First, the rules for locations, assignment and sequence are standard [Win93]:*

$$\dfrac{\varsigma(\boldsymbol{X}) = \mathbf{t}}{\boldsymbol{X},\ \varsigma\ \Downarrow\ \mathbf{t}} \qquad \dfrac{\mathbf{t},\ \varsigma\ \Downarrow\ \mathbf{u}}{\boldsymbol{X} := \mathbf{t},\ \varsigma\ \Downarrow\ \varsigma[\boldsymbol{X}/\mathbf{u}]} \qquad \dfrac{\mathbf{t},\ \varsigma\ \Downarrow\ \varsigma' \qquad \mathbf{u},\ \varsigma'\ \Downarrow\ \mathbf{d}}{\mathbf{t};\,\mathbf{u},\ \varsigma\ \Downarrow\ \mathbf{d}}$$

The interesting new rules are those for intension and extension:

$$\dfrac{}{\text{¡}\mathbf{t},\ \varsigma\ \Downarrow\ \text{¡}\mathbf{t}} \qquad \dfrac{\mathbf{t},\ \varsigma\ \Downarrow\ \text{¡}\mathbf{u} \qquad \mathbf{u},\ \varsigma\ \Downarrow\ \mathbf{d}}{!\mathbf{t},\ \varsigma\ \Downarrow\ \mathbf{d}}$$

The rules for supplementary operators are standard, and omitted here.

The intuition for the rules for intension and extension are as follows:

- intension "holds back" the computation of a term,
- extension "induces" computation.

Note that there are *no side-effects in* $\boldsymbol{AC}$; if a term is of type $\boldsymbol{\tau} \neq \mathbf{S}$ then it only results in a value. This is further discussed in §3.5.

Lemma 3.4. *Computation is* unique and deterministic, *i.e., there is at most one derivation for any* $\mathbf{t}, \varsigma$.

We return to our running example ([3]). Its operational interpretation is as follows. For any ς, $\varsigma' = \varsigma[\boldsymbol{P}/\text{¡}\boldsymbol{X}]$ and $\varsigma'' = \varsigma[\boldsymbol{P}/\text{¡}\boldsymbol{X}][\boldsymbol{X}/\mathbf{1}]$,

$$\dfrac{\dfrac{\text{¡}\boldsymbol{X},\ \varsigma\ \Downarrow\ \text{¡}\boldsymbol{X}}{\boldsymbol{P} := \text{¡}\boldsymbol{X},\ \varsigma\ \Downarrow\ \varsigma'} \qquad \dfrac{\dfrac{\mathbf{1},\ \varsigma'\ \Downarrow\ \mathbf{1}}{\boldsymbol{X} := \mathbf{1},\ \varsigma'\ \Downarrow\ \varsigma''} \qquad \dfrac{\dfrac{\varsigma''(\boldsymbol{P}) = \text{¡}\boldsymbol{X}}{\boldsymbol{P},\ \varsigma''\ \Downarrow\ \text{¡}\boldsymbol{X}} \qquad \dfrac{\varsigma''(\boldsymbol{X}) = \mathbf{1}}{\boldsymbol{X},\ \varsigma''\ \Downarrow\ \mathbf{1}}}{!\boldsymbol{P},\varsigma''\ \Downarrow\ \mathbf{1}}}{\boldsymbol{X} := \mathbf{1};\ !\boldsymbol{P},\ \varsigma''\ \Downarrow\ \mathbf{1}}}{\boldsymbol{P} := \text{¡}\boldsymbol{X};\ \boldsymbol{X} := \mathbf{1};\ !\boldsymbol{P},\ \varsigma\ \Downarrow\ \mathbf{1}}$$

3.4 Rigidity, Modal Closedness, and Canonicity

Definition 3.5 (Operational rigidity). *A term* t *is called* operationally rigid *iff there is a* u *s.t. all stores* ς *give* $\mathsf{t}, \varsigma \Downarrow \mathsf{u}$*. A term for which this is not the case is called operationally* non-rigid*.*

To provide a *syntactic* approximation to rigidity, we follow Montague and define the set of *modally closed* terms as follows:

Definition 3.6 (Modal Closedness). *The set of modally closed terms* $\boldsymbol{MC}$*, ranged over by* mc*, is generated by*

$$\mathsf{mc} \ ::= \ \mathsf{b} \mid \mathsf{n} \mid {}^{\mathsf{i}}\mathsf{t} \mid \mathsf{mc}_1 \boldsymbol{+} \mathsf{mc}_2 \mid \ldots$$

with similar clauses for other arithmetic, comparison and boolean operators.

Lemma 3.7. $\mathsf{t} \in \boldsymbol{MC} \Longrightarrow \mathsf{t}$ *is operationally rigid.*

Modal closedness captures the intuitive notion of a completed imperative computation, but it can leave arithmetic and boolean computations "unfinished". Terms in which all of these computations are also complete are called *canonical* and are defined by:

Definition 3.8 (Canonical terms). *The set of canonical terms* $\boldsymbol{Can}$*, ranged over by* c*, is generated by:*

$$\mathsf{c} \ ::= \ \mathsf{b} \mid \mathsf{n} \mid {}^{\mathsf{i}}\mathsf{t}$$

Clearly, $\boldsymbol{Can} \subseteq \boldsymbol{MC}$. Hence:

$$\mathsf{t} \in \boldsymbol{Can} \quad \Rightarrow \quad \mathsf{t} \in \boldsymbol{MC} \quad \Rightarrow \quad \mathsf{t} \text{ is operationally rigid.}$$

Definition 3.9 (Properness of Stores). *A store* ς *is* proper *if it maps locations only to canonical terms.*

The main result of this section is that the operational semantics is well defined, i.e., it only produces canonical terms or proper stores:

Theorem 3.10 (Properness of Operational Semantics). *If* ς *is proper, then for any* t *where* t, ς *converges:*

1. *If* t *is of type* S *then there is a proper store* ς' *s.t.* $\mathsf{t}, \varsigma \Downarrow \varsigma'$*;*
2. *Otherwise, there is a (canonical)* c *s.t.* $\mathsf{t}, \varsigma \Downarrow \mathsf{c}$*.*

Proof. By induction on derivations. Details in [Ben10].

3.5 State Backtracking

A significant difference between $\boldsymbol{AC}$ and traditional imperative languages is that there are *no side-effects.* In $\boldsymbol{AC}$, terms represent either *stores* (effects), if they are of type S, or *values* if they are of some other type.

In this respect, note that a language based on side-effects can easily be translated into $\boldsymbol{AC}$. Further, the absence of side-effects in $\boldsymbol{AC}$ leads to an interesting phenomenon: *state backtracking*.

State backtracking, or *non-persistent* or *local state update*, occurs when we have terms of the form

$$\mathsf{t}\,;\mathsf{u}^{\boldsymbol{\tau}} \quad \text{where} \quad \boldsymbol{\tau} \neq \mathsf{S} \tag{4}$$

because state changes caused by t are "lost," "localized" or "unrolled" when the value of u is returned. Consider the following example:

$$\boldsymbol{Y} := (\boldsymbol{X} := \mathbf{1};\ \boldsymbol{X}).$$

Changes to $\boldsymbol{X}$ are *local to the computation of* $\boldsymbol{Y}$, so in fact the above term is equivalent to '$\boldsymbol{Y} := \mathbf{1}$', a result which can be confirmed easily by the reader (for details, see [Ben10]).

The inclusion of terms of the form (4), makes possible a clean *rewriting system* for $\boldsymbol{AC}$: it gives us a way to "push assignments into terms". We discuss this further in §4; for now, consider this example: $\boldsymbol{X} := \mathbf{1};\ \ \boldsymbol{X} := (\boldsymbol{X} + \boldsymbol{X})$. By intuition and the operational semantics, we know that this term is equivalent to $\boldsymbol{X} := \mathbf{2}$. But how can it be possible, without overly complicated rules, to *rewrite* the former to the latter? By admitting terms like (4), we can express *intermediate steps* of the computation that could not otherwise be written. Using the rewriting rules (Definition 4.1),

$$\begin{aligned} \boldsymbol{X} := \mathbf{1};\ \boldsymbol{X} := (\boldsymbol{X} + \boldsymbol{X}) \ &\Rrightarrow\ \boldsymbol{X} := (\boldsymbol{X} := \mathbf{1};\ (\boldsymbol{X} + \boldsymbol{X})) \\ &\Rrightarrow\ \boldsymbol{X} := ((\boldsymbol{X} := \mathbf{1};\ \boldsymbol{X}) + (\boldsymbol{X} := \mathbf{1};\ \boldsymbol{X})) \\ &\Rrightarrow\ \boldsymbol{X} := (\mathbf{1} + \mathbf{1}) \ \Rrightarrow\ \boldsymbol{X} := \mathbf{2} \end{aligned}$$

Thus, based on its usefulness and the simplicity of the resulting rewriting rules, we believe that

State backtracking is a natural part of imperative computation.

4 Term Rewriting

In this section we explore $\boldsymbol{AC}$ by examining meaning-preserving transformations of terms. What we will develop is essentially the "calculus" part of Assignment Calculus—a term rewriting system whose rules are meant to capture and exploit its essential equivalences.[1]

4.1 Rewrite Rules and Properties

In order to make the rewriting definitions simpler, we adopt the convention that terms are syntactically identical regardless of parenthesization of the sequencing

[1] In developing these rules, we find significant guidance in Janssen's [Jan86] and Hung's [Hun90] work on *state-switcher reductions.*

operator; to wit, $(\mathsf{t}\,;\mathsf{u})\,;\mathsf{v} \;\equiv\; \mathsf{t}\,;(\mathsf{u}\,;\mathsf{v})$ This convention makes it much easier to express rewriting rules that govern the interaction of assignment operators.

The heart of the rewriting system is the rewriting function $\mathrel{\mid\!\!\Rrightarrow}\, : \boldsymbol{Term} \to \boldsymbol{Term}$. Recall the definition (3.6) of modally closed terms mc.

Definition 4.1. *The rewriting function* $\mathrel{\mid\!\!\Rrightarrow}$ *is given by*

$$\begin{array}{lll}
1. & !¡\mathsf{t} & \mathrel{\mid\!\!\Rrightarrow} \; \mathsf{t} \\
2. & \boldsymbol{X} := \mathsf{mc}_1;\ \mathsf{mc}_2 & \mathrel{\mid\!\!\Rrightarrow} \; \mathsf{mc}_2 \\
3. & \boldsymbol{X} := \mathsf{t};\ \boldsymbol{X} & \mathrel{\mid\!\!\Rrightarrow} \; \mathsf{t} \\
4. & \boldsymbol{X} := \mathsf{mc};\ \boldsymbol{Y} & \mathrel{\mid\!\!\Rrightarrow} \; \boldsymbol{Y} \\
5. & \boldsymbol{X} := \mathsf{mc};\ \boldsymbol{X} := \mathsf{u} & \mathrel{\mid\!\!\Rrightarrow} \; \boldsymbol{X} := (\boldsymbol{X} := \mathsf{mc};\ \mathsf{u}) \\
6. & \boldsymbol{X} := \mathsf{mc};\ \boldsymbol{Y} := \mathsf{t} & \mathrel{\mid\!\!\Rrightarrow} \; \boldsymbol{Y} := (\boldsymbol{X} := \mathsf{mc};\ \mathsf{t});\ \boldsymbol{X} := \mathsf{mc} \\
7. & \boldsymbol{X} := \mathsf{mc};\ !\mathsf{t} & \mathrel{\mid\!\!\Rrightarrow} \; \boldsymbol{X} := \mathsf{mc};\ !(\boldsymbol{X} := \mathsf{mc};\ \mathsf{t})
\end{array}$$

Definition 4.2. *The rewrite relation* $\Rrightarrow \;\subset (\boldsymbol{Term} \times \boldsymbol{Term})$ *is defined by:* $\mathsf{t} \Rrightarrow \mathsf{u}$ *iff* u *results from applying* $\mathrel{\mid\!\!\Rrightarrow}$ *to a subterm of* t*. If* $\mathsf{t} \Rrightarrow \cdots \Rrightarrow \mathsf{u}$ *(including if* $\mathsf{t} \equiv \mathsf{u}$*), then we write* $\mathsf{t} \mathrel{\Rrightarrow\!\!\!\!\Rrightarrow} \mathsf{u}$*; that is,* $\mathrel{\Rrightarrow\!\!\!\!\Rrightarrow}$ *is the reflexive-transitive closure of* $\Rrightarrow$*.*

Some brief remarks are in order to explain the rewrite rules. Rule 1 expresses a basic property of Montague's intension and extension operators. In our setting, it embodies the *execution of a procedure.* Rule 7 is very important: it is the *recursion rule.* It may be difficult to see immediately why we identify this rule with recursion; the following special case, which combines the use of rules 7, 3 and 1, illustrates the concept more clearly:

$$\boldsymbol{X} := ¡\mathsf{t}\,;!\boldsymbol{X} \;\mathrel{\Rrightarrow\!\!\!\!\Rrightarrow}\; \boldsymbol{X} := ¡\mathsf{t}\,;\mathsf{t}.$$

This amounts to simple substitution of a procedure body for its identifier, while "keeping a copy" of the procedure body available for further substitutions.

Our first order of business is to show that the rewrite function does not change the meaning of a term.

Theorem 4.3 (Validity of rewrite rules). $\mathsf{t} \mathrel{\mid\!\!\Rrightarrow} \mathsf{u} \implies (\mathsf{t}, \varsigma \Downarrow \mathsf{d} \;\Leftrightarrow\; \mathsf{u}, \varsigma \Downarrow \mathsf{d})$

Proof. By cases on the rewrite rules. This proof also provides constructive "rules of replacement" for parts of operational derivation trees: each rewrite rule corresponds to a transformation on derivation trees. □

We can gain some valuable insight into how to use the rewriting rules by using them to interpret our running example (3):

$$\begin{array}{rl}
\boldsymbol{P} := ¡\boldsymbol{X};\ \boldsymbol{X} := 1;\ !\boldsymbol{P} & \Rrightarrow\; \boldsymbol{X} := 1;\ \boldsymbol{P} := ¡\boldsymbol{X};\ !\boldsymbol{P} \\
& \Rrightarrow\; \boldsymbol{X} := 1;\ \boldsymbol{P} := ¡\boldsymbol{X};\ !(\boldsymbol{P} := ¡\boldsymbol{X};\ \boldsymbol{P}) \\
& \Rrightarrow\; \boldsymbol{X} := 1;\ \boldsymbol{P} := ¡\boldsymbol{X};\ !¡\boldsymbol{X} \\
& \Rrightarrow\; \boldsymbol{X} := 1;\ \boldsymbol{P} := ¡\boldsymbol{X};\ \boldsymbol{X} \;\mathrel{\Rrightarrow\!\!\!\!\Rrightarrow}\; \mathbf{1}
\end{array}$$

4.2 Equivalence of Interpretations

In this subsection we demonstrate that the rewriting system provided by the $\Rrightarrow$ relation is equivalent to the operational interpretation. Before stating this theorem, however, we need to address some technicalities.

In order to use the rewriting rules to arrive at the same result as the operational semantics, we need to take into account the *store*. That means that we need a way to take the required information from the store and *actualize* it as a term. For example, take the term $\mathsf{t} \equiv \boldsymbol{X} + \boldsymbol{X}$ and a store ς that maps $\boldsymbol{X}$ to $\mathbf{1}$; then, $\mathsf{t}, \varsigma \Downarrow \mathbf{2}$. We can accomplish this in rewriting by *prepending* t with $\boldsymbol{X} := \mathbf{1}$, which gives $\boldsymbol{X} := \mathbf{1};\ (\boldsymbol{X} + \boldsymbol{X}) \Rrightarrow \mathbf{2}$ as needed. For technical convenience, we take the set of locations $\boldsymbol{Loc}$ to be finite ([Ben10] extends the treatment to a countable set).

Definition 4.4 (Store terms, store actualization and store abstraction). *The store-actualization of ς, assuming $\boldsymbol{Loc} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$, is defined as*

$$\langle\varsigma\rangle \underset{\text{def}}{=} \boldsymbol{X}_1 := \varsigma(\boldsymbol{X}_1)\ ;\ \ldots\ ;\ \boldsymbol{X}_n := \varsigma(\boldsymbol{X}_n).$$

Terms like those above we call store-terms. The set of store-terms **STerm***, ranged over by* s*, is the set of terms of the form*

$$\boldsymbol{X}_1 := \mathsf{c}_1\ ;\ \ldots\ ;\ \boldsymbol{X}_n := \mathsf{c}_n$$

We also define an inverse to the store-actualization operator, store abstraction, which takes a store term and returns the corresponding store, such that $[\langle\varsigma\rangle] = \varsigma$ and $\langle[\mathsf{s}]\rangle$ is s with its assignment statements reordered into some arbitrary canonical ordering.

A convenient lemma shows the operational soundness of the above notions.

Lemma 4.5. $\mathsf{t}, \varsigma \Downarrow \mathsf{d} \iff \forall\varsigma' \cdot (\langle\varsigma\rangle \mathbin{;} \mathsf{t}), \varsigma' \Downarrow \mathsf{d}$. □

We now present the first part of the proof of equivalence between the operational semantics and term rewriting.

Theorem 4.6 (Operational adequacy).

1. *If* $\mathsf{t}, \varsigma \Downarrow \mathsf{c}$*, then* $\langle\varsigma\rangle \mathbin{;} \mathsf{t} \Rrightarrow \mathsf{c}$*;*
2. *If* $\mathsf{t}, \varsigma \Downarrow \varsigma'$*, then* $\langle\varsigma\rangle \mathbin{;} \mathsf{t} \Rrightarrow \langle\varsigma'\rangle$*.*

Proof. By induction on derivations. Details can be found in [Ben10]. □

The above theorem only works in one direction, in that it only shows that the rewriting system is at least as strong as the operational semantics. In fact it is (syntactically speaking) even stronger; as a simple demonstration of this, consider the term $\text{¡}(\mathbf{1} + \mathbf{1})$. Operationally, it is inert: it returns itself. However, the rewrite rules allow us to "reach inside" the intension and rewrite it to $\text{¡}\mathbf{2}$.[2]

[2] This was not an issue in [Ben10] because the equivalence proof there was stated in terms of denotational semantics; here we strive for a stronger result about the syntax itself.

The operational semantics and rewriting system are therefore syntactically equivalent only *up to rewriting of unevaluated intensions.* The following theorem completes the equivalence result.

Theorem 4.7 (Rewriting adequacy).

1. *If* $\mathsf{t} \Rrightarrow \mathsf{c}$*, then there is a* c' *s.t.* $\mathsf{c} \Rrightarrow \mathsf{c}'$ *and for any* ς'*,* $\mathsf{t}, \varsigma' \Downarrow \mathsf{c}'$*.*
2. *If* $\mathsf{t} \Rrightarrow \mathsf{s}$*, then there is a* s' *s.t.* $\mathsf{s} \Rrightarrow \mathsf{s}'$ *and for any* ς''*,* $\mathsf{t}, \varsigma'' \Downarrow [\mathsf{s}']$*.*

Proof. By induction on the length of the rewrite sequence $\mathsf{t} \Rrightarrow \mathsf{c}$ or $\mathsf{t} \Rrightarrow \mathsf{s}$. The inductive step consists roughly of identifying the affected parts of the operational derivation tree and substituting a suitably modified tree (using Theorem 4.3); or, if the rewrite step affects only an uninterpreted (i.e., bound by intension) part of the term, to add the rewrite to $\mathsf{c} \Rrightarrow \mathsf{c}'$ or $\mathsf{s} \Rrightarrow \mathsf{s}'$ as applicable. □

By combining Theorems 4.6 and 4.7, using Lemma 4.5, we obtain our desired equivalence result.

We conclude this section by mentioning that we have also attained a confluence result for our term rewriting system [Ben10, App. B]. It will be discussed in detail in a forthcoming publication.

Theorem 4.8 (Confluence). *If* $\mathsf{t} \Rrightarrow \mathsf{u}$ *and* $\mathsf{t} \Rrightarrow \mathsf{v}$*, then there is a* w *s.t.* $\mathsf{u} \Rrightarrow \mathsf{w}$ *and* $\mathsf{v} \Rrightarrow \mathsf{w}$*.*

5 Conclusion

We hope to have convinced the reader, in the last four sections, that ***AC*** does indeed possess the desirable properties listed in §2.1. (a) It is *small*, *elegant* and (hopefully) *intuitive*: as discussed in §2, ***AC***'s set of four basic operators is simple and understandable; (b) its operators represent well-understood, fundamental concepts: by taking only assignment, sequence, procedure formation and procedure invocation as basic, we remain close to practical imperative languages; (c) we can *rewrite* its terms using simple rules: Definition 4.1 demonstrates this; and (d) it has multiple forms of semantics that are equivalent: in this paper we demonstrated this equivalence for operational semantics and term rewriting, [Ben10] extends this equivalence to denotational semantics as well.

We believe that the above points show that Assignment Calculus is a realization of our goal to develop a true *imperative computation language.*

5.1 Related Work

The final step in our presentation is to explore related and otherwise relevant work. First, note that there has been no other attempt, as far as the authors are aware, to define a pure imperative computation language as we have done. Therefore all of the other work that we will examine is only indirectly related to our aims; nevertheless there are certainly interesting connections to be explored.

The first and perhaps most striking language of interest that can be found in the literature is (unfortunately!) nameless; it is defined in the seminal report of Strachey and Scott [SS71, §5]. Here we find a language that has features that closely resemble those of $\boldsymbol{AC}$: there are operators for referencing and dereferencing, and operators for treating a procedure as an expression and an expression as a procedure. This language does not appear to have been revisited in subsequent work.

Insofar as *rewriting systems for imperative-style languages*, the prime example is the work of Felleisen [FH92]. He adds facilities for handling state and control operators to Plotkin's call-by-value λ-calculus, which results in a quite elegant system. There could be a good deal of interesting work in comparing our work with Felleisen's; the main problem that we immediately encounter is that his work depends fundamentally on the λ-calculus, which is precisely what we have tried to avoid incorporating into $\boldsymbol{AC}$.

5.2 Future Work

We now discuss interesting avenues of further research. First, we should like to continue exploring, expanding, and improving the rewriting system of §4. Our present goal was to arrive at a small set of rules that was sufficient to achieve our equivalence proof; however, it would be useful to develop more powerful rules that might be more intuitive in terms of real-world use. In fact we have already made significant progress in this direction while developing the proof of confluence in [Ben10, App. B]; the work will be elaborated in forthcoming work.

It would be interesting to examine more carefully the concept of *state backtracking* in $\boldsymbol{AC}$. As mentioned in §3.5, we believe that state backtracking is a fundamental part of imperative computation; therefore, we would like to provide an improved analysis of what it comprises and how it takes part in and affects imperative computation. Along these lines, it is important to explore connections with Separation Logic [Rey02], particularly its interesting store model, and with Felleisen's work as mentioned above.

The aim of this paper was to provide an analysis of imperative computation. We have done this on multiple levels: from a philosophical dissection of the concept of imperative computation in §2, to developing the actual types and syntax of $\boldsymbol{AC}$, and finally to $\boldsymbol{AC}$'s operational and rewriting-based meanings. We believe that this broad approach leaves $\boldsymbol{AC}$ well-prepared for further investigations, and we hope that it will stimulate future work in what we consider to be an exciting new approach to an old paradigm.

References

[Bac78] Backus, J.: Can programming be liberated from the von Neumann style?: A functional style and its algebra of programs. Commun. ACM 21(8), 613–641 (1978)

[Bar84] Barendregt, H.P.: The Lambda Calculus: Its Syntax and Semantics. North Holland (1984)

[Ben10] Bender, M.: Assignment Calculus: A Pure Imperative Reasoning Language. PhD thesis, McMaster University (2010)

[Car47] Carnap, R.: Meaning and Necessity. University of Chicago (1947)

[Chu40] Church, A.: A formulation of the simple theory of types. Journal of Symbolic Logic 5(2), 56–68 (1940)

[Chu41] Church, A.: The Calculi of Lambda Conversion. Princeton (1941)

[Chu51] Church, A.: A formulation of the logic of sense and denotation. In: Henle, P. (ed.) Structure, Method and Meaning: Essays in Honor of Henry M. Sheffer, pp. 3–24. Liberal Arts Press (1951)

[FH88] Field, A.J., Harrison, P.G.: Functional Programming. Addison-Wesley (July 1988)

[FH92] Felleisen, M., Hieb, R.: The revised report on the syntactic theories of sequential control and state. Theor. Comput. Sci. 103(2), 235–271 (1992)

[Fre92] Frege, G.: Über Sinn und Bedeutung. Zeitschrift für Philosophie und philosophische Kritik 100, 25–50 (1892)

[HHJW07] Hudak, P., Hughes, J., Jones, S.P., Wadler, P.: A history of Haskell: being lazy with class. In: HOPL III: Proc. 3rd ACM SIGPLAN Conf. on History of Prog. Languages, pp. 12-1–12-55. ACM, New York (2007)

[Hun90] Hung, H.-K.: Compositional Semantics and Program Correctness for Procedures with Parameters. PhD thesis, SUNY-Buffalo (1990)

[HZ91] Hung, H.-K., Zucker, J.: Semantics of pointers, referencing and dereferencing with intensional logic. In: Proc. 6th Annual IEEE Symposium on Logic in Computer Science, pp. 127–136 (1991)

[Jan86] Janssen, T.M.V.: Foundations and Applications of Montague Grammar: Part 1: Philosophy, Framework, Computer Science. Centrum voor Wiskunde en Informatica, Amsterdam (1986)

[JvEB77] Janssen, T.M.V., van Emde Boas, P.: On the Proper Treatment or Referencing, Dereferencing and Assignment. In: Salomaa, A., Steinby, M. (eds.) ICALP 1977. LNCS, vol. 52, pp. 282–300. Springer, Heidelberg (1977)

[Kri59] Kripke, S.A.: A completeness theorem in modal logic. Journal of Symbolic Logic 24, 1–14 (1959)

[Mon70] Montague, R.: Universal grammar. Theoria 36, 373–398 (1970)

[Mon73] Montague, R.: The proper treatment of quantification in ordinary English. In: Hintikka, K.J.J., Moravcsik, J.M.E., Suppes, P. (eds.) Approaches to Natural Language, pp. 221–242. Reidel (1973)

[Qui60] Quine, W.V.: Word and Object. MIT Press, Cambridge (1960)

[Rey02] Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: LICS, pp. 55–74. IEEE Computer Society (2002)

[Sco70] Scott, D.S.: Outline of a mathematical theory of computation. Tech. Monograph PRG–2, Oxford University Computing Laboratory (1970)

[SS71] Scott, D., Strachey, C.: Toward a mathematical semantics for computer languages. In: Fox, J. (ed.) Proc. Symp. on Computers and Automata, Brooklyn, N.Y., vol. XXI, pp. 19–46. Polytechnic Press (April 1971)

[Tur36] Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. 2(42), 230–265 (1936)

[Win93] Winskel, G.: The formal semantics of programming languages: an introduction. MIT Press, Cambridge (1993)

Multiplayer Cost Games with Simple Nash Equilibria

Thomas Brihaye[1], Julie De Pril[1], and Sven Schewe[2]

[1] University of Mons - UMONS, Belgium
{thomas.brihaye,julie.depril}@umons.ac.be
[2] University of Liverpool, United Kingdom
sven.schewe@liverpool.ac.uk

Abstract. Multiplayer games with selfish agents naturally occur in the design of distributed and embedded systems. As the goals of selfish agents are usually neither equivalent nor antagonistic to each other, such games are non zero-sum games. We study such games and show that a large class of these games, including games where the individual objectives are mean- or discounted-payoff, or quantitative reachability, and show that they do not only have a solution, but a *simple* solution. We establish the existence of Nash equilibria that are composed of k memoryless strategies for each agent in a setting with k agents, one main and $k - 1$ minor strategies. The main strategy describes what happens when all agents comply, whereas the minor strategies ensure that all other agents immediately start to co-operate against the agent who first deviates from the plan. This simplicity is important, as rational agents are an idealisation. Realistically, agents have to decide on their moves with very limited resources, and complicated strategies that require exponential—or even non-elementary—implementations cannot realistically be implemented. The existence of simple strategies that we prove in this paper therefore holds a promise of implementability.

1 Introduction

The construction of correct and efficient computer systems (both hard- and software) is recognised to be an extremely difficult task. Formal methods have been exploited with some success in the design and verification of such systems. Mathematical logic, automata theory [16], and model-checking [11] have contributed much to the success of formal methods in this field. However, traditional approaches aim at systems with qualitative specifications like LTL, and rely on the fact that these specifications are either satisfied or violated by the system.

Unfortunately, these techniques do not trivially extend to complex systems, such as embedded or distributed systems. A main reason for this is that such systems often consist of multiple independent components with individual objectives. These components can be viewed as selfish agents that may cooperate and compete at the same time. It is difficult to model the interplay between these components with traditional finite state machines, as they cannot reflect the intricate quantitative valuation of an agent on how well he has met his goal. In particular, it is not realistic to assume that these components are always cooperating to satisfy a common goal, as it is, e.g., assumed in works that distinguish between an environment and a system. We argue that it is more realistic to assume that all components act like selfish agents that try to achieve their own objectives and are either unconcerned about the effect this has on the other components or consider this effect to be secondary. It is indeed a recent trend to enhance the system models used in the classical approach of verification by quantitative cost and gain functions, and to exploit the well established game-theoretic framework [20,21] for their formal analysis.

The first steps towards the extension of computational models with concepts from classical game theory were taken by advancing from boolean to general two-player zero-sum games played on graphs [14]. Like their qualitative counter parts, those games are adequate to model controller-environment interaction problems [23,24]. As usual in control theory, one can distinguish between moves of a control player, who plays actions to control a system to meet a control objective, and an antagonistic environment player. In the classical setting, the control player has a qualitative objective—he might, for example, try to enforce a temporal specification—whereas the environment tries to prevent this. In the extension to quantitative games, the controller instead tries to maximise its gain, while the environment tries to minimise it. This extension lifts the controller synthesis problem from a constructive extension of a decision problem to a classical optimisation problem.

However, this extension has not lifted the restriction to purely antagonist interactions between a controller and a hostile environment. In order to study more complex systems with more than two components, and with objectives that are not necessarily antagonist, we resort to multiplayer non zero-sum games. In this context, *Nash equilibria* [20] take the place that winning and optimal strategies take in qualitative and quantitative two-player games zero-sum games, respectively. Surprisingly, qualitative objectives have so far prevailed in the study of Nash equilibria for distributed systems. However, we argue that Nash equilibria for selfish agents with quantitative objectives—such as reaching a set of target states quickly or with a minimal consumption of energy—are natural objectives that aught to be studied alongside (or instead of) traditional qualitative objectives.

Consequently, we study *Nash equilibria* for *multiplayer non zero-sum* games played on graphs with *quantitative* objectives.

Our Contribution. In this paper, we study turn-based multiplayer non zero-sum games played on finite graphs with quantitative objectives, expressed through a cost function for each player (*cost games*). Each cost function assigns, for every play of the game, a value that represents the cost that is incurred for a player by this play. Cost functions allow to express classical quantitative objectives such as *quantitative reachability* (i.e., the player aims at reaching a subset of states as soon as possible), or *mean-payoff* objectives. In this framework, all players are supposed to be rational: they want to minimise their own cost or, equivalently, maximise their own gain. This invites the use of Nash equilibria as the adequate concept for cost games.

Our results are twofold. Firstly, we prove the *existence* of Nash equilibria for a large class of cost games that includes quantitative reachability and mean-payoff objectives. Secondly, we study the complexity of these Nash equilibria in terms of the *memory* needed in the strategies of the individual players in these Nash equilibria. More precisely, we ensure existence of Nash equilibria whose strategies only requires a number of memory states that is *linear* in the size of the game for a wide class of cost games, including games with quantitative reachability and mean-payoff objectives.

The general philosophy of our work is as follows: we try to derive existence of Nash equilibria in multiplayer non zero-sum quantitative games (and characterization of their complexity) through determinacy results (and characterization of the optimal strategies) of several well-chosen two-player quantitative games derived from the multiplayer game. These ideas were already successfully exploited in the qualitative framework [15], and in the case of limit-average objectives [25].

Related Work. Several recent papers have considered *two-player zero-sum* games played on finite graphs with regular objectives enriched by some *quantitative* aspects. Let us mention some of them: games with finitary objectives [9], mean-payoff parity games [10], games with prioritised requirements [1], request-response games where the waiting times between the requests and the responses are minimized [17,27], games whose winning conditions are expressed via quantitative languages [2], and recently, cost-parity and cost-Streett games [12].

Other work concerns *qualitative non zero-sum* games. In [15], general criteria ensuring existence of Nash equilibria and subgame perfect equilibria (resp. secure equilibria) are provided for multiplayer (resp. 2-player) games, as well as complexity results. The complexity of Nash equilibria in multiplayer concurrent games with Büchi objectives has been discussed in [5]. [4] studies the existence of Nash equilibria for timed games with qualitative reachability objectives.

Finally, there is a series of recent results on the combination of *non zero-sum* aspects with *quantitative objectives*. In [3], the authors study games played on graphs with terminal vertices where quantitative payoffs are assigned to the players. In [18], the authors provide an algorithm to decide the existence of Nash equilibria for concurrent priced games with quantitative reachability objectives. In [22], the authors prove existence of a Nash equilibrium in Muller games on finite graphs where players have a preference ordering on the sets of the Muller table. Let us also notice that the existence of a Nash equilibrium in cost games with quantitative reachability objectives we study in this paper has already been established in [6]. The new proves we provide are simpler and significantly improve the complexity of the strategies constructed from exponential to linear in the size of the game.

Organization of the Paper. In Section 2, we present the model of multiplayer cost games and define the problems we study. The main results are given in Section 3. Finally, in Section 4, we apply our general result on particular cost games with classical objectives. Omitted proofs and additional materials can be found in [8, Appendix].

2 General Background

In this section, we define our model of *multiplayer cost game*, recall the concept of Nash equilibrium and state the problems we study.

Definition 1. *A* multiplayer cost game *is a tuple* $\mathcal{G} = (\Pi, V, (V_i)_{i\in\Pi}, E, (\mathsf{Cost}_i)_{i\in\Pi})$ *where*

- *Π is a finite set of* players*,*
- *$G = (V, E)$ is a* finite directed graph *with vertices V and edges $E \subseteq V \times V$,*
- *$(V_i)_{i \in \Pi}$ is a partition of V such that V_i is the set of vertices controlled by player i,*
- $\mathsf{Cost}_i : \mathsf{Plays} \to \mathbb{R} \cup \{+\infty, -\infty\}$ *is the* cost function *of player i, where* Plays *is the set of* plays *in $\mathcal{G}$, i.e. the set of infinite paths through G. For every play $\rho \in$* Plays*, the value* $\mathsf{Cost}_i(\rho)$ *represents the amount that player i loses for this play.*

Cost games are *multiplayer turn-based quantitative non zero-sum* games. We assume that the players are rational: they play in a way to minimise their own cost.

Note that minimising cost or maximising gain are essentially[1] equivalent, as maximising the gain for player i can be modelled by using Cost_i to be minus this gain and then minimising the cost. This is particularly important in cases where two players have antagonistic goals, as it is the case in all two-player zero-sum games. To cover these cases without changing the setting, we sometimes refer to maximisation in order to preserve the connection to such games in the literature.

For the sake of simplicity, we assume that each vertex has at least one outgoing edge. Moreover, it is sometimes convenient to specify an initial vertex $v_0 \in V$ of the game. We then call the pair $(\mathcal{G}, v_0)$ an *initialised multiplayer cost game*. This game is played as follows. First, a token is placed on the initial vertex v_0. Whenever a token is on a vertex $v \in V_i$ controlled by player i, player i chooses one of the outgoing edges $(v, v') \in E$ and moves the token along this edge to v'. This way, the players together determine an *infinite* path through the graph G, which we call a *play*. Let us remind that Plays is the set of all plays in $\mathcal{G}$.

A *history* h of $\mathcal{G}$ is a *finite* path through the graph G. We denote by Hist the set of histories of a game, and by ϵ the empty history. In the sequel, we write $h = h_0 \ldots h_k$, where $h_0, \ldots, h_k \in V$ ($k \in \mathbb{N}$), for a history h, and similarly, $\rho = \rho_0\rho_1 \ldots$, where $\rho_0, \rho_1, \ldots \in V$, for a play ρ. A *prefix* of length $n + 1$ (for some $n \in \mathbb{N}$) of a play $\rho = \rho_0\rho_1 \ldots$ is the history $\rho_0 \ldots \rho_n$, and is denoted by $\rho[0, n]$.

Given a history $h = h_0 \ldots h_k$ and a vertex v such that $(h_k, v) \in E$, we denote by hv the history $h_0 \ldots h_k v$. Moreover, given a history $h = h_0 \ldots h_k$ and a play $\rho = \rho_0\rho_1 \ldots$ such that $(h_k, \rho_0) \in E$, we denote by $h\rho$ the play $h_0 \ldots h_k\rho_0\rho_1 \ldots$.

The function Last (resp. First) returns, for a given history $h = h_0 \ldots h_k$, the last vertex h_k (resp. the first vertex h_0) of h. The function First naturally extends to plays.

A *strategy* of player i in $\mathcal{G}$ is a function $\sigma : \mathsf{Hist} \to V$ assigning to each history $h \in \mathsf{Hist}$ that ends in a vertex $\mathsf{Last}(h) \in V_i$ controlled by player i, a successor $v = \sigma(h)$ of $\mathsf{Last}(h)$. That is, $\big(\mathsf{Last}(h), \sigma(h)\big) \in E$. We say that a play $\rho = \rho_0\rho_1 \ldots$ of $\mathcal{G}$ is *consistent* with a strategy σ of player i if $\rho_{k+1} = \sigma(\rho_0 \ldots \rho_k)$ for all $k \in \mathbb{N}$ such that $\rho_k \in V_i$. A *strategy profile* of $\mathcal{G}$ is a tuple $(\sigma_i)_{i \in \Pi}$ of strategies, where σ_i refers to a strategy for player i. Given an initial vertex v, a strategy profile determines a unique play of $(\mathcal{G}, v)$ that is consistent with all strategies σ_i. This play is called the *outcome* of $(\sigma_i)_{i \in \Pi}$ and denoted by $\langle (\sigma_i)_{i \in \Pi} \rangle_v$. We say that a player *deviates* from a strategy (resp. from a play) if he does not carefully follow this strategy (resp. this play).

A *finite strategy automaton* for player $i \in \Pi$ over a game $\mathcal{G} = (\Pi, V, (V_i)_{i \in \Pi}, E, (\mathsf{Cost}_i)_{i \in \Pi})$ is a Mealy automaton $\mathcal{A}_i = (M, m_0, V, \delta, \nu)$ where:

[1] Sometimes the translation implies minor follow-up changes, e.g., the replacement of $\liminf$ by $\limsup$ and vice versa.

– M is a non-empty, finite set of memory states,
– $m_0 \in M$ is the initial memory state,
– $\delta : M \times V \to M$ is the memory update function,
– $\nu : M \times V_i \to V$ is the transition choice function, such that $(v, \nu(m, v)) \in E$ for all $m \in M$ and $v \in V_i$.

We can extend the memory update function δ to a function $\delta^* : M \times \mathsf{Hist} \to M$ defined by $\delta^*(m, \epsilon) = m$ and $\delta^*(m, hv) = \delta(\delta^*(m, h), v)$ for all $m \in M$ and $hv \in \mathsf{Hist}$. The strategy $\sigma_{\mathcal{A}_i}$ computed by a finite strategy automaton $\mathcal{A}_i$ is defined by $\sigma_{\mathcal{A}_i}(hv) = \nu(\delta^*(m_0, h), v)$ for all $hv \in \mathsf{Hist}$ such that $v \in V_i$. We say that σ is a *finite-memory strategy* if there exists[2] a finite strategy automaton $\mathcal{A}$ such that $\sigma = \sigma_{\mathcal{A}}$. Moreover, we say that $\sigma = \sigma_{\mathcal{A}}$ has a memory of size at most $|M|$, where $|M|$ is the number of states of $\mathcal{A}$. In particular, if $|M| = 1$, we say that σ is a *positional strategy* (the current vertex of the play determines the choice of the next vertex). We call $(\sigma_i)_{i\in\Pi}$ a strategy profile with memory m if for all $i \in \Pi$, the strategy σ_i has a memory of size at most m. A strategy profile $(\sigma_i)_{i\in\Pi}$ is called *positional* or *finite-memory* if each σ_i is a positional or a finite-memory strategy, respectively.

We now define the notion of *Nash equilibria* in this quantitative framework.

Definition 2. *Given an initialised multiplayer cost game $(\mathcal{G}, v_0)$, a strategy profile $(\sigma_i)_{i\in\Pi}$ is a* Nash equilibrium *in $(\mathcal{G}, v_0)$ if, for every player $j \in \Pi$ and for every strategy σ'_j of player j, we have:*

$$\mathsf{Cost}_j(\rho) \le \mathsf{Cost}_j(\rho')$$

where $\rho = \langle(\sigma_i)_{i\in\Pi}\rangle_{v_0}$ and $\rho' = \langle\sigma'_j, (\sigma_i)_{i\in\Pi\setminus\{j\}}\rangle_{v_0}$.

This definition means that, for all $j \in \Pi$, player j has no incentive to deviate from σ_j since he cannot strictly decrease his cost when using σ'_j instead of σ_j. Keeping notations of Definition 2 in mind, a strategy σ'_j such that $\mathsf{Cost}_j(\rho) > \mathsf{Cost}_j(\rho')$ is called a *profitable deviation* for player j w.r.t. $(\sigma_i)_{i\in\Pi}$.

Example 3. Let $\mathcal{G} = (\Pi, V, V_1, V_2, E, \mathsf{Cost}_1, \mathsf{Cost}_2)$ be the two-player cost game whose graph $G = (V, E)$ is depicted in Figure 1. The states of player 1 (resp. 2) are represented by circles (resp. squares). Thus, according to Figure 1, $V_1 = \{A, C, D\}$ and $V_2 = \{B\}$. In order to define the cost functions of both players, we consider a price function $\pi : E \to \{1, 2, 3\}$, which assigns a price to each edge of the graph. The price function[3] π is as follows (see the numbers in Figure 1): $\pi(A, B) = \pi(B, A) = \pi(B, C) = 1$, $\pi(A, D) = 2$ and $\pi(C, B) = \pi(D, B) = 3$. The cost function Cost_1 of player 1 expresses a *quantitative reachability objective*: he wants to reach the vertex C (shaded vertex) while minimising the sum of prices up to this vertex. That is, for every play $\rho = \rho_0\rho_1 \ldots$ of $\mathcal{G}$:

$$\mathsf{Cost}_1(\rho) = \begin{cases} \sum_{i=1}^{n} \pi(\rho_{i-1}, \rho_i) & \text{if } n \text{ is the \textit{least} index s.t. } \rho_n = C, \\ +\infty & \text{otherwise.} \end{cases}$$

[2] Note that there exist several finite strategy automata such that $\sigma = \sigma_{\mathcal{A}}$.

[3] Note that we could have defined a different price function for each player. In this case, the edges of the graph would have been labelled by couples of numbers.

As for the cost function Cost_2 of player 2, it expresses a *mean-payoff objective*: the cost of a play is the long-run average of the prices that appear along this play. Formally, for any play $\rho = \rho_0\rho_1 \ldots$ of $\mathcal{G}$:

$$\mathsf{Cost}_2(\rho) = \limsup_{n \to +\infty} \frac{1}{n} \cdot \sum_{i=1}^{n} \pi(\rho_{i-1}, \rho_i).$$

Each player aims at minimising the cost incurred by the play. Let us insist on the fact that the players of a cost game may have different kinds of cost functions (as in this example).

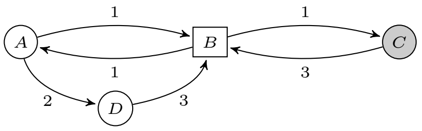


Fig. 1. A two-player cost game $\mathcal{G}$

An example of a play in $\mathcal{G}$ can be given by $\rho = (AB)^\omega$, leading to the costs $\mathsf{Cost}_1(\rho) = +\infty$ and $\mathsf{Cost}_2(\rho) = 1$. In the same way, the play $\rho' = A(BC)^\omega$ induces the following costs: $\mathsf{Cost}_1(\rho) = 2$ and $\mathsf{Cost}_2(\rho) = 2$.

Let us fix the initial vertex v_0 at the vertex A. The play $\rho = (AB)^\omega$ is the outcome of the positional strategy[4] profile (σ_1, σ_2) where $\sigma_1(A) = B$ and $\sigma_2(B) = A$. Moreover, this strategy profile is in fact a *Nash equilibrium*: player 2 gets the least cost he can expect in this game, and player 1 has no incentive to choose the edge (A, D) (it does not allow the play to pass through vertex C).

We now consider the positional strategy profile (σ'_1, σ'_2) with $\sigma'_1(A) = B$ and $\sigma'_2(B) = C$. Its outcome is the play $\rho' = A(BC)^\omega$. However, this strategy profile is *not* a Nash equilibrium, because player 2 can strictly lower his cost by always choosing the edge (B, A) instead of (B, C), thus lowering his cost from 2 to 1. In other words, the strategy σ_2 (defined before) is a *profitable deviation* for player 2 w.r.t. (σ'_1, σ'_2).

The questions studied in this paper are the following ones:

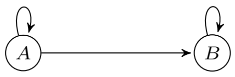
Problem 1. *Given a multiplayer cost game $\mathcal{G}$, does there exist a Nash equilibrium in $\mathcal{G}$?*

Problem 2. *Given a multiplayer cost game $\mathcal{G}$, does there exist a finite-memory Nash equilibrium in $\mathcal{G}$?*

Obviously enough, if we make no restrictions on our cost games, the answer to Problem 1 (and thus to Problem 2) is negative (see Example 4). Our first goal in this paper is to identify a large class of cost games for which the answer to Problem 1 is positive. Then we also positively reply to Problem 2 for subclasses of the previously identified class of cost games. Both results can be found in Section 3.

[4] Note that player 1 has no choice in vertices C and D, that is, $\sigma_1(hv)$ is necessarily equal to B for $v \in \{C, D\}$ and $h \in \mathsf{Hist}$.

Example 4. Let $(\mathcal{G}, A)$ be the initialised one-player cost game depicted below, whose cost function Cost_1 is defined by $\mathsf{Cost}_1(A^n B^\omega) = \frac{1}{n}$ for $n \in \mathbb{N}_0$ and $\mathsf{Cost}_1(A^\omega) = +\infty$. One can be convinced that there is no Nash equilibrium in this initialised game.



In order to our class of cost games, we need the notions of *Min-Max cost games*, *determinacy* and *optimal strategies*. The following two definitions are inspired by [26].

Definition 5. *A* Min-Max cost game *is a tuple* $\mathcal{G} = (V, V_{Min}, V_{Max}, E, \mathsf{Cost}_{Min}, \mathsf{Gain}_{Max})$*, where*

- $G = (V, E)$ *is a* finite directed graph *with vertices V and edges $E \subseteq V \times V$,*
- (V_{Min}, V_{Max}) *is a partition of V such that V_{Min} (resp. V_{Max}) is the set of vertices controlled by player Min (resp. Max), and*
- $\mathsf{Cost}_{Min} : \mathsf{Plays} \to \mathbb{R} \cup \{+\infty, -\infty\}$ *is the* cost function *of player Min, that represents the amount that he loses for a play, and* $\mathsf{Gain}_{Max} : \mathsf{Plays} \to \mathbb{R} \cup \{+\infty, -\infty\}$ *is the* gain function *of player Max, that represents the amount that he wins for a play.*

In such a game, player Min wants to *minimise* his cost, while player Max wants to *maximise* his gain. So, a Min-Max cost game is a particular case of a two-player cost game. Let us stress that, according to this definition, a Min-Max cost game is *zero-sum* if $\mathsf{Cost}_{\mathrm{Min}} = \mathsf{Gain}_{\mathrm{Max}}$, but this might not always be the case[5]. We also point out that Definition 5 allows to take completely unrelated functions $\mathsf{Cost}_{\mathrm{Min}}$ and $\mathsf{Gain}_{\mathrm{Max}}$, but usually they are similar (see Definition 15). In the sequel, we denote by Σ_{Min} (resp. Σ_{Max}) the set of strategies of player Min (resp. Max) in a Min-Max cost game.

Definition 6. *Given a Min-Max cost game $\mathcal{G}$, we define for every vertex $v \in V$ the* upper value $\mathsf{Val}^*(v)$ *as:*

$$\mathsf{Val}^*(v) = \inf_{\sigma_1 \in \Sigma_{Min}} \sup_{\sigma_2 \in \Sigma_{Max}} \mathsf{Cost}_{Min}(\langle \sigma_1, \sigma_2 \rangle_v) \,,$$

and the lower value $\mathsf{Val}_*(v)$ *as:*

$$\mathsf{Val}_*(v) = \sup_{\sigma_2 \in \Sigma_{Max}} \inf_{\sigma_1 \in \Sigma_{Min}} \mathsf{Gain}_{Max}(\langle \sigma_1, \sigma_2 \rangle_v) \,.$$

The game $\mathcal{G}$ is determined *if, for every $v \in V$, we have* $\mathsf{Val}^*(v) = \mathsf{Val}_*(v)$*. In this case, we say that the game $\mathcal{G}$ has a* value*, and for every $v \in V$,* $\mathsf{Val}(v) = \mathsf{Val}^*(v) = \mathsf{Val}_*(v)$*. We also say that the strategies $\sigma_1^\star \in \Sigma_{Min}$ and $\sigma_2^\star \in \Sigma_{Max}$ are* optimal strategies *for the respective players if, for every $v \in V$, we have that*

$$\inf_{\sigma_1 \in \Sigma_{Min}} \mathsf{Gain}_{Max}(\langle \sigma_1, \sigma_2^\star \rangle_v) = \mathsf{Val}(v) = \sup_{\sigma_2 \in \Sigma_{Max}} \mathsf{Cost}_{Min}(\langle \sigma_1^\star, \sigma_2 \rangle_v) \,.$$

If $\sigma_1^\star$ is an optimal strategy for player Min, then he loses at most $\mathsf{Val}(v)$ when playing according to it. On the other hand, player Max wins at least $\mathsf{Val}(v)$ if he plays according to an optimal strategy $\sigma_2^\star$ for him.

Examples of classical determined Min-Max cost games can be found in Section 4.

[5] For an example, see the average-price game in Definition 15.

3 Results

In this section, we first define a large class of cost games for which Problem 1 can be answered positively (Theorem 10). Then, we study existence of simple Nash equilibria (Theorems 13 and 14). To define this interesting class of cost games, we need the concepts of *cost-prefix-linear* and *coalition-determined* cost games.

Definition 7. *A multiplayer cost game* $\mathcal{G} = (\Pi, V, (V_i)_{i\in\Pi}, E, (\mathsf{Cost}_i)_{i\in\Pi})$ *is* cost-prefix-linear *if, for every player* $i \in \Pi$*, every vertex* $v \in V$ *and history* $hv \in \mathsf{Hist}$*, there exists* $a \in \mathbb{R}$ *and* $b \in \mathbb{R}^+$ *such that, for every play* $\rho \in \mathsf{Plays}$ *with* $\mathsf{First}(\rho) = v$*, we have:*

$$\mathsf{Cost}_i(h\rho) = a + b \cdot \mathsf{Cost}_i(\rho)\,.$$

Let us now define the concept of *coalition-determined* cost games.

Definition 8. *A multiplayer cost game* $\mathcal{G} = (\Pi, V, (V_i)_{i\in\Pi}, E, (\mathsf{Cost}_i)_{i\in\Pi})$ *is* (positionally/finite-memory) coalition-determined *if, for every player* $i \in \Pi$*, there exists a gain function* $\mathsf{Gain}^i_{Max} : \mathsf{Plays} \to \mathbb{R} \cup \{+\infty, -\infty\}$ *such that*

- $\mathsf{Cost}_i \geq \mathsf{Gain}^i_{Max}$*, and*
- *the Min-Max cost game* $\mathcal{G}^i = (V, V_i, V \setminus V_i, E, \mathsf{Cost}_i, \mathsf{Gain}^i_{Max})$*, where player* i *(player Min) plays against the coalition* $\Pi \setminus \{i\}$ *(player Max), is determined and has (positional/finite-memory) optimal strategies for both players. That is:* $\exists\, \sigma^\star_i \in \Sigma_{Min}$*,* $\exists\, \sigma^\star_{-i} \in \Sigma_{Max}$ *(both positional/finite-memory) such that* $\forall v \in V$

$$\inf_{\sigma_i \in \Sigma_{Min}} \mathsf{Gain}^i_{Max}(\langle \sigma_i, \sigma^\star_{-i}\rangle_v) = \mathsf{Val}^i(v) = \sup_{\sigma_{-i} \in \Sigma_{Max}} \mathsf{Cost}_i(\langle \sigma^\star_i, \sigma_{-i}\rangle_v)\,.$$

Given $i \in \Pi$, note that $\mathcal{G}^i$ does not depend on the cost functions Cost_j, with $j \neq i$.

Example 9. Let us consider the two-player cost game $\mathcal{G}$ of Example 3, where player 1 has a quantitative reachability objective (Cost_1) and player 2 has a mean-payoff objective (Cost_2). We show that $\mathcal{G}$ is positionally coalition-determined.

Let us set $\mathsf{Gain}^1_{\mathrm{Max}} = \mathsf{Cost}_1$ and study the Min-Max cost game $\mathcal{G}^1 = (V, V_1, V_2, E, \mathsf{Cost}_1, \mathsf{Gain}^1_{\mathrm{Max}})$, where player Min (resp. Max) is player 1 (resp. 2) and wants to minimise Cost_1 (resp. maximise $\mathsf{Gain}^1_{\mathrm{Max}}$). This game is positionally determined [26,13]. We define positional strategies $\sigma^\star_1$ and $\sigma^\star_{-1}$ for player 1 and player 2, respectively, in the following way: $\sigma^\star_1(A) = B$ and $\sigma^\star_{-1}(B) = A$. From A, their outcome is $\langle(\sigma^\star_1, \sigma^\star_{-1})\rangle_A = (AB)^\omega$, and $\mathsf{Cost}_1((AB)^\omega) = \mathsf{Gain}^1_{\mathrm{Max}}((AB)^\omega) = +\infty$. One can check that the strategies $\sigma^\star_1$ and $\sigma^\star_{-1}$ are optimal in $\mathcal{G}^1$. Note that the positional strategy $\tilde{\sigma}^\star_1$ defined by $\tilde{\sigma}^\star_1(A) = D$ is also optimal (for player 1) in $\mathcal{G}^1$. With this strategy, we have that $\langle(\tilde{\sigma}^\star_1, \sigma^\star_{-1})\rangle_A = (ADB)^\omega$, and $\mathsf{Cost}_1((ADB)^\omega) = \mathsf{Gain}^1_{\mathrm{Max}}((ADB)^\omega) = +\infty$.

We now examine the Min-Max cost game $\mathcal{G}^2 = (V, V_2, V_1, E, \mathsf{Cost}_2, \mathsf{Gain}^2_{\mathrm{Max}})$, where $\mathsf{Gain}^2_{\mathrm{Max}}$ is defined as Cost_2 but with $\liminf$ instead of $\limsup$. In this game, player Min (resp. Max) is player 2 (resp. 1) and wants to minimise Cost_2 (resp. maximise $\mathsf{Gain}^2_{\mathrm{Max}}$). This game is also positionally determined [26,13]. Let $\sigma^\star_2$ and $\sigma^\star_{-2}$

be the positional strategies for player 2 and player 1, respectively, defined as follows: $\sigma_2^\star(B) = C$ and $\sigma_{-2}^\star(A) = D$. From A, their outcome is $\langle(\sigma_2^\star, \sigma_{-2}^\star)\rangle_A = AD(BC)^\omega$, and $\mathsf{Cost}_2(AD(BC)^\omega) = \mathsf{Gain}^2_{\text{Max}}(AD(BC)^\omega) = 2$. We claim that $\sigma_2^\star$ and $\sigma_{-2}^\star$ are the only positional optimal strategies in $\mathcal{G}^2$.

Theorem 10 positively answers Problem 1 for cost-prefix-linear, coalition-determined cost games.

Theorem 10. *In every initialised multiplayer cost game that is cost-prefix-linear and coalition-determined, there exists a Nash equilibrium.*

Proof. Let $(\mathcal{G} = (\Pi, V, (V_i)_{i\in\Pi}, E, (\mathsf{Cost}_i)_{i\in\Pi}), v_0)$ be an initialised multiplayer cost game that is cost-prefix-linear and coalition-determined. Thanks to the latter property, we know that, for every $i \in \Pi$, there exists a gain function $\mathsf{Gain}^i_{\text{Max}}$ such that the Min-Max cost game $\mathcal{G}^i = (V, V_i, V \setminus V_i, E, \mathsf{Cost}_i, \mathsf{Gain}^i_{\text{Max}})$ is determined and there exist optimal strategies $\sigma_i^\star$ and $\sigma_{-i}^\star$ for player i and the coalition $\Pi \setminus \{i\}$ respectively. In particular, for $j \neq i$, we denote by $\sigma_{j,i}^\star$ the strategy of player j derived from the strategy $\sigma_{-i}^\star$ of the coalition $\Pi \setminus \{i\}$.

The idea is to define the required Nash equilibrium as follows: each player i plays according to his strategy $\sigma_i^\star$ and punishes the first player $j \neq i$ who deviates from his strategy $\sigma_j^\star$, by playing according to $\sigma_{i,j}^\star$ (the strategy of player i derived from $\sigma_{-j}^\star$ in the game $\mathcal{G}^j$).

Formally, we consider the outcome of the optimal strategies $(\sigma_i^\star)_{i\in\Pi}$ from v_0, and set $\rho := \langle(\sigma_i^\star)_{i\in\Pi}\rangle_{v_0}$. We need to specify a punishment function $P : \mathsf{Hist} \to \Pi \cup \{\perp\}$ that detects who is the first player to deviate from the play ρ, i.e. who has to be punished. For the initial vertex v_0, we define $P(v_0) = \perp$ (meaning that nobody has deviated from ρ) and for every history $hv \in \mathsf{Hist}$, we let:

$$P(hv) := \begin{cases} \perp & \text{if } P(h) = \perp \text{ and } hv \text{ is a prefix of } \rho, \\ i & \text{if } P(h) = \perp, hv \text{ is not a prefix of } \rho, \text{ and } \mathsf{Last}(h) \in V_i, \\ P(h) & \text{otherwise } (P(h) \neq \perp). \end{cases}$$

Then the definition of the Nash equilibrium $(\tau_i)_{i\in\Pi}$ in $\mathcal{G}$ is as follows. For all $i \in \Pi$ and $h \in \mathsf{Hist}$ such that $\mathsf{Last}(h) \in V_i$,

$$\tau_i(h) := \begin{cases} \sigma_i^\star(h) & \text{if } P(h) = \perp \text{ or } i, \\ \sigma_{i,P(h)}^\star(h) & \text{otherwise.} \end{cases}$$

Clearly the outcome of $(\tau_i)_{i\in\Pi}$ is the play ρ $(= \langle(\sigma_i^\star)_{i\in\Pi}\rangle_{v_0})$.

Now we show that the strategy profile $(\tau_i)_{i\in\Pi}$ is a Nash equilibrium in $\mathcal{G}$. As a contradiction, let us assume that there exists a profitable deviation τ_j' for some player $j \in \Pi$. We denote by $\rho' := \langle\tau_j', (\tau_i)_{i\in\Pi\setminus\{j\}}\rangle_{v_0}$ the outcome where player j plays according to his profitable deviation τ_j' and the players of the coalition $\Pi \setminus \{j\}$ keep their strategies $(\tau_i)_{i\in\Pi\setminus\{j\}}$. Since τ_j' is a profitable deviation for player j w.r.t. $(\tau_i)_{i\in\Pi}$, we have that:

$$\mathsf{Cost}_j(\rho') < \mathsf{Cost}_j(\rho). \quad (1)$$

As both plays ρ and ρ' start from vertex v_0, there exists a history $hv \in \mathsf{Hist}$ such that $\rho = h\langle(\tau_i)_{i\in\Pi}\rangle_v$ and $\rho' = h\langle\tau_j', (\tau_i)_{i\in\Pi\setminus\{j\}}\rangle_v$ (remark that h could be empty).

Among the common prefixes of ρ and ρ', we choose the history hv of maximal length. By definition of the strategy profile $(\tau_i)_{i\in\Pi}$, we can write in the case of the outcome ρ that $\rho = h\langle(\sigma_i^\star)_{i\in\Pi}\rangle_v$. Whereas in the case of the outcome ρ', player j does not follow his strategy $\sigma_j^\star$ any more from vertex v, and so, the coalition $\Pi \setminus \{j\}$ punishes him by playing according to the strategy $\sigma_{-j}^\star$ after history hv, and so $\rho' = h\langle\tau_j', \sigma_{-j}^\star\rangle_v$ (see Figure 2).

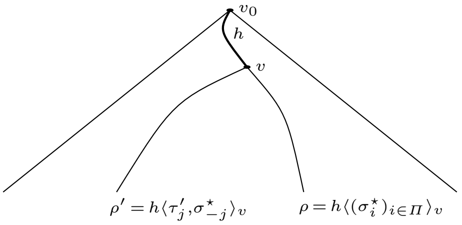

v_0
h
v
$\rho' = h\langle\tau_j',\sigma_{-j}^\star\rangle_v$
$\rho = h\langle(\sigma_i^\star)_{i\in\Pi}\rangle_v$

Fig. 2. Sketch of the tree representing the unravelling of the game $\mathcal{G}$ from v_0

Since $\sigma_{-j}^\star$ is an optimal strategy for the coalition $\Pi\setminus\{j\}$ in the determined Min-Max cost game $\mathcal{G}^j$, we have:

$$\begin{aligned}\mathsf{Val}^j(v) &= \inf_{\sigma_j\in\Sigma_{\text{Min}}} \mathsf{Gain}^j_{\text{Max}}(\langle\sigma_j, \sigma_{-j}^\star\rangle_v)\\ &\le \mathsf{Gain}^j_{\text{Max}}(\langle\tau_j', \sigma_{-j}^\star\rangle_v)\\ &\le \mathsf{Cost}_j(\langle\tau_j', \sigma_{-j}^\star\rangle_v)\,. \end{aligned} \tag{2}$$

The last inequality comes from the hypothesis $\mathsf{Cost}_j \ge \mathsf{Gain}^j_{\text{Max}}$ in the game $\mathcal{G}^j$.

Moreover, the game $\mathcal{G}$ is cost-prefix-linear, and then, when considering the history hv, there exist $a \in \mathbb{R}$ and $b \in \mathbb{R}^+$ such that

$$\mathsf{Cost}_j(\rho') = \mathsf{Cost}_j(h\langle\tau_j', \sigma_{-j}^\star\rangle_v) = a + b\cdot\mathsf{Cost}_j(\langle\tau_j', \sigma_{-j}^\star\rangle_v)\,. \tag{3}$$

As $b \ge 0$, Equations (2) and (3) imply:

$$\mathsf{Cost}_j(\rho') \ge a + b\cdot\mathsf{Val}^j(v)\,. \tag{4}$$

Since h is also a prefix of ρ, we have:

$$\mathsf{Cost}_j(\rho) = \mathsf{Cost}_j(h\langle(\sigma_i^\star)_{i\in\Pi}\rangle_v) = a + b\cdot\mathsf{Cost}_j(\langle(\sigma_i^\star)_{i\in\Pi}\rangle_v)\,. \tag{5}$$

Furthermore, as $\sigma_j^\star$ is an optimal strategy for player j in the Min-Max cost game $\mathcal{G}^j$, it follows that:

$$\begin{aligned}\mathsf{Val}^j(v) &= \sup_{\sigma_{-j}\in\Sigma_{\text{Max}}} \mathsf{Cost}_j(\langle\sigma_j^\star, \sigma_{-j}\rangle_v)\\ &\ge \mathsf{Cost}_j(\langle(\sigma_i^\star)_{i\in\Pi}\rangle_v)\,. \end{aligned} \tag{6}$$

Then, Equations (5) and (6) imply:

$$\mathsf{Cost}_j(\rho) \leq a + b \cdot \mathsf{Val}^j(v) \,. \tag{7}$$

Finally, Equations (4) and (7) lead to the following inequality:

$$\mathsf{Cost}_j(\rho) \leq a + b \cdot \mathsf{Val}^j(v) \leq \mathsf{Cost}_j(\rho') \,,$$

which contradicts Equation (1). The strategy profile $(\tau_i)_{i \in \Pi}$ is then a Nash equilibrium in the game $\mathcal{G}$. □

Remark 11. The proof of Theorem 10 remains valid for cost functions $\mathsf{Cost}_i : \mathsf{Plays} \to K$, where $\langle K, +, \cdot, 0, 1, \leq \rangle$ is an ordered field. This allows for instance to consider non-standard real costs and enjoy infinitesimals to model the costs of a player.

Example 12. Let us consider the initialised two-player cost game $(\mathcal{G}, A)$ of Example 3, where player 1 has a quantitative reachability objective (Cost_1) and player 2 has a mean-payoff objective (Cost_2). One can show that $\mathcal{G}$ is cost-prefix-linear. Since we saw in Example 9 that this game is also positionally coalition-determined, we can apply the construction in the proof of Theorem 10 to get a Nash equilibrium in $\mathcal{G}$. The construction from this proof may result in two different Nash equilibria, depending on the selection of the strategies $\sigma_1^\star/\tilde{\sigma}_1^\star$, $\sigma_{-1}^\star$, $\sigma_2^\star$ and $\sigma_{-2}^\star$ as defined in Example 9.

The first Nash equilibrium (τ_1, τ_2) with outcome $\rho = \langle \sigma_1^\star, \sigma_2^\star \rangle_A = A(BC)^\omega$ is given, for any history h, by:

$$\tau_1(hA) := \begin{cases} B & \text{if } P(hA) = \{\perp, 1\} \\ D & \text{otherwise} \end{cases} \quad ; \quad \tau_2(hB) := \begin{cases} C & \text{if } P(hB) = \{\perp, 2\} \\ A & \text{otherwise} \end{cases}$$

where the punishment function P is defined as in the proof of Theorem 10 and depends on the play ρ. The cost for this finite-memory Nash equilibrium is $\mathsf{Cost}_1(\rho) = 2 = \mathsf{Cost}_2(\rho)$.

The strategy $\tilde{\tau}_1$ of the second Nash equilibrium $(\tilde{\tau}_1, \tau_2)$ with outcome $\tilde{\rho} = \langle \tilde{\sigma}_1^\star, \sigma_2^\star \rangle_A = AD(BC)^\omega$ is given by $\tilde{\tau}_1(hA) := D$ for all history h. The cost for this finite-memory Nash equilibrium is $\mathsf{Cost}_1(\tilde{\rho}) = 6$ and $\mathsf{Cost}_2(\tilde{\rho}) = 2$, respectively.

Note that there is no positional Nash equilibrium with outcome ρ (resp. $\tilde{\rho}$).

The two following theorems provide results about the complexity of the Nash equilibrium defined in the latter proof. Applications of these theorems to specific classes of cost games are provided in Section 4.

Theorem 13. *In every initialised multiplayer cost game that is cost-prefix-linear and* positionally *coalition-determined, there exists a Nash equilibrium with memory (at most)* $|V| + |\Pi|$.

Theorem 14. *In every initialised multiplayer cost game that is cost-prefix-linear and* finite-memory *coalition-determined, there exists a Nash equilibrium with finite memory.*

The proofs of these two theorems rely on the construction of the Nash equilibrium provided in the proof of Theorem 10.

4 Applications

In this section, we exhibit several classes of *classical objectives* that can be encoded in our general setting. The list we propose is far from being exhaustive.

4.1 Qualitative Objectives

Multiplayer games with qualitative (win/lose) objectives can naturally be encoded via multiplayer cost games; for instance via cost functions $\mathsf{Cost}_i : \mathsf{Plays} \to \{1, +\infty\}$, where 1 (resp. $+\infty$) means that the play is won (resp. lost) by player i. Let us now consider the subclass of qualitative games with prefix-independent[6] Borel objectives. Given such a game $\mathcal{G}$, we have that $\mathcal{G}$ is coalition-determined, as a consequence of the Borel determinacy theorem [19]. Moreover the prefix-independence hypothesis obviously guarantees that $\mathcal{G}$ is also cost-prefix-linear (by taking $a = 0$ and $b = 1$). By applying Theorem 10, we obtain the existence of a Nash equilibrium for qualitative games with prefix-independent Borel objectives. Let us notice that this result is already present in [15].

When considering more specific subclasses of qualitative games enjoying a positional determinacy result, such as parity games [14], we can apply Theorem 13 and ensure existence of a Nash equilibrium whose memory is (at most) linear.

4.2 Classical Quantitative Objectives

We here give four well-known kinds of Min-Max cost games and see later that they are determined. For each sort of game, the cost and gain functions are defined from a price function (and a reward function in the last case), which labels the edges of the game graph with prices (and rewards).

Definition 15 ([26]). *Given a game graph $G = (V, V_{Min}, V_{Max}, E)$, a price function $\pi : E \to \mathbb{R}$ that assigns a price to each edge, a diverging[7] reward function $\vartheta : E \to \mathbb{R}$ that assigns a reward to each edge, and a play $\rho = \rho_0\rho_1 \dots$ in G, we define the following Min-Max cost games:*

1. *a* reachability-price game *is a Min-Max cost game $\mathcal{G} = (G, RP_{Min}, RP_{Max})$ together with a given goal set $\mathsf{Goal} \subseteq V$, where*

$$RP_{Min}(\rho) = RP_{Max}(\rho) = \begin{cases} \pi(\rho[0,n]) & \text{if } n \text{ is the least index s.t. } \rho_n \in \mathsf{Goal}, \\ +\infty & \text{otherwise,} \end{cases}$$

with $\pi(\rho[0,n]) = \sum_{i=1}^{n} \pi(\rho_{i-1}, \rho_i)$;

[6] An objective $\Omega \subseteq V^\omega$ is prefix-independent if only if for every play $\rho = \rho_0\rho_1 \dots \in V^\omega$, we have that $\rho \in \Omega$ iff for every $n \in \mathbb{N}$, $\rho_n\rho_{n+1} \dots \in \Omega$.

[7] For all plays $\rho = \rho_0\rho_1 \dots$ in G, it holds that $\lim_{n\to\infty} |\sum_{i=1}^{n} \vartheta(\rho_{i-1}, \rho_i)| = +\infty$. This is equivalent to requiring that every cycle has a positive sum of rewards.

2. *a* discounted-price game *is a Min-Max cost game* $\mathcal{G} = (G, DP_{Min}(\lambda), DP_{Max}(\lambda))$ *together with a given discount factor* $\lambda \in \,]0,1[$, *where*

$$DP_{Min}(\lambda)(\rho) = DP_{Max}(\lambda)(\rho) = (1-\lambda) \cdot \sum_{i=1}^{+\infty} \lambda^{i-1} \pi(\rho_{i-1}, \rho_i)\,;$$

3. *an* average-price game[8] *is a Min-Max cost game* $\mathcal{G} = (G, AP_{Min}, AP_{Max})$, *where*

$$AP_{Min}(\rho) = \limsup_{n \to +\infty} \frac{\pi(\rho[0,n])}{n} \quad \textit{and} \quad AP_{Max}(\rho) = \liminf_{n \to +\infty} \frac{\pi(\rho[0,n])}{n}\,;$$

4. *a* price-per-reward-average game *is a Min-Max cost game* $\mathcal{G} = (G, PRAvg_{Min}, PRAvg_{Max})$, *where*

$$PRAvg_{Min}(\rho) = \limsup_{n \to +\infty} \frac{\pi(\rho[0,n])}{\vartheta(\rho[0,n])} \quad \textit{and} \quad PRAvg_{Max}(\rho) = \liminf_{n \to +\infty} \frac{\pi(\rho[0,n])}{\vartheta(\rho[0,n])}\,,$$

with $\vartheta(\rho[0,n]) = \sum_{i=1}^{n} \vartheta(\rho_{i-1}, \rho_i)$.

An average-price game is then a particular case of a price-per-reward-average game. Let us remark that, in Example 3, the cost function Cost_1 (resp. Cost_2) corresponds to $\mathrm{RP_{Min}}$ with $\mathsf{Goal} = \{C\}$ (resp. $\mathrm{AP_{Min}}$). The game $\mathcal{G}^1$ (resp. $\mathcal{G}^2$) of Example 9 is a reachability-price (resp. average-price) game.

The following theorem is a well-known result about the particular cost games described in Definition 15.

Theorem 16 ([26,13]). *Reachability-price games, discounted-price games, average-price games, and price-per-reward games are determined and have positional optimal strategies.*

This result implies that a multiplayer cost game where each cost function is $\mathrm{RP_{Min}}$, $\mathrm{DP_{Min}}$, $\mathrm{AP_{Min}}$ or $\mathrm{PRAvg_{Min}}$ is positionally coalition-determined. Moreover, one can show that such a game is cost-prefix-linear. Theorem 17 then follows from Theorem 13.

Theorem 17. *In every initialised multiplayer cost game* $\mathcal{G} = (\Pi, V, (V_i)_{i \in \Pi}, E, (\mathsf{Cost}_i)_{i \in \Pi})$ *where the cost function* Cost_i *belongs to* $\{RP_{Min}, DP_{Min}, AP_{Min}, PRAvg_{Min}\}$ *for every player* $i \in \Pi$, *there exists a Nash equilibrium with memory (at most)* $|V| + |\Pi|$.

Note that the existence of finite-memory Nash equilibria in cost games with quantitative reachability objectives has already been established in [6,7]. Even if not explicitly stated in the previous papers, one can deduce from the proof of [7, Lemma 16] that the provided Nash equilibrium has a memory (at least) exponential in the size of the cost game. Thus, Theorem 17 significantly improves the complexity of the strategies constructed in the case of cost games with quantitative reachability objectives.

[8] When the cost function of a player is $\mathrm{AP_{Min}}$, we say that he has a mean-payoff objective.

4.3 Combining Qualitative and Quantitative Objectives

Multiplayer cost games allow to encode games combining both qualitative and quantitative objectives, such as *mean-payoff parity games* [10]. In our framework, where each player aims at minimising his cost, the mean-payoff parity objective could be encoded as follows: $\mathsf{Cost}_i(\rho) = \mathrm{AP}_{\mathrm{Min}}(\rho)$ if the parity condition is satisfied, $+\infty$ otherwise.

The determinacy of mean-payoff parity games, together with the existence of optimal strategies (that could require infinite memory) have been proved in [10]. This result implies that multiplayer cost games with mean-payoff parity objectives are coalition-determined. Moreover, one can prove that such a game is also cost-prefix-linear (by taking $a = 0$ and $b = 1$). By applying Theorem 10, we obtain the existence of a Nash equilibrium for multiplayer cost games with mean-payoff parity objectives. As far as we know, this is the first result about the existence of a Nash equilibrium in cost games with mean-payoff parity games.

Remark 18. Let us emphasise that Theorem 10 applies to cost games where the players have different kinds of cost functions (as in Example 3). In particular, one player could have a qualitative Büchi objective, a second player a discounted-price objective, a third player a mean-payoff parity objective,...

We would like to thank Quentin Menet for some helpful discussions. This work has been partly supported by a grant from the National Bank of Belgium, and the ARC project (number AUWB-2010-10/15-UMONS-3). The second author is supported by a grant from F.R.S.-FNRS. The third author is supported by the Engineering and Physical Science Research Council (EPSRC) through grant EP/H046623/1 ‘Synthesis and Verification in Markov Game Structures’.

References

1. Alur, R., Kanade, A., Weiss, G.: Ranking Automata and Games for Prioritized Requirements. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 240–253. Springer, Heidelberg (2008)
2. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better Quality in Synthesis through Quantitative Objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009)
3. Boros, E., Gurvich, V.: Why chess and back gammon can be solved in pure positional uniformly optimal strategies. Rutcor Research Report 21-2009, Rutgers University (2009)
4. Bouyer, P., Brenguier, R., Markey, N.: Nash Equilibria for Reachability Objectives in Multiplayer Timed Games. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 192–206. Springer, Heidelberg (2010)
5. Bouyer, P., Brenguier, R., Markey, N., Ummels, M.: Nash equilibria in concurrent games with Büchi objectives. In: FSTTCS. LIPIcs, vol. 13, pp. 375–386. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
6. Brihaye, T., Bruyère, V., De Pril, J.: Equilibria in Quantitative Reachability Games. In: Ablayev, F., Mayr, E.W. (eds.) CSR 2010. LNCS, vol. 6072, pp. 72–83. Springer, Heidelberg (2010)
7. Brihaye, T., Bruyère, V., De Pril, J.: On equilibria in quantitative games with reachability/safety objectives. CoRR, abs/1205.4889 (2012)

8. Brihaye, T., De Pril, J., Schewe, S.: Multiplayer Cost Games with Simple Nash Equilibria. Technical report, arXiv:1210.3548 [cs.GT] (2012)
9. Chatterjee, K., Henzinger, T.A.: Finitary Winning in ω-Regular Games. In: Hermanns, H. (ed.) TACAS 2006. LNCS, vol. 3920, pp. 257–271. Springer, Heidelberg (2006)
10. Chatterjee, K., Henzinger, T.A., Jurdzinski, M.: Mean-payoff parity games. In: LICS, pp. 178–187. IEEE Computer Society (2005)
11. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (2000)
12. Fijalkow, N., Zimmermann, M.: Cost-parity and cost-streett games. CoRR, abs/1207.0663 (2012)
13. Filar, J., Vrieze, K.: Competitive Markov decision processes. Springer (1997)
14. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS, vol. 2500. Springer, Heidelberg (2002)
15. Grädel, E., Ummels, M.: Solution concepts and algorithms for infinite multiplayer games. In: New Perspectives on Games and Interaction. Texts in Logic and Games, vol. 4, pp. 151–178. Amsterdam University Press (2008)
16. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Co., Reading (1979)
17. Horn, F., Thomas, W., Wallmeier, N.: Optimal Strategy Synthesis in Request-Response Games. In: Cha, S(S.), Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 361–373. Springer, Heidelberg (2008)
18. Klimoš, M., Larsen, K.G., Štefaňák, F., Thaarup, J.: Nash Equilibria in Concurrent Priced Games. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 363–376. Springer, Heidelberg (2012)
19. Martin, D.A.: Borel determinacy. Ann. of Math. (2) 102(2), 363–371 (1975)
20. Nash, J.: Equilibrium points in n-person games. Proceedings of the National Academy of Sciences of the United States of America 36(1), 48–49 (1950)
21. Osborne, M., Rubinstein, A.: A course in game theory. MIT Press, Cambridge (1994)
22. Paul, S., Simon, S., Kannan, R., Kumar, K.: Nash equilibrium in generalised muller games. In: FSTTCS. LIPIcs, vol. 4, pp. 335–346. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2009)
23. Thomas, W.: On the Synthesis of Strategies in Infinite Games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)
24. Thomas, W.: Church's Problem and a Tour through Automata Theory. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 635–655. Springer, Heidelberg (2008)
25. Thuijsman, F., Raghavan, T.E.S.: Perfect information stochastic games and related classes. International Journal of Game Theory 26(3), 403–408 (1998)
26. Trivedi, A.: Competative optimisation on timed automata. PhD thesis, University of Warwick (2009)
27. Zimmermann, M.: Time-Optimal Winning Strategies for Poset Games. In: Maneth, S. (ed.) CIAA 2009. LNCS, vol. 5642, pp. 217–226. Springer, Heidelberg (2009)

Forward Chaining for Hybrid ASP

Alex Brik[*] and Jeffrey B. Remmel

Department of Mathematics, University of California, San Diego,
La Jolla, CA 92093-0112

Abstract. In this paper, we define an analogue of the Forward Chaining (FC) algorithm due to Marek, Nerode, and Remmel [12] for Hybrid Answer Set Programming (H-ASP). The FC algorithm for normal logic programs takes as an input a well ordering $\prec$ of the non-Horn clauses of a normal logic program P and produces a stable model $D^{\prec}$ for a subprogram $A^{\prec}$ of P. It is the case that for every stable model M of P, there is a well ordering $\prec$ such that $D^{\prec} = M$ and $A^{\prec} = P$. Thus the search for a stable model of P becomes a search for a well ordering $\prec$ such that $A^{\prec} = P$. We show that a similar result hold in case of FC for H-ASP. H-ASP is an extension of normal logic programming or Answer Set Programming (ASP), introduced by the authors in [2] that allows users to combine ASP type rules and numerical algorithms. The MFC algorithm, introduced by the authors in [1] is a Monte Carlo algorithm that combines the FC algorithm and the Metropolis-Hastings algorithm to search for stable models of normal logic programs. We shall briefly discuss how one can produce an analogue of the MFC algorithm to search for stable models of H-ASP programs.

1 Introduction

In previous paper [1], the authors developed a Monte Carlo type algorithm called the Metropolized Forward Chaining (MFC) algorithm to solve the following two problems.
(1) *Given a finite propositional logic program P which has a stable model, find a stable model M of P.*
(2) *Given a finite proposition logic program P which has no stable model, find a maximal program $P' \subseteq P$ which has a stable model and find a stable model M' of P'.*

The MFC algorithm combines the Forward Chaining algorithm of Marek, Nerode, and Remmel [12] and the Metropolis algorithm introduced by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller [13].

Marek, Nerode and Remmel [12] developed the Forward Chaining (FC) algorithm to solve both problems 1 and 2. The basic idea of the FC algorithm was to start with a finite normal logic program P and divide P into is its monotonic part $mon(P)$, consisting of the set of Horn clauses in P, and its non-monotonic part $nmon(P)$, consisting of those clauses in P which contain negations in its

[*] Currently at Google Inc.; this work was completed before the change in affiliation.

body. Then for any ordering or permutation σ of $nmon(P)$, the FC algorithm produces a subset $mon(P) \subseteq A^{\sigma} \subseteq P$ and a stable model D^{σ} of A^{σ}. Marek, Nerode, and Remmel showed that for each stable model M of P, there is a permutation σ_M of $nmon(P)$ such that $A^{\sigma_M} = P$ and $D^{\sigma_M} = M$. Thus every stable model can be computed by the FC algorithm relative to a suitable ordering of the nonmonotonic rules in P. Marek, Nerode, and Remmel also noted that in the case where P has no stable models, the FC algorithm automatically produces a subprogram of P which does have a stable model.

The Metropolis algorithm is a widely applicable procedure for drawing samples from a specified distribution on large finite set. That is, let X be a finite set and $\pi(x)$ be a probability distribution on X. The algorithm requires that one specifies a connected, aperiodic Markov chain $K(x, y)$ on X. In the Metropolis algorithm $K\,(x, y)$ needs to be symmetric. However, the algorithm was later generalized to the Metropolis-Hastings algorithm by Hastings [8]. In the Metropolis-Hastings algorithm $K(x, y)$ need not be symmetric but it must be the case that $K(x, y) > 0$ if and only if $K(y, x) > 0$. The chain K is then modified by an auxiliary coin tossing procedure to create a new chain M with stationary distribution π. That is, if the chain is currently at x, one chooses y from $K(x, y)$. Then one defines an acceptance ratio by

$$A(x, y) = \frac{\pi(y)K(y, x)}{\pi(x)K(x, y)} \tag{1}$$

If $A(x, y) \geq 1$, then the chain moves to y. If $A(x, y) < 1$, then we flip a coin with probability of heads equal to $A(x, y)$. If the coin comes up heads, then we move to y and, if the coin comes up tails, we stay at x. See the survey article by Diaconis and Saloff-Coste [4] for a survey about what is known about the Metropolis algorithm.

From the point of view of the FC algorithm, the search for a stable model of a finite normal logic program P is a search through the set of possible well-orderings of $nmon(P)$ until one finds a suitable well ordering $\prec$ such that FC algorithm relative to $\prec$ produces a stable model of P. Thus the Metropolis algorithm in MFC uses an appropriate Markov chain $K(x, y)$ defined on the space of permutations of $nmon(P)$. In [1], we defined several possible Markov chains that could be employed and ran computer experiments to find good Markov chains for certain classes of programs. We showed that in some cases, MFC could find stable models of programs that were not found by ASP search engines *clasp* [6] and *smodels* [14].

The main goal of this paper is to develop an analogue of the FC algorithm for a far reaching extension of the logic programming called Hybrid Answer Set Programming (H-ASP). H-ASP, introduced by the authors in [2], is an extension of Answer Set Programming (ASP) that allows users to combine ASP type rules and numerical algorithms. The goal of H-ASP is to allow users to reason about dynamical systems that exhibit both discrete and continuous aspects. The unique feature of H-ASP is that H-ASP rules can be thought of as general input-output devices. In particular, H-ASP programs allow users to include ASP type rules that act as controls for when to apply a given algorithm to advance the system to the next position.

FC for H-ASP programs is more complicated than FC for normal logic programs. First, H-ASP rules allow for nondeterministic outputs which is not compatible with FC. Thus, besides a well ordering $\prec$ of the nonmonotonic rules one has to have an additional parameter F which specifies a selector function. F has the effect of choosing a fixed deterministic output for each rule. Second, stable models are dependent on fixing an initial condition I. Thus in the analogue of the MFC algorithm of H-ASP programs, one has to define an appropriate Markov chain on triples $(\prec, I, F)$ or, if we fix an initial condition I, on pairs $(\prec, F)$.

In this paper we will be considering a finite version of FC for H-ASP. That is we will assume that every H-ASP program considered has a finite set of rules, that every algorithm on every invocation produces a finite set as an output and that every stable model of a program is finite.

The outline of this paper is as follows. In section 2, we shall give the basic definitions for both normal logic programs and H-ASP programs. In section 3, we shall review the FC algorithm of Marek, Nerode, and Remmel [12]. In section 4, we shall define the FC algorithm for H-ASP programs. In section 5, we shall define a possible analogue of the MFC algorithm for H-ASP programs. Finally, in section 6, we shall state conclusions and discuss directions for further research.

2 Hybrid ASP Programs

We will now give a brief overview of normal logic programs and then a brief overview of H-ASP programs.

A *normal propositional logic programming rule* is an expression of the form

$$C = p \leftarrow q_1, \ldots, q_m, \text{not } r_1, \ldots, \text{not } r_n \tag{2}$$

where $p, q_1, \ldots, q_m, r_1, \ldots, r_n$ are atoms from a fixed set of atoms At. The atom p in the rule above is called the *head* of C (*head*(C)), and the expression $q_1, \ldots, q_m, \text{not } r_1, \ldots, \text{not } r_n$, with ‘,’ interpreted as conjunction, is called the *body* of C (*body*(C)). The set $\{q_1, \ldots, q_m\}$ is called the *positive part of the body* of C (*posBody*(C)) (or *premises* of C) and the set $\{r_1, \ldots, r_n\}$ is called the *negative part of the body* of C (*negBody*(C)) (or *constraints* of C). Given any set $M \subseteq At$ and atom a, we say that M satisfies a (not a), written $M \models a$ ($M \models \text{not } a$), if $a \in M$ ($a \notin M$). For a rule C of the form (2), we say that M satisfies the body of C if M satisfies q_i for $i = 1, ..., m$ and M satisfies not r_j for $j = 1, ..., n$. We say that M satisfies C, written $M \models C$, if whenever M satisfies the body of C, then M satisfies the head of C. A normal logic program P is a set of rules of the form of (2). We say that $M \subseteq At$ is a model of P, written $M \models P$, if M satisfies every rule of P.

A propositional Horn rule is a logic programming rule of the form

$$H = p \leftarrow q_1, \ldots, q_m$$

where $p, q_1, \ldots, q_m \in At$. Thus in a Horn rule, the negative part of its body is empty. A Horn program P is a set of Horn rules. Each Horn program P has a least model under inclusion, LM_P, which can be defined using the one-step provability operator T_P. For any set A, let $\mathcal{P}(A)$ denote the set of all subsets of A. The one-step provability operator $T_P : \mathcal{P}(At) \to \mathcal{P}(At)$ associated with the Horn program P [15] is defined by setting

$$T_P(M) = M \cup \{p : \exists C \in P(p = \mathit{head}(C) \wedge M \models \mathit{body}(C))\}$$

for any $M \in \mathcal{P}(A)$. We define $T_P^n(M)$ by induction by setting $T_P^0(M) = M$, $T_P^1(M) = T_P(M)$ and $T_P^{n+1}(M) = T_P(T_P^n(M))$. Then the least model LM_P can be computed as $LM_P = T_P(\emptyset) \uparrow \omega = \bigcup_{n \geq 0} T_P^n(\emptyset)$.

If P is a normal logic program and $M \subseteq At$, then the Gelfond-Lifschitz reduct of P with respect to M [7] is the Horn program P^M which results by eliminating those rules C of the form (2) such that $r_i \in M$ for some i and replacing C by $p \leftarrow q_1, \ldots, q_n$ otherwise. We then say that M is a *stable model* for P if M equals the least model of P^M.

An *answer set programming rule* is an expression of the form (2) where $p, q_1, \ldots, q_m, r_1, \ldots, r_n$ are classical literals, i.e., either positive atoms or atoms preceded by the classical negation sign $\neg$. Answer sets are defined in analogy to stable models, but taking into account that atoms may be preceded by classical negation.

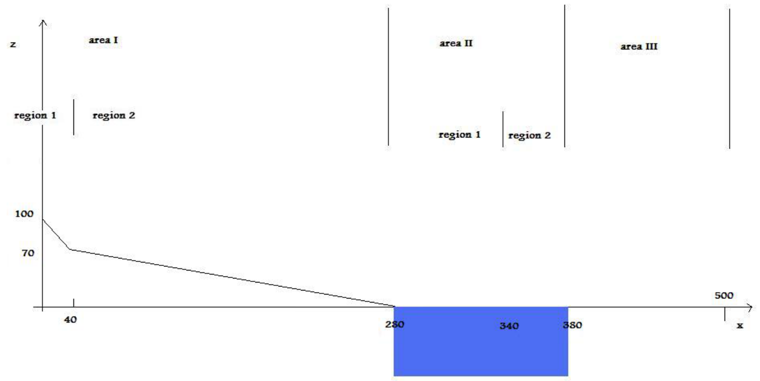


Fig. 1. A cross section of the regions to be traversed by Secret Agent 00111

Next we shall present the basic definitions of Hybrid ASP programs and define an analogue of stable models for such programs. To help motivate our definitions, we shall consider the following toy example. Imagine that Secret Agent 00111 (the agent, for short) needs to move through a domain consisting of 3 areas: Area I, Area II, and Area III. The domain's vertical cross section is shown on the diagram 1. Area I is a mountain, Area II is a lake, and Area III is a desert.

The agent needs to descend down the mountain in his car until he reaches the lake at which point the car can be reconfigured so that it can be used as a boat that can navigate across the lake. We shall assume that the lake has a water current moving with a constant speed and a constant direction. If the agent is pursued by the evil agents, then he will attempt to travel through the lake as fast as possible. If the agent is not pursued by the evil agents then he would like to exit the lake at a point with a y-coordinate being close to the y-coordinate of the point of his entrance into the lake. To accomplish this, the agent will be able to steer the boat in directions which make various angles to the x-axis.

A H-ASP program P has an underlying parameter space S. Elements of S are of the form $\mathbf{p} = (t, x_1, \ldots, x_m)$ where t is time and x_i are parameter values. We shall let $t(\mathbf{p})$ denote t and $x_i(\mathbf{p})$ denote x_i for $i = 1, \ldots, m$. We refer to the elements of S as *generalized positions.* For example, in our secret agent example, a generalized position would naturally have continuous parameters such as time, position, velocity, and acceleration as well as discrete parameters such as is the car currently configured as a car or as a boat and a parameter to tell us whether the agent is being pursed by evil agents or not. Let At be a set of atoms of P. Then the universe of P is $At \times S$.

If $M \subseteq At \times S$, we let $GP(M) = \{\mathbf{p} \in S : (\exists a \in At)((a, \mathbf{p}) \in M)\}$. For $I \in S$ let $GP_I(M) = GP(M) \cup \{I\}$. A block B is an object of the form $B = a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m$ where $a_1, \ldots, a_n, b_1, \ldots, b_m \in At$. Given $M \subseteq At \times S$ and $\mathbf{p} \in S$, we say that M satisfies B at the generalized position $\mathbf{p}$, written $M \models (B, \mathbf{p})$, if $(a_i, \mathbf{p}) \in M$ for $i = 1, \ldots, n$ and $(b_j, \mathbf{p}) \notin M$ for $j = 1, \ldots, m$. If B is empty, then $M \models (B, \mathbf{p})$ automatically holds. We define $B^- = \text{not } b_1, \ldots, \text{not } b_m$.

There are two types of rules in H-ASP.

Advancing rules are of the form

$$r = \frac{B_1; B_2; \ldots; B_k : A, O}{a} \tag{3}$$

where A is an algorithm, for each i, $B_i = a_{1,i}, \ldots, a_{n_i,i}, \text{not } b_{1,i}, \ldots, \text{not } b_{m_i,i}$ where $a_{1,i}, \ldots, a_{n_i,i}, b_{1,i}, \ldots, b_{m_i,i}$ are atoms, and O is a subset of S^k such that if $(\mathbf{p}_1, \ldots, \mathbf{p}_k) \in O$, then $t(\mathbf{p}_1) < \cdots < t(\mathbf{p}_k)$ and $A(\mathbf{p}_1, \ldots, \mathbf{p}_k)$ is a subset of S such that for all $\mathbf{q} \in A(\mathbf{p}_1, \ldots, \mathbf{p}_k)$, $t(\mathbf{q}) > t(\mathbf{p}_k)$. Here and in the next rule, we allow n_i or m_i to be equal to 0 for any given i. O is called the *constraint parameter set* of the rule r and will be denoted by $CPS(r)$. A is called the *advancing algorithm* of the rule r and is denoted by $alg(r)$. The constraint atom set of r, $CAS(r)$, is defined to be $\bigcup_{i=1}^{k}\{b_{1,i}, \ldots, b_{m_i,i}\}$. The arity of rule r, $N(r)$, is equal to k.

The idea is that if $(\mathbf{p}_1, \ldots, \mathbf{p}_k) \in O$ and for each i, B_i is satisfied at the generalized position $\mathbf{p}_i$, then the algorithm A can be applied to $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$ to produce a set of generalized positions O' such that if $\mathbf{q} \in O'$, then $t(\mathbf{q}) > t(\mathbf{p}_k)$ and $(a, \mathbf{q})$ holds. Thus advancing rules are like input-output devices in that the algorithm A allows the user do derive possible successor generalized positions as well as certain atoms a which are to hold at such positions. Here the advancing algorithm A can be quite arbitrary in that it may involve algorithms for solving differential or integral equations, solving a set of linear equations or linear programming equations, solving an optimization problem, etc. For example, in [3], we incorporated algorithms for Markov Decision Processes into our advancing algorithms to construct H-ASP programs that can be used to create efficient, robust representations of dynamic domains and to compute optimal finite horizon policies for the agents acting in such domains.

Stationary rules are of the form

$$r = \frac{B_1; B_2; \ldots; B_k : H, O}{a} \tag{4}$$

where for each i, $B_i = a_{1,i}, \ldots, a_{n_i,i}, \text{not } b_{1,i}, \ldots, \text{not } b_{m_i,i}$ where $a_{1,i}, \ldots, a_{n_i,i}$, $b_{1,i}, \ldots, b_{m_i,i}$ are atoms, H is a Boolean algorithm, and $O \subseteq S^k$. As in advancing rules, we let $CPS(r) = O$, $alg(r) = H$, $CAS(r) = \bigcup_{i=1}^{k}\{b_{1,i}, \ldots, b_{m_i,i}\}$, and $N(r) = k$.

The idea is that if $(\mathbf{p}_1, \ldots, \mathbf{p}_k) \in O$ and for each i, B_i is satisfied at the generalized position $\mathbf{p}_i$, and $H((\mathbf{p}_1, \ldots, \mathbf{p}_k))$ is true, then $(a, \mathbf{p}_k)$ holds. Stationary rules are much like normal logic programming rules in that they allow us to derive new atoms at a given generalized position $\mathbf{p}_k$. The difference is that a derivation with our stationary rules can be based on what happens at multiple times in the past and the Boolean algorithm H can be any sort of algorithm which gives either true or false as on output.

For a model $M \subseteq At \times S$ and an initial condition $I \in S$, a state corresponds to a generalized position $\mathbf{p} \in GP_I(M)$ together with the set of all the atoms that hold at $\mathbf{p}$ relative to M, i.e. $\{a \mid \exists (a, \mathbf{p}) \in M\}$. Given a generalized position $\mathbf{p}$ at time t, we use advancing rules to generate the set of possible "next" generalized positions at time $t + \Delta$ for some $\Delta > 0$. For example, in the Secret Agent example, if the agent is in the lake, then each "next" generalized position $\mathbf{p}'$ would specify not only the position at $t + \Delta$ but also a different steering angle to be used from time t to $t + \Delta$. Then for such a "next" generalized position $\mathbf{p}'$, stationary rules can be applied to a pair $(\mathbf{p}, \mathbf{p}')$ to derive more atoms at $\mathbf{p}'$ or to enforce constraints.

A H-ASP program P is a collection of H-ASP advancing and stationary rules. To define the notion of a stable model of P, we first must define the notion of a H-ASP Horn program and the one-step provability operator for H-ASP Horn programs.

A *H-ASP Horn program* is a H-ASP program which does not contain any negated atoms. Let P be a Horn H-ASP program and $I \in S$ be an initial condition. Then the one-step provability operator $T_{P,I}$ is defined so that given $M \subseteq At \times S$, $T_{P,I}(M)$ consists of M together with the set of all $(a, J) \in At \times S$ such that

(1) there exists a stationary rule $r = \frac{B_1; B_2; \ldots; B_k : H, O}{a}$ and $(\mathbf{p}_1, \ldots, \mathbf{p}_k) \in O \cap (GP_I(M))^k$ such that $(a, J) = (a, \mathbf{p}_k)$, $M \models (B_i, \mathbf{p}_i)$ for $i = 1, \ldots, k$, and $H(\mathbf{p}_1, \ldots, \mathbf{p}_k) = 1$ or

(2) there exists an advancing rule $r = \frac{B_1;B_2;\ldots;B_k:A,O}{a}$ and $(\mathbf{p}_1,\ldots,\mathbf{p}_k) \in O \cap (GP_I(M))^k$ such that $J \in A(\mathbf{p}_1,\ldots,\mathbf{p}_k)$ and $M \models (B_i,\mathbf{p}_i)$ for $i = 1,\ldots,k$.

The stable model semantics for H-ASP programs is defined as follows. Let $M \subseteq At \times S$ and I be an initial condition in S. An H-ASP rule $C = \frac{B_1;\ldots,B_k:A,O}{a}$ is *inapplicable* for (M,I) if for all $(\mathbf{p}_1,\ldots,\mathbf{p}_k) \in O \cap (GP_I(M))^k$, either (i) there is an i such that $M \not\models (B_i^-,\mathbf{p}_i)$, (ii) $A(\mathbf{p}_1,\ldots,\mathbf{p}_r) \cap GP_I(M) = \emptyset$ if A is an advancing algorithm, or (iii) $A(\mathbf{p}_1,\ldots,\mathbf{p}_k) = 0$ if A is a Boolean algorithm. Then we form the Gelfond-Lifschitz reduct of P over M and I, $P^{M,I}$ as follows.
(1) Eliminate all rules which are inapplicable for (M,I).
(2) If the advancing rule $r = \frac{B_1;\ldots,B_k:A,O}{a}$ is not eliminated by (1), then replace it by $\frac{B_1^+;\ldots,B_k^+:A^+,O^+}{a}$ where for each i, B_i^+ is the result of removing all the negated atoms from B_i, O^+ is equal to the set of all $(\mathbf{p}_1,\ldots,\mathbf{p}_k)$ in $O \cap (GP_I(M))^k$ such that $M \models (B_i^-,\mathbf{p}_i)$ for $i = 1,\ldots,k$ and $A(\mathbf{p}_1,\ldots,\mathbf{p}_k) \cap GP_I(M) \neq \emptyset$, and $A^+(\mathbf{p}_1,\ldots,\mathbf{p}_k)$ is defined to be $A(\mathbf{p}_1,\ldots,\mathbf{p}_k) \cap GP_I(M)$.
(3) If the stationary rule $r = \frac{B_1;\ldots,B_k:H,O}{a}$ is not eliminated by (1), then replace it by $\frac{B_1^+;\ldots,B_k^+:H|_{O^+},O^+}{a}$ where for each i, B_i^+ is the result of removing all the negated atoms from B_i, O^+ is equal to the set of all $(\mathbf{p}_1,\ldots,\mathbf{p}_k)$ in $O \cap (GP_I(M))^k$ such that $M \models (B_i^-,\mathbf{p}_i)$ for $i = 1,\ldots,k$ and $H(\mathbf{p}_1,\ldots,\mathbf{p}_k) = 1$.

We then say that M is a *stable model of* P *with initial condition* I if $\bigcup_{k=0}^{\infty} T^k_{P^{M,I},I}(\emptyset) = M$.

3 The Forward Chaining Algorithm for Normal Logic Programs

Let P be a normal logic program. We let $H(P)$ denote the Herbrand base of P, i.e. the underlying set of atoms of the program. We let $mon(P)$ denote the set of all Horn clauses of P and $nmon(P) = P \setminus mon(P)$. The elements of $nmon(P)$ will be called *nonmonotonic* clauses. The Forward Chaining algorithm of [12] applies to programs of arbitrary cardinality. It takes as an input a program P and a well-ordering $\prec$ of $nmon(P)$. The principal output of the Forward Chaining construction will be a subset $D^{\prec}$ of $H(P)$. Although such subset is not, necessarily, a stable model of P, it will be a stable model of $A^{\prec}$ for a subset $A^{\prec} \subseteq P$. This subset, $A^{\prec}$, is also computed out of Forward Chaining construction and is the maximal set of clauses for which $D^{\prec}$ is a stable model.

For any set $S \subseteq H(P)$, the monotonic closure of S relative of $mon(P)$, $cl_{mon}(S)$ is defined by

$$cl_{mon}(S) = T_{mon(P)}(S) \uparrow \omega. \tag{5}$$

The general Forward Chaining algorithm is the following.

Forward Chaining Algorithm
Let P be a normal logic program and let $\prec$ be a well-ordering of $nmon(P)$.

Two sequences of sets of atoms from $H(P)$, $\langle D_\xi \rangle_{\xi \in \alpha^+}$ and $\langle R_\xi \rangle_{\xi \in \alpha^+}$ are defined, where α^+ is the least cardinal greater than the ordinal α determined by the well ordering $\prec$. The set D_ξ is the set of atoms *derived* by stage ξ and R_ξ is the set of atoms *rejected* by the stage ξ.
1. Set $D_0^\prec = cl_{mon}(\emptyset)$ and $R_0^\prec = \emptyset$.

2. If $\gamma = \beta + 1$, then $C \in nmon(P)$ is an **applicable clause** at stage γ if (a) $\text{posBody}(C) \subseteq D_\beta^\prec$, (b) $(\{\text{head}(C)\} \cup \text{negBody}(C)) \cap D_\beta^\prec = \emptyset$, and (c) $cl_{mon}(D_\beta^\prec \cup \{\text{head}(C)\}) \cap (\text{negBody}(C) \cup R_\beta^\prec) = \emptyset$.

If there is no applicable clause at stage γ, then let $D_\gamma^\prec = D_\beta^\prec$ and $R_\gamma^\prec = R_\beta^\prec$. Otherwise, let C_γ be the $\prec$-first applicable clause and set

$$D_\gamma^\prec = cl_{mon}(D_\beta^\prec \cup \{\text{head}(C_\gamma)\}) \quad R_\gamma^\prec = R_\beta^\prec \cup \text{negBody}(C_\gamma).$$

3. If γ is a limit ordinal, then set $D_\gamma^\prec = \bigcup_{\xi<\gamma} D_\xi^\prec$ and $R_\gamma^\prec = \bigcup_{\xi<\gamma} R_\xi^\prec$.
4. Finally let

$$D^\prec = D_{\alpha^+}^\prec = \bigcup_{\xi \in \alpha^+} D_\xi^\prec \quad \text{and} \quad R^\prec = R_{\alpha^+}^\prec = \bigcup_{\xi \in \alpha^+} R_\xi^\prec.$$

The sets $D^\prec$ and $R^\prec$ are called the sets of *derived* atoms and *rejected* atoms of the Forward Chaining construction relative to $\prec$ respectively.

C is *inconsistent* relative to $\prec$ if $(\{\text{head}(C)\} \cup \text{negBody}(C)) \cap D^\prec = \emptyset$, $\text{posBody}(C) \subseteq D^\prec$, but $cl_{mon}(D^\prec \cup \{\text{head}(C)\}) \cap (\text{negBody}(C) \cup R^\prec) \neq \emptyset$. We let $I^\prec = \{C : C \text{ is inconsistent}\}$ and $A^\prec = P \setminus I^\prec$.

Marek, Nerode, and Remmel [12] proved the following results.

Theorem 1. *Let P be a normal propositional logic program.*

1. *Let $\prec$ be a well-ordering of $nmon(P)$. Then $D^\prec$ is a stable model of $A^\prec$. Hence if $I^\prec = \emptyset$, then $D^\prec$ is a stable model of P.*
2. *If M is a stable model of P, then there exists a well-ordering $\prec$ of $nmon(P)$ such that $D^\prec = M$. In fact, for every well-ordering $\prec$ such that*

$$NG(M,P) = \{C \in nmon(P) : \text{posBody}(C) \subseteq M, \text{negBody}(C) \cap M = \emptyset\}$$

forms an initial segment of $\prec$, $D^\prec = M$ and $A^\prec = P$.

For finite programs a well ordering of nmon(P) is determined by taking a permutation σ of $nmon(P)$.

Example 1. Let $H = \{a, b, c, d, e, f\}$ and let P consist of the following clauses:

$$\begin{array}{lll} 1.\ a \leftarrow & 2.\ b \leftarrow c & 3.\ c \leftarrow a, \neg d \\ 4.\ d \leftarrow b, \neg c & 5.\ e \leftarrow c, \neg f & 6.\ f \leftarrow c, \neg e. \end{array}$$

Here, $mon(P)$ consists of clauses (1) and (2), whereas $nmon(P)$ consists of clauses (3), (4), (5), and (6).

Let $\prec$ of $nmon(P)$ be defined as (3) $\prec$ (4) $\prec$ (5) $\prec$ (6). Then the construction of sets $D_n^{\prec}$ and $R_n^{\prec}$ is as follows:
Stage 0 $D_0^{\prec} = cl_{mon}(\emptyset) = \{a\}$, $R_0^{\prec} = \emptyset$.
Stage 1 $r_1 = (3)$, $D_1^{\prec} = cl_{mon}(\{a\} \cup \{c\}) = \{a, b, c\}$, $R_1^{\prec} = \{d\}$.
Stage 2 $r_2 = (5)$, $D_2^{\prec} = \{a, b, c, e\}$, and $R_2^{\prec} = \{d, f\}$.
Stage 3 At this stage our construction stabilizes.
It is easy to see that $I^{\prec} = \emptyset$ so $D^{\prec} = D_2^{\prec}$ is an stable model of P.

Now, let $\prec'$ be an ordering of $nmon(P)$ as follows: (4) $\prec'$ (3) $\prec'$ (6) $\prec'$ (5). Here, the construction of $D^{\prec'}$ produces these stages:
Stage 0 $D_0^{\prec'} = cl_{mon}(\emptyset) = \{a\}$, $R_0^{\prec'} = \emptyset$.
Stage 1 $r_1 = (3)$, $D_1^{\prec'} = cl_{mon}(\{a\} \cup \{c\}) = \{a, b, c\}$, $R_1^{\prec'} = \{d\}$.
Stage 2 $r_2 = (6)$, $D_2^{\prec'} = \{a, b, c, f\}$, and $R_2^{\prec'} = \{d, e\}$.
Stage 3 At this stage our construction stabilizes.
Again, it is easy to see that $I^{\prec'} = \emptyset$ so $D^{\prec'} = D_2^{\prec'}$ is a stable model of P. These are the only stable models of P and one can check that the Forward Chaining construction will produce one of these two stable models for any well-ordering of $nmon(P)$. □

4 The Forward Chaining for Hybrid ASP Programs

In this section, we shall describe how one can adapt the FC algorithm to H-ASP programs. The main difficulty is that advancing rules in H-ASP are inherently nondeterministic because advancing algorithms are set-valued functions. We have used this feature in applications where one wants to reason about all possible trajectories of a dynamical system rather than a single trajectory; see [3] for an example. Unfortunately, the FC algorithm requires that we do something deterministic at any given stage. To remedy this problem, we introduce the notion of selector function F for an H-ASP program Π which essentially makes a deterministic choice for each advancing rule. Then we can define a FC algorithm relative to a selector function F.

Let Π be a H-ASP program whose underlying space of generalized positions is S and whose underlying set of atoms is At. Let $I \in S$ be an initial condition.

A *selector function* for Π is a map F which given any advancing rule r of the form (3) and any k-tuple of generalized positions $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$ in $O = CPS(r)$, specifies a set $F(r, (\mathbf{p}_1, \ldots, \mathbf{p}_k))$ contained in $A(\mathbf{p}_1, \ldots, \mathbf{p}_k)$ where $A = adv(r)$. The idea is that the selector function tells us exactly which pairs $(a, \mathbf{q})$ we can conclude if we apply rule r at the generalized positions $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$, namely, the set of all $(a, \mathbf{q})$ such that $\mathbf{q} \in F(r, (\mathbf{p}_1, \ldots, \mathbf{p}_k))$.

The monotonic part of Π, $mon(\Pi)$ consists of all stationary rules $r \in \Pi$ such that the constraint atom set of r, $CAS(r)$, is empty. The nonmonotonic part of Π, $nmon(P)$, is $\Pi - mon(\Pi)$. Note that $mon(\Pi)$ is always a Horn H-ASP program so that we define the monotonic closure of a set $M \subseteq At \times S$, $cl_{mon,\Pi,I}(M)$, relative to Π and an initial condition I to be

$$cl_{mon,\Pi,I}(M) = T_{mon(\Pi),I}(M) \uparrow \omega.$$

The FC algorithm for H-ASP depends on the program Π and initial condition I, a well ordering $\prec$ on

$$Z(\Pi) = \{(r, \overrightarrow{\mathbf{p}}) : r \in nmon(\Pi) \;\&\; \overrightarrow{\mathbf{p}} \in CPS(r)\},$$

and a selector function F for Π. The algorithm will generate three sequences of sets $\langle D_\xi \rangle_{\xi \in \alpha^+}$, $\langle R_\xi \rangle_{\xi \in \alpha^+}$, and $\langle P_\xi \rangle_{\xi \in \alpha^+}$ where α^+ is the least cardinal greater than the ordinal α determined by the well ordering $\prec$. Here $D_\xi \subseteq At \times S$ will be the analogue of the derived atoms at stage ξ in the FC algorithm for normal logic programs, $R_\xi \subseteq At \times S$ will be the analogue of the rejected atoms at stage ξ in the FC algorithm for normal logic programs, and $P_\xi \subseteq S$ is the set of rejected generalized positions at stage ξ which has no analogue in the FC algorithm for normal logic programs. At any given stage ξ, our construction will ensure that $D_\xi \cap R_\xi = \emptyset$ and that $GP_I(D_\xi) \cap P_\xi = \emptyset$.

The FC algorithm for H-ASP is then defined as follows.

H-ASP Forward Chaining

1. Set $D_0 = cl_{mon,\Pi,I}(\emptyset)$ and $R_0 = \emptyset$, $P_0 = \emptyset$.
2. We say that a pair $(r, \overrightarrow{\mathbf{p}})$ where $r = \frac{B_1;B_2;\ldots;B_k:H,O}{a}$ is a stationary rule and $\overrightarrow{\mathbf{p}} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)$ is **applicable** at stage $\xi + 1$ if (i) $\overrightarrow{\mathbf{p}} \in (GP_I(D_\xi))^k \cap O$, (ii) $H(\overrightarrow{\mathbf{p}}) = 1$, (iii) $\forall i = 1, \ldots, k$, $D_\xi \models (B_i, \mathbf{p}_i)$, (iv) $(a, \mathbf{p}_k) \notin D_\xi$, and (v) if $Q = cl_{mon,\Pi,I}(D_\xi \cup \{(a, \mathbf{p}_k)\})$, then $Q \cap R_\xi = \emptyset$ and $\forall i = 1, \ldots, k$, $Q \models (B_i^-, \mathbf{p}_i)$. Thus $(r, \overrightarrow{\mathbf{p}})$ is applicable at stage $\xi + 1$, if D_ξ allows us to apply rule r at the generalized position tuple $\overrightarrow{\mathbf{p}}$, we have not already derived $(a, \mathbf{p}_k)$ and if adding $(a, \mathbf{p}_k)$ to D_ξ and taking the monotonic closure with respect Π and I does not generate any rejected pairs $(b, \mathbf{p})$ and does stop us from applying rule r at $\overrightarrow{\mathbf{p}}$.

We say that a pair $(r, \overrightarrow{\mathbf{p}})$ where $r = \frac{B_1;B_2;\ldots;B_k:A,O}{a}$ is an advancing rule and $\overrightarrow{\mathbf{p}} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)$ is **applicable** at stage $\xi + 1$ if (i) $\overrightarrow{\mathbf{p}} \in (GP_I(D_\xi))^k \cap O$, (ii) $\forall i = 1, \ldots, k$, $D_\xi \models (B_i, \mathbf{p}_i)$, (iii) $F(r, \overrightarrow{\mathbf{p}}) \cap P_\xi = \emptyset$, (iv) there is at least one generalized position $\mathbf{p} \in F(r, \overrightarrow{\mathbf{p}})$ such that $(a, \mathbf{p}) \notin D_\xi$, and (v) if $Q = cl_{mon,\Pi,I}\left(D_\xi \cup \{(a, \mathbf{p}) : \mathbf{p} \in F(r, \overrightarrow{\mathbf{p}})\}\right)$, then $Q \cap R_\xi = \emptyset$, $GP_I(Q) \cap P_\xi = \emptyset$, and $\forall i = 1, \ldots, k$, $Q \models (B_i^-, \mathbf{p}_i)$. Thus $(r, \overrightarrow{\mathbf{p}})$ is applicable at stage $\xi + 1$, if D_ξ allows us to apply rule r at the generalized position tuple $\overrightarrow{\mathbf{p}}$ and if adding $(a, \mathbf{p})$ to D_ξ for all $\mathbf{p} \in F(r, \overrightarrow{\mathbf{p}})$ and taking the monotonic closure with respect Π and I does not generate any rejected pairs $(b, \mathbf{p})$ or rejected generalized positions and does not stop us from applying rule r at $\overrightarrow{\mathbf{p}}$.

(a) If there is no pair $(r, \overrightarrow{\mathbf{p}})$ which is applicable at stage $\xi + 1$, then set $D_{\xi+1} = D_\xi$, $R_{\xi+1} = R_\xi$ and $P_{\xi+1} = P_\xi$.
(b) Otherwise, let $(r, \overrightarrow{\mathbf{p}}) = \left(r_{\xi+1}, \overrightarrow{\mathbf{p}}_{\xi+1}\right)$ (where $\overrightarrow{\mathbf{p}} = (\mathbf{p}_1, \ldots, \mathbf{p}_k)$) be the $\prec$-least applicable pair at stage $\xi + 1$.
If $r = \frac{B_1;\ldots;B_k:H,O}{a}$ is a stationary rule, set $P_{\xi+1} = P_\xi$,
$D_{\xi+1} = cl_{mon,\Pi,I}(D_\xi \cup \{(a, \mathbf{p}_k)\})$, and
$R_{\xi+1} = R_\xi \cup \{(b, \mathbf{p}_i) \,|\, b \in B_i^-, \, i = 1, \ldots, k\}$.
If $r = \frac{B_1;\ldots;B_k:A,O}{a}$ is an advancing rule, set $P_{\xi+1} = P_\xi \cup (A(\overrightarrow{\mathbf{p}}) \setminus F(r, \overrightarrow{\mathbf{p}})$,

$D_{\xi+1} = cl_{mon,\Pi,I}\left(D_\xi \cup \left\{(a,\mathbf{p})\,|\mathbf{p} \in F\left(r,\overrightarrow{\mathbf{p}}\right)\right\}\right)$, and
$R_{\xi+1} = R_\xi \cup \{(b,\mathbf{p}_i)\,|b \in B_i^-,\ i=1,...,k\}$.

Finally, we set $D^{\prec}_{F,I} = \bigcup_{\xi\in\alpha^+} D_\xi$, $R^{\prec}_{F,I} = \bigcup_{\xi\in\alpha^+} R_\xi$, $P^{\prec}_{F,I} = \bigcup_{\xi\in\alpha^+} P_\xi$. We shall call $D^{\prec}_{F,I}$ the set of derived pairs relative to $\prec$, I, and F, $R^{\prec}_{F,I}$ the set of rejected pairs relative to $\prec$, I, and F, and $P^{\prec}_{F,I}$ the set rejected generalized positions relative to $\prec$, I, and F.

We say that a pair $(r,\overrightarrow{\mathbf{p}})$ where $r = \frac{B_1;B_2;...;B_k:H,O}{a}$ is a stationary rule and $\overrightarrow{\mathbf{p}} = (\mathbf{p}_1,...,\mathbf{p}_k)$ is **inconsistent relative to $\prec$, I, and F** if (i) $H\left(\overrightarrow{\mathbf{p}}\right)=1$, (ii) $\overrightarrow{\mathbf{p}} \in \left(GP_I\left(D^{\prec}_{F,I}\right)\right)^k \cap O$, (iii) $\forall i=1,...,k$, $D^{\prec}_{F,I} \models (B_i,\mathbf{p}_i)$, (iv) $(a,\mathbf{p}_k) \notin D^{\prec}_{F,I}$, and (v) if $Q = clmon_{\Pi,I}\left(D^{\prec}_{F,I} \cup \{(a,\mathbf{p}_k)\}\right)$, then either $Q \cap R^{\prec}_{F,I} \neq \emptyset$ or there is an i with $1 \le i \le k$ such that $Q \not\models \left(B_i^-,\mathbf{p}_i\right)$. Thus $(r,\overrightarrow{\mathbf{p}})$ is inconsistent if $D^{\prec}_{F,I}$ allows us to apply rule r at $\overrightarrow{\mathbf{p}}$ and we have not already derived $(a,\mathbf{p}_k)$, but adding $(a,\mathbf{p}_k)$ to $D^{\prec}_{F,I}$ and taking the monotonic closure with respect to Π and I either generates a rejected pair $(b,\mathbf{p})$ or stops us from applying rule r at $\overrightarrow{\mathbf{p}}$.

We say that a pair $(r,\overrightarrow{\mathbf{p}})$ where $r = \frac{B_1;B_2;...;B_k:A,O}{a}$ is an advancing rule and $\overrightarrow{\mathbf{p}} = (\mathbf{p}_1,...,\mathbf{p}_k)$ is **inconsistent relative to $\prec$, I, and F** if (i) $\overrightarrow{\mathbf{p}} \in \left(GP_I\left(D^{\prec}_{F,I}\right)\right)^k \cap O$, (ii) $\forall i=1,...,k$, $D^{\prec}_{F,I} \models (B_i,\mathbf{p}_i)$, (iii) $F(r,\overrightarrow{\mathbf{p}}) \cap P^{\prec}_{F,I} = \emptyset$, (iv) there is at least one generalized position $\mathbf{p} \in F(r,\overrightarrow{\mathbf{p}})$ such that $(a,\mathbf{p}) \notin D^{\prec}_{F,I}$, and (v) if $Q = clmon_{\Pi,I}\left(D^{\prec}_{F,I} \cup \{(a,\mathbf{p}) : \mathbf{p} \in F(r,\overrightarrow{\mathbf{p}})\}\right)$ then either $Q \cap R^{\prec}_{F,I} \neq \emptyset$, $GP_I(Q) \cap P^{\prec}_{F,I} \neq \emptyset$, or there is an i such that $1 \le i \le k$ and $Q \not\models \left(B_i^-,\mathbf{p}_i\right)$. Thus $(r,\overrightarrow{\mathbf{p}})$ is inconsistent if $D^{\prec}_{F,I}$ allows us to apply rule r at $\overrightarrow{\mathbf{p}}$ but adding $(a,\mathbf{p})$ to $D^{\prec}_{F,I}$ for all $\mathbf{p} \in F(r,\overrightarrow{\mathbf{p}})$ and taking the monotonic closure with respect Π and I either generates a rejected pair $(b,\mathbf{p})$ or a rejected generalized position or stops us from applying rule r at $\overrightarrow{\mathbf{p}}$.

For each stationary rule $r = \frac{B_1;B_2;...;B_k:H,O}{a} \in nmon(\Pi)$, we let $RPos_{\prec,I,F}(r)$ denote the set all $\overrightarrow{\mathbf{p}} \in O$ such that $(r,\overrightarrow{\mathbf{p}})$ is inconsistent relative to $\prec$, I, and F. We then let $new(r)_{\prec,I,F} = \frac{B_1;B_2;...;B_k:H',O'}{a}$ where
$O' = O - RPos_{\prec,I,F}(r)$ and H' equals H restricted to O' if $O' \neq \emptyset$ and let $new(r)_{\prec,I,F}$ be empty if $O' = \emptyset$.

For each advancing rule $r = \frac{B_1;B_2;...;B_k:A,O}{a} \in nmon(\Pi)$, we let $RPos_{\prec,I,F}(r)$ denote the set all $\overrightarrow{\mathbf{p}} \in O$ such that $(r,\overrightarrow{\mathbf{p}})$ is inconsistent relative to $\prec$, I, and F. We then let $new(r)_{\prec,I,F} = \frac{B_1;B_2;...;B_k:A',O'}{a}$ where $O' = O - RPos_{\prec,I,F}(r)$ and where for each $\overrightarrow{\mathbf{p}} \in O'$, $A'(\overrightarrow{\mathbf{p}}) = F(r,\overrightarrow{\mathbf{p}})$ if $O' \neq \emptyset$ and let $new(r)_{\prec,I,F}$ be empty if $O' = \emptyset$. We define

$$A^{\prec}_{F,I} = mon(\Pi) \cup \{new_{\prec,I,F}(r) : r \in nmon(\Pi)\ \&\ new_{\prec,I,F}(r) \neq \emptyset\},$$

$$IP^{\prec}_{F,I} = \left\{\left(r,\overrightarrow{\mathbf{p}}\right) \mid r \in nmon(\Pi)\ \&\ \overrightarrow{\mathbf{p}} \in RPos_{\prec,I,F}\left(r\right)\right\}.$$

Theorem 2. *Let Π be a H-ASP program whose underlying set of generalized positions is S.*

1. *For any selector function F relative to Π, any well ordering $\prec$ of $Z(\Pi)$ and an initial condition $I \in S$, $D^{\prec}_{F,I}$ is a stable model of $A^{\prec}_{F,I}$ with the initial condition I.*
2. *For any stable model M of Π with the initial condition I, there is a well ordering $\prec$ of $Z(\Pi)$ and selector function F relative to Π such that $D^{\prec}_{F,I} = M$ and $A^{\prec}_{F,I} = \Pi$.*
   In fact if $F\left(r, \overrightarrow{\mathbf{p}}\right) = alg\left(r\right)\left(\overrightarrow{\mathbf{p}}\right) \cap GP_I\left(M\right)$ and $\prec$ is any well ordering of $Z\left(\Pi\right)$ such that the set

$$\{\left(r, \overrightarrow{\mathbf{p}}\right)|r = \frac{B_1; ...; B_k : A, O}{a} \text{ and } \forall i = 1, ..., k,\ M \models \left(B_i, \overrightarrow{\mathbf{p}}_i\right) \text{ and } \overrightarrow{\mathbf{p}} \in$$

$$\left(GP_I\left(M\right)\right)^k \cap CPS\left(r\right) \text{ and if } r \text{ is a stationary rule then } A\left(\overrightarrow{\mathbf{p}}\right) = 1\}$$

   forms an initial segment of $\prec$, then $D^{\prec}_{F,I} = M$.

5 The FC for H-ASP Based Monte Carlo Methods

In this section, we shall briefly review the Metropolis-Hastings algorithm and then we will outline Metropolized Forward Chaining (MFC) algorithm to find stable models of H-ASP programs. Our MFC algorithm for H-ASP programs will be a modification of the MFC algorithm for normal logic programs described by the authors in [1].

5.1 The Metropolis-Hastings Algorithm

The presentation in this section is based on the description of Markov Chains and the Metropolis algorithm found in [5], [9], and [10].

A sequence of random variables x_0, x_1, x_2, ... defined on a finite state space X is called a Markov chain if it satisfies the Markov property:

$$\forall t \geq 0 \left(P\left(x_{t+1} = y|\ x_t = x, ..., x_0 = z\right) = P\left(x_{t+1} = y|\ x_t = x\right)\right).$$

In this paper we are considering only Markov chains with the additional property: $P\left(x_{t+1} = y|\ x_t = x\right) = P\left(x_{s+1} = y|\ x_s = x\right)$ for all $t, s \geq 0$. Hence, we can record such probabilities as a transition function $M\left(x, y\right) = P\left(x_1 = y|\ x_0 = x\right)$. We let $M\left(\cdot, \cdot\right)$ denote the matrix which records this transition function and let $M^n\left(\cdot, \cdot\right)$ denote the n-th power of the matrix $M(\cdot, \cdot)$ for any $n \geq 1$. It then follows that for all $n \geq 1$, $P\left(x_n = y|\ x_0 = x\right) = M^n\left(x, y\right)$.

A *probability distribution* on X is a function $\pi : X \rightarrow [0, 1]$ such that $\sum_{x \in X} \pi\left(x\right) = 1$. We say that π is a *stationary distribution* for M if for all $x \in X$, $\sum_{y \in X} \pi\left(y\right) M\left(y, x\right) = \pi\left(x\right)$.

Given a Markov chain $K(\cdot,\cdot)$, called the proposal chain, and a probability distribution $\pi(\cdot)$, let $G(x,y) = \pi(y) K(y,x) / \pi(x) K(x,y)$. The Metropolis-Hastings algorithm defines a new Markov chain $M(\cdot,\cdot)$, where the probability $M(x,y)$ is equal to the probability of drawing $x_{t+1} = y$ given $x_t = x$ using the following procedure:

1. given current state $x_t = x$, draw y based on the Markov chain $K(x,\cdot)$;
2. draw U from the uniform distribution on $[0,1]$;
3. set $x_{t+1} = y$ if $U \leq G(x_t, y)$ and set $x_{t+1} = x_t$ otherwise.

The key result about the Metropolis-Hasting algorithm is the following.

Proposition 1. *Let X be a finite set and $K(\cdot,\cdot)$ be a proposal chain on X such that $\forall x,y \in X$ $K(x,y) > 0$ iff $K(y,x) > 0$. Let $\pi(\cdot)$ be a probability distribution on X. Let $M(\cdot,\cdot)$ be the Metropolis-Hastings chain as defined above. Then $\pi(x) M(x,y) = \pi(y) M(y,x)$ for all x, y. In particular, for all x, $y \in X$ $\lim_{n\to\infty} M^n(x,y) = \pi(y)$.*

5.2 Using FC for H-ASP with the Metropolis-Hastings Algorithm

The following is an outline of how FC for H-ASP can be combined with the Metropolis-Hastings algorithm to solve the following problem: given a H-ASP program Π which has a stable model with respect to the initial condition I, find a stable model M of Π with the initial condition I.

To apply the Metropolis-Hastings algorithm it is necessary to specify the state space X, the Markov chain $K(\cdot,\cdot)$ and the sampling distribution $\pi(\cdot)$.

In MFC algorithm for a finite normal propositional program P described in [1], the state space X is the set $perm(P)$ of well-orderings of $nmon(P)$. To define $K(\cdot,\cdot)$, an integer k is fixed with $2 \leq k \leq |nmon(P)|$. For σ, $\tau \in perm(P)$, $K(\sigma,\tau)$ is the probability that starting with σ, τ is produced by picking k elements $1 \leq i_1 < i_2 < ... < i_k \leq |nmon(P)|$ uniformly at random, then picking a permutation γ of i_1,...,i_k uniformly at random and then creating a new permutation by replacing $\sigma_{i_1}, ..., \sigma_{i_k}$ in σ by $\sigma_{\gamma(i_1)}, ..., \sigma_{\gamma(i_k)}$. Since $K(\sigma,\tau) = K(\tau,\sigma)$ the acceptance ratio for the Metropolis-Hastings algorithm is $G(\sigma,\tau) = \frac{\pi(\sigma)}{\pi(\tau)}$. Finally, to define $\pi(\cdot)$ we let $r(\sigma)$ equal $|I^\sigma|$ - the number of inconsistent rules produced by the FC algorithm when it is used on P with the well-ordering σ. We then choose θ and m, with $0 < \theta < 1$ and $m \geq 1$ independent of $|perm(P)|$ and set

$$\pi(\sigma) \propto \theta^{r(\sigma)^m}$$

Since the denominator of $\pi(\sigma)$ is not needed for the computation of the acceptance ratio $G(\sigma,\tau)$, we only need to know $\theta^{r(\sigma)^m}$ for the computational purposes.

A similar approach can be used to combine the FC algorithm for a H-ASP program P and the Metropolis-Hasting algorithm, to produce an MFC algorithm for H-ASP. Given a H-ASP program Π, let $perm(Z(\Pi))$ be the set of all the well-orderings on $Z(\Pi)$. Unlike MFC algorithm for normal logic programs where only a well ordering has to be chosen in order to use the FC algorithm, for H-ASP programs, we must also specify a selector function F as well as a well-ordering

$\prec$ to apply the FC algorithm. Thus our state space X will consist of pairs (F,σ) where F is a selector function for Π and σ is a well-ordering on $Z(\Pi)$.

Our Markov chain $K(\cdot,\cdot)$ will be defined analogously to that for MFC except that now changes to the selector function will need to be considered. We will choose integer parameters k with $2 \leq k \leq |Z(\Pi)|$ and $s,t,u \geq 1$. For two selector functions F and H and two well orderings on $Z(\Pi)$, σ and τ, $K((F,\sigma),(H,\tau))$ is the probability that starting with (F,σ), we produce (H,τ) by the following procedure.

1. Choose s advancing rules $r_1, r_2, ..., r_s$ from $nmon(\Pi)$ uniformly at random. For each rule r_i choose t elements $\overrightarrow{\mathbf{p}_1}, ..., \overrightarrow{\mathbf{p}_t}$ from $CPS(r_i)$ uniformly at random. For each $\overrightarrow{\mathbf{p}_j}$, choose an integer w uniformly at random where $0 \leq w \leq u$ and w elements $\mathbf{p}_1, ..., \mathbf{p}_w$ from $alg(r_i)\left(\overrightarrow{\mathbf{p}_j}\right)$ uniformly at random. For each $\mathbf{p}_k$ chosen for a $\overrightarrow{\mathbf{p}_j}$, if $\mathbf{p}_k \in F\left(r_i, \overrightarrow{\mathbf{p}_j}\right)$, then $\mathbf{p}_k \notin H\left(r_i, \overrightarrow{\mathbf{p}_j}\right)$ and if $\mathbf{p}_k \notin F\left(r_i, \overrightarrow{\mathbf{p}_j}\right)$, then $\mathbf{p}_k \in H\left(r_i, \overrightarrow{\mathbf{p}_j}\right)$. H is identical to F in all other cases.
2. Pick k elements $1 \leq i_1 < ... < i_k \leq |Z(\Pi)|$ of σ uniformly at random. Pick a permutation γ of $i_1, ..., i_k$ uniformly at random and create τ by replacing $\sigma_{i_1}, ..., \sigma_{i_k}$ with $\sigma_{\gamma(i_1)}, ..., \sigma_{\gamma(i_k)}$.

To specify $\pi(\cdot)$, we define $r((F,\sigma)) = \left|IP^{\sigma}_{F,I}\right|$. Choose a parameter θ where $0 < \theta < 1$ and a parameter $m \leq 1$. As in MFC we set

$$\pi((F,\sigma)) \propto \theta^{r((F,\sigma))^m}$$

It is the case that $K((F,\sigma),(H,\tau)) = K((H,\tau),(F,\sigma))$ and so the acceptance ratio for the Metropolis-Hastings algorithm is

$$G((F,\sigma),(H,\tau)) = \frac{\pi((F,\sigma))}{\pi((H,\tau))} = \theta^{r((F,\sigma))^m - r((H,\tau))^m}$$

The Metropolis-Hastings algorithm for $K(\cdot,\cdot)$ and π can thus be used to find a pair (F,σ) that minimizes $r((F,\sigma)) = \left|IP^{\sigma}_{F,I}\right|$. In the case that Π has a stable model Metropolis-Hastings algorithm will eventually find it (when $r((F,\sigma)) = 0$). In the case that Π does not have a stable model, the Metropolis-Hastings algorithm will eventually find a subprogram $A^{\sigma}_{F,I}$ with the minimal $r((F,\sigma))$ that has a stable model $D^{\sigma}_{F,I}$

6 Conclusions and Future Research

In this paper, we have defined an analogue of the Forward Chaining algorithm for normal logic programs due to Marek, Nerode and Remmel [12] and discussed an analogue of the Metropolized Forward Chaining Algorithm due to the authors [1] for H-ASP programs.

There are several questions for future research. For example, Marek, Nerode and Remmel [11] define an analogue of Rieter's normal default theories for normal logic programs which are logic programs P where the FC algorithm produces a stable model for every well ordering of $nmon(P)$. There should be a similar analogue of normal default theories for H-ASP programs. Also, there are several issues that need to be resolved for our MFC algorithm for H-ASP programs to be practical. For example, we must decide how one specifies a selector function F and how one specifies a well-ordering σ on Q? The difficulty here lies in the fact that Q may be too large to enumerate. One approach to resolving the above two issues is to specify F and σ implicitly by providing a procedure that is able to access specific elements of a well-ordering σ on $Z(\Pi)$ without explicitly enumerating σ. Such issues will be the subject of future research.

References

1. Brik, A., Remmel, J.B.: Computing stable models of logic programs using metropolis type algorithms. In: Workshop Proceedings. ICLP 2011 Workshop on Answer Set Programming and Other Computing Paradigms (2011)
2. Brik, A., Remmel, J.B.: Hybrid ASP. In: Gallagher, J.P., Gelfond, M. (eds.) ICLP (Technical Communications). LIPIcs, vol. 11, pp. 40–50. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
3. Brik, A., Remmel, J.B.: Computing a Finite Horizon Optimal Strategy Using Hybrid ASP. In: Workshop Proceedings. NMR (2012)
4. Diaconis, P., Saloff-Coste, L.: What Do We Know About the Metropolis Algorithm? Journal of Computer and System Sciences 57(1), 20–36 (1998)
5. Diaconis, P.: The Markov chain Monte Carlo revolution. Bull. Amer. Math. Soc. 46, 179–205 (2009)
6. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In: IJCAI, pp. 386–373 (2007)
7. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, pp. 1070–1080 (1988)
8. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1), 97–109 (1970)
9. Karlin, S., Taylor, H.M.: A First Course in Stochastic Processes, 2nd edn. Academic Press (April 1975)
10. Liu, J.S.: Monte Carlo Strategies in Scientific Computing, corrected edn. Springer (January 2008)
11. Marek, V.W., Nerode, A., Remmel, J.B.: Context for belief revision: Fc-normal nomonotonic rule systems. Ann. Pure Appl. Logic 67(1-3), 269–324 (1994)
12. Marek, V.W., Nerode, A., Remmel, J.B.: Logic programs, well-orderings, and forward chaining. Ann. Pure Appl. Logic 96(1-3), 231–276 (1999)
13. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics 21(6), 1087–1092 (1953)
14. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. 138(1-2), 181–234 (2002)
15. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. J. ACM 23(4), 733–742 (1976)

Effectivity Questions for Kleene's Recursion Theorem

John Case[1], Sanjay Jain[2,⋆], and Frank Stephan[3,⋆⋆]

[1] Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716-2586, USA
case@cis.udel.edu
[2] Department of Computer Science
National University of Singapore, Singapore 117417
sanjay@comp.nus.edu.sg
[3] Department of Mathematics and Department of Computer Science
National University of Singapore, Singapore 119076
fstephan@comp.nus.edu.sg

Abstract. The present paper explores the interaction between two recursion-theoretic notions: program self-reference and learning partial recursive functions in the limit. Kleene's Recursion Theorem formalises the notion of program self-reference: It says that given a partial-recursive function ψ_p there is an index e such that the e-th function ψ_e is equal to the e-th slice of ψ_p. The paper studies constructive forms of Kleene's recursion theorem which are inspired by learning criteria from inductive inference and also relates these constructive forms to notions of learnability. For example, it is shown that a numbering can fail to satisfy Kleene's Recursion Theorem, yet that numbering can still be used as a hypothesis space when learning explanatorily an arbitrary learnable class. The paper provides a detailed picture of numberings separating various versions of Kleene's Recursion Theorem and learnability.

Keywords: inductive inference, Kleene's Recursion Theorem, Kolmogorov complexity, optimal numberings.

1 Introduction

Program self-reference is the ability of a program to make use of its own source code in its computations. This notion is formalized by Kleene's Recursion Theorem.[1] Intuitively, this theorem asserts that, for each preassigned algorithmic

⋆ Supported in part by NUS grant numbers R252-000-420-112 and C252-000-087-001.
⋆⋆ Supported in part by NUS grant number R252-000-420-112.

[1] Other "recursion theorems" do not so well capture the notion of program self-reference. For example, consider the (quasi-fixed-point) recursion theorem as formalised by Rogers [Rog67, Theorem 11-I]. In contrast to Kleene's Recursion Theorem, Rogers' recursion theorem is not strong enough to guarantee that a numbering of partial-recursive functions satisfying it has a self-reproducing program which outputs its own index [CM09].

task, there exists a program e that computes exactly the e-th slice of this algorithmic task. The theorem is stated below, following some necessary definitions.

Let $\mathbb{N}$ be the set of natural numbers, $\{0, 1, 2, \ldots\}$. Let $\mathcal{P}$ be the collection of all partial recursive functions from $\mathbb{N}$ to $\mathbb{N}$. Let $\langle \cdot, \cdot \rangle$ be Cantor's pairing function [Rog67, page 64]: $\langle x, y \rangle = (x + y)(x + y + 1)/2 + y$, which is a recursive, order preserving bijection $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ [Rog67, page 64]; here order preserving means that $x \leq x' \wedge y \leq y' \Rightarrow \langle x, y \rangle \leq \langle x', y' \rangle$. For each $\psi \in \mathcal{P}$ and $p \in \mathbb{N}$, let ψ_p be shorthand for $\psi(\langle p, \cdot \rangle)$. An *effective numbering* of $\mathcal{P}$ is a $\psi \in \mathcal{P}$ such that

$$(\forall \alpha \in \mathcal{P})(\exists p \in \mathbb{N})[\psi_p = \alpha]. \tag{1}$$

For this paper, we shall be concerned only with numberings that are effective, and that number the elements of $\mathcal{P}$. Hence, we shall generally omit the phrases "effective" and "of $\mathcal{P}$".

The following is the formal statement of Kleene's Recursion Theorem.

Definition 1 (Kleene [Kle38]). For each numbering ψ, *Kleene's Recursion Theorem holds in* $\psi \Leftrightarrow$

$$(\forall p \in \mathbb{N})(\exists e \in \mathbb{N})[\psi_e = \psi_p(\langle e, \cdot \rangle)]. \tag{2}$$

Equation (2) can be interpreted as follows: Suppose the ψ-program p represents an arbitrary, algorithmic task to perform; then the equation says that there is a ψ-program e such that ψ_e is equal to the e-th slice of this algorithmic task. This is often used in diagonalizations by defining ψ_e in a way implicitly employing parameter e (in effect, a self-copy of e) in some algorithmic task ψ_p.

The following constructive form of Kleene's Recursion Theorem has been well-studied. For reasons that will become apparent shortly, we call this form of the theorem FinKrt.

Definition 2 (Kleene, see [Ric80, Ric81, Roy87]). A numbering ψ is called a FinKrt-*numbering* $\Leftrightarrow$

$$(\exists \text{ recursive } r : \mathbb{N} \to \mathbb{N})(\forall p)[\psi_{r(p)} = \psi_p(\langle r(p), \cdot \rangle)]. \tag{3}$$

In (3), ψ-program $r(p)$ plays the role played by e in (2). In this sense, the function r *finds* a ψ-program $r(p)$ such that $\psi_{r(p)}$ is equal to the $r(p)$-th slice of ψ_p.

In this paper, additional constructive forms of the theorem are considered. Each is inspired by a Gold-style criterion for learning partial recursive functions in the limit. The Gold-style criteria differ in when a learning device is considered to have learned a target partial recursive function. However, the following is common to all. The learning device is fed the elements of the graph of a partial recursive function α.[2] After being fed each such element, the device outputs either '?' or a hypothesis, i.e., a program, possibly corresponding to the partial

[2] The device may also be fed one or more instances of the pause symbol '#'. This allows that graph of the target partial recursive function to be empty, i.e., in such a case, the device is fed nothing but #.

recursive function α. In the present paper, the device is expected to be algorithmic, that is, definable by a computer program.

For the finite (Fin) learning criterion, the device is considered to have learned the target partial recursive function α iff the device outputs finitely many '?' immediately followed by a hypothesis corresponding to α. The constructive form of Kleene's Recursion Theorem, given in Definition 2, may be viewed in a similar way. A device is fed a program p for a preassigned task. After finitely many steps, that device outputs a program e that uses its own source code in the manner prescribed by p.

A numbering ψ is said to be optimal for Fin-learning iff every Fin-learnable class of partial recursive functions can be Fin-learned using ψ as the hypothesis space. A numbering ψ is said to be effectively optimal for Fin-learning iff one can effectively translate every Fin-learning device into a Fin-learning device that uses ψ as its hypothesis space [JS10, Jai11].

Not every numbering is optimal for Fin-learning [JS10], let alone effectively optimal. Similarly, not every numbering is a FinKrt-numbering [Ric80, Ric81]. Hence, one might ask: is every FinKrt-numbering optimal for Fin-learning? Conversely, if a numbering is optimal for Fin-learning, then is it necessarily a FinKrt numbering?

Additional Gold-style learning criteria are introduced in Section 2 below and will be familiar to most readers familiar with inductive inference. These criteria, which are successively less stringent in when a learning device is considered to have learned a target partial recursive function, are: single mind-change explanatory (Ex_1), explanatory (Ex), vacillatory (Vac) and behaviorally correct (Bc). Section 2 also introduces additional constructive forms of Kleene's Recursion Theorem (ExKrt, VacKrt and BcKrt). Each is inspired by one of the just mentioned learning criteria. Our results include the following.

- There is a numbering which does not satisfy Kleene's Recursion Theorem, but which is optimal for Fin-learning and effectively optimal for Ex, Vac and Bc-learning (Theorem 7).
- There is a FinKrt-numbering which is not optimal for any of the learning criteria Fin, Ex, Vac, Bc (Theorem 8).
- There is an ExKrt-numbering which is not a FinKrt-numbering and which is effectively optimal for Ex-learning, but not optimal for Fin or Bc-learning (Theorem 10).
- There is a VacKrt-numbering which is not an ExKrt-numbering and which is effectively optimal for Vac-learning but not optimal for Fin, Ex or Bc-learning (Theorem 11).
- There is a BcKrt-numbering which is not a VacKrt-numbering and which is effectively optimal for Bc-learning, but not optimal for Fin, Ex or Vac-learning (Theorem 13).
- There is a numbering satisfying Kleene's Recursion Theorem which is not a BcKrt-numbering and which is not optimal for any of the learning criteria Fin, Ex, Vac, Bc (Theorem 14).

– There is a numbering satisfying Kleene's Recursion Theorem which is not a BcKrt-numbering, but which is effectively optimal for Ex, Vac and Bc-learning (Theorem 15).

The remainder of this paper is organized as follows. Section 2 covers preliminaries. Section 3 presents our results concerning numberings that do not satisfy Kleene's Recursion Theorem. Section 4 presents our results concerning numberings that satisfy Kleene's Recursion Theorem in an effective way. Section 5 presents our results concerning numberings that satisfy Kleene's Recursion Theorem, but not in an effective way.

2 Preliminaries

Recursion-theoretic concepts not covered below are treated as by Rogers [Rog67].

Lowercase math-italic letters (e.g., a, b, c) range over elements of $\mathbb{N}$, unless stated otherwise. Uppercase math-italic italicized letters (e.g., A, B, C) range over subsets of $\mathbb{N}$, unless stated otherwise. Lowercase Greek letters (e.g., α, β, γ) range over partial functions from $\mathbb{N}$ to $\mathbb{N}$, unless stated otherwise.

For each non-empty X, $\min X$ denotes the minimum element of X. We let $\min \emptyset \overset{\text{def}}{=} \infty$. For each non-empty, finite X, $\max X$ denotes the maximum element of X. We let $\max \emptyset \overset{\text{def}}{=} -1$. $D_0, D_1, D_2, \ldots$ denotes a recursive canonical enumeration of all finite subsets of $\mathbb{N}$.

The pairing function $\langle \cdot, \cdot \rangle$ was introduced in Section 1. Note that $\langle 0, 0 \rangle = 0$ and, for each x and y, $\max\{x, y\} \leq \langle x, y \rangle$.

For each one-argument partial function α and $x \in \mathbb{N}$, $\alpha(x)\downarrow$ denotes that $\alpha(x)$ converges; $\alpha(x)\uparrow$ denotes that $\alpha(x)$ diverges. We use $\uparrow$ to denote the value of a divergent computation. So, for example, $\lambda x \centerdot \uparrow$ denotes the everywhere divergent partial function.

$\mathbb{N}_{\#} \overset{\text{def}}{=} \mathbb{N} \cup \{\#\}$ and $\mathbb{N}_{?} \overset{\text{def}}{=} \mathbb{N} \cup \{?\}$. For each partial function f (of arbitrary type), $\mathrm{rng}(f)$ denotes the range of f. A *text* is a total (not necessarily recursive) function of type $\mathbb{N} \to \mathbb{N}_{\#}$. For each text T and $i \in \mathbb{N}$, $T[i]$ denotes the initial segment of T of length i. Init denotes the set of all finite initial segments of all texts. For each text T and partial function α, T is a text *for* α iff $\mathrm{rng}(T) - \{\#\}$ is the graph of α as coded by $\langle \cdot, \cdot \rangle$, i.e., $\mathrm{rng}(T) - \{\#\} = \{\langle x, y \rangle \mid \alpha(x) = y \wedge x, y \in \mathbb{N}\}$. For a total function f, we often identify f with its canonical text, that is, the text T with $T(i) = \langle i, f(i) \rangle$. Thus, $f[n]$ represents the initial segment of length n of this canonical text.

A numbering φ is *acceptable* iff for each numbering ψ, there exists a recursive function $t : \mathbb{N} \to \mathbb{N}$ such that, for each p, $\varphi_{t(p)} = \psi_p$ [Rog67, Ric80, Ric81, Roy87]. Let φ be any fixed acceptable numbering satisfying $\varphi_0 = \lambda x \centerdot \uparrow$. For each p, $W_p \overset{\text{def}}{=} \{x \mid \varphi_p(x)\downarrow\}$. K denotes the diagonal halting problem with respect to φ, i.e., $\{x \mid x \in W_x\}$. Let $\mathrm{pad} : \mathbb{N}^2 \to \mathbb{N}$ be a recursive function such that, for each e and y, $\varphi_{\mathrm{pad}(e,y)} = \varphi_e$ and $\mathrm{pad}(e, y) < \mathrm{pad}(e, y + 1)$, where we assume $\mathrm{pad}(0, 0) = 0$.

The following are the Gold-style learning criteria considered in this paper.

Definition 3. Let α be any partial recursive function. For each recursive function $M : \text{Init} \to \mathbb{N}_?$ and each numbering ψ, (a)–(e) below.

(a) [Gol67] M Fin_ψ*-learns* α iff for each text T for α, there exist i_0 and e such that

$$(\forall i < i_0)\big[M(T[i]) = ?\big] \ \wedge\ (\forall i \geq i_0)\big[M(T[i]) = e\big] \ \wedge\ \psi_e = \alpha. \tag{4}$$

(b) [CS83] M $\mathsf{Ex}_{\psi,1}$*-learns* α iff for each text T for α, there exist i_0, i_1, e_0 and e_1 such that

$$\begin{aligned}(\forall i < i_0)\big[M(T[i]) = ?\big] \ \wedge\ (\forall i \in \{i_0, \ldots, i_1 - 1\})\big[M(T[i]) = e_0\big]\\ \wedge\ (\forall i \geq i_1)\big[M(T[i]) = e_1\big] \ \wedge\ \psi_{e_1} = \alpha.\end{aligned} \tag{5}$$

(c) [Gol67] M Ex_ψ*-learns* α iff for each text T for α, there exist i_0 and e such that

$$(\forall i \geq i_0)\big[M(T[i]) = e\big] \ \wedge\ \psi_e = \alpha. \tag{6}$$

(d) [Cas99] M Vac_ψ*-learns* α iff for each text T for α, there exist i_0 and a finite set E such that

$$(\forall i \geq i_0)\big[M(T[i]) \in E\big] \ \wedge\ (\forall e \in E)[\psi_e = \alpha]. \tag{7}$$

(e) [Bar74, OW82] M Bc_ψ*-learns* α iff for each text T for α, there exists an i_0 such that

$$M(T[i_0]) \neq ? \text{ and } (\forall i \geq i_0)(\forall e)\big[M(T[i]) = e \ \Rightarrow\ \psi_e = \alpha\big]. \tag{8}$$

Let $I \in \{\mathsf{Fin}, \mathsf{Ex}_1, \mathsf{Ex}, \mathsf{Vac}, \mathsf{Bc}\}$ and let $\mathcal{S}$ be a class of partial recursive functions. *M I_ψ-learns $\mathcal{S}$* iff M I_ψ-learns each partial recursive function in $\mathcal{S}$. We say that $\mathcal{S}$ is I_ψ-learnable if some M I_ψ-learns $\mathcal{S}$. In above definitions, we omit the subscript ψ when ψ is the fixed acceptable numbering φ.

Definition 4 (Jain & Stephan [JS10]). Let φ be an acceptable numbering. For each $I \in \{\mathsf{Fin}, \mathsf{Ex}_1, \mathsf{Ex}, \mathsf{Vac}, \mathsf{Bc}\}$ and each numbering ψ, (a) and (b) below.

(a) ψ is *optimal for I-learning* iff each I_φ-learnable class is I_ψ-learnable.
(b) ψ is *effectively optimal for I-learning* iff there exists a recursive function $t : \mathbb{N} \to \mathbb{N}$ such that, for each p and each class of partial recursive functions $\mathcal{S}$, if φ_p I_φ-learns $\mathcal{S}$, then $\varphi_{t(p)}$ I_ψ-learns $\mathcal{S}$.

Note that while for learning criteria and the below constructive versions of KRT, the implications $\mathsf{Fin} \to \mathsf{Ex}_1 \to \mathsf{Ex} \to \mathsf{Vac} \to \mathsf{Bc}$ hold, the corresponding implications do not always hold with respect to numberings being optimal or effectively optimal for I-learning. For example, there are numberings which are optimal for Vac-learning but not optimal for Bc-learning [JS10]. However, if a numbering is effectively optimal for Fin-learning, then it is effectively optimal for Ex, Vac and Bc-learning. Furthermore, if a numbering is effectively optimal for Ex-learning then it is effectively optimal for Vac-learning [JS10].

The following are the constructive forms of Kleene's Recursion Theorem considered in this paper. The reader will note the similarity to Definition 3.

Definition 5. [Moe09] For each numbering ψ, (a)–(d) below.

(a) ψ is a FinKrt-*numbering* iff there exists a recursive function $r : \mathbb{N} \to \mathbb{N}$ such that, for each p,
$$\psi_{r(p)} = \psi_p\big(\langle r(p), \cdot\rangle\big). \tag{9}$$

(b) ψ is an ExKrt-*numbering* iff there exists a recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ such that, for each p, there exist i_0 and e such that
$$(\forall i \geq i_0)[f(p,i) = e] \ \wedge \ \psi_e = \psi_p(\langle e, \cdot\rangle). \tag{10}$$

(c) ψ is a VacKrt-*numbering* iff there exists a recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ such that, for each p, there exist i_0 and a finite set E such that
$$(\forall i \geq i_0)[f(p,i) \in E] \ \wedge \ (\forall e \in E)[\psi_e = \psi_p(\langle e, \cdot\rangle)]. \tag{11}$$

(d) ψ is a BcKrt-*numbering* iff there exists a recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ such that, for each p, there exists an i_0 such that
$$(\forall i \geq i_0)(\forall e)[f(p,i) = e \ \Rightarrow \ \psi_e = \psi_p(\langle e, \cdot\rangle)]. \tag{12}$$

Definition 6. For each numbering ψ, (a) and (b) below.

(a) ψ is Ex_1-*acceptable* iff there exists a recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ such that, for each p, there exist i_0, e_0 and e_1 such that
$$(\forall i < i_0)[f(p,i) = e_0] \ \wedge \ (\forall i \geq i_0)[f(p,i) = e_1] \ \wedge \ \psi_{e_1} = \varphi_p. \tag{13}$$

(b) **(Case, Jain and Suraj [CJS02])** ψ is Ex-*acceptable* iff there exists a recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ such that, for each p, there exist i_0 and e such that
$$(\forall i \geq i_0)[f(p,i) = e] \ \wedge \ \psi_e = \varphi_p. \tag{14}$$

We use the convention that, for each y, $\log(y) \stackrel{\text{def}}{=} \min\{x \mid 2^x \geq y\}$. So, for example, $\log(0) = 0$ and $\log(3) = 2$. For each e, $C(e)$ denotes the plain Kolmogorov complexity of e [LV08, Nie09]. Note that there exists an approximation $\lambda s, e \centerdot C_s(e)$ such that, for each e, $C(e) = \lim_s C_s(e)$. Further note that, for each 1–1 recursive sequence $e_0, e_1, e_2, \ldots$, there exists a constant c such that, for each i, $C(e_{i+1}) < C(e_i) + c$.

3 When Kleene's Recursion Theorem Is Absent

This section presents our results concerning numberings that do not satisfy Kleene's Recursion Theorem. Note that every acceptable numbering is a FinKrt-numbering [Kle38]. However, as the next result shows, this does not generalize to other criteria of acceptability. In particular, there is an Ex_1-acceptable numbering that does not satisfy Kleene's Recursion Theorem.

Theorem 7. *There exists a numbering ψ satisfying (a)–(d) below.*

(a) ψ does not satisfy Kleene's Recursion Theorem.
(b) ψ is an Ex_1-acceptable numbering.
(c) ψ effectively optimal for Ex, Vac and Bc-learning.
(d) ψ is optimal for Fin-learning.

Proof. Let ψ be such that, for each e and x,

$$\psi_e(x) = \begin{cases} \varphi_e(x), & \text{if } \mathrm{rng}(\varphi_e) \not\subseteq \{e, e+1, e+2, \ldots\}; \\ \uparrow, & \text{if } \mathrm{rng}(\varphi_e) \subseteq \{e, e+1, e+2, \ldots\}. \end{cases} \tag{15}$$

The numbering ψ does not satisfy Kleene's Recursion Theorem: There is no e such that $\psi_e = \lambda x \centerdot e$; hence, there is no e such that $\psi_e = \alpha(\langle e, \cdot\rangle)$ when $\alpha = \lambda\langle e, x\rangle \centerdot e$.

To show that ψ is Ex_1-acceptable: there exists a translator which behaves as follows. On input e, the translator first conjectures 0 for $\psi_0 = \varphi_0 = \lambda x \centerdot \uparrow$. Then, in the case that $\varphi_e(x) = y$, for some x and y, the translator outputs $\mathrm{pad}(e, y)$. Note that $y < \mathrm{pad}(e, y)$. Hence, it follows from the definition of ψ that $\psi_{\mathrm{pad}(e,y)} = \varphi_e$.

To show that ψ is effectively optimal for Ex, Vac, and Bc-learning: given a Bc-learner M, the new learner N first conjectures 0 for $\lambda x \centerdot \uparrow$. If, however, a datum (x, y) is ever seen, then, from that point onward, N simulates M and translates each conjecture e of M into $\mathrm{pad}(e, y)$.

To show that ψ is optimal for Fin-learning: suppose that M is a Fin-learner for a class not containing $\lambda x \centerdot \uparrow$. Then, the new learner N waits for the first pair (x, y); from that point onward, N simulates M and translates each conjecture e of M into $\mathrm{pad}(e, y)$. On the other hand, suppose that M is a Fin-learner for a class containing $\lambda x \centerdot \uparrow$. Then, this class contains no other partial functions. Hence, N can just ignore all input and output 0 as $\psi_0 = \lambda x \centerdot \uparrow$. Hence, ψ is optimal for Fin-learning.[3] □ (**Theorem 7**)

4 When Kleene's Recursion Theorem Is Effective

This section presents our results concerning numberings that satisfy Kleene's Recursion Theorem in an effective way. These results include the following. First, a numbering can be a FinKrt-numbering, yet not be optimal for learning (Theorem 8). Second, there exists an ExKrt-numbering that is not a FinKrt-numbering (Theorem 10). Third, there exists a VacKrt-numbering that is not an ExKrt-numbering (Theorem 11). Finally, there exists a BcKrt-numbering that is not a VacKrt-numbering (Theorem 13).

Theorem 8. *There exists a numbering ψ satisfying (a) and (b) below.*

(a) ψ is a FinKrt-numbering.
(b) ψ is not optimal for any of the learning criteria Fin, Ex, Vac, Bc.

[3] Note that, as ψ is not acceptable, ψ cannot be effectively optimal for Fin-learning [JS10]. Hence, the non-uniform case distinction in this proof is unavoidable.

Proof. The construction of ψ is in two parts. First, we construct a numbering ϑ such that the set $\{e : \vartheta_e \text{ has finite domain}\}$ is dense simple relative to K, i.e., the function that maps n to the n-th index of a function with infinite domain dominates every K-recursive function.[4] From ϑ, we construct ψ.

For each e, s, let $F_{e,s}(\cdot)$ be a uniformly recursive sequence of recursive functions such that, for $F_e(x) = \lim_{s\to\infty} F_{e,s}(x)$,

(a) for all e, x, $F_{e,s}(x) \leq F_{e,s+1}(x)$;
(b) if $\varphi_e^K(x)\downarrow$, then $F_e(x)\downarrow \geq \varphi_e^K(x)$.
(c) if $\varphi_e^K(x)\uparrow$, then $F_e(x)\uparrow$.

Note that such $F_{e,s}$ exist and are uniformly recursive from e, s. Furthermore, each of $F_0, F_1, F_2, \ldots$ is a partial K-recursive function. It should also be noted that, for each e, φ_e^K is majorized by F_e. Hence, the function $n \mapsto \max\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\}$ dominates every K-recursive function.

Let ϑ be such that, for each n, m and x,

$$\vartheta_{\langle n,m\rangle}(x) = \begin{cases} \varphi_n(x), & \text{if } (\exists s > x)\big[\quad (\exists e \leq n)[F_{e,s}(n) = m] \\ & \qquad\qquad \wedge\ (\forall d \leq n)[\quad F_{d,s}(n) \leq m \\ & \qquad\qquad\qquad\qquad \vee\ F_{d,s}(n) > F_{d,x}(n)]\big]; \\ \uparrow, & \text{otherwise.} \end{cases} \tag{16}$$

We show that, for each n and m, $\vartheta_{\langle n,m\rangle}$ has infinite domain iff φ_n has infinite domain and $m = \max\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\}$. To see this, let n and m be given and consider the following four cases.

Case 1: φ_n has finite domain. Clearly, for each n and m, φ_n extends $\vartheta_{\langle n,m\rangle}$. Hence, if φ_n has finite domain, then so does $\vartheta_{\langle n,m\rangle}$.

Case 2: $\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\} = \emptyset$. Let w be so large that, for each $e \leq n$,

$$F_{e,w}(n) > m. \tag{17}$$

Then, for each $x \geq w$, there is no $s > x$ such that $(\exists e \leq n)[F_{e,s}(n) = m]$. Hence, for almost all x, $\vartheta_{\langle n,m\rangle}(x)\uparrow$.

Case 3: $\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\} \neq \emptyset$ and $m \neq \max\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\} < \infty$. Let w be so large that, for each $e \leq n$,

$$F_e(n)\downarrow > m \ \Rightarrow \ F_{e,w}(n) = F_e(n), \tag{18}$$

and

$$F_e(n)\uparrow \ \Rightarrow \ F_{e,w}(n) > m. \tag{19}$$

Then, for each $x \geq w$, there is no $s > x$ such that $(\exists e \leq n)[F_{e,s}(n) = m]$ and $(\forall d \leq n)[F_{d,s}(n) \leq m \ \vee \ F_{d,s}(n) > F_{d,x}(n)]$. Hence, for almost all x, $\vartheta_{\langle n,m\rangle}(x)\uparrow$.

Case 4: φ_n has infinite domain and $m = \max\{F_e(n) : e \leq n \wedge F_e(n)\downarrow\}$. Then, for each x, one can find an $s > x$ such that $(\exists e \leq n)[F_{e,s}(n) = m]$ and, for each $d \leq n$,

$$[F_d(n)\downarrow \ \Rightarrow \ F_{d,s}(n) = F_d(n)] \ \wedge \ [F_d(n)\uparrow \ \Rightarrow \ F_{d,s}(n) > F_{d,x}(n)]. \tag{20}$$

[4] The existence of such numberings is a well-known folklore result.

Hence, for each x, if $\varphi_n(x)\downarrow$, then $\vartheta_{\langle n,m\rangle}(x)\downarrow$.

From Case 4 it also follows that, ϑ is a numbering for $\mathcal{P}$. As the function $\lambda n\,.\,\max\{F_e(n) : e \leq n \ \wedge \ F_e(n)\downarrow\}$ dominates every K-recursive function, the set of all pairs $\langle n, m\rangle$ where $\vartheta_{\langle n,m\rangle}$ has a finite domain is dense simple relative to K.

Now, let ψ be such that, for each p, i and x,

$$\psi_{\langle p,0\rangle}(x) = \vartheta_p(x); \tag{21}$$

$$\psi_{\langle p,i+1\rangle}(x) = \psi_{\langle p,i\rangle}(\langle\langle p, i+1\rangle, x\rangle) \tag{22}$$

Note that ψ is defined such that $\psi_{\langle p,i+1\rangle}$ coincides with the $\langle p, i+1\rangle$-th row of $\psi_{\langle p,i\rangle}$. Hence, ψ is a FinKrt-numbering.

To show that ψ is not optimal for any of the criteria Fin, Ex, Vac, Bc: consider the class $\mathcal{S} = \{f_0, f_1, f_2, \ldots\}$ where, for each n and x, $f_n(x) = n + x$. $\mathcal{S}$ is Fin-learnable and, hence, is also Ex, Vac and Bc-learnable.

Note that, if $\psi_{\langle p,i\rangle} = f_n$, then, by induction over j for all $j > i$, $\mathrm{rng}(\psi_{p,i}) - \mathrm{rng}(\psi_{p,j})$ is infinite and hence $\psi_{p,j} \neq f_m$ for all m. Thus, the following claim holds.

Claim 9. *For each p, there exists at most one i such that $\psi_{\langle p,i\rangle} \in \mathcal{C}$.*

We first show $\mathcal{S}$ is not Vac_ψ-learnable. Now a Vac-learner for $\mathcal{C}$ would, for any n, output only finitely many indices while learning the function f_n. Hence, there exists an index e such that F_e is a K-recursive (i.e., total) function and, for each n, $F_e(n)$ is larger than all the indices output by the learner while learning f_n. It follows that $F_e(n)$ is greater than at least one pair $\langle p, i\rangle$ such that $\psi_{\langle p,i\rangle} = f_n$. Using Claim 9, it follows that ϑ has $n+1$ distinct indices of functions with infinite domain below the value $\max\{F_e(0), F_e(1), \ldots, F_e(n)\}$. But this contradicts the fact that ϑ is a numbering in which the set of indices of partial functions with finite domain is dense simple relative to K. Hence, $\mathcal{C}$ is not Vac_ψ-learnable, and thus neither Fin_ψ nor Ex_ψ-learnable.

Now, assume by way of contradiction that there exists a Bc_ψ-learner M for $\mathcal{C}$. By Claim 9, it follows that, for each p and n, M outputs only finitely many different indices of the form $\langle p, i\rangle$ while learning f_n. Furthermore, by an argument similar to that of the previous paragraph, it can be shown that, for each n, the set $\{p \mid (\exists i \in \mathbb{N})[M \text{ outputs } \langle p, i\rangle \text{ while learning } f_n]\,\}$ is finite. Hence, the overall number of indices output by the learner while learning an f_n is finite. It follows that M is actually a Vac_ψ-learner for $\mathcal{C}$. But such a learner does not exist as shown in the previous paragraph. □ (**Theorem 8**)

The next result shows, in part, that there exist ExKrt-numberings that are not FinKrt-numberings.

Theorem 10. *There exists a numbering ψ satisfying (a)–(e) below.*

(a) ψ is an Ex-acceptable numbering.
(b) ψ is an ExKrt-numbering.
(c) ψ is not a FinKrt-numbering.

(d) ψ is effectively optimal for Ex*-learning.*
(e) ψ is neither optimal for Fin *nor for* Bc*-learning.*

Proof. Let ψ be such that, for each e and x,

$$\psi_e(x) = \begin{cases} \varphi_e(x), & \text{if } (\exists s > x)(\exists n)[n^2 \leq C_s(e) \leq n^2 + n]; \\ \uparrow, & \text{otherwise.} \end{cases} \tag{23}$$

First, we show that the set of indices of partial functions with infinite domain is immune. Let E be any infinite r.e. set of indices and let $e_0, e_1, e_2, \ldots$ be any ascending recursive sequence of elements of E. Then, there exists a constant c such that, for each i, $C(e_{i+1}) < C(e_i) + c$. Let n be so large that $n > C(e_0)$ and $n > c$. As there are only finitely many indices with Kolmogorov complexity below $n^2 + n$, there exists a largest i such that $C(e_i) \leq n^2 + n$. Note that

$$n^2 + n < C(e_{i+1}) < C(e_i) + c < n^2 + 2n < (n+1)^2. \tag{24}$$

It follows that $\psi_{e_{i+1}}$ has a finite domain. Hence, the set of indices of partial functions with infinite domain is immune.

To show that ψ is Ex-acceptable: let e be given. It follows by an argument similar to that of the previous paragraph that there exist n and y such that $n^2 \leq C(\text{pad}(e,y)) \leq n^2 + n$. Furthermore, one can find from e the least such y in the limit. One then has that $\psi_{\text{pad}(e,y)} = \varphi_e$.

To show that ψ is an ExKrt-numbering: let $E_0, E_1, E_2, \ldots$ be a uniformly r.e. family of infinite sets such that, for each p and each $e \in E_p$, $\varphi_e = \psi_p(\langle e, \cdot \rangle)$. One can construct a machine M to witness that ψ is an ExKrt-numbering as follows. Given p, M finds (in the limit) the least $e \in E_p$ for which there exists an n such that $n^2 \leq C(e) \leq n^2 + n$. (The existence of such an e follows by an argument similar to that of the first paragraph.) Then, $\psi_e = \varphi_e = \psi_p(\langle e, \cdot \rangle)$.

To show that ψ is not a FinKrt-numbering: assume by way of contradiction otherwise. Let e_0 be such that $\psi_{e_0} = \lambda x.x$. Then, enumerate $e_1, e_2, e_3, \ldots$ such that, for each n, $\psi_{e_{n+1}} = \psi_{e_n}(\langle e_{n+1}, \cdot \rangle)$. One can show by induction that, for each n, $\text{rng}(\psi_{e_{n+1}})$ is a proper subset of $\text{rng}(\psi_{e_n})$. Hence, $\{e_0, e_1, e_2, \ldots\}$ is an infinite r.e. set of ψ-indices of total functions. But this would contradict the fact that the set of indices of partial functions with infinite domain is immune.

To show that ψ is optimal for Ex-learning: it was shown above that ψ is Ex-acceptable. It is known that Ex-acceptable numberings are effectively optimal for Ex-learning [JS10].

To show that ψ is not optimal for Fin-learning: consider the class of all constant functions. This class of functions is Fin-learnable. However, if this class could be Fin_ψ-learned, then there would be an infinite r.e. set consisting only of indices of total functions. Again, this would contradict the fact that the set of indices of functions with infinite domain is immune.

In order to see that ψ is not optimal for Bc-learning, it can be shown that every Bc_ψ-learner can be transformed into a Vac_ψ-learner. However, as there are Bc-learnable classes of partial functions which are not Vac-learnable, the numbering ψ cannot be optimal for Bc-learning. □ (**Theorem 10**)

Theorem 11. *There exists a numbering ψ satisfying (a)–(d) below.*

(a) ψ is a VacKrt*-numbering.*
(b) ψ is not an ExKrt*-numbering.*
(c) ψ is effectively optimal for Vac*.*
(d) ψ is not optimal for any of the learning criteria Fin*,* Ex*,* Bc*.*

Proof. For this proof, let $(C_s)_{s\in\mathbb{N}}$ be a sequence of uniformly recursive approximations to C^K, such that $C^K(d) = \limsup_{s\to\infty} C_s(d)$. Here we assume that the approximation is such that, for any s and any e, there are at most 2^e many d such that $C_s(d) < e$. Let, for all d, e,

$$\psi_{\langle d,e\rangle}(x) = \begin{cases} \varphi_e(\langle\langle d,e\rangle, x\rangle), & \text{if } [\log(d) \leq e+1] \text{ and } (\exists s > x)[C_s(d) \geq e]; \\ \uparrow, & \text{otherwise.} \end{cases}$$

Let g, h be recursive functions such that, for all p, x, $\varphi_{g(e)}(x) = \psi_e(x)$ and $\varphi_{h(e)}(\langle p, x\rangle) = \varphi_e(x)$. Note that there exist such g, h.

Claim 12. *(i) If $C^K(d) < e$ or $\log(d) > e+1$, then $\psi_{\langle d,e\rangle}$ is a finite function.*
(ii) For all e, for all d such that, $\log(d) \leq e+1$ and $C^K(d) \geq e$, the following holds: $(\forall x)[\psi_{\langle d,e\rangle}(x) = \varphi_e(\langle\langle d,e\rangle, x\rangle)]$.
Here, note that for all e, there exists a d such that $\log(d) \leq e+1$ and $C^K(d) \geq e$.
(iii) For all e, for all d such that, $\log(d) \leq h(e)+1$ and $C^K(d) \geq h(e)$, the following holds: $(\forall x)[\psi_{\langle d,h(e)\rangle}(x) = \varphi_{h(e)}(\langle\langle d,h(e)\rangle, x\rangle) = \varphi_e(x)]$.
(iv) For all e, for all d such that $\log(d) \leq g(e)+1$ and $C^K(d) \geq g(e)$, the following holds: $(\forall x)[\psi_{\langle d,g(e)\rangle}(x) = \varphi_{g(e)}(\langle\langle d,g(e)\rangle, x\rangle) = \psi_e(\langle\langle d,g(e)\rangle, x\rangle)]$.

Parts (i) and (ii) follow immediately from the construction. Parts (iii) and (iv) follow using part (ii) and definitions of g and h.

By part (ii) of Claim 12 it follows that ψ is a numbering of all the partial recursive functions. We now show the different parts of the theorem.

(a) Let $f(e,s) = \langle d, g(e)\rangle$ such that $\log(d) \leq g(e)+1$ and $C_s(d) \geq g(e)$. By part (iv) of Claim 12, we have that f witnesses that ψ satisfies VacKrt.

(b) Suppose $\langle d_0, e_0\rangle$ is such that, for all x, $\psi_{\langle d_0,e_0\rangle}(x) = x$. Suppose by way of contradiction that H witness ExKrt for ψ, that is, for all i, x, $\psi_{H(i)}(x) = \psi_i(\langle H(i), x\rangle)$. Then for each n, let $\langle d_{n+1}, e_{n+1}\rangle = H(\langle d_n, e_n\rangle)$. Thus,

$$(\forall n, x)\,[\psi_{\langle d_{n+1},e_{n+1}\rangle}(x) = \psi_{\langle d_n,e_n\rangle}(\langle\langle d_{n+1}, e_{n+1}\rangle, x\rangle)]. \tag{25}$$

Now, for all n, ψ_{d_n,e_n} is total. Furthermore, $range(\psi_{d_{n+1},e_{n+1}}) \subset range(\psi_{d_n,e_n})$. Thus, $\langle d_n, e_n\rangle$ are pairwise different for different n. Thus, for each $a \in \mathbb{N}$, one can effectively find an n_a with $d_{n_a} \geq a \wedge e_{n_a} \geq a$. For sufficiently large a, $C^K(d_{n_a}) \leq 2\log(a)$ and $e_{n_a} \geq a$. But then, for sufficiently large a, by Claim 12, $\psi_{\langle d_{n_a},e_{n_a}\rangle}$, would be finite function. A contradiction.

(c) To see that the numbering is effectively optimal for vacillatory learning note that, by Claim 12 and definition of h, for all e, for all but finitely many s, for the least d such that $\log(d) \leq h(e)+1$ and $C_s(d) \geq h(e)$, we have $\psi_{\langle d,h(e)\rangle}(x) =$

$\varphi_e(x)$. Thus, one can just convert a Vac-learner M using φ as the hypothesis space to a Vac-learner M' using ψ as the hypothesis space by having $M'(f[n]) = \langle d, h(M(f[n])\rangle$, where d is least such that $\log(d) \leq h(M(f[n]) + 1$ and $C_n(d) \geq h(M(f[n])$.

(d) Let $\mathcal{S}$ be a class of total functions which is Bc-learnable by some learner M using ψ as the hypothesis space. For any total f, let $E_f = \{M(f[n]) : n \in \mathbb{N}\}$. We claim that E_f is finite for each $f \in \mathcal{S}$. Suppose by way of contradiction that for some $f \in \mathcal{S}$, E_f is infinite. Note that E_f is an r.e. set. Let $\eta(e) = d_e$, for the first pair $\langle d_e, e\rangle$ enumerated in E_f, if any. Now, $\eta(e)$ is defined on infinitely many e, and thus $C^K(d_e) \leq 2\log(e)$ for infinitely many e in the domain of η. But then, by Claim 12, $\psi_{\langle d_e,e\rangle}$ is a finite function for infinitely many e in the domain of η, a contradiction to M Bc-learning f. Thus, M is also a Vac-learner for $\mathcal{S}$. As there are classes of total functions which are Bc-learnable but not Vac-learnable [CS83], ψ is not optimal for Bc-learning.

Now, suppose by way of contradiction that M Ex-learns all constant functions using the numbering ψ. Thus, for each a, there exists a constant c such that, for some d_a, e_a, for all but finitely many n, $M(c^\infty[n]) = \langle d_a, e_a\rangle$, with $\min\{d_a, e_a\} \geq a$. Note that one such pair of values d_a, e_a can be computed using the oracle K. Then, for almost all a, $C^K(d_a) \leq 2\log(a)$ and $e_a \geq a$. Hence, by Claim 12, for all but finitely many a, $\psi_{\langle d_a,e_a\rangle}$ is a finite function. Thus, M does not Ex-learn the class of all constant functions using the numbering ψ. It follows that ψ is not optimal for Fin and Ex-learning. $\square$ (**Theorem 12**)

The final result of this section establishes, in part, that there exist BcKrt-numberings that are not VacKrt-numberings.

Theorem 13. *There exists a numbering ψ satisfying (a)–(d) below.*

(a) ψ is a BcKrt*-numbering.*
(b) ψ is not a VacKrt*-numbering.*
(c) ψ is not optimal for any of the learning criteria Fin, Ex, Vac.
(d) ψ is effectively optimal for Bc*-learning.*

Proof. Let ψ be such that, for each e and x,

$$\psi_e(x) = \begin{cases} \varphi_e(x), & \text{if } [\quad \varphi_{e,x}(0)\uparrow \\ & \quad\quad \vee\ |\mathrm{rng}(\varphi_e)| \geq 2 \\ & \quad\quad \vee\ [\varphi_e(0)\downarrow\ \wedge\ C(\varphi_e(0)) < \log(\varphi_e(0))] \\ & \quad\quad \vee\ [\varphi_e(0)\downarrow\ \wedge\ |W_{\log(\varphi_e(0)),x}| < e] \\ & \quad\quad \vee\ [\varphi_e(0)\downarrow\ \wedge\ |W_{\log(\varphi_e(0))}| > x] \quad]; \\ \uparrow, & \text{otherwise.} \end{cases} \tag{26}$$

To show that ψ is a BcKrt-numbering: let $E_0, E_1, E_2, \ldots$ be a uniformly r.e. family of infinite sets such that, for each p and each $e \in E_p$, $\varphi_e = \psi_p(\langle e, \cdot\rangle)$. One can construct a machine M to witness that ψ is a BcKrt-numbering as follows. Suppose that M is given p. Then, at stage s, M outputs the first element e in some canonical enumeration of E_p such that

$$\begin{array}{ll} & \varphi_{e,s}(0)\uparrow \\ \vee & |\mathrm{rng}(\varphi_{e,s})| \geq 2 \\ \vee & [\varphi_{e,s}(0)\downarrow \ \wedge \ C_s(\varphi_{e,s}(0)) < \log(\varphi_{e,s}(0))] \\ \vee & e > s. \end{array} \tag{27}$$

Consider the following two cases.

Case 1: There exists an $e \in E_p$ such that $\varphi_e(0)\uparrow$, $|\mathrm{rng}(\varphi_e)| \geq 2$, or $\varphi_e(0) = y$ for some y with $C(y) < \log(y)$. Then, M converges to the first such e in the canonical enumeration of E_p. Furthermore, for this e, it holds that $\psi_e = \varphi_e = \psi_p(\langle e, \cdot\rangle)$.

Case 2: Not Case 1. Then, the set $F = \{\varphi_e(0) : M \text{ outputs } e\}$ has an empty intersection with the simple set $\{d : C(d) < \log(d)\}$ and, hence, is finite. Let $c = \max\{|W_{\log(d)}| : d \in F \ \wedge \ |W_{\log(d)}| < \infty\}$. As F is finite, this maximum c is taken over only finitely many numbers and, hence, $c < \infty$. Furthermore, as Case 1 does not apply, M outputs each index in E_p only finitely often. Hence, M outputs almost always some index $e > c$. If, for such an e, $W_{\log(\varphi_e(0))}$ is finite, then, for each x, $|W_{\log(\varphi_e(0)),x}| \leq c < e$. On the other hand, if $W_{\log(\varphi_e(0))}$ is infinite, then, for each x, $|W_{\log(\varphi_e(0))}| > x$. Either way, M outputs almost always an e such that $\psi_e = \psi_p(\langle e, \cdot\rangle)$.

It follows from the case distinction that M witnesses that ψ is a BcKrt-numbering.

To show that ψ is not a VacKrt-numbering: assume by way of contradiction otherwise, as witnessed by M. We show that, under this assumption, one can decide membership in $\{x \mid W_x \text{ is finite}\}$ using an oracle for K (which is impossible). It is known that, for almost all x, there exist distinct y and z such that $\log(y) = \log(z) = x$, but $C(y) \geq x$ and $C(z) \geq x$.[5] Given x, one can find such y and z using an oracle for K. One can then determine p such that, for each v and w,

$$\varphi_p(\langle v, w\rangle) = \begin{cases} y, & \text{if } v \text{ is even;} \\ z, & \text{if } v \text{ is odd.} \end{cases} \tag{28}$$

Note that $|\mathrm{rng}(\varphi_p)| = 2$ and, hence, $\psi_p = \varphi_p$. One can then run M on input p and, using the oracle for K, determine the largest e among the finitely many indices output by M. Hence, for some $v < e$, ψ_v is either the constantly y function, or the constantly z function. It follows that either $|W_{\log(\varphi_v(0))}| < v$ or $|W_{\log(\varphi_v(0))}|$ is infinite. If the former, then

$$|W_x| = |W_{\log(\varphi_v(0))}| < v < e. \tag{29}$$

If the latter, then

$$|W_x| = |W_{\log(\varphi_v(0))}| \geq e. \tag{30}$$

[5] Recall from Section 2 that, for each y, $\log(y) \overset{\text{def}}{=} \min\{x \mid 2^x \geq y\}$. For each $x \geq 1$, there are 2^{x-1} many numbers y with $\log(y) = x$ and only $2^{x-1} + 1$ many numbers y with $C(y) < x$. Furthermore, for sufficiently large x, there will exist three or more programs less than $2^{x-1} + 1$ that either produce no output, or produce the same output as programs less than themselves. Hence, for sufficiently large x, such y and z exist.

Hence, W_x is finite iff $|W_x| < e$. As $|W_x| < e$ can be decided using an oracle for K, this allows one to determine whether W_x is finite. Since this is impossible, it follows that M does not witness that ψ is a VacKrt-numbering and, more generally, that ψ is not a VacKrt-numbering.

To show that ψ is not optimal for any of the learning criteria Fin, Ex, Vac: note that the class of constant functions is Fin-learnable. But it can be shown that this class is neither Fin_ψ, Ex_ψ nor Vac_ψ-learnable using a proof-idea similar to that of the previous paragraph. Assume by way of contradiction otherwise, as witnessed by M. Then, given x, one can use an oracle for K to find a y such that $\log(y) = x$ and $C(y) \geq x$. Then, when M is fed a text for the constantly y function, M outputs finitely many indices whose maximum is some e. Using this e and the oracle for K, one can determine whether W_x is finite as in the previous paragraph (a contradiction). Hence, ψ is not optimal for any of the learning criteria Fin, Ex, Vac.

To show that ψ is effectively optimal for Bc-learning: suppose that M is a Bc-learner that uses φ as its hypothesis space. Further suppose that M is fed a text for a partial recursive function α and that $e_0, e_1, e_2, \ldots$ is the sequence of indices output by M on this text. Without loss of generality, suppose that this sequence is monotonically increasing, e.g., due to padding. We show that, for almost all i, $\psi_{e_i} = \varphi_{e_i}$. Consider the following three cases.

Case 1: $\alpha(0)\uparrow$. Then, for almost all i, $\varphi_{e_i}(0)\uparrow$ and, hence, $\psi_{e_i} = \varphi_{e_i}$.

Case 2: $\alpha(0)\downarrow$ and $|W_{\log(\alpha(0))}|$ is infinite. Then, for almost all i, $|W_{\log(\varphi_{e_i}(0))}|$ is infinite and, hence, $\psi_{e_i} = \varphi_{e_i}$.

Case 3: $\alpha(0)\downarrow$ and $|W_{\log(\alpha(0))}|$ is finite. Then, as $e_0, e_1, e_2, \ldots$ is monotonically increasing, for almost all i, $|W_{\log(\varphi_{e_i}(0))}| < e_i$. Hence, for almost all i, $\psi_{e_i} = \varphi_{e_i}$.

This case distinction shows that Bc_φ-learners that output successively larger indices are also Bc_ψ-learners. Hence, the numbering ψ is effectively optimal for Bc-learning. □ (**Theorem 13**)

5 When Kleene's Recursion Theorem Is Ineffective

This section presents our results concerning numberings that satisfy Kleene's Recursion Theorem, but not in an effective way. Moelius [Moe09, Theorem 4.1] showed that there exist numberings that are not BcKrt-numberings, but in which Kleene's Recursion Theorem holds. Hence, in such numberings, Kleene's Recursion Theorem is extremely ineffective. Theorems 14 and 15 expand on Moelius's result by showing that there exist such numberings that are optimal for learning and such numberings that are not optimal for learning (respectively).

Theorem 14. *There exists a numbering ψ satisfying (a)–(c) below.*

(a) ψ satisfies Kleene's Recursion Theorem.
(b) ψ is not a BcKrt*-numbering.*
(c) ψ is not optimal for any of the learning criteria Fin, Ex, Vac, Bc.

Theorem 15. *There exists a numbering ψ satisfying (a)–(c) below.*

(a) ψ satisfies Kleene's Recursion Theorem.
(b) ψ is not a BcKrt*-numbering.*
(c) ψ is effectively optimal for Ex, Vac *and* Bc*-learning.*

Acknowledgment. The authors would like to thank Samuel E. Moelius III for discussions and support in writing this paper.

References

[Bar74] Bārzdiņš, J.: Two theorems on the limiting synthesis of functions. Theory of Algorithms and Programs 1, 82–88 (1974)

[Cas99] Case, J.: The power of vacillation in language learning. SIAM Journal on Computing 28, 1941–1969 (1999)

[CJS02] Case, J., Jain, S., Suraj, M.: Control structures in hypothesis spaces: The influence on learning. Theoretical Computer Science 270(1-2), 287–308 (2002)

[CM09] Case, J., Moelius III, S.E.: Program Self-reference in Constructive Scott Subdomains. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) CiE 2009. LNCS, vol. 5635, pp. 89–98. Springer, Heidelberg (2009)

[CS83] Case, J., Smith, C.H.: Comparison of identification criteria for machine inductive inference. Theoretical Computer Science 25, 193–220 (1983)

[Gol67] Mark Gold, E.: Language identification in the limit. Information and Control 10, 447–474 (1967)

[Jai11] Jain, S.: Hypothesis spaces for learning. Information and Computation 209(3), 513–527 (2011)

[JS10] Jain, S., Stephan, F.: Numberings optimal for learning. Journal of Computer and System Sciences 76(3-4), 233–250 (2010)

[Kle38] Kleene, S.C.: On notation for ordinal numbers. Journal of Symbolic Logic 3(4), 150–155 (1938)

[LV08] Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Texts in Computer Science. Springer (2008)

[Moe09] Moelius III, S.E.: Program Self-Reference. PhD thesis, University of Delaware (2009)

[Nie09] Nies, A.: Computability and Randomness. Oxford Logic Guides. Oxford University Press (2009)

[OW82] Osherson, D.N., Weinstein, S.: Criteria of language learning. Information and Control 52, 123–138 (1982)

[Ric80] Riccardi, G.A.: The Independence of Control Structures in Abstract Programming Systems. PhD thesis, SUNY Buffalo (1980)

[Ric81] Riccardi, G.A.: The independence of control structures in abstract programming systems. Journal of Computer and System Sciences 22(2), 107–143 (1981)

[Rog67] Rogers Jr., H.: Theory of Recursive Functions and Effective Computability. McGraw Hill (1967); Reprinted. MIT Press (1987)

[Roy87] Royer, J.S.: A Connotational Theory of Program Structure. LNCS, vol. 273. Springer, Heidelberg (1987)

Sub-computable Bounded Pseudorandomness[⋆]

Douglas Cenzer[1] and Jeffrey B. Remmel[2]

[1] Department of Mathematics, University of Florida, P.O. Box 118105, Gainesville, Florida 32611
cenzer@math.ufl.edu

[2] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112
jremmel@ucsd.edu

Abstract. This paper defines a new notion of bounded pseudorandomness for certain classes of sub-computable functions where one does not have access to a universal machine for that class within the class. In particular, we define such a version of randomness for the class of primitive recursive functions and a certain subclass of $PSPACE$ functions. Our new notion of primitive recursive bounded pseudorandomness is robust in that there are equivalent formulations in terms of (1) Martin-Löf tests, (2) Kolmogorov complexity, and (3) martingales.

Keywords: algorithmic randomness, complexity, computability.

1 Introduction

The study of algorithmic randomness has flourished over the past century. The main topic of study in this paper is the randomness of a single real number which, for our purposes, can be thought of as an infinite sequence $X = (X(0), X(1), \dots)$ from $\{0,1\}^\omega$. Many interesting notions of algorithmic randomness for real numbers have been investigated in recent years. The most well-studied notion, Martin-Löf randomness [24] or 1-randomness, is usually defined in terms of measure. A real X is 1-random if it is *typical*, that is, X does not belong to any effective set of measure zero in the sense of Martin-Löf [24]. A second definition of 1-randomness may be given in terms of information content. X is 1-random if it is *incompressible*, that is, the initial segments $(X(0), X(1), \dots, X(n))$ have high Kolmogorov [18] or Levin-Chaitin [10,20] complexity. A third definition may be given in terms of martingales. X is 1-random if it is unpredictable, that is, there is no effective martingale for which one can obtain unbounded capital by betting on the values of X [27]. These three versions have been shown by Schnorr [26] to be equivalent. This demonstrates the robustness of the concept of Martin-Löf randomness. Many other notions of algorithmic randomness have been studied and in most cases, formulations are only given for one or perhaps two versions. For a thorough study of the area of algorithmic randomness, the reader is directed to three excellent recently published books: Downey and Hirschfeldt [15], Nies [25] and Li and Vitanyi [21].

[⋆] Cenzer was partially supported by the NSF grant DMS-652372.

In this paper we present a notion of bounded pseudorandomness for certain classes of sub-computable functions where one does not have access to a universal machine for that class within the class. We will first state our definitions for the class of primitive recursive functions and define a new notion of *bounded primitive recursive pseudorandomness* (BP randomness). We shall show that there are three equivalent definitions of BP randomness, one in terms of measure, one in terms of compressibility, and one in terms of martingales. For measure, a *bounded primitive recursive test* will be a primitive recursive sequence of clopen sets $(U_n)_{n\geq 0}$ such that U_n has measure $\leq 2^{-n}$ and we define X to be BP random if it does not belong to $\bigcap_{n\geq 0} U_n$ for any such test. For compressibility, we say that X is BP compressed by a primitive recursive machine M if there is a primitive recursive function f such that $C_M(X \upharpoonright f(c)) \leq f(c) - c$ for all c where C_M is a primitive recursive analogue of Kolomogrov complexity. We will show that X is BP random if and only if X is not compressible by any primitive recursive machine. For martingales, we say that a primitive recursive martingale d succeeds on a sequence X if there is a primitive recursive function f such that $d(X \upharpoonright f(n)) \geq 2^n$ for each n. Thus d makes us rich betting on X and f tells us how fast this happens. We will show that X is BP random if and only if there is no primitive recursive martingale which succeeds on X. These definitions can easily be adapted to define a notion of bounded pseudorandomness for other classes of sub-computable functions. As an example, we will define a notion of bounded $PSPACE$ pseudorandomness.

The terms *bounded randomness* or *finite randomness* are sometimes used to refer to versions of randomness given by tests in which the c.e. open sets are in fact clopen. Thus our notion of BP randomness is "bounded" in this sense. The term "finite" comes from the fact that any clopen set U is the finite union of intervals $U = [\sigma_1] \cup \cdots \cup [\sigma_k]$. Kurtz randomness [19], also refered to as weak randomness, falls into this category. A real X is Kurtz random if it does not belong to any Π^0_1class Q of measure zero. But any Π^0_1class may be effectively expressed as a decreasing intersection of clopen sets $Q = \bigcap_n Q_n$ where the clopen sets Q_n are unions of intervals of length n. If $\mu(Q) = 0$, it is easy to find a subsequence $U_i = Q_{n_i}$ with $\mu(U_i) \leq 2^{-i}$ and thus $(U_n)_{n\geq 0}$ is a bounded Martin-Löf test. Another special type of bounded randomness was recently studied by Brodhead, Downey and Ng [8].

As shown by Wang [28], Kurtz random reals need not be stochastic in the sense of Church. For example, it need not be the case that the number of occurrences of 0's in a Kurtz random sequence X tends to 1/2 in the limit. In such a situation, one often uses the term pseudorandom instead of randomness. Our BP random reals are pseudorandom in this sense. That is, we will construct a recursive real which is BP random but not stochastic. However, we will show that BP random sets satisfy only a weak version of the stochastic property.

A lot of work has been done on various notions of resource-bounded randomness. One of the first approaches to resource-bounded randomness was via the stochastic property of typical reals [12]. It is expected that for a random real, the relative density of the occurrences of 0 and of 1 should be equal in the limit.

We identify a set A of natural numbers with its characteristic function and in those terms we expect that $\lim_n \frac{card(A \cap [[n]])}{n} = \frac{1}{2}$ where $[[n]] = \{0, 1, \ldots, n\}$ for any $n \in \mathbb{N}$. Levin [20] defined a notion of primitive randomness for a set A to mean that for every primitive recursive set B, $A \cap B$ is stochastic relative to B and constructed a recursive set that is primitive random. Di Paola [13] studied similar notions of randomness in the Kalmar hierarchy of elementary functions. Wilber [29] defined a set A to be P-random if, for every $PTIME$ set B, A and B agree on a set of density $\frac{1}{2}$ and constructed an exponential time computable P-random set.

The literature of computational complexity contains many papers on random number generators and cryptography which examine various notions of pseudorandomness. For example, Blum and Micali [7] gave a weak definition of pseudorandom sequences in which a randomly generated sequence is said to be pseudorandom if it meets all $PTIME$ statistical tests. Ko [17] gave definitions of randomness with respect to polynomial time and space complexity which are in the tradition of algorithmic randomness as established by Levin, Martin-Löf and Chaitin. One of the notions of Ko has equivalent formulations in terms of tests and in terms of compressibility and has bounds on the compressibility that are similar in nature to those presented in this paper. Ko's definitions are based on computation from a universal machine M and, in particular, states that X is ($PSPACE$) compressed with polynomial bounding function f if, for every k, there exists infinitely many n such that $K_M(X \upharpoonright n) < n - (log\ n)^k$. In contrast, our definitions are not based on the existence of a universal machine.

Lutz [22] defined an important notion of resource-bounded randomness in terms of martingales. Here a real is, say, $PSPACE$ random if there is no $PSPACE$ martingale which succeeds on X. One can also say that a set $\mathcal{X}$ of reals has $PSPACE$ measure one if there is no $PSPACE$ martingale which succeeds on every element of $\mathcal{X}$. Then almost every $EXPSPACE$ real is random and this can be used to study properties of $EXPSPACE$ reals by examining whether the set of $EXPSPACE$ reals with the property has measure one. Buhrman and Longpre [9] gave a rather complicated equivalent formulation of $PSPACE$ randomness in terms of compressibility. Lutz's notion of complexity theoretic randomness concept has had great impact on complexity theory [1,2,3]. Shen et al. [11] have recently studied on-line complexity and randomness.

There are several important properties of Martin-Löf random reals that are regarded as fundamental such as Ville's theorem which states that any effective subsequence of a random sequence is also random. We will prove an analogue of Ville's theorem for BP randomness. Another fundamental property for random reals is van Lambalgen's theorem, which states that the join $A \oplus B$ of two random sets is random if and only if A is random relative to B and B is random. We define a notion of relative BP randomness which still has three equivalent formulations, and prove an analogue of van Lambalgen's theorem for this notion. Our formulation is a type of truth-table reducibility similar to that of Miyabe [23].

For the case of bounded $PSPACE$ randomness, we give two different notions, one which has equivalent versions for compression and for measure and the other of which has equivalent versions for measure and for martingales. These notions are actually a hybrid of polynomial time and space.

We note that we get a notion of bounded computable pseudorandomness by replacing primitive recursive functions by computable functions in our definitions. In such a case, our definitions are equivalent to Kurtz randomness which has nice equivalent formulations in all three settings. This was previously shown by Wang [28] for the martingale definition and by Downey, Griffiths and Reid [14] and also Bienvenu and Merkle [6] for the compression definition.

Jockusch [16] showed that Kurtz random sets are *immune*, that is, they do not have infinite c. e. subsets. We will consider an analogue of immunity for our notion of bounded pseudorandom sets.

We will normally work with the usual alphabet $\Sigma = \{0,1\}$ and the corresponding set $\{0,1\}^*$ of finite strings and the Cantor space $\{0,1\}^\omega$ of infinite sequences, but our results hold for any finite alphabet.

The outline of this paper is as follows. In section 2, we study BP randomness and show the equivalence of our three versions. We construct a computable real which is BP random. We prove an analogue of Ville's theorem for primitive recursive subsequences of BP random reals. We will also define a notion of relative randomness and prove an analogue of van Lambalgen's theorem. In section 3, we consider two notions of bounded $PSPACE$ pseudorandomness and give two equivalent definitions for each notion. Finally, in section 4, we will state our conclusions and some directions for further research.

2 Bounded Primitive Recursive Randomness

In this section, we will define the three notions of primitive recursive randomness, Kolmogorov BP randomness, Martin-Löf BP randomness, and martingale BP randomness and show their equivalence. Hence, we will say that a real X is BP random if it satisfies one of these three definitions. We will then prove analogues of Ville's Theorem and van Lambalgen's Theorem of BP random reals.

We will work with the family of primitive recursive functions $M : \Sigma^* \to \Sigma^*$, where Σ is a finite alphabet (normally $\{0,1\}$). Note that we can code finite strings as numbers in order to define these primitive recursive functions and that the coding and decoding functions are all primitive recursive.

Martin-Löf BP Randomness

In what follows, the code $c(\sigma)$ of a finite sequence $\sigma = \sigma_1 \dots \sigma_n \in \{0,1\}^*$ is just the natural number whose binary expansion is $1\sigma_1 \dots \sigma_n$. Given a nonempty finite set $S = \{\sigma^{(1)}, \dots, \sigma^{(k)}\}$ of strings in $\{0,1\}^*$ such that $c(\sigma^{(1)}) < \dots < c(\sigma^{(k)})$, the code $C(S)$ of S is defined be the natural number n whose ternary expansion is $2c(\sigma^{(1)})2 \dots 2c(\sigma^{(k)})$. We let 0 be the code of the empty set. We say a sequence $\{U_n : n \in \mathbb{N}\}$ of clopen sets is a primitive recursive sequence if there is a primitive recursive function f such that for all n, $f(n)$ is a code of a finite set $G_n = \{\sigma_{1,n}, \dots, \sigma_{k(n),n}\}$ such that $U_n = [G_n]$. Here for any string $\sigma \in \{0,1\}^*$ and $X \in \{0,1\}^\omega$, we write $\sigma \sqsubset X$ if σ is an initial segment of X and we let $[\sigma] = \{X \in \{0,1\}^{\mathbb{N}}$ such that $\sigma \sqsubset X\}$. For any set G of strings in $\{0,1\}^*$, we let $[G] = \bigcup\{[\sigma] : \sigma \in G\}$.

Since we can recover $\{\sigma_{1,n}, \dots, \sigma_{k(n),n}\}$ from $f(n)$ in polynomial time, it is easy to see that given any primitive recursive sequence $\{U_n : n \in \mathbb{N}\}$, we can produce a primitive recursive function g such that $g(n)$ is a code of a finite set $G_n = \{\tau_{1,n}, \dots, \tau_{r(n),n}\} subseteq \{0,1\}^{\ell(n)}$ where $U_n = [G_n]$ and r and ℓ are also primitive recursive functions.

We define a primitive recursive test to be a primitive recursive sequence $(U_n)_{n \geq 0}$ of clopen sets such that, for each n, $\mu(U_n) \leq 2^{-n}$. It follows that there is a primitive recursive function m such that $m(n)$ codes the measure $\mu(U_n)$ as a dyadic rational. Since the measures $\mu(U_n)$ may be computed, one could equivalently consider a primitive recursive sequence $\{V_n : n \in \mathbb{N}\}$ such that $lim_n \; \mu(V_n) = 0$ and there is a primitive recursive function f such that, for each p, $\mu(V_{f(p)}) \leq 2^{-p}$.

We observe here that $\bigcap_n U_n$ will be a Π^0_1 class of measure 0 so that any primitive recursive test is a Kurtz test and, hence, is also a Schnorr test.

We say that an infinite sequence $X \in \{0,1\}^\omega$ is *Martin-Löf BP random* if X *passes* every primitive recursive test, that is, for every primitive recursive test $(U_n)_{n \geq 0}$, there is some n such that $X \notin U_n$.

By the remarks above, every Kurtz random real is Martin-Löf BP random.

Proposition 1. *X is Martin-Löf BP random if and only if there is no primitive recursive sequence $(U_n)_{n \geq 0}$ of clopen sets with $\mu(U_n) = 2^{-n}$ such that $X \in \bigcap_n U_n$.*

Proof. The if direction is immediate. Now suppose that there is a primitive recursive sequence $(V_n)_{n \geq 0}$ such that $\mu(V_n) \leq 2^{-n}$ and $X \in \bigcap_n V_n$. Let $V_n = \bigcup_{\sigma \in G_n} [\sigma]$ where $G_n \subseteq \{0,1\}^{\ell(n)}$ for some primitive recursive function $\ell(n)$ where $\ell(n) \geq n$ for all $n \geq 0$. Then $\mu(V_n) = \frac{card(G_n)}{2^{\ell(n)}} \leq 2^{-\ell(n)}$. Now define H_n to be G_n together with $2^{\ell(n)-n} - card(G_n)$ additional strings of length $\ell(n)$ and let $U_n = \bigcup_{\tau \in H_n} [\tau]$. Then for each n, $X \in U_n$ and $\mu(U_n) = 2^{-n}$.

We will also need the notion of a *weak* primitive recursive test. A *weak* primitive recursive test $(U_n)_{n \geq 0}$ is a primitive recursive sequence $(G_n)_{n \geq 0}$ of finite sets of strings, where there is a primitive recursive function ℓ such that for each n, $U_n = [G_n]$ and, for all $\tau \in G_n$, $|\tau| = \ell(n)$ and $\mu(U_{n+1} \cap [\tau]) \leq \frac{1}{2}\mu([\tau])$.

We can convert each primitive recursive test $(U_n)_{n \geq 0}$ into a weak primitive recursive test as follows. First, we may assume that $U_{n+1} \subseteq U_n$ for each n, since the sequence given by $W_n = \bigcap_{i \leq n} U_i$ is also a primitive recursive test with $\mu(W_n) \leq \mu(U_n) \leq 2^{-n}$. Next suppose $U_n = [\tau_{1,n}] \cup \dots \cup [\tau_{k(n),n}]$ where there is a primitive recursive function ℓ such that $|\tau_{i,n}| = \ell(n)$ for $1 \leq i \leq k(n)$. Thus each interval $[\tau_{i,n}]$ has measure exactly $2^{-\ell(n)}$. Now the clopen set $U_{\ell(n)+1}$ has a total measure $\leq 2^{-\ell(n)-1}$, so that the relative measure of $\mu(U_{\ell(n)+1} \cap [\tau_{i,n}]) \leq \frac{1}{2}\mu([\tau_{i,n}])$. Then we can define a primitive recursive weak test $(V_n)_{n \geq 0}$ as follows. Let $h(0) = 0$ and let $V_0 = U_0$. Then let $h(1) = \ell(0)+1$ and $V_1 = U_{h(1)}$. In general for $n > 1$, we let $h(n+1) = \ell(h(n)) + 1$ and let $V_{n+1} = U_{h(n+1)}$. Then the sequence $V_0, V_1, \ldots$ will be a weak primitive recursive test. Since the sequence $\{V_n : n \in \mathbb{N}\}$ is a subsequence of the original sequence $\{U_m : m \in \mathbb{N}\}$, it follows that $\bigcap_n V_n = \bigcap_n U_n$, so that X passes the weak test $\{V_n : n \in \mathbb{N}\}$ if and only if it passes the original test.

It follows that a real X passes every primitive recursive test, then it certainly passes every weak primitive recursive test. Conversely, if X fails some primitive recursive test, then the argument above shows that it also fails some weak test. Hence we conclude the following.

Proposition 2. *X is Martin-Löf BP random if and only if it passes every weak primitive recursive test.* □

Kolmogorov BP Random

Let $C_M(\tau)$ be the length $|\sigma|$ of the shortest string σ such that $M(\sigma) = \tau$, that is, the length of the shortest M-description of τ. Notice that we are using plain and not prefix-free complexity. We say that X is primitive recursively compressed by M if there exist primitive recursive functions M and f such that, for every $c \in \mathbb{N}$, $C_M(X \upharpoonright f(c)) \leq f(c) - c$. Our definition of primitive recursive compression is a natural analogue for primitive recursive functions of the usual definition of Kolmogorov compression which says that, for every $c \in \mathbb{N}$, there exists n such that $C_M(X \upharpoonright n) \leq n - c$. Of course, one defines Kolmogorov randomness in terms of prefix-free complexity K_M since there are no infinite Kolmogorov random sequences for plain complexity. We use plain complexity here since every primitive recursive function is total so that there are no prefix-free primitive recursive machines.

We say that an infinite sequence $X \in \{0,1\}^\omega$ is *Kolmogorov BP random* if it cannot be primitive recursively compressed by any primitive recursive machine M. A notion of prefix-free complexity for primitive recursive functions may be obtained by allowing primitive recursive functions M such that $M(\sigma)$ may diverge. This can be done by introducing a new symbol ∞ as a possible output of $M(\sigma)$ to signify that $M(\sigma)$ diverges. It is not hard to show that this makes no difference.

Proposition 3. *A real X is BP random if and only if it is prefix-free BP random.*

Martingale BP Random

A martingale d is a function $d : \{0,1\}^* \to \mathbb{Q} \cap [0, \infty]$ such that for all $\sigma \in \{0,1\}^*$, $d(\sigma) = \sum_{a \in \{0,1\}} d(\sigma^\frown a)/2$. Of course, any primitive recursive martingale is also a computable martingale. We say that the martingale d *succeeds primitive recursively on X* if there is a primitive recursive function f such that, for all n, $d(X \upharpoonright f(n)) \geq 2^n$. (Of course, we could replace 2^n here with any primitive recursive function which is increasing to infinity.) In general, a martingale d is said to succeed on X if $\limsup_n d(X \upharpoonright n) = \infty$, that is, for every n, there exists m such that $d(X \upharpoonright m) \geq 2^n$. Thus our definition is an effectivization of the usual definition where there is a primitive recursive function f which witnesses that d will return 2^n at some point for every n. We say that X is *martingale BP random* if there is no primitive recursive martingale which succeeds primitive recursively on X. If X is *not* martingale BP random, then there is a computable martingale which succeeds primitive recursively on X and thus certainly succeeds on X, so that X is not computably random. Hence every computably random real is also a martingale BP random real.

Our definition of martingale BP random real has the following equivalent formulations.

Proposition 4. *The following are equivalent.*

(1) *X is martingale BP random.*
(2) *There do not exist a primitive recursive martingale d and a primitive recursive function f such that, for every n, there exists $m \leq f(n)$ such that $d(X \restriction m) \geq 2^n$.*
(3) *There do not exist a primitive recursive martingale d and a primitive recursive function f such that $d(X \restriction m) \geq 2^n$ for all n and all $m \geq f(n)$.*

Proof. Our proof uses the idea of a *savings account* as formulated in [15,25]. That is, if we have a martingale and function as in (2), then we can modify the martingale so that whenever $d(\tau) \geq 2^{n+1}$ but $d(\sigma) < 2^{n+1}$ for all proper initial segments of τ, then we put aside 2^n and only bet with the other half of our capital. This means that we can never drop below 2^n in the future. Thus if we use the function $f'(n) = f(n+1)$, we will satisfy condition (3) and hence satisfy (1) as well.

Our main result in this section is to show the three versions of BP random described above are equivalent.

Theorem 1. *The following statements are equivalent for $X \in \{0,1\}^\omega$.*

(1) *X is Martin-Löf BP random.*
(2) *X is Kolmogorov BP random.*
(3) *X is martingale BP random.*

Proof. We shall show the equivalence of (1) with both (2) and (3).

(1) implies (2): Suppose X is not Kolmogorov BP random. Then there exist primitive recursive M and f such that $C_M(X \restriction f(c)) \leq f(c) - c - 1$ for every $c \in \mathbb{N}$.

Let $U_c = \{X : C_M(X \restriction f(c)) \leq f(c) - c - 1\}$. This is certainly a uniformly primitive recursive sequence of clopen sets. That is, given c, compute $M(\sigma)$ for all σ with $|\sigma| \leq f(c) - c - 1$ and let

$$G_c = \{M(\sigma) : \sigma \in \{0,1\}^{\leq f(c)-c-1}\} \cap \{0,1\}^{f(c)}$$

and $U_c = \bigcup_{\tau \in G_c} [\tau]$. Clearly, $(U_c)_{c \geq 0}$ is a primitive recursive sequence of clopen sets.

We claim that $\mu(U_c) \leq 2^{-c}$. That is, fix c and let $U_c = [\tau_1] \cup [\tau_2] \cup \cdots \cup [\tau_k]$, for distinct $\tau_i \in \{0,1\}^{f(c)}$. Thus there exist $\sigma_1, \ldots, \sigma_k$ such that, for $i = 1, \ldots, k$,

$|\sigma_i| \leq f(c) - c - 1$ and such that $M(\sigma_i) = \tau_i$. Since there are only $2^{f(c)-c} - 1$ strings of length $\leq f(c) - c - 1$, it follows that $k \leq 2^{f(c)-c}$. Since for each i, $\mu([\tau_i]) = 2^{-f(c)}$, it follows that $\mu(U_c) = k \cdot 2^{-f(c)} \leq 2^{f(c)-c} \cdot 2^{-f(c)} = 2^{-c}$. By assumption, $X \in U_c$ for all $c \geq 0$ so that X is not Martin-Löf BP random.

(2) implies (1): Suppose that X is not Martin-Löf BP random. Thus there exist primitive recursive functions g, k, and f so that for all $c \geq 0$, $g(c)$ is a code of a finite sets of strings $\{\tau_{i,c} : 1 \leq i \leq k(c)\} \subseteq \{0,1\}^{f(c)}$ such that if $U_c = [\tau_{1,c}] \cup [\tau_{2,c}] \cup \cdots \cup [\tau_{k(c),c}]$, then $\mu(U_c) \leq 2^{-c}$ and $X \in \bigcap_{c \geq 0} U_c$. We may assume without loss of generality that for each c, $f(c+1) - (c+1) > f(c) - c$. This is because we may always break each $[\tau_i]$ into $[\tau_i {}^\frown 0] \cup [\tau_i {}^\frown 1]$ to increase $f(c)$ by one, if necessary.

We will define a primitive recursive function M such that for all $c \in \mathbb{N}$, $C_M(X \upharpoonright f(c)) \leq f(c) - c$. Since $\mu(U_c) = k(c) \cdot 2^{-f(c)}$, it follows that $k(c) \leq 2^{f(c)-c}$. Now take the lexicographically first $k(c)$ strings $\sigma_{1,c}, \ldots, \sigma_{k(c),c}$ of length $f(c) - c$ and define $M(\sigma_{i,c}) = \tau_{i,c}$. To make M a total function, the remaining strings of length $f(c) - c$ may all be mapped to 0 and all strings not of length $f(c) - c$ for any c are also mapped to 0.

By assumption $X \in U_c$ for every $c \in \mathbb{N}$ so that $X \upharpoonright f(c) = \tau_{i,c}$ for some i. Hence $M(\sigma_{i,c}) = \tau_{i,c} = X \upharpoonright f(c)$. Since $|\sigma_{i,c}| = f(c) - c$, it follows that $C_M(X \upharpoonright f(c)) = f(c) - c$.

It remains to be checked that M is indeed a primitive recursive function. Observe that since $f(c+1) - c - 1 > f(c) - c > 0$ for all $c \in \mathbb{N}$, we have by induction that $f(c) - c > c$ for all c. Thus, given a string σ of length m, we need only check $c < m$ to see whether $m = f(c) - c$ for some c. This can be done primitive recursively. That is, if $m = f(c) - c$, then it is a bounded search to determine whether $\sigma = \sigma_{i,c}$ where $M(\sigma_{i,c}) = \tau_{i,c}$ or not. If not, or if $m \neq f(c) - c$ for any c, then we just let $M(\sigma) = 0$.

Hence, X is not Kolmogorov BP random.

(1) implies (3): Suppose that X is not martingale BP random. Then there is a primitive recursive martingale d which succeeds primitive recursively on X so that there is a primitive recursive function f such that, for all n, $d(X \upharpoonright f(n)) \geq 2^n$. Let $G_n = \{\tau \in \{0,1\}^{f(n)} : d(\tau) \geq 2^n\}$ and let $U_n = \bigcup_{\tau \in G_n} [\tau]$. Since d and f are primitive recursive, it follows that the sequence $(U_n)_{n \geq 0}$ is a primitive recursive sequence of clopen sets. Certainly $X \in \bigcap_n U_n$.

Recall that for all martingales with $d(\emptyset) = 1$, $\sum_{|\tau|=m} d(\tau) \leq 2^m$. It follows that there are at most $2^{f(n)-n}$ strings $\tau \in \{0,1\}^{f(n)}$ such that $d(\tau) \geq 2^n$. For each such τ, $\mu([\tau]) = 2^{-f(n)}$. Thus $\mu(U_n) \leq 2^{f(n)-n} \cdot 2^{-f(n)} = 2^{-n}$.

Hence $(U_n)_{n \geq 0}$ is a primitive recursive test so that X is not Martin-Löf BP random.

(3) implies (1): Suppose X is not Martin-Löf BP random. Then $X \in \bigcap_n U_n$ where $(U_n)_{n \geq 0}$ is a weak primitive recursive test. We may assume that there are primitive recursive functions ℓ and k such that $U_n = [\tau_{1,n}] \cup \cdots \cup [\tau_{k(n),n}]$ where $|\tau_{i,n}| = \ell(n)$. Let $G_n = \{\tau_{1,n}, \ldots, \tau_{k(n),n}\}$.

We recursively define our martingale d as follows. For $n = 1$, we let $d(\tau_{i,1}) = \frac{2^{\ell(1)}}{k(1)}$ for $i = 1, \ldots, k(1)$. If $\tau \in \{0,1\}^{\ell(1)} - \{\tau_{1,1}, \ldots, \tau_{k(1),1}\}$, then we let $d(\tau) = 0$. Since $\mu(U_1) \leq \frac{1}{2}$, it follows that $k(1) \leq 2^{\ell(1)-1}$ and therefore $d(\tau_{i,1}) \geq 2$ for each i. Moreover, $\sum_{\tau \in \{0,1\}^{\ell(1)}} d(\tau) = k(1) \cdot \frac{2^{\ell(1)}}{k(1)} = 2^{\ell(1)}$. Now work backwards using the martingale equation $d(\sigma) = \frac{1}{2}(d(\sigma^\frown 0) + d(\sigma^\frown 1))$ to define $d(\sigma)$ for all σ of length $\leq f(1)$. It follows by induction that for all $j \leq f(1)$, $\sum_{\tau \in \{0,1\}^j} d(\tau) = 2^j$ so that, in particular, $d(\emptyset) = 1$.

Now suppose that we have defined $d(\tau)$ for all τ with $|\tau| \leq \ell(n)$ so that $d(\tau) \geq 2^n$ for all $\tau \in G_n$. Then we will show how to extend d to strings of length $\leq \ell(n+1)$. For σ of length $\ell(n)$, we will define $d(\sigma\tau)$, where $\sigma\tau = \sigma^\frown\tau$, for all τ of length $\ell(n+1) - \ell(n)$. If $d(\sigma) = 0$, then we simply let $d(\sigma\tau) = 0$ for all τ. Now fix $\sigma \in G_n$ with $d(\sigma) \geq 2^n$ and consider $G = \{\tau : \sigma\tau \in G_{n+1}\}$. Since we have begun with a weak test, it follows that $\mu([G]) \leq \frac{1}{2}$. Thus we may proceed as in the first case where $n = 1$ to define a martingale m such that $m(\sigma) = 1$ and $m(\tau) \geq 2$ for all $\tau \in G$. Now extend the definition of d to the strings below σ by defining $d(\sigma\tau) = d(\sigma) \cdot m(\tau)$. Since $d(\sigma) \geq 2^n$ and, for $\tau \in G$, $m(\tau) \geq 2$, it follows that for $\sigma\tau \in G_{n+1}$, $d(\sigma\tau) \geq 2^{n+1}$. It is easy to see that this extension obeys the martingale equality, since, for any τ,

$$d(\sigma\tau) = d(\sigma) \cdot m(\tau) = d(\sigma) \cdot \frac{1}{2}(m(\tau^\frown 0) + m(\tau^\frown 1)) = \frac{1}{2} \cdot (d(\sigma\tau^\frown 0) + d(\sigma\tau^\frown 1)).$$

Since $X \in \bigcap_n U_n$, it follows that $d(X \upharpoonright \ell(n)) \geq 2^n$ for each n and hence d succeeds primitive recursively on X.

It is clear that from a given string σ, this defines a primitive recursive procedure to compute $d(\sigma)$. The first step is to compute $\ell(n)$ for $n \leq |\sigma|$ until we find n so that $|\sigma| \leq \ell(n)$. Then we consider all extensions τ of σ of length $\ell(n)$. We can follow the procedure outlined above to compute $d(\sigma \upharpoonright \ell(i))$ for $i \leq n$, and, hence, compute $d(\tau)$ for all extensions τ of σ of length $\ell(n)$. Finally we backtrack using the martingale inequality to compute $d(\sigma)$ from the values of such $d(\tau)$. Thus d is a primitive recursive martingale so that X is not martingale BP random. □

We should note one could alternatively prove Theorem 1 by modifying proofs that have already appeared in the literature. For example Downey, Griffiths and Reid [14] proved the equivalence of parts (1) and (3) in the setting of Kurtz randomness and their proof can be modified to give a proof of the equivalence of parts (1) and (3) in Theorem 1. Similarly, Bienvenu and Merkle [6] gave a proof the equivalence of parts (1) and (2) in the setting of Kurtz randomness and their proof can easily be modified to give a proof of the equivalence of parts (1) and (2) in Theorem 1.

Given Theorem 1, we define an $X \in \{0,1\}^\omega$ to be *BP random* if and only if X is Martin-Löf BP random. Since every BP test is also a Kurtz test and a computable test, it follows that all Kurtz random and all computably random reals are BP random.

It is clear that no primitive recursive set can be BP random. It was shown by Jockusch [16] that Kurtz random sets are *immune*, that is, they do not include any c.e. subsets. Here is a version of that result for BP randomness.

Proposition 5. *If A is BP random, then for any increasing primitive recursive function f, A does not contain the range of f.*

Proof. Suppose for the contrapositive that A contains the range of f. For each n, let $G_n = \{\sigma \in \{0,1\}^{f(n)} : (\forall i < n)(\sigma(f(i)) = 1)\}$. Then let $U_n = \bigcup_{\tau \in G_n} [\tau]$. It is clear that $\mu([U_n]) = 2^{-n}$ so that $(U_n)_{n \geq 0}$ is a primitive recursive test. But then A belongs to each U_n so that A is not BP random.

Theorem 2. *There is a recursive real which is BP random.*

Proof. Let $(g_e, \ell_e)_{e \geq 0}$ enumerate all pairs of primitive recursive functions. For any e, let $G_{n,e} = \{\sigma_{1,n}, \ldots, \sigma_{k_e(n),n}\}$ be the finite set whose code is $g_e(n)$ and $U_{n,e} = [G_{e,n}]$.

Our goal is to construct an increasing recursive functions r and t and recursive sequence $X = (X(0), X(1), \ldots)$ such that for all s, either
(I) it is not the case that for all $1 \leq i \leq r(e)$, $\mu(U_{i,e}) \leq 2^{-i}$, $|\sigma| = \ell_e(i)$ for all $\sigma \in G_{i,e}$, and $U_{0,e} \supseteq U_{1,e} \supseteq \cdots \supseteq U_{r(e),e}$ or
(II) **(I)** fails and $(X(0), \ldots, X(t(e)-1)) \notin U_{r(e),e}$.
That is, either g_e and ℓ_e do not specify a primitive recursive test of the proper form or $(X(0), \ldots, X(s(e)-1)) \notin U_{r(e),e}$. This will ensure that X is a recursive real which is not BP random.

We construct r, t, and X in stages as follows.

Stage 0. Compute $\ell_0(1)$, $G_{1,0}$ and $U_{1,0}$. If it is not the case, that $\mu(U_{1,0}) \leq \frac{1}{2}$ and $|\sigma| = \ell_0(1)$ for all $\sigma \in G_{1,0}$, then set $r(0) = s(0) = 1$ and $X(0) = 0$. Otherwise let σ be the lexicographically least string τ of length $\ell_0(1)$ such that $\tau \notin U_{1,0}$. Note that σ must exists since $\mu(U_{1,0}) \leq \frac{1}{2}$. Then set $r(0) = 1$, $(X(0), \ldots, X(\ell_0(1)-1)) = \sigma$, and $t(0) = \ell_0(1)$.

Stage s+1. Assume that we have defined $r(0) < \cdots < r(s)$, $t(0) < \cdots < t(s)$, and $(X(0), \ldots, X(t(s)-1))$ such that either **(I)** or **(II)** hold for $e \leq s$.

Then let $r(s+1) = t(s) + 2$. If it is the case that for all $1 \leq i \leq r(s+1)$, $\mu(U_{i,s+1}) \leq 2^{-i}$, $|\sigma| = \ell_{s+1}(i)$ for all $\sigma \in G_{i,s+1}$, and $U_{0,s+1} \supseteq U_{1,s+1} \supseteq \cdots \supseteq U_{r(s+1),s+1}$, then we know that $\mu(U_{r(s+1),s+1}) \leq 2^{-(t(s)+2)}$ and $\ell_{s+1}(r(s+1)) \geq t(s)+2$. Thus there must exist an extension τ of $(X(0), \ldots, X(t(s)-1))$ of length $\ell_{s+1}(r(s+1))$ such that $\tau \notin U_{r(s+1),s+1}$. Then we let $t(s+1) = \ell_{s+1}(r(s+1))$ and $(X(0), \ldots, X(t(s+1)-1))$ be lexicographically least such τ. Otherwise, we let $t(s+1) = t(s) + 1$ and $X(t(s+1)-1) = 0$.

It is easy to see that our construction is completely effective so that $X = (X(0), X(1), \ldots)$ will be a computable real which is not BP random.

Next we show that BP random reals satisfy the following analogue of Ville's Theorem.

Theorem 3. *Let $X \in \{0,1\}^n$ be BP random and let g be a primitive recursive increasing function. Then the sequence $(X(g(0)), X(g(1)), X(g(2)), \ldots)$ is also BP random.*

Proof. Let $Y(n) = X(g(n))$ and suppose by way of contradiction that Y is not BP random. Let $(U_n)_{n\geq 0}$ be a primitive recursive test such that $Y \in U_n$ for all n. That is, suppose that there are primitive recursive functions a, b, and c such that

1. $U_{n+1} \subseteq U_n$ for all n,
2. $\mu(U_n)$, is $\leq 2^{-n}$ for all n, and
3. for all n, $a(n)$ is the code of a finite set $G_n = \{\sigma_{1,n}, \ldots, \sigma_{b(n),n}\}$ of strings such that $U_n = [G_n]$ and $|\sigma_{i,n}| = c(n)$ for all $1 \leq i \leq b(n)$.

For any string $\tau_{i,n} = \tau_1 \ldots \tau_{c(n)}$ in G_n, let $\tau_{i,n}^{(1)}, \ldots, \tau_{i,n}^{(2^{g(c(n))-c(n)})}$ be a list of the $2^{g(c(n))-c(n)}$ strings of length $g(c(n))$ such that $(\tau^{(i)}_{g(1)}\tau^{(i)}_{g(2)} \cdots \tau^{(i)}_{g(c(n))}) = \tau_{i,n}$. Then define

$$V_n = \{X : (X(g(1), \ldots, X(g(c(n)))\} \in U_n\} = \bigcup_{i=1}^{b(n)} \bigcup_{j=1}^{2^{g(c(n))-c(n)}} [\tau_{i,n}^{(j)}].$$

It is easy to see that $\mu(U_n) = \mu(V_n)$ and the $(V_n)_{n\geq 1}$ is a primitive recursive test. But then $X \in \bigcap_{n\geq 1} V_n$ which would violate the fact that X is BP random. Thus Y must be BP random.

2.1 Statistical Tests

It is important to see to what extent the BP random sets are statistically random. We begin with a positive result.

Theorem 4. *Let A be a BP random set. For any increasing primitive recursive function f and any $\epsilon > 0$, there is some n such that $|\frac{card(A\cap[[f(n)]])}{f(n)} - \frac{1}{2}| \leq \epsilon$.*

Proof. This follows from the law of large numbers (Chernoff's Lemma [21], p.61).

Corollary 1. *For any BP random set A, if $\lim_n \frac{card(A\cap[[n]])}{n}$ exists, then it equals $\frac{1}{2}$.*

On the other hand, BP random sets do not have to be stochastic.

Theorem 5. *There exists a computable BP random set A such that $lim_n \frac{card(A\cap[[n]])}{n}$ does not exist.*

Proof. To construct such a set A, just modify the proof of Theorem 2 by adding long strings of 0's and long strings of 1's (in alternation) after satisfying each requirement. Then we can make the density go below $\frac{1}{3}$ and then above $\frac{2}{3}$ infinitely often.

2.2 Relative Randomness

Primitive recursive functions may be defined with inputs from $\mathbb{N}$ and also from $\mathbb{N}^{\mathbb{N}}$ by simply adding the basic evaluation functional Ev where $Ev(n, X) = X(n)$ to the basic constant, successor, and projection functions and closing under composition and primitive recursion. If X is a fixed subset of $\mathbb{N}$, then we obtain the X-primitive recursive functions using the characteristic function of X as a fixed oracle.

Relative Martin-Löf BP Randomness

We define a primitive recursive oracle test to consists of a primitive recursive function $g : \mathbb{N} \times \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ and a primitive recursive function f such that for all $Z \in \mathbb{N}$, $g(n, Z) = g_n^Z$ is the code of a finite sequences of strings $G_n^Z = \{\sigma_{1,n}^Z, \ldots, \sigma_{k^Z(n),n}^Z\} \subseteq \{0,1\}^{f(n)}$ such that if $U_n^Z = [G_n^Z]$, then for all n, $\mu(U_n^Z) \leq 2^{-n}$. Thus a primitive recursive oracle test must uniformly give a Z-primitive recursive test for all $Z \subseteq \mathbb{N}$. Then we say that X is *Martin-Löf BP random relative to* $Y \subseteq \mathbb{N}$ if, for any primitive recursive oracle test G, $X \notin [G_n^Y]$ for some n.

Relative Kolmogorov BP Random

Let $M : \{0,1\}^* \times \mathbb{N}^{\mathbb{N}} \to \{0,1\}^*$ be a primitive recursive oracle function. Then for any $Y \subseteq \mathbb{N}$, we let $C_M^Y(\tau)$ be the length $|\sigma|$ of the shortest string σ such that $M^Y(\sigma) = \tau$, i.e., the length of the shortest M^Y-description of τ.

An infinite sequence X is *Kolmogorov BP random relative to* Y if it cannot be primitive recursively compressed by any function M^Y which is primitive recursive in Y. Here we say that X is primitive recursively compressed relative to Y if there exist a primitive recursive oracle function $M : \{0,1\}^* \times \mathbb{N}^{\mathbb{N}} \to \{0,1\}^*$ and a primitive recursive function f such that, for every $c \in N$, $C_M^Y(X \upharpoonright f(c)) \leq f(c) - c$. Notice that f is still primitive recursive although the function M^Y is only primitive recursive in Y.

Relative Martingale BP Random

Let $D : \{0,1\}^* \times \mathbb{N}^{\mathbb{N}} \to \mathbb{Q} \cap [0, \infty]$ be a primitive recursive oracle function. We say that D is a primitive recursive oracle martingale if the function $D^Z : \{0,1\}^* \to \mathbb{Q} \cap [0, \infty]$ is a martingale for all $Z \subseteq \mathbb{N}$ where for any $\sigma \in \{0,1\}^*$, $D^Z(\sigma) = D(\sigma, Z)$. Then X is *BP random relative to* Y if there is no primitive recursive oracle martingale D such that D^Y succeeds primitive recursively on X.

It is easy to see that we can simply relativize the proof of Theorem 1 to prove that X is Kolmogorov BP random relative to Y if and only if X is Martin-Löf BP random relative to Y if and only if X is martingale BP random relative to Y.

With these definitions, we can modify the proof of van Lambalgen's Theorem to prove the following theorem. Recall that if $A, B \subseteq \mathbb{N}$, then $A \oplus B = \{2x : x \in A\} \cup \{2x + 1 : x \in B\}$.

Theorem 6. *For any sets $A, B \subseteq \mathbb{N}$, $A \oplus B$ is BP random if and only if B is BP random relative to A and A is BP random.*

3 Polynomial-Space Bounded Pseudorandomness

In this section, we shall briefly outline how one might modify the definitons of BP random to define a notion of random relative to $PSPACE$ functions. In particular, we shall define two possible analogues of BP random reals for $PSPACE$. Let $PSPACE^*$ be the family of functions computable in polynomial space where we include the space needed to write the output. Thus the output of a $PSPACE^*$ function has length which is a polynomial of the length of the input.

To obtain a robust analogue of primitive recursive randomness, we specified primitive recursive bounding functions f which, for example, specified that $d(X \upharpoonright f(n)) \geq 2^n$ for a primitive recursive martingale to succeed. The key to versions of $PSPACE$ random reals is that we require similar bounding functions which are polynomial time functions $f : \{1\}^* \to \{1\}^*$. That is, we have the following notions of $PSPACE$ BP random reals.

Martin-Löf BPS Random
A $PSPACE$ test $(U_n)_{n \geq 0}$ is specified by a pair of functions (G, f) such that $G : \{1\}^* \times \{0,1\}^* \to \{0,1\}$ is a $PSPACE$-function and $f : \{1\}^* \to \{1\}^*$ is a strictly length increasing $PTIME$ function such that for each n,

$$G_{n,f} = \{\tau \in \{0,1\}^{\leq |f(1^n)|} : G(1^n, \tau) = 1\} = \{\sigma_{1,n}, \ldots, \sigma_{k(n),n}\}$$

is a set of strings of length $\leq |f(1^n)|$ such that $U_n = [\sigma_{1,n}] \cup \cdots \cup [\sigma_{k(n),n}]$ is a clopen set with measure $\leq 2^{-n}$.

A *weak* $PSPACE$ test $(U_n)_{n \geq 0}$ is specified by a pair (G, f) as above with the additional property that for each n, $\mu(U_{n+1} \cap [\sigma_{i,n}]) \leq \frac{1}{2}\mu([\sigma_{i,n}])$.

We say that X is *Martin-Löf BPS random* if X passes every $PSPACE$ test and is *weakly Martin-Löf BPS random* if X passes every weak $PSPACE$ test.

Kolmogorov BPS Random
An infinite sequence X is *Kolmogorov BPS random* if there do not exist a $PSPACE^*$ function $M : \{0,1\}^* \to \{0,1\}^*$ and a $PTIME$ function $f : \{1\}^* \to \{1\}^*$ such that, for every $n \in \mathbb{N}$, $C_M(X \upharpoonright |f(1^n)|) \leq |f(1^n)| - n$.

Martingale BPS Random
A $PSPACE^*$ martingale $d : \{0,1\} \to \mathbb{Q} \cap [0, \infty]$ succeeds on X if there is a $PTIME$ function $f : \{1\}^* \to \{1\}^*$ such that, for all n, there is some $m \leq |f(1^n)|$ such that $d(X \upharpoonright m) \geq 2^n$. Here we shall think of $\mathbb{Q} \cap [0, \infty]$ as the set of all strings $\sigma 2 \tau$ where $\sigma.\tau$ is the binary expansion of a rational number $r \in \mathbb{Q} \cap [0, \infty]$. We say that X is *martingale BPS random* if no $PSPACE^*$ martingale succeeds on X.

By suitably modifying the proof of Theorem [1], we can prove the following.

Theorem 7. *For any $X \in \{0,1\}^\omega$,*

(1) X is Kolmogorov BPS random if and only if X is Martin-Löf BPS random and

(2) X is martingale BPS random if and only if X is weakly Martin-Löf BPS random.

By modifying the proof of Theorem 2, we can prove the following.

Theorem 8. *There is a $DSPACE(2^{2^n})$ real which is Martin-Löf BPS random.*

4 Conclusions and Future Work

In this paper, we defined a robust notion of primitive recursive and $PSPACE$ bounded pseudorandom reals in that each definition could be framed in at least two of the three versions of algorithmically random reals via measure, Kolmogorov complexity, or martingales. We view the work of this paper as a possible model for defining algorithmically random reals relative to several other classes of sub-computable functions. In future work, we will define similar notions of bounded pseudorandom reals for other classes of sub-computable functions such as elementary, on-line, or $EXPSPACE$.

A theory of algorithmic randomness for trees and effectively closed sets was developed in a series of papers by Barmpalias, Cenzer, Remmel et al [4,5]. One can adapt our definitions of primitive recursive bounded pseudorandomness to define similar notions of bounded pseudorandom trees and effectively closed sets for various classes of sub-computable functions. This will appear in future papers.

References

1. Allender, E., Strauss, M.: Measure on small complexity classes with applications for BPP. In: Proceedings of the 35th Symposium on Foundations of Computer Science, pp. 807–818. IEEE Computer Society (1994)
2. Allender, E., Strauss, M.: Measure on P: Robustness of the Notion. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969, pp. 129–138. Springer, Heidelberg (1995)
3. Ambos-Spies, K., Mayordomo, E.: Resource-bounded measure and randomness. In: Sorbi, A. (ed.) Complexity, Logic and Recursion Theory. Lecture Notes in Pure and Applied Mathematics, pp. 1–47. Marcel Dekker, New York (1997)
4. Barmpalias, G., Brodhead, P., Cenzer, D., Dashti, S., Weber, R.: Algorithmic randomness of closed sets. J. Logic and Computation 17, 1041–1062 (2007)
5. Barmpalias, G., Brodhead, P., Cenzer, D., Remmel, J.B., Weber, R.: Algorithmic Randomness of Continuous Functions. Archive for Mathematical Logic 46, 533–546 (2008)
6. Bienvenu, L., Merkle, W.: Reconciling Data Compression and Kolmogorov Complexity. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 643–654. Springer, Heidelberg (2007)
7. Blum, M., Micali, S.: How to Generate Cryptographically Strong Sequences of Pseudorandom Bits. Siam J. Computing 13, 850–864 (1984)
8. Brodhead, P., Downey, R., Ng, K.M.: Bounded Randomness. In: Dinneen, M.J., Khoussainov, B., Nies, A. (eds.) WTCS 2012 (Calude Festschrift). LNCS, vol. 7160, pp. 59–70. Springer, Heidelberg (2012)

9. Buhrman, H., Longpre, L.: Compressibility and resource bounded measure. SIAM Journal on Computing 31(3), 876–886 (2002)
10. Chaitin, G.: On the length of programs for computing finite binary sequences. J. Assoc. Comp. Mach. 13, 547–569 (1966)
11. Chernov, A., Shen, A., Vereshchagin, N., Vovk, V.: On-Line Probability, Complexity and Randomness. In: Freund, Y., Györfi, L., Turán, G., Zeugmann, T. (eds.) ALT 2008. LNCS (LNAI), vol. 5254, pp. 138–153. Springer, Heidelberg (2008)
12. Church, A.: On the concept of random sequences. Bull. Amer. Math. Soc. 46, 130–135 (1940)
13. Di Paola, R.: Random sets in subrecursive hierarchies. J. Assoc. Comp. Mach. 16, 621–630 (1969)
14. Downey, R.G., Griffiths, E.J., Reid, S.: On Kurtz randomness. Theoretical Computer Science 321, 249–270 (2004)
15. Downey, R., Hirschfeldt, D.: Algorithmic Randomness and Complexity. Springer (2011)
16. Kautz, S.: Degrees of Random Sets. Ph.D. Thesis, Cornell University (1991)
17. Ko, K.: On the notion of infinite pseudorandom sequences. Theoretical Computer Science 48, 9–33 (1986)
18. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems of Information Transmission 1, 1–7 (1965)
19. Kurtz, S.: Randomness and Genericity in the Degrees of Unsolvability. Ph.D. Thesis, University of Illinois at Urbana (1981)
20. Levin, L.: On the notion of a random sequence. Soviet Math. Doklady 14, 1413–1416 (1973)
21. Li, M., Vitanyi, P.: An introduction to Kolmogorov Complexity and Its Applications, 3rd edn. Springer (2008)
22. Lutz, J.H.: Category and measure in complexity classes. SIAM Journal on Computing 19, 1100–1131 (1990)
23. Miyabe, K.: Truth-table Schnorr randomness and truth-table reducible randomness. Math. Logic Quarterly 57, 323–338 (2011)
24. Martin-Löf, P.: The definition of random sequences. Information and Control 9, 602–619 (1966)
25. Nies, A.: Computability and Randomness. Oxford University Press (2009)
26. Schnorr, C.P.: A unified approach to the definition of random sequences. Mathematical Systems Theory 5, 246–258 (1971)
27. Ville, J.: Étude Critique de la Notion de Collectif. Gauthier-Villars, Paris (1939)
28. Wang, Y.: Resource bounded randomness and computational complexity. Theoretical Computer Science 237, 33–55 (2000)
29. Wilber, R.: Randomness and the density of hard problems. In: Proc. 24th IEEE Symposium on Foundations of Computer Science, pp. 335–342 (1983)

Automated Support for the Investigation of Paraconsistent and Other Logics[⋆]

Agata Ciabattoni[1], Ori Lahav[2], Lara Spendier[1], and Anna Zamansky[1]

[1] Vienna University of Technology
[2] Tel Aviv University

Abstract. We automate the construction of analytic sequent calculi and effective semantics for a large class of logics formulated as Hilbert calculi. Our method applies to infinitely many logics, which include the family of paraconsistent C-systems, as well as to other logics for which neither analytic calculi nor suitable semantics have so far been available.

1 Introduction

Non-classical logics are often introduced using Hilbert systems. Intuitionistic, modal and paraconsistent logics are just a few cases in point. The usefulness of such logics, however, strongly depends on two essential components. The first is an intuitive *semantics*, which can provide insights into the logic. A desirable property of such semantics is *effectiveness*, in the sense that it naturally induces a decision procedure for the logic. Examples of such semantics include finite-valued matrices, and their generalizations: non-deterministic finite-valued matrices (*Nmatrices*) and partial Nmatrices (*PNmatrices*) (see [5,6]). The second component is a corresponding *analytic calculus*, i.e. a calculus whose proofs only consist of concepts already contained in the result. Analytic calculi are useful for establishing various properties of the corresponding logics, and are also the key for developing automated reasoning methods for them.

In this paper we provide both methodologies and practical tools for an *automatic generation* of analytic sequent calculi and effective semantics for a large class **H** of Hilbert systems. This is a concrete step towards a systematization of the vast variety of existing non-classical logics and the developement of tools for designing new application-oriented logics, see e.g. [11].

The calculi in **H** are obtained (i) by extending the language of CL$^+$, the positive fragment of classical logic, to a language $\mathcal{L}_\mathcal{U}$ which includes also a finite set $\mathcal{U}$ of unary connectives, and (ii) by adding to a Hilbert axiomatization HCL^+ of CL$^+$ axioms over $\mathcal{L}_\mathcal{U}$ of a certain general form. **H** contains infinitely many systems, which include well-known Hilbert calculi, the simplest and best known of which is the standard calculus for classical logic, obtained by adding to HCL^+ the usual axioms for negation. Another example of calculi in **H** is the family of *paraconsistent logics* known as C-systems [8,10].

⋆ Work supported by the Israel Science Foundation (grant no. 280-10), the EC Marie Curie Fellowship (NDMANCL-252314), and FWF START Y544-N23.

Given a system $H \in \mathbf{H}$, our algorithm proceeds in two steps. First we introduce a sequent calculus G equivalent to H. This is done by suitably adapting the procedure in [9], where certain Hilbert axioms are transformed into equivalent (sequent and hypersequent) structural rules. In contrast to [9], however, here the rules extracted from the axioms of H are logical rules in Gentzen's terminology, that is they introduce logical connectives. The analyticity of the resulting calculus depends on the interaction between these rules. This is not anymore a local check and needs instead a *"global view"* on the obtained calculus, which is provided by the semantics constructed in the second step. This semantics is given in the framework of PNmatrices – a generalization of usual many-valued matrices in which each entry in the truth-tables of the logical connectives consists of a *possibly empty set* of options (see [6]). This framework allows non-deterministic semantics, and also, using empty sets of options makes it possible to forbid some combinations of truth values. However, it is still effective, as it guarantees the decidability of the corresponding sequent calculus. As a corollary it follows that each system $H \in \mathbf{H}$ is decidable. Furthermore, we show that the PNmatrix constructed for H is an Nmatrix (i.e., it has no empty sets in the truth-tables) iff G enjoys a certain generalized analyticity property.

Related Work: A semi-automated procedure to define semantics and analytic calculi for the family of C-systems was introduced in [4]. A corresponding Nmatrix was constructed there for each system in the family, and was then used for introducing a corresponding analytic sequent calculus. However, the construction of Nmatrices out of the Hilbert calculi is done manually, and it requires some ingenuity. In this paper we provide a full automation of the generation of effective semantics and analytic calculi for all the systems considered in [4], which have finite-valued semantics. Our method also applies to infinitely many other extensions of CL$^+$, which had so far no available semantics or adequate calculi. These include some logics defined in [1], finding semantics for which was left as an open problem. It should be noted that our algorithm reverses the steps taken in [4]: it first extracts suitable sequent rules from the axioms of $\mathbf{H}$, and uses them to "read off" the semantics.

Implementation: Our method is implemented in the Prolog system *Paralyzer*, available at `www.logic.at/people/lara/paralyzer.html`. For any set of axioms over $\mathcal{L}_{\mathcal{U}}$ of a certain general form *Paralyzer* (PARAconsistent (and other) logics anaLYZER) outputs: (a) a set of corresponding sequent rules, and (b) the associated PNmatrix. The user can choose whether to start as basic system with HCL^+ or with the system BK from [4], obtained by augmenting HCL^+ with the axioms ($\mathbf{n_1}$), ($\mathbf{b}$) and ($\mathbf{k}$) (cf. Fig. 1). In the latter case, by exploiting the invertibility of the sequent rules for $\circ$, (a) and (b) for the C-systems having finite-valued semantics coincide with the results in [4].

2 Step 1: From Hilbert Systems to Sequent Calculi

The first step of our method consists of a mapping from a family $\mathbf{H}$ of Hilbert systems into a family $\mathbf{G}$ of "well-behaved" sequent calculi.

2.1 The Family H

In what follows, $\mathcal{L}$ denotes a propositional language, and $wff_{\mathcal{L}}$ is its set of formulas. We assume that the atomic formulas of $\mathcal{L}$ are $\{p_1, p_2, \ldots\}$. $\mathcal{L}^+_{cl}$ is the language of CL^+, the positive fragment of (propositional) classical logic, consisting of the binary connectives $\wedge, \vee$ and $\supset$. We consider languages that extend $\mathcal{L}^+_{cl}$ with finitely many new unary connectives (such as $\neg$ and $\circ$). Henceforth $\mathcal{U}$ denotes an arbitrary finite set of unary connectives, and $\mathcal{L}_{\mathcal{U}}$ denotes the extension of $\mathcal{L}^+_{cl}$ with the connectives of $\mathcal{U}$. For a Hilbert system H, we write $\Gamma \vdash_H \varphi$ if φ is provable in H from a finite set Γ of formulas. HCL^+ denotes any Hilbert calculus for $\mathcal{L}^+_{cl}$, which is sound and complete for CL^+. **H** is a family of axiomatic extensions of HCL^+, each of which is in the language $\mathcal{L}_{\mathcal{U}}$ for some $\mathcal{U}$. These systems are obtained by augmenting HCL^+ with axioms[1] of the form defined below.

Definition 1. *Let $\mathcal{U} = \{\star_1, \ldots, \star_n\}$. $\mathbf{Ax}_{\mathcal{U}}$ is the set of $\mathcal{L}_{\mathcal{U}}$-formulas generated by the following grammar (where S is the initial variable):*

$$\begin{array}{ll}
S = R_p \mid R_1 \mid R_2 & P_1 = (P_1 \diamond P_1) \mid \star p_1 \mid p_1 \mid p_2 \mid \ldots \\
R_p = (R_p \diamond P_1) \mid (P_1 \diamond R_p) \mid \star p_1 & P_2 = (P_2 \diamond P_2) \mid \star p_1 \mid \star p_2 \mid p_1 \mid p_2 \mid p_3 \mid \ldots \\
R_1 = (R_1 \diamond P_1) \mid (P_1 \diamond R_1) \mid \star\star p_1 & \diamond = \wedge, \vee, \supset \\
R_2 = (R_2 \diamond P_2) \mid (P_2 \diamond R_2) \mid \star(p_1 \diamond p_2) & \star = \star_1 \mid \ldots \mid \star_n
\end{array}$$

N :	$(\mathbf{n_1})$	$p_1 \vee \neg p_1$	$(\mathbf{n_2})$	$p_1 \supset (\neg p_1 \supset p_2)$
	$(\mathbf{c})$	$\neg\neg p_1 \supset p_1$	$(\mathbf{e})$	$p_1 \supset \neg\neg p_1$
	$(\mathbf{n^l_\wedge})$	$\neg(p_1 \wedge p_2) \supset (\neg p_1 \vee \neg p_2)$	$(\mathbf{n^r_\wedge})$	$(\neg p_1 \vee \neg p_2) \supset \neg(p_1 \wedge p_2)$
	$(\mathbf{n^l_\vee})$	$\neg(p_1 \vee p_2) \supset (\neg p_1 \wedge \neg p_2)$	$(\mathbf{n^r_\vee})$	$(\neg p_1 \wedge \neg p_2) \supset \neg(p_1 \vee p_2)$
	$(\mathbf{n^l_\supset})$	$\neg(p_1 \supset p_2) \supset (p_1 \wedge \neg p_2)$	$(\mathbf{n^r_\supset})$	$(p_1 \wedge \neg p_2) \supset \neg(p_1 \supset p_2)$
C :	$(\mathbf{b})$	$p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2))$	$(\mathbf{r}_\diamond)$	$\circ(p_1 \diamond p_2) \supset (\circ p_1 \vee \circ p_2)$
	$(\mathbf{k})$	$\circ p_1 \vee (p_1 \wedge \neg p_1)$	$(\mathbf{i})$	$\neg\circ p_1 \supset (p_1 \wedge \neg p_1)$
	$(\mathbf{o^1_\diamond})$	$\circ p_1 \supset \circ(p_1 \diamond p_2)$	$(\mathbf{o^2_\diamond})$	$\circ p_2 \supset \circ(p_1 \diamond p_2)$
	$(\mathbf{a}_\diamond)$	$(\circ p_1 \wedge \circ p_2) \supset \circ(p_1 \diamond p_2)$	$(\mathbf{a}_\neg)$	$\circ p_1 \supset \circ\neg p_1$

Fig. 1. Examples of formulas in $\mathbf{Ax}_{\{\neg,\circ\}}$ ($\diamond \in \{\vee, \wedge, \supset\}$)

Definition 2. *A Hilbert calculus H for a language $\mathcal{L}_{\mathcal{U}}$ is called a $\mathcal{U}$-*extension *of HCL^+ if it is obtained by augmenting HCL^+ with a finite set of axioms from $\mathbf{Ax}_{\mathcal{U}}$. We denote by **H** the family of all $\mathcal{U}$-extensions of HCL^+ for some $\mathcal{U}$.*

The family **H** contains infinitely many systems, which include many well-known Hilbert calculi. The most important member of **H** is the standard calculus for (propositional) classical logic, obtained by adding $(\mathbf{n_1})$ and $(\mathbf{n_2})$ to HCL^+ (cf. Fig. 1). Other important examples include various systems for paraconsistent logics [4,7,8,10].

Remark 1. Paraconsistent logics are logics which are tolerant of inconsistent theories, i.e. there are some formulas ψ, φ, such that: $\psi, \neg\psi \nvdash \varphi$. One well-known

[1] By *axioms* we actually mean *axiom schemata.*

family of paraconsistent logics, formulated in terms of Hilbert calculi, is known as C-systems [4,7,8,10]. In this family the notion of consistency is internalized into the object language by employing a unary consistency operator $\circ$, the intuitive meaning of $\circ\psi$ being "ψ is consistent". Clearly, a system which includes the standard axiom for negation ($\mathbf{n_2}$) (Fig. 1) cannot induce a paraconsistent logic. Many C-systems include instead the weaker axiom ($\mathbf{b}$), and in addition also the axiom ($\mathbf{n_1}$). Furthermore, different C-systems employ different subsets of the axioms from the set $\mathbf{C}$ (Fig. 1), which express various properties of the operator $\circ$. For instance, axiom ($\mathbf{a}_\vee$) says that the consistency of two formulas implies the consistency of their disjunction. The axiom ($\mathbf{o}^{\mathbf{1}}_\vee$) expresses another form of consistency propagation: the consistency of a formula implies the consistency of its disjunction with any other formula. By adding to HCL^+ various combinations of axioms from Fig. 1, we obtain a wider family of systems (not all of them paraconsistent), many of which are studied in [2,4].

2.2 The Family G

The sequent calculi we will consider, formulated label-style, are as follows:

Definition 3. *1. A* labelled $\mathcal{L}$-formula *has the form $b : \psi$, where $b \in \{f, t\}$ and $\psi \in wff_{\mathcal{L}}$. An* $\mathcal{L}$-sequent *is a finite set of labelled $\mathcal{L}$-formulas. The usual sequent notation $\psi_1, \dots, \psi_n \Rightarrow \varphi_1, \dots, \varphi_m$ is interpreted as the set $\{f : \psi_1, \dots, f : \psi_n, t : \varphi_1, \dots, t : \varphi_m\}$.*

2. An $\mathcal{L}$-substitution *is a function $\sigma : wff_{\mathcal{L}} \to wff_{\mathcal{L}}$, such that $\sigma(\diamond(\psi_1, \dots, \psi_n)) = \diamond(\sigma(\psi_1), \dots, \sigma(\psi_n))$ for every n-ary connective $\diamond$ of $wff_{\mathcal{L}}$. $\mathcal{L}$-substitutions are naturally extended to labelled $\mathcal{L}$-formulas and $\mathcal{L}$-sequents.*

3. An $\mathcal{L}$-rule *is an expression of the form Q/s, where Q is a finite set of $\mathcal{L}$-sequents (called* premises*) and s is an $\mathcal{L}$-sequent (called* conclusion*). An* application *of an $\mathcal{L}$-rule Q/s is any inference step inferring the $\mathcal{L}$-sequent $\sigma(s) \cup c$ from the set of $\mathcal{L}$-sequents $\{\sigma(q) \cup c \mid q \in Q\}$, where σ is an $\mathcal{L}$-substitution, and c is an $\mathcal{L}$-sequent.*

4. A sequent calculus G for $\mathcal{L}$ consists of a finite set of $\mathcal{L}$-rules. We write $\mathcal{S} \vdash_G s$ whenever the $\mathcal{L}$-sequent s is derivable from the set $\mathcal{S}$ of $\mathcal{L}$-sequents in G.

Example 1. Formulated according to Def. 3, the standard sequent calculus LK^+ for CL^+ is the set of $\mathcal{L}^+_{cl}$-rules consisting of the following elements:

(id)	$\emptyset/\{f : p_1, t : p_1\}$	(cut)	$\{\{f : p_1\}, \{t : p_1\}\}/\emptyset$
$(W\Rightarrow)$	$\{\emptyset\}/\{f : p_1\}$	$(\Rightarrow W)$	$\{\emptyset\}/\{t : p_1\}$
$(\wedge \Rightarrow)$	$\{\{f : p_1, f : p_2\}\}/\{f : p_1 \wedge p_2\}$	$(\Rightarrow \wedge)$	$\{\{t : p_1\}, \{t : p_2\}\}/\{t : p_1 \wedge p_2\}$
$(\vee \Rightarrow)$	$\{\{f : p_1\}, \{f : p_2\}\}/\{f : p_1 \vee p_2\}$	$(\Rightarrow \vee)$	$\{\{t : p_1, t : p_2\}\}/\{t : p_1 \vee p_2\}$
$(\supset\Rightarrow)$	$\{\{t : p_1\}, \{f : p_2\}\}/\{f : p_1 \supset p_2\}$	$(\Rightarrow\supset)$	$\{\{f : p_1, t : p_2\}\}/\{t : p_1 \supset p_2\}$

$\mathbf{G}$ is a family of sequent calculi, each of which is in the language $\mathcal{L}_{\mathcal{U}}$ for some $\mathcal{U}$. These calculi are obtained by augmenting LK^+ with *simple* rules:

$(\Rightarrow \neg)$	$\{\{f : p_1\}\}/\{t : \neg p_1\}$	$\dfrac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\varphi, \Delta}$
$(\circ \Rightarrow)$	$\{\{t : p_1\}, \{t : \neg p_1\}\}/\{f : \circ p_1\}$	$\dfrac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma \Rightarrow \neg\varphi, \Delta}{\Gamma, \circ\varphi \Rightarrow \Delta}$
$(\neg\neg \Rightarrow)$	$\{\{f : p_1\}\}/\{f : \neg\neg p_1\}$	$\dfrac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma, \neg\neg\varphi \Rightarrow \Delta}$
$(\Rightarrow \neg\wedge)_1$	$\{\{t : \neg p_1\}\}/\{t : \neg(p_1 \wedge p_2)\}$	$\dfrac{\Gamma \Rightarrow \neg\varphi, \Delta}{\Gamma \Rightarrow \neg(\varphi \wedge \psi), \Delta}$

Fig. 2. Examples of $\mathcal{L}_{\{\neg,\circ\}}$-rules and their applications forms

Definition 4. *A $\mathcal{U}_n$-premise ($n = 1, 2$) is an $\mathcal{L}_{\mathcal{U}}$-sequent of the form $\{b : p_n\}$ or $\{b : \star p_n\}$, where $b \in \{f, t\}$ and $\star \in \mathcal{U}$. An $\mathcal{L}_{\mathcal{U}}$-rule Q/s is ($b \in \{f, t\}$, $\star, \triangleright \in \mathcal{U}$ and $\diamond \in \{\wedge, \vee, \supset\}$):*

– primitive *if $s = \{b : \star p_1\}$ and Q consists only of $\mathcal{U}_1$-premises.*
– onevar *if $s = \{b : \star \triangleright p_1\}$ and Q consists only of $\mathcal{U}_1$-premises.*
– twovar *if $s = \{b : \star(p_1 \diamond p_2)\}$ and Q consists only of $\mathcal{U}_1$-premises and $\mathcal{U}_2$-premises.*
– simple *if it is either a primitive, a onevar or a twovar rule.*

Example 2. $(\Rightarrow \neg)$ is primitive, $(\neg\neg \Rightarrow)$ onevar, and $(\Rightarrow \neg\wedge)_1$ twovar (cf. Fig. 2).

Distinguishing between the types of rules above will be crucial for the semantic definitions of Section 3.2. As we shall see, rules of different types will play different semantic roles: the primitive rules will determine the truth values in the PNmatrices, while the onevar and twovar rules will dictate the truth-tables of the unary and binary connectives respectively.

Definition 5. *A sequent calculus G for $\mathcal{L}_{\mathcal{U}}$ is called a $\mathcal{U}$-*extension of *LK^+ if it is obtained by augmenting LK^+ with a finite set of simple $\mathcal{L}_{\mathcal{U}}$-rules. We denote by* **G** *the family of all $\mathcal{U}$-extensions of LK^+ for some $\mathcal{U}$.*

2.3 Mapping from H to G

Given a Hilbert system $H \in \mathbf{H}$ we show how to construct a sequent calculus $G_H \in \mathbf{G}$ which is equivalent in the following sense:

Definition 6. *A sequent calculus G is* equivalent *to a Hilbert system H if for every finite set $\Gamma \cup \{\varphi\}$ of formulas: $\Gamma \vdash_H \varphi$ iff $\vdash_G \Gamma \Rightarrow \varphi$.*

Fact 1. *LK^+ is equivalent to HCL^+.*

We denote by $H \cup \{\varphi\}$ the Hilbert system obtained from H by adding the axiom φ, and by $G \cup R$ the sequent calculus extending G with the set R of rules.

Definition 7. *Let R and R' be two sets of $\mathcal{L}$-rules, and G be a sequent calculus for $\mathcal{L}$. R and R' are* equivalent *in G if $Q \vdash_{G \cup R'} s$ for every $Q/s \in R$, and $Q \vdash_{G \cup R} s$ for every $Q/s \in R'$.*

Definition 8. *An $\mathcal{L}_{\mathcal{U}}$-rule Q/s is* invertible *in G if $s \vdash_G q$ for every $q \in Q$.*

The key observations for our transformation procedure are: (i) the invertibility of the rules for $\wedge, \vee$ and $\supset$ in LK^+, (ii) Lemma 1, known as Ackermann's lemma and used, e.g. in [9] for substructural logics, and (iii) Lemma 2, which allows the generated rules to obey a (weaker form of) subformula property.

Lemma 1. *Let G be a sequent calculus for $\mathcal{L}$ extending LK^+. Let s be an $\mathcal{L}$-sequent, and γ be a labelled formula in s. The $\mathcal{L}$-rule $\emptyset/s$ is equivalent in G to the rule $r = \{\{\overline{b} : \varphi\} \mid b : \varphi \in s \setminus \{\gamma\}\}/\{\gamma\}$ (where $\overline{f} = t$ and $\overline{t} = f$).*

Proof. $\{\{\overline{b} : \varphi\} \mid b : \varphi \in s \setminus \{\gamma\}\} \vdash_{G \cup \{\emptyset/s\}} \gamma$ is obtained by applying the rule $\emptyset/s$ and then have multiple applications of (cut) (preceded by suitable applications of $(W\Rightarrow)$ and $(\Rightarrow W)$). To prove $\vdash_{G \cup \{r\}} s$ we first use (id) to obtain $\{f : \psi, t : \psi\}$ for every $\psi \in \{\varphi \mid b : \varphi \in s \setminus \{\gamma\}\}$ followed by suitable applications of $(W\Rightarrow)$ and $(\Rightarrow W)$. The claim then follows by applying r. □

Lemma 2. *Let G be a sequent calculus for $\mathcal{L}$ extending LK^+. Let s be an $\mathcal{L}$-sequent, and let $s' = s \cup \{b : p\}$, where $b \in \{f, t\}$ and p is an atomic formula that does not occur in s. Then, $\vdash_{G \cup \{\emptyset/s'\}} \Gamma \Rightarrow \varphi$ iff $\vdash_{G \cup \{\emptyset/s\}} \Gamma \Rightarrow \varphi$, for every sequent $\Gamma \Rightarrow \varphi$.*

Proof. Clearly, $\vdash_{G \cup \{\emptyset/s'\}} \subseteq \vdash_{G \cup \{\emptyset/s\}}$ (applications of $\emptyset/s'$ can be simulated using $(W\Rightarrow)$ or $(\Rightarrow W)$, and $\emptyset/s$). For the converse direction, we distinguish two cases according to b. If $b = f$ then every application of $\emptyset/s$ deriving $\sigma(s)$ can be simulated in $G \cup \{\emptyset/s'\}$ by using (cut) on $\sigma(s) \cup \{f : p_1 \supset p_1\}$ (obtained by $\emptyset/s'$ in which p is substituted with $p_1 \supset p_1$) and $\sigma(s) \cup \{t : p_1 \supset p_1\}$, derivable in LK^+. If $b = t$ we need a proof transformation: every application of $\emptyset/s$ in a derivation of $\Gamma \Rightarrow \varphi$ is replaced with an application of $\emptyset/s'$, in which p is substituted with φ. $t : \varphi$ is then propagated till the end sequent. □

Theorem 1. *Every $H \in \mathbf{H}$ has an equivalent sequent calculus $G_H \in \mathbf{G}$.*

Proof. Follows by repeatedly applying the following procedure (starting from HCL^+ and LK^+). Let $H \in \mathbf{H}$ and $G \in \mathbf{G}$ be an equivalent sequent calculus for $\mathcal{L}_{\mathcal{U}}$ and let $\psi \in \mathbf{Ax}_{\mathcal{U}}$. We show how to construct a finite (possibly empty) set R' of simple $\mathcal{L}_{\mathcal{U}}$-rules such that $H \cup \{\psi\}$ is equivalent to $G \cup R'$.

First, it is easy to see that $H \cup \{\psi\}$ is equivalent to $G \cup \{r_\psi\}$, where r_ψ is the rule $\emptyset/\{t : \psi\}$. For the right-to-left direction consider a proof of a sequent $\Gamma \Rightarrow \varphi$ in $G \cup \{r_\psi\}$, and transform it into a proof of $\Gamma, \psi \Rightarrow \varphi$ in G, by replacing every application of r_ψ with the identity axiom $\{f : \psi, t : \psi\}$, and propagating $f : \psi$ through the derivation till the end sequent. The equivalence of H and G entails that $\Gamma, \psi \vdash_H \varphi$, and it immediately follows that $\Gamma \vdash_{H \cup \{\psi\}} \varphi$.

Now, starting from r_ψ and using the invertibility of the rules for $\wedge, \vee$ and $\supset$, we obtain a finite set of rules R, such that (i) R is equivalent to $\{r_\psi\}$ in G, and (ii) each $r \in R$ has the form $\emptyset/s$, where s has one of the following forms, according to whether ψ is generated by R_p, R_1 or R_2 in the grammar of Def. 1:

1. s consists of at least one labelled formula of the form $b : \star p_1$ ($b \in \{f, t\}$, $\star \in \mathcal{U}$), and any number of labelled formulas $b : p_i$ ($b \in \{f, t\}$, $i \geq 1$).
2. s consists of exactly one labelled formula of the form $b : \star \triangleright p_1$ ($b \in \{f, t\}$, $\star, \triangleright \in \mathcal{U}$), and any number of labelled formulas of the form $b : p_i$ or $b : \star p_1$ ($b \in \{f, t\}$, $i \geq 1$, and $\star \in \mathcal{U}$).
3. s consists of exactly one labelled formula of the form $b : \star(p_1 \diamond p_2)$ ($b \in \{f, t\}$, $\star \in \mathcal{U}$, $\diamond \in \{\wedge, \vee, \supset\}$), and any number of labelled formulas of the form $b : p_i$, $b : \star p_1$, or $b : \star p_2$ ($b \in \{f, t\}$, $i \geq 1$, and $\star \in \mathcal{U}$).

Obviously, we can discard all rules $\emptyset/s$ of R for which $\{f : p_i, t : p_i\} \subseteq s$ for some $i \geq 1$. By Lemma 2, for each rule $\emptyset/s$ left in R: if s has the form 1 or 2 above, we can omit from s all labelled formulas of the form $b : p_i$ for $i > 1$, and similarly, if s has the form 3, all labelled formulas of the form $b : p_i$ for $i > 2$. By Lemma 1 the resulting rules can be transformed into equivalent simple $\mathcal{L}_\mathcal{U}$-rules. □

The proof above is constructive, and induces an algorithm to extract simple $\mathcal{L}_\mathcal{U}$-rules out of axioms in $\mathbf{Ax}_\mathcal{U}$.

Example 3. Let (**b**) be the axiom $p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2))$. Consider the rule $\emptyset/\{t : p_1 \supset (\neg p_1 \supset (\circ p_1 \supset p_2))\}$. Using the invertibility of $(\Rightarrow\supset)$ we obtain an equivalent rule $\emptyset/\{f : p_1, f : \neg p_1, f : \circ p_1, t : p_2\}$. By Lemma 2 we get $\emptyset/\{f : p_1, f : \neg p_1, f : \circ p_1\}$. The primitive rule $\{\{t : p_1\}, \{t : \neg p_1\}\}/\{f : \circ p_1\}$ (or $\{\{t : p_1\}, \{t : \circ p_1\}\}/\{f : \neg p_1\}$) then follows by Lemma 1.

3 Step 2: Extracting Semantics

We define finite-valued semantics, using partial non-deterministic matrices, for every calculus in $\mathbf{G}$.

3.1 Partial Non-deterministic Matrices

Partial non-deterministic matrices were introduced in [6] in the context of labelled sequent calculi. They generalize the notion of non-deterministic matrices by allowing *empty* sets of options in the truth-tables of the logical connectives. This feature makes it possible to semantically characterize every $G \in \mathbf{G}$. Below we shortly reproduce and adapt to our context the basic definitions from [6].

Definition 9. *A* partial non-deterministic matrix *(*PNmatrix*) $\mathcal{M}$ for $\mathcal{L}$ consists of: (i) a set $\mathcal{V}_\mathcal{M}$ of truth values, (ii) a subset $\mathcal{D}_\mathcal{M} \subseteq \mathcal{V}_M$ (designated truth values), and (iii) a truth-table $\diamond_\mathcal{M} : {\mathcal{V}_\mathcal{M}}^n \to P(\mathcal{V}_\mathcal{M})$ for every n-ary connective $\diamond$ of $\mathcal{L}$.*

Definition 10. *Let $\mathcal{M}$ be a PNmatrix for $\mathcal{L}$, and $\mathcal{W}$ be a set of $\mathcal{L}$-formulas closed under subformulas.*

1. *A $\mathcal{W}$-*valuation *is a function v from $\mathcal{W}$ to some set $\mathcal{V}$ (of* truth values*). A wff$_\mathcal{L}$-valuation is also called* an $\mathcal{L}$-valuation.
2. *A $\mathcal{W}$-*valuation *v is called $\mathcal{M}$-legal if its range is $\mathcal{V}_\mathcal{M}$, and it respects the truth-tables of $\mathcal{M}$, i.e. $v(\diamond(\psi_1, \ldots, \psi_n)) \in \diamond_\mathcal{M}(v(\psi_1), \ldots, v(\psi_n))$ for every compound formula $\diamond(\psi_1, \ldots, \psi_n) \in \mathcal{W}$.*
3. *A $\mathcal{W}$-valuation v satisfies an $\mathcal{L}$-sequent s for $\mathcal{M}$ (denoted by $v \models_\mathcal{M} s$) if either $v(\varphi) \in \mathcal{D}_\mathcal{M}$ for some $t : \varphi \in s$, or $v(\varphi) \notin \mathcal{D}_\mathcal{M}$ for some $f : \varphi \in s$.*
4. *Given an $\mathcal{L}$-sequent s, $\vdash^\mathcal{W}_\mathcal{M} s$ if $v \models_\mathcal{M} s$ for every $\mathcal{M}$-legal $\mathcal{W}$-valuation v. We write $\vdash_\mathcal{M} s$ instead of $\vdash^{wff_\mathcal{L}}_\mathcal{M} s$.*

Clearly, every (ordinary) matrix can be identified with a PNmatrix, in which all truth-tables take only singletons.

Example 4. The (positive fragment of the) standard classical matrix can be identified with the PNmatrix $\mathcal{M}_{LK^+}$ defined as:

1. $\mathcal{V}_{\mathcal{M}_{LK^+}} = \{f, t\}$, $\mathcal{D}_{\mathcal{M}_{LK^+}} = \{t\}$.
2. $\wedge_{\mathcal{M}_{LK^+}}$, $\vee_{\mathcal{M}_{LK^+}}$, and $\supset_{\mathcal{M}_{LK^+}}$ are defined according to the classical truth-tables (singletons are used instead of values, e.g. $\wedge_{\mathcal{M}_{LK^+}}(t, f) = \{f\}$).

Fact 2. *$\mathcal{M}_{LK^+}$ is sound and complete for LK^+ (i.e. $\vdash_{LK^+} s$ iff $\vdash_{\mathcal{M}_{LK^+}} s$).*

3.2 PNmatrices for $\mathcal{U}$-extensions of LK^+

Until the end of this section, let G be some $\mathcal{U}$-extension of LK^+. The main idea behind the construction of a PNmatrix $\mathcal{M}_G$ for G is to use truth values as "information carriers" (along the lines of [1]) in the following sense. In addition to determining whether φ is "true", the truth value of φ contains also information about the "truth/falsity" of all the formulas of the form $\star\varphi$ for $\star \in \mathcal{U}$. To this end, instead of using the truth values $\{f, t\}$, we use extended truth values, which are tuples over $\{f, t\}$ of size $|\mathcal{U}| + 1$. The first element of each such a tuple u, denoted by u^0, is reserved for representing the "truth/falsity" of φ. Each connective $\star \in \mathcal{U}$ is then (arbitrarily) allocated one of the remaining elements. We shall denote by $u^\star$ the element of u allocated for $\star \in \mathcal{U}$. Thus whenever φ is assigned the truth value u, φ is "true" iff $u^0 = t$, and for each $\star \in \mathcal{U}$, $\star\varphi$ is "true" iff $u^\star = t$. However, in constructing $\mathcal{M}_G$ not all the possible tuples will be used as truth values: only those that "respect" the primitive rules of G (cf. Def. 4). This is formalized as follows:

Notation 1. *We denote by $\mathsf{V}_\mathcal{U}$ the set of all $(|\mathcal{U}| + 1)$-tuples over $\{f, t\}$.*

Definition 11. *A tuple $u \in \mathsf{V}_\mathcal{U}$* satisfies *a $\mathcal{U}_1$-premise q, if either $q = \{u^0 : p_1\}$, or $q = \{u^\star : \star p_1\}$ for some $\star \in \mathcal{U}$. u* respects *a primitive rule $Q/\{b : \star p_1\}$ if $u^\star = b$ whenever u satisfies every $q \in Q$.*

Definition 12. *$\mathcal{V}_{\mathcal{M}_G}$ (the set of truth values of the PNmatrix $\mathcal{M}_G$) is the set of all tuples in $\mathsf{V}_\mathcal{U}$ which respect all primitive rules of G. In addition, the set of designated truth values $\mathcal{D}_{\mathcal{M}_G}$ is $\{u \in \mathcal{V}_{\mathcal{M}_G} \mid u^0 = t\}$.*

Example 5. Suppose that $\mathcal{U} = \{\neg\}$, and that the only primitive rule of G is $\{\{f : p_1\}\}/\{t : \neg p_1\}$. A pair $u \in \mathcal{V}_{\mathcal{M}_G}$ respects $(\Rightarrow \neg)$ iff $u^{\neg} = t$ whenever $u^0 = f$. Thus we obtain $\mathcal{V}_{\mathcal{M}_G} = \{\langle f, t\rangle, \langle t, f\rangle, \langle t, t\rangle\}$ (here $u^{\neg}$ is the second component of each pair). The designated values are: $\mathcal{D}_{\mathcal{M}_G} = \{\langle t, f\rangle, \langle t, t\rangle\}$.

Having defined the truth values of $\mathcal{M}_G$, we proceed to providing a truth-table $\triangleright_{\mathcal{M}_G}$ for each (unary) connective $\triangleright \in \mathcal{U}$. This is done according to the *onevar rules* of G of the form $Q/\{b : \star \triangleright p_1\}$.

Definition 13. *Let $\triangleright \in \mathcal{U}$. For every $u_1 \in \mathcal{V}_{\mathcal{M}_G}$, $\triangleright_{\mathcal{M}_G}(u_1)$ is the set of all tuples $u \in \mathcal{V}_{\mathcal{M}_G}$ such that: (i) $u^0 = u_1^{\triangleright}$; and (ii) for every onevar rule of G of the form $Q/\{b : \star \triangleright p_1\}$, if u_1 satisfies every $q \in Q$ then $u^{\star} = b$.*

Intuitively, condition (i) forces the information about the "truth/falsity" of $\triangleright\varphi$ carried in the truth value of $\triangleright\varphi$ (in the first bit of this tuple) to be equal to the one carried in the truth value of φ.

Example 6. Following Example 5, suppose that G's only onevar rule of the form $Q/\{b : \star\neg p_1\}$ is $\{\{f : p_1\}\}/\{f : \neg\neg p_1\}$. Let us explain, e.g., how $\neg_{\mathcal{M}_G}(\langle t, f\rangle)$ is obtained. The only tuple from $\mathcal{V}_{\mathcal{M}_G} = \{\langle f, t\rangle, \langle t, f\rangle, \langle t, t\rangle\}$ satisfying condition (i) (that is, whose first component is $\langle t, f\rangle^{\neg} = f$) is $u = \langle f, t\rangle$. Condition (ii) holds trivially for u, as $\langle t, f\rangle$ does not satisfy the premise $\{f : p_1\}$ of the above rule. Thus we obtain: $\neg_{\mathcal{M}_G}(\langle t, f\rangle) = \{\langle f, t\rangle\}$. Similarly, we get $\neg_{\mathcal{M}_G}(\langle f, t\rangle) = \{\langle t, f\rangle\}$, and $\neg_{\mathcal{M}_G}(\langle t, t\rangle) = \{\langle t, f\rangle, \langle t, t\rangle\}$.

To complete the construction of $\mathcal{M}_G$, we provide the truth-tables of the binary connectives, using the *twovar rules*.

Definition 14. *A pair of tuples $\langle u_1, u_2\rangle \in \mathsf{V}_{\mathcal{U}}{}^2$* satisfies *a $\mathcal{U}_1$-premise q, if u_1 satisfies q. $\langle u_1, u_2\rangle$* satisfies *a $\mathcal{U}_2$-premise q, if u_2 satisfies q.*

Definition 15. *Let $\diamond \in \{\wedge, \vee, \supset\}$. For every $u_1, u_2 \in \mathcal{V}_{\mathcal{M}_G}$, $\diamond_{\mathcal{M}_G}(u_1, u_2)$ is the set of all tuples $u \in \mathcal{V}_{\mathcal{M}_G}$ satisfying: (i) $u^0 \in \diamond_{\mathcal{M}_{LK+}}(u_1^0, u_2^0)$; and (ii) for every twovar rule of G of the form $Q/\{b : \star(p_1 \diamond p_2)\}$, if $\langle u_1, u_2\rangle$ satisfies every $q \in Q$ then $u^{\star} = b$.*

Intuitively, condition (i) ensures that $\diamond$ behaves as the corresponding classical connective, and condition (ii) provides the correspondence between the truth-table of $\diamond$ and the twovar rules that involve $\diamond$.

Example 7. Following Example 5, suppose that G's only twovar rule of the form $Q/\{b : \star(p_1 \wedge p_2)\}$ is $(\Rightarrow \neg\wedge)_1$ (see Fig. 2). A pair of values $\langle u_1, u_2\rangle \in \mathcal{V}_{\mathcal{M}_G}{}^2$ satisfies the premise of $(\Rightarrow \neg\wedge)_1$ iff $u_1^{\neg} = t$. In this case we require that for every $u \in \wedge_{\mathcal{M}_G}(u_1, u_2)$ we have $u^{\neg} = t$. Thus we obtain the following table for $\wedge$:

$\widetilde{\wedge}$	$\langle f, t\rangle$	$\langle t, f\rangle$	$\langle t, t\rangle$
$\langle f, t\rangle$	$\{\langle f, t\rangle\}$	$\{\langle f, t\rangle\}$	$\{\langle f, t\rangle\}$
$\langle t, f\rangle$	$\{\langle f, t\rangle\}$	$\{\langle t, f\rangle, \langle t, t\rangle\}$	$\{\langle t, f\rangle, \langle t, t\rangle\}$
$\langle t, t\rangle$	$\{\langle f, t\rangle\}$	$\{\langle t, t\rangle\}$	$\{\langle t, t\rangle\}$

3.3 Soundness and Completeness

We turn to prove the correctness of the construction of $\mathcal{M}_G$. We establish strong forms of soundness and completeness, to be used later in the characterization of analyticity of G. The main idea is to maintain a correlation between the formulas used in the derivation, and the formulas from the domain of the valuations. In what follows $\mathcal{W}$ is an arbitrary set of $\mathcal{L}_\mathcal{U}$-formulas closed under subformulas. We use the following additional notations and definitions:

Notation 2. *Let s be an $\mathcal{L}_\mathcal{U}$-sequent.*

1. *$sub[s]$ denotes the set of subformulas of all $\mathcal{L}_\mathcal{U}$-formulas occurring in s.*
2. *s is called a* $\mathcal{W}$-sequent *if $sub[s] \subseteq \mathcal{W}$.*
3. *We write $\vdash_G^{\mathcal{W}} s$ if there exists a proof of s in G consisting only of $\mathcal{W}$-sequents.*

Definition 16. *The sets $\mathcal{U}^+(\mathcal{W})$ and $\mathcal{U}^-(\mathcal{W})$ are defined as follows:*
$\mathcal{U}^-(\mathcal{W}) = \mathcal{W} \backslash \{\star\psi \in \mathcal{W} \mid \star \in \mathcal{U}, \star\psi \textit{ is not a proper subformula of a formula in } \mathcal{W}\}$
$\mathcal{U}^+(\mathcal{W}) = \mathcal{W} \cup \{\star\psi \mid \star \in \mathcal{U}, \psi \in \mathcal{U}^-(\mathcal{W})\}$

Example 8. For $\mathcal{U} = \{\neg\}$ and $\mathcal{W} = \{p_1, p_2, \neg p_1, \neg p_2, p_1 \vee p_2, \neg p_1 \vee p_2, \neg(p_1 \vee p_2)\}$, we have $\mathcal{U}^-(\mathcal{W}) = \{p_1, p_2, \neg p_1, p_1 \vee p_2, \neg p_1 \vee p_2\}$, and $\mathcal{U}^+(\mathcal{W}) = \mathcal{W} \cup \{\neg\neg p_1, \neg(\neg p_1 \vee p_2)\}$.

Remark 2. Note that $\psi \in \mathcal{U}^-(\mathcal{W})$ whenever $\star\psi \in \mathcal{U}^+(\mathcal{W})$ for some $\star \in \mathcal{U}$.

The weaker notion of satisfaction, introduced in the following definition, is needed later to characterize (a generalized form of) analyticity.

Definition 17. *A $\mathcal{U}^-(\mathcal{W})$-valuation $v : \mathcal{U}^-(\mathcal{W}) \to \mathsf{V}_\mathcal{U}$* w-satisfies *a $\mathcal{U}^+(\mathcal{W})$-sequent s if there exists some labelled formula $b : \psi \in s$, such that either (i) ψ does not have the form $\star\varphi$ and $v(\psi)^0 = b$; or (ii) $\psi = \star\varphi$ (for some $\star \in \mathcal{U}$ and $\varphi \in \mathcal{U}^-(\mathcal{W})$) and $v(\varphi)^\star = b$.*

Theorem 2 (Soundness). *Let s be a $\mathcal{W}$-sequent. If $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s$, then every $\mathcal{M}_G$-legal $\mathcal{U}^-(\mathcal{W})$-valuation w-satisfies s.*

Proof. It suffices to show that whenever an $\mathcal{M}_G$-legal $\mathcal{U}^-(\mathcal{W})$-valuation w-satisfies the premises of some application of an $\mathcal{L}_\mathcal{U}$-rule $r = Q/s$ of G consisting solely of formulas from $\mathcal{U}^+(\mathcal{W})$, it also w-satisfies its conclusion. Consider such an application of r inferring $\sigma(s) \cup c$ from the set $\{\sigma(q) \cup c \mid q \in Q\}$, where c is an $\mathcal{L}_\mathcal{U}$-sequent, and σ is an $\mathcal{L}_\mathcal{U}$-substitution. Assume that $\sigma(p_1) = \psi_1$ and $\sigma(p_2) = \psi_2$. Let v be an $\mathcal{M}_G$-legal $\mathcal{U}^-(\mathcal{W})$-valuation, and suppose that v w-satisfies $\sigma(q) \cup c$ for every $q \in Q$. We prove that v w-satisfies $\sigma(s) \cup c$. Clearly, if v w-satisfies c, then we are done. Suppose otherwise. Then our assumption entails that it w-satisfies $\sigma(q)$ for every $q \in Q$. We show that in this case v w-satisfies $\sigma(s)$ (and so it w-satisfies $\sigma(s) \cup c$). For $r \in LK^+$ the claim is easy and left for the reader. Otherwise, r is a simple rule. Three cases can occur:

- Suppose that $r = Q/\{b : \triangleright p_1\}$ is a primitive rule. Note that since we only consider applications of r consisting solely of formulas from $\mathcal{U}^+(\mathcal{W})$, we have that $\triangleright\psi_1 \in \mathcal{U}^+(\mathcal{W})$ and so $\psi_1 \in \mathcal{U}^-(\mathcal{W})$. The fact v w-satisfies $\sigma(q)$ for every $q \in Q$ implies that $v(\psi_1)$ satisfies every $q \in Q$. To see this, consider the following cases:
  - Assume that $q = \{b : p_1\}$, and ψ_1 does not have the form $\star\varphi$. Since v w-satisfies $\sigma(q)$, $v(\psi_1)^0 = b$.
  - Assume that $q = \{b : p_1\}$, and ψ_1 has the form $\star\varphi$. Since v w-satisfies $\sigma(q)$, $v(\varphi)^\star = b$. Since v is $\mathcal{M}_G$-legal, $v(\star\varphi)^0 = b$.
  - Assume that $q = \{b : \star p_1\}$. Since v w-satisfies $\sigma(q)$, $v(\psi_1)^\star = b$.

  In all cases, we obtain that $v(\psi_1)$ satisfies q. Now, since $v(\psi_1) \in \mathcal{V}_{\mathcal{M}_G}$, $v(\psi_1)$ respects r, and so $v(\psi_1)^\triangleright = b$. Thus v w-satisfies $\{b : \triangleright\psi_1\}$.
- Suppose that $r = Q/\{b : \star \triangleright p_1\}$ is a onevar rule. As in the previous case, $v(\psi_1)$ satisfies every $q \in Q$. Thus, since $v(\triangleright\psi_1) \in \triangleright_{\mathcal{M}_G}(v(\psi_1))$, we have $v(\triangleright\psi_1)^\star = b$. It follows that v w-satisfies $\{b : \star \triangleright \psi_1\}$.
- Suppose that $r = Q/\{b : \star(p_1 \diamond p_2)\}$ is a twovar rule. Similarly to the previous cases, we have $\langle v(\psi_1), v(\psi_2)\rangle$ satisfies every $q \in Q$. Thus, since $v(\psi_1 \diamond \psi_2) \in \diamond_{\mathcal{M}_G}(v(\psi_1), v(\psi_2))$, we have that $v(\psi_1 \diamond \psi_2)^\star = b$. It follows that v w-satisfies $\{b : \star(\psi_1 \diamond \psi_2)\}$. □

Theorem 3 (Completeness). *If every $\mathcal{M}_G$-legal $\mathcal{U}^-(\mathcal{W})$-valuation w-satisfies a $\mathcal{W}$-sequent s_0, then $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_0$.*

Proof. Suppose that $\nvdash_G^{\mathcal{U}^+(\mathcal{W})} s_0$. We construct an $\mathcal{M}_G$-legal $\mathcal{U}^-(\mathcal{W})$-valuation v that does not w-satisfy s_0. Call a set Ω of labelled $\mathcal{L}_\mathcal{U}$-formulas *maximal* if it satisfies the following conditions: (i) Ω consists of labelled $\mathcal{L}_\mathcal{U}$-formulas of the form $b : \psi$ for $\psi \in \mathcal{U}^+(\mathcal{W})$; (ii) $\nvdash_G^{\mathcal{U}^+(\mathcal{W})} s$ for every $\mathcal{L}_\mathcal{U}$-sequent $s \subseteq \Omega$; and (iii) For every formula $\psi \in \mathcal{U}^+(\mathcal{W})$ and $b \in \{f, t\}$, if $b : \psi \notin \Omega$ then $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s \cup \{b : \psi\}$ for some $\mathcal{L}_\mathcal{U}$-sequent $s \subseteq \Omega$. It is straightforward to construct a maximal set Ω that extends s_0.

Note that the availability of the cut rule implies that for every $\psi \in \mathcal{U}^+(\mathcal{W})$, either $f : \psi \in \Omega$ or $t : \psi \in \Omega$ (otherwise, we would have $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_1 \cup \{f : \psi\}$ and $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_2 \cup \{t : \psi\}$ for $s_1, s_2 \subseteq \Omega$, and by applying weakenings (the rules $(W \Rightarrow)$ and $(\Rightarrow W)$) and (cut) we could obtain $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_1 \cup s_2$). Similarly, the availability of the identity axiom implies that for every $\psi \in \mathcal{U}^+(\mathcal{W})$, either $f : \psi \notin \Omega$ or $t : \psi \notin \Omega$ (otherwise, the fact that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} \{f : \psi, t : \psi\}$ would contradict Ω's properties).

Let $v : \mathcal{U}^-(\mathcal{W}) \to \mathsf{V}_\mathcal{U}$ be a $\mathcal{U}^-(\mathcal{W})$-valuation defined by: $v(\psi)^0 = t$ iff $f : \psi \in \Omega$, and for every $\star \in \mathcal{U}$: $v(\psi)^\star = t$ iff $f : \star\psi \in \Omega$. Thus we have that for every $\psi \in \mathcal{U}^-(\mathcal{W})$ and $b \in \{f, t\}$, $v(\psi)^0 = b$ iff $b : \psi \notin \Omega$, and for every $\star \in \mathcal{U}$ $v(\psi)^\star = b$ iff $b : \star\psi \notin \Omega$. We show that v does not w-satisfy s_0. Let $b : \psi \in s_0$ such that ψ does not have the form $\star\varphi$. Thus $\psi \in \mathcal{U}^-(\mathcal{W})$, and since $s_0 \subseteq \Omega$, $v(\psi)^0 \neq b$. Similarly, let $b : \psi \in s_0$ such that ψ does have the form $\psi = \star\varphi$ (for some $\star \in \mathcal{U}$ and $\mathcal{L}_\mathcal{U}$-formula φ). Thus $\varphi \in \mathcal{U}^-(\mathcal{W})$, and since $s_0 \subseteq \Omega$, $v(\varphi)^\star \neq b$.

To show that v is $\mathcal{M}_G$-legal, we use the following properties:

$(*)$ Let σ be an $\mathcal{L}_{\mathcal{U}}$-substitution, such that $\sigma(p_1) \in \mathcal{U}^-(\mathcal{W})$. If $v(\sigma(p_1))$ satisfies a $\mathcal{U}_1$-premise q then $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s \cup \sigma(q)$ for some $\mathcal{L}_{\mathcal{U}}$-sequent $s \subseteq \Omega$.

To see this, note that if $v(\sigma(p_1))$ satisfies q then one of the following holds:

– $q = b : p_1$ and $v(\sigma(p_1))^0 = b$. Thus $b : \sigma(p_1) \notin \Omega$, and since $\sigma(p_1) \in \mathcal{U}^+(\mathcal{W})$, we obtain that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s \cup \{b : \sigma(p_1)\}$ for some $\mathcal{L}_{\mathcal{U}}$-sequent $s \subseteq \Omega$.
– $q = b : \star p_1$ and $v(\sigma(p_1))^\star = b$. Thus $b : \star\sigma(p_1) \notin \Omega$, and since $\star\sigma(p_1) \in \mathcal{U}^+(\mathcal{W})$, we obtain that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s \cup \{b : \star\sigma(p_1)\}$ for some $\mathcal{L}_{\mathcal{U}}$-sequent $s \subseteq \Omega$.

Similarly, we have the following:

$(**)$ Let q be a $\mathcal{U}_1$-premise or a $\mathcal{U}_2$-premise, and σ be an $\mathcal{L}_{\mathcal{U}}$-substitution, such that $\sigma(p_1), \sigma(p_2) \in \mathcal{U}^-(\mathcal{W})$. If $\langle v(\sigma(p_1)), v(\sigma(p_2))\rangle$ satisfies q, then $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s \cup \sigma(q)$ for some $\mathcal{L}_{\mathcal{U}}$-sequent $s \subseteq \Omega$.

We show that $\mathcal{V}_{\mathcal{M}_G}$ is the range of v. Let $\psi \in \mathcal{U}^-(\mathcal{W})$. To prove that $v(\psi) \in \mathcal{V}_{\mathcal{M}_G}$, we show that $v(\psi)$ respects all primitive rules of G. Consider a primitive rule of G, $r = Q/\{b : \star p_1\}$. Suppose that $v(\psi)$ satisfies every $q \in Q$. We show that $v(\psi)^\star = b$. Let σ be any $\mathcal{L}_{\mathcal{U}}$-substitution, assigning ψ to p_1. By $(*)$, for every $q \in Q$, there exists some $\mathcal{L}_{\mathcal{U}}$-sequent $s_q \subseteq \Omega$ such that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_q \cup \sigma(q)$. By applying weakenings and the rule r, we obtain that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} \bigcup_{q\in Q} s_q \cup \{b : \star\psi\}$ (here we use the fact that $\star\psi \in \mathcal{U}^+(\mathcal{W})$ since $\psi \in \mathcal{U}^-(\mathcal{W})$). This implies that $b : \star\psi \notin \Omega$, and so $v(\psi)^\star = b$.

Finally, we show that v respects the truth-tables of $\mathcal{M}_G$:

(1) Let $\triangleright\psi \in \mathcal{U}^-(\mathcal{W})$ (where $\triangleright \in \mathcal{U}$). We show that $v(\triangleright\psi) \in \triangleright_{\mathcal{M}_G}(v(\psi))$. By the construction of $\triangleright_{\mathcal{M}_G}$, it suffices to show: (i) $v(\triangleright\psi)^0 = v(\psi)^\triangleright$; and (ii) $v(\triangleright\psi)^\star = b$ for every onevar rule $r = Q/\{b : \star \triangleright p_1\}$ of G for which $v(\psi)$ satisfies every $q \in Q$. (i) trivially holds using the definition of v. For (ii), let $r = Q/\{b : \star \triangleright p_1\}$ be a onevar rule of G, and suppose that $v(\psi)$ satisfies every $q \in Q$. We prove that $v(\triangleright\psi)^\star = b$. Let σ be any $\mathcal{L}_{\mathcal{U}}$-substitution, assigning ψ to p_1. By $(*)$ (note that $\psi \in \mathcal{U}^-(\mathcal{W})$ since $\mathcal{U}^-(\mathcal{W})$ is closed under subformula), for every $q \in Q$, there exists some $\mathcal{L}_{\mathcal{U}}$-sequent $s_q \subseteq \Omega$ such that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_q \cup \sigma(q)$. By applying weakenings and the rule r, we obtain that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} \bigcup_{q\in Q} s_q \cup \{b : \star \triangleright \psi\}$ (note that $\star \triangleright \psi \in \mathcal{U}^+(\mathcal{W})$ since $\triangleright\psi \in \mathcal{U}^-(\mathcal{W})$). This implies that $b : \star \triangleright \psi \notin \Omega$, and so $v(\triangleright\psi)^\star = b$.

(2) Let $\psi_1 \diamond \psi_2 \in \mathcal{U}^-(\mathcal{W})$ for $\diamond \in \{\wedge, \vee, \supset\}$. We show that $v(\psi_1 \diamond \psi_2) \in \diamond_{\mathcal{M}_G}(v(\psi_1), v(\psi_2))$. Here it suffices to show: (i) $v(\psi_1 \diamond \psi_2)^\star = b$ for every twovar rule $r = Q/\{b : \star(p_1 \diamond p_2)\}$ of G for which $\langle v(\psi_1), v(\psi_2)\rangle$ satisfies every $q \in Q$; and (ii) $v(\diamond(\psi_1, \psi_2))^0 \in \diamond_{\mathcal{M}_{LK+}}(v(\psi_1)^0, v(\psi_2)^0)$. We prove (i) and leave (ii) to the reader. Let $r = Q/\{b : \star(p_1 \diamond p_2)\}$ be a twovar rule of G, and suppose that $\langle v(\psi_1), v(\psi_2)\rangle$ satisfies every $q \in Q$. We prove that $v(\psi_1 \diamond \psi_2)^\star = b$. Let σ be any $\mathcal{L}_{\mathcal{U}}$-substitution, assigning ψ_1 to p_1, and ψ_2 to p_2. By $(**)$, for every $q \in Q$, there exists some $\mathcal{L}_{\mathcal{U}}$-sequent $s_q \subseteq \Omega$ such that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} s_q \cup \sigma(q)$. By applying weakenings and the rule r, we obtain that $\vdash_G^{\mathcal{U}^+(\mathcal{W})} \bigcup_{q\in Q} s_q \cup \{b : \star(\psi_1 \diamond \psi_2)\}$

(note that $\star(\psi_1 \diamond \psi_2) \in \mathcal{U}^+(\mathcal{W})$ since $\psi_1 \diamond \psi_2 \in \mathcal{U}^-(\mathcal{W})$). This implies that $b : \star(\psi_1 \diamond \psi_2) \notin \Omega$, and so $v(\psi_1 \diamond \psi_2)^\star = b$. □

Corollary 1. *For every $\mathcal{L}_\mathcal{U}$-sequent s, $\vdash_G s$ iff $\vdash_{\mathcal{M}_G} s$.*

Proof. The claim follows by choosing $\mathcal{W} = \mathit{wff}_{\mathcal{L}_\mathcal{U}}$ in Thm. 2 and Thm. 3 (in this case $\mathcal{U}^+(\mathcal{W}) = \mathcal{U}^-(\mathcal{W}) = \mathcal{W}$). Note that an $\mathcal{M}_G$-legal $\mathcal{L}_\mathcal{U}$-valuation v w-satisfies an $\mathcal{L}_\mathcal{U}$-sequent iff $v \models_{\mathcal{M}_G} s$ (since $v(\star\psi)^0 = v(\psi)^\star$ for every $\mathcal{L}_\mathcal{U}$-formula $\star\psi$). □

4 Semantics at Work

Let us take stock of what we have achieved so far. Given a Hilbert calculus $H \in \mathbf{H}$ we introduced an equivalent sequent calculus $G_H \in \mathbf{G}$ and extracted a suitable semantics out of it (the PNmatrix $\mathcal{M}_{G_H}$). In this section we show how to use $\mathcal{M}_{G_H}$ to prove the decidability of H and to check whether G_H is analytic (in the sense defined below). If G_H is not analytic, $\mathcal{M}_{G_H}$ is used to define a family of cut-free calculi for H.

Corollary 2 (Decidability). *Given a Hilbert system $H \in \mathbf{H}$ and a finite set $\Gamma \cup \{\varphi\}$ of formulas, it is decidable whether $\Gamma \vdash_H \varphi$ or not.*

Proof. Follows by the soundness and completeness of $\mathcal{M}_{G_H}$ for G_H, Thm. 1, and the fact, proved in [6], that each logic characterized by a finite PNmatrix is decidable. □

Roughly speaking, a sequent calculus is analytic if whenever a sequent s is provable in it, it can also be proven using only the "syntactic material available within s". Usually this "material" is taken to consist of all subformulas occurring in s (in this case 'analyticity' is just another name for the global subformula property). However, weaker variants have also been considered in the literature, especially in modal logic. In this paper we use the following:

Definition 18. *A $\mathcal{U}$-extension G of LK^+ is* $\mathcal{U}$-analytic *if for every $\mathcal{L}_\mathcal{U}$-sequent s: $\vdash_G s$ implies that $\vdash_G^{\mathcal{U}^+(sub[s])} s$.*

Next, we show that $\mathcal{M}_G$ can be easily used to check whether G is $\mathcal{U}$-analytic.

Definition 19. *A PNmatrix $\mathcal{M}$ for $\mathcal{L}$ is called* proper *if $\mathcal{V}_\mathcal{M}$ is non-empty and $\diamond_\mathcal{M}(x_1, \dots, x_n) \neq \emptyset$ for every n-ary connective $\diamond$ of $\mathcal{L}$ and $x_1, \dots, x_n \in \mathcal{V}_\mathcal{M}$.*

Theorem 4. *A $\mathcal{U}$-extension G of LK^+ is $\mathcal{U}$-analytic iff $\mathcal{M}_G$ is proper.*

Proof. ($\Rightarrow$) Suppose that $\mathcal{M}_G$ is not proper. First, if $\mathcal{V}_{\mathcal{M}_G}$ is empty, then $\vdash_{\mathcal{M}_G} \emptyset$, and so (by Cor. 1), $\vdash_G \emptyset$. But, $\mathcal{U}^+(\emptyset) = \emptyset$, and clearly there is no derivation in G that does not contain any formula. It follows that G is not $\mathcal{U}$-analytic in this case. Otherwise, there exist either some $\triangleright \in \mathcal{U}$ and $u \in \mathcal{V}_{\mathcal{M}_G}$ such that $\triangleright_{\mathcal{M}_G}(u) = \emptyset$, or some $\diamond \in \{\wedge, \vee, \supset\}$ and $u_1, u_2 \in \mathcal{V}_{\mathcal{M}_G}$ such that $\diamond_{\mathcal{M}_G}(u_1, u_2) = \emptyset$. We consider here only the first case and leave the second to the reader. Define the $\mathcal{L}_\mathcal{U}$-sequent $s = \{\overline{u^0} : p_1\} \cup \{\overline{u^\star} : \star p_1 \mid \star \in \mathcal{U}\}$ (where $\overline{t} = f$ and $\overline{f} = t$).

We first prove that $\vdash_G s$. By Cor. 1 it suffices to show $\vdash_{\mathcal{M}_G} s$. Suppose otherwise, and let v be an $\mathcal{M}_G$-legal $\mathcal{L}_\mathcal{U}$-valuations such that $v \not\models_{\mathcal{M}_G} s$. Then, $v(p_1)^0 = u^0$ and $v(\star p_1)^0 = u^\star$ for every $\star \in \mathcal{U}$. Since v is $\mathcal{M}_G$-legal, we have that $v(p_1)^\star = u^\star$ for every $\star \in \mathcal{U}$. It follows that $v(p_1) = u$. Since v is $\mathcal{M}_G$-legal, we have $v(\triangleright p_1) \in \triangleright_{\mathcal{M}_G}(v(p_1))$. Clearly, this is not possible under the assumption that $\triangleright_{\mathcal{M}_G}(u) = \emptyset$. Next we claim that $\not\vdash_G^{\mathcal{U}^+(sub[s])} s$ (and so G is not $\mathcal{U}$-analytic). To see this, note that the $\{p_1\}$-valuation defined by $v(p_1) = u$ is an $\mathcal{M}_G$-legal $\mathcal{U}^-(sub[s])$-valuation that does not w-satisfy s. By Thm. 2, $\not\vdash_G^{\mathcal{U}^+(sub[s])} s$.

($\Leftarrow$) Assume that $\mathcal{M}_G$ is proper and $\not\vdash_G^{\mathcal{U}^+(sub[s])} s$ for some $\mathcal{L}_\mathcal{U}$-sequent s. We prove that $\not\vdash_G s$. By Thm. 3, there exists an $\mathcal{M}_G$-legal $\mathcal{U}^-(sub[s])$-valuation v that does not w-satisfy s. Being $\mathcal{M}_G$ proper, it is straightforward to extend v to a (full) $\mathcal{M}_G$-legal $\mathcal{L}_\mathcal{U}$-valuation v'. Note that $v' \not\models_{\mathcal{M}_G} s$ (since $v(\star\psi)^0 = v'(\psi)^\star$ for every $\mathcal{L}_\mathcal{U}$-formula $\star\psi$). Cor. 1 then entails that $\not\vdash_G s$. $\square$

There are, however, calculi in **G** which are *not* $\mathcal{U}$-analytic. This is the case, e.g., for the extension of HCL^+ by axioms $(\mathbf{n_1})$, $(\mathbf{n}^\mathbf{r}_\wedge)$, $(\mathbf{b})$ and $(\mathbf{o}^\mathbf{1}_\wedge)$ (cf. Fig. 1). Its corresponding sequent calculus induces a PNmatrix which is not proper (this can be verified in the system *Paralyzer*), hence it is not $\{\neg, \circ\}$-analytic. When $G \in \mathbf{G}$ is not $\mathcal{U}$-analytic, we start by transforming $\mathcal{M}_G$ into a *finite family* of *proper* PNmatrices, which satisfy the following property:

Definition 20. *([6]) Let $\mathcal{M}$ and $\mathcal{N}$ be PNmatrices for $\mathcal{L}$. We say that $\mathcal{N}$ is a* simple refinement *of $\mathcal{M}$ if $\mathcal{V}_\mathcal{N} \subseteq \mathcal{V}_\mathcal{M}$, $\mathcal{D}_\mathcal{N} = \mathcal{D}_\mathcal{M} \cap \mathcal{V}_\mathcal{N}$, and $\diamond_\mathcal{N}(x_1, \ldots, x_n) \subseteq \diamond_\mathcal{M}(x_1, \ldots, x_n)$ for every n-ary connective $\diamond$ of $\mathcal{L}$ and $x_1, \ldots, x_n \in \mathcal{V}_\mathcal{N}$.*

Theorem 5. *For every finite PNmatrix $\mathcal{M}$ for $\mathcal{L}$, there exists $\mathcal{M}_1 \ldots \mathcal{M}_n$, finite proper simple refinements of $\mathcal{M}$, such that $\vdash_\mathcal{M} = \vdash_{\cap \mathcal{M}_i}$ for $i = 1, \ldots, n$.*

Proof (Outline). Let $\mathcal{M}$ be a PNmatrix for $\mathcal{L}$. Choose $\mathcal{M}_1, \ldots, \mathcal{M}_n$ to be all simple refinements of $\mathcal{M}$ which are proper PNmatrices. Based on the results in [6], we show that $\vdash_\mathcal{M} = \vdash_{\cap \mathcal{M}_i}$. ($\Rightarrow$) By Prop. 1 in [6], $\vdash_\mathcal{M} \subseteq \vdash_\mathcal{N}$ for every simple refinement $\mathcal{N}$ of $\mathcal{M}$. Therefore, $\vdash_\mathcal{M} \subseteq \vdash_{\cap \mathcal{M}_i}$.

($\Leftarrow$) Suppose that $\not\vdash_\mathcal{M} s$. Thus $v \not\models_\mathcal{M} s$ for some $\mathcal{M}$-legal $\mathcal{L}$-valuation v. Thm. 1 in [6] ensures that there exists some $\mathcal{M}_i$, such that v is $\mathcal{M}_i$-legal. The fact that $v \not\models_\mathcal{M} s$ easily entails that $v \not\models_{\mathcal{M}_i} s$, and so $\not\vdash_{\mathcal{M}_i} s$. $\square$

We can now apply the method of [3], which produces a cut-free sequent calculus G which is sound and complete for any proper PNmatrix $\mathcal{M}$, whose set of designated truth values ($\mathcal{D}_\mathcal{M}$) is a non-empty proper subset of the set of its truth values ($\mathcal{V}_\mathcal{M}$), provided that its language satisfies the following slightly reformulated condition of [3]:

Definition 21. *Let $\mathcal{M}$ be a proper PNmatrix for $\mathcal{L}$. We say that $\mathcal{L}$ is* sufficiently expressive *for $\mathcal{M}$ if for any $x \in \mathcal{V}_\mathcal{M}$, there exists a set $\mathcal{S}$ of $\mathcal{L}$-sequents, each of which has the form $\{b : \psi\}$, for some $b \in \{f, t\}$ and $\psi \in wff_\mathcal{L}$ in which p_1 is the only atomic variable, such that the following condition holds:*

– *For any $\mathcal{M}$-legal $\mathcal{L}$-valuation v and $\varphi \in wff_{\mathcal{L}}$, $v(\varphi) = x$ iff v satisfies every $\mathcal{L}$-sequent in $\sigma(\mathcal{S})$ for $\mathcal{M}$ for any $\mathcal{L}$-substitution σ such that $\sigma(p_1) = \varphi$.*

Corollary 3. *Let $G \in \mathbf{G}$ be a $\mathcal{U}$-extension of LK^+ that is not $\mathcal{U}$-analytic. We can construct a family of cut-free sequent calculi F_G, such that for every sequent s: $\vdash_G s$ iff $\vdash_{G'} s$ for every $G' \in \mathsf{F}_G$.*

Proof. We start by constructing $\mathcal{M}_G$. If $\mathcal{D}_{\mathcal{M}_G} = \emptyset$ or $\mathcal{D}_{\mathcal{M}_G} = \mathcal{V}_{\mathcal{M}_G}$, $\mathcal{M}_G$ has a trivial corresponding cut-free calculus. For the rest of the cases, the claim follows by Thm. 5 using the method of [3]. Note that $\mathcal{L}_{\mathcal{U}}$ is sufficiently expressive for any simple refinement of $\mathcal{M}_G$. Indeed, for $x \in \mathcal{V}_{\mathcal{M}_G}$, define $\mathcal{S}_x = \{x^0 : p_1\} \cup \{x^\star : \star p_1 \mid \star \in \mathcal{U}\}$. Let $\mathcal{M}$ be a simple refinement of $\mathcal{M}_G$ and let v be an $\mathcal{M}$-legal $\mathcal{L}_{\mathcal{U}}$-valuation. The required condition is met by the fact that for every $\star \in \mathcal{U}$ and $\theta \in wff_{\mathcal{L}_{\mathcal{U}}}$, $v(\star\theta)^0 = v(\theta)^\star$ (by condition (i) in Def. 13). □

References

1. Avron, A.: Logical non-determinism as a tool for logical modularity: An introduction. In: We Will Show Them: Essays in Honor of Dov Gabbay, pp. 105–124. College Publications (2005)
2. Avron, A.: Non-deterministic Semantics for Families of Paraconsistent Logics. In: Handbook of Paraconsistency. Studies in Logic, vol. 9, pp. 285–320. College Publications (2007)
3. Avron, A., Ben-Naim, J., Konikowska, B.: Cut-free ordinary sequent calculi for logics having generalized finite-valued semantics. Logica Universalis 1, 41–69 (2006)
4. Avron, A., Konikowska, B., Zamansky, A.: Modular construction of cut-free sequent calculi for paraconsistent logics. In: Proceedings of LICS 2012, pp. 85–94. IEEE (2012)
5. Avron, A., Zamansky, A.: Non-deterministic semantics for logical systems - A survey. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, vol. 16, pp. 227–304. Springer (2011)
6. Baaz, M., Lahav, O., Zamansky, A.: Effective finite-valued semantics for labelled calculi. In: Proceedings of IJCAR 2012, pp. 52–66 (2012)
7. Carnielli, W.A., Coniglio, M.E., Marcos, J.: Logics of formal inconsistency. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 14, 2nd edn., pp. 15–107. Springer (2007)
8. Carnielli, W.A., Marcos, J.: A taxonomy of C-systems. In: Carnielli, W.A., Coniglio, M.E., D'Ottaviano, I. (eds.) Paraconsistency: The Logical Way to the Inconsistent. Lecture Notes in Pure and Applied Mathematics, vol. 228, pp. 1–94. Marcel Dekker (2002)
9. Ciabattoni, A., Galatos, N., Terui, K.: From axioms to analytic rules in nonclassical logics. In: Proceedings of LICS 2008, pp. 229–240. IEEE (2008)
10. da Costa, N.C.A.: On the theory of inconsistent formal systems. Notre Dame Journal of Formal Logic 15, 497–510 (1974)
11. Ohlbach, J.: Computer support for the development and investigation of logics. Journal of the IGPL 4, 109–127 (1994)

A Modal BI Logic for Dynamic Resource Properties[⋆]

J.R. Courtault and D. Galmiche

Université de Lorraine – LORIA UMR 7503
Campus Scientifique, BP 239
Vandœuvre-lès-Nancy, France

Abstract. The logic of Bunched implications (BI) and its variants or extensions provide a powerful framework to deal with resources having static properties. In this paper, we propose a modal extension of BI logic, called DBI, which allows us to deal with dynamic resource properties. After defining a Kripke semantics for DBI, we illustrate the interest of DBI for expressing some dynamic properties and then we propose a labelled tableaux calculus for this logic. This calculus is proved sound and complete w.r.t. the Kripke semantics. Moreover, we also give a method for countermodel generation in this logic.

1 Introduction

The notion of *resource* is an important notion in computer science. The location, ownership, access to and, indeed, consumption of, resources are central concerns in the design of systems, such as networks, and in the design of programs, which access memory and manipulate data structures like pointers. We are interested in studying such notions on resources through logics with an emphasis on usable semantics and proof-theory. In this context we can mention Linear Logic (LL) [5] that focuses on resource consumption and the logic of Bunched Implications (BI) [13] that mainly focuses on resource sharing and separation. The BI logic and its variants, like Boolean BI (BBI) [11,13], can be seen as the logical kernel of so-called separation logics, that provides a concrete way of understanding the connectives in the context of program verification [7,14]. Some recent results on BI and BBI concern new semantics [4], proof-search with labelled tableaux and resource graphs [3,4] and (un)decidability of these logics [4,9]. Some extensions or refinements have led to separation logics, like BI's pointer logic (PL) [7] that allows us to express properties on pointers or BiLoc [1] that is based on resource trees and captures the notion of place. In this context MBI logic [12] extends BI with modalities and a calculus à la Hennessy-Milner [10] dealing with processes and resources.

We can remark that two kinds of dynamic are captured by BI, BBI and their extensions. On the one hand, there are logics that transform resources into other resources, which is a first kind of dynamic. On the other hand, there are logics where properties of resources can change (called here dynamic properties) or not (called here static properties). For example, in BI logic the resource properties are static because if a resource satisfies a property, it will always satisfies this property. The dynamic, that corresponds

⋆ This work is supported by the ANR grant DynRes on Dynamic Resources and Separation and Update Logics (project no. ANR-11-BS02-011).

to the transformation of resources, is captured in LL by proofs and in PL by a calculus à la Hoare [6]. Moreover in MBI, the dynamic is also based on resource transformation because of a calculus à la Hennessy-Milner with judgements of the form $R,E \xrightarrow{a} R',E'$, which means that a process E performs an action a on a resource R in order to obtain a resource R' and then becomes a process E'. But the modalities à la Hennessy-Milner can only express properties on R' and E', directly at the next state, but not on any reachable resource and process (or state), knowing that reachable means after performing any action.

In this paper, we are interested in expressing some dynamic properties on resources directly at level of formulae, on future states (and not only on the next ones) and in dealing with interacting systems. Then we define a modal extension of BI, called DBI (Dynamic Bunched Implications logic), in order to model some dynamic properties of resources. We define a Kripke semantics for this logic, which is an extension of Kripke semantics for BI with state constraints (a set of states with a preorder) introduced in addition to resource constraints. We also give a labelled tableaux calculus in the spirit of works on BI logic [3,4] but dealing with both resource graphs and state graphs. This calculus is proved sound and complete w.r.t. this semantics, with generation of countermodels in case of non-validity in DBI.

2 The DBI logic

BI logic is a logic that expresses sharing and separation properties on resources [11,13]. We present here a modal extension of BI, called DBI, which allows us to express some dynamic properties on resources. The language $\mathcal{L}$ of DBI is obtained by adding two modalities $\Box$ and $\Diamond$ to the BI language [13].

Let *Prop* be a countable set of propositional symbols, the language $\mathcal{L}$ of DBI is defined as follows, where $p \in Prop$:

$$X ::= p \mid \top \mid \bot \mid \mathrm{I} \mid X \wedge X \mid X \vee X \mid X \rightarrow X \mid X * X \mid X \mathrel{-\!\!*} X \mid \Diamond X \mid \Box X$$

The negation is defined by: $\neg X \equiv X \rightarrow \bot$. We now define a Kripke semantics that can be seen as an extension of the Kripke semantics of BI [4] based on a resource monoid. In the case of DBI we consider a dynamic resource monoid with an explicit inconsistency, and also a preorder set of states with an accessibility relation between states.

Definition 1 (Dynamic resource monoid). *A* dynamic resource monoid *is a structure* $\mathcal{M} = (R, \bullet, e, \pi, \sqsubseteq, S, \preceq)$ *such that:*

- *R is a set of* resources *and S is a set of* states
- *$e \in R$ and $\pi \in R$*
- *$\bullet : R \times R \rightarrow R$ such that:*
  - *Neutral element: $\forall r \in R,\ r \bullet e = e \bullet r = r$*
  - *Associativity: $\forall r_1, r_2, r_3 \in R,\ r_1 \bullet (r_2 \bullet r_3) = (r_1 \bullet r_2) \bullet r_3$*
  - *Commutativity: $\forall r_1, r_2 \in R,\ r_1 \bullet r_2 = r_2 \bullet r_1$*
- *$\sqsubseteq \subseteq R \times R$ is a preorder (on resources):*
  - *Reflexivity: $\forall r \in R,\ r \sqsubseteq r$*
  - *Transitivity: $\forall r_1, r_2, r_3 \in R$, if $r_1 \sqsubseteq r_2$ and $r_2 \sqsubseteq r_3$ then $r_1 \sqsubseteq r_3$*

- *$\pi \in R$ is the greatest element: $\forall r \in R$, $r \sqsubseteq \pi$ and $\forall r \in R$, $r \bullet \pi = \pi$.*
- *$\preceq \subseteq S \times S$ is a preorder (on states)*
- *Compatibility (P): $\forall r_1, r_2, r_3 \in R$, if $r_1 \sqsubseteq r_2$ then $r_1 \bullet r_3 \sqsubseteq r_2 \bullet r_3$*

We note $\mathbb{P}(E)$ the powerset of the set E, namely the set of sets built from E. We call e the *unit resource* (empty resource), π the *inconsistent resource* and $\bullet$ the *resource composition*. A preordered set $(S, \preceq)$ is added to the Kripke's BI semantics with S that can be viewed as the states of a system and $\preceq$ as the accessibility (through transitions) of states of the system.

Definition 2 (Dynamic interpretation). *A* dynamic interpretation *is a function $[\![\cdot]\!]$: $Prop \to \mathbb{P}(R \times S)$, that verifies the following properties, for any $s \in S$ and $p \in Prop$:*

- *Monotonicity (K): $\forall r, r' \in R$ such that $r \sqsubseteq r'$, if $(r,s) \in [\![p]\!]$ then $(r',s) \in [\![p]\!]$*
- *Inconsistency (BC): $\forall r \in R$ such that $\pi \sqsubseteq r$, $(r,s) \in [\![p]\!]$*

As we see the dynamic interpretation makes the resource properties non static: the interpretation of a propositional symbol is not only a set of resources (as BI), but a set of pairs of resources and states.

Definition 3 (Dynamic resource model). *A* dynamic resource model *is a triple $\mathcal{K} = (\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ such that $\mathcal{M}$ is a dynamic resource monoid, $[\![\cdot]\!]$ is a dynamic interpretation and $\vDash_{\mathcal{K}}$ is a forcing relation on $R \times S \times \mathcal{L}$ defined as follows:*

- *$r,s \vDash_{\mathcal{K}} p$ iff $(r,s) \in [\![p]\!]$*
- *$r,s \vDash_{\mathcal{K}} \mathrm{I}$ iff $e \sqsubseteq r$*
- *$r,s \vDash_{\mathcal{K}} \top$ always*
- *$r,s \vDash_{\mathcal{K}} \bot$ iff $\pi \sqsubseteq r$*
- *$r,s \vDash_{\mathcal{K}} \phi \wedge \psi$ iff $r,s \vDash_{\mathcal{K}} \phi$ and $r,s \vDash_{\mathcal{K}} \psi$*
- *$r,s \vDash_{\mathcal{K}} \phi \vee \psi$ iff $r,s \vDash_{\mathcal{K}} \phi$ or $r,s \vDash_{\mathcal{K}} \psi$*
- *$r,s \vDash_{\mathcal{K}} \phi \to \psi$ iff $\forall r' \in R \cdot (r \sqsubseteq r'$ and $r',s \vDash_{\mathcal{K}} \phi) \Rightarrow r',s \vDash_{\mathcal{K}} \psi$*
- *$r,s \vDash_{\mathcal{K}} \phi * \psi$ iff $\exists r', r'' \in R \cdot r' \bullet r'' \sqsubseteq r$ and $r',s \vDash_{\mathcal{K}} \phi$ and $r'',s \vDash_{\mathcal{K}} \psi$*
- *$r,s \vDash_{\mathcal{K}} \phi \mathrel{-\!\!*} \psi$ iff $\forall r' \in R \cdot r',s \vDash_{\mathcal{K}} \phi \Rightarrow r \bullet r',s \vDash_{\mathcal{K}} \psi$*
- *$r,s \vDash_{\mathcal{K}} \Diamond\phi$ iff $\exists s' \in S \cdot s \preceq s'$ and $r,s' \vDash_{\mathcal{K}} \phi$*
- *$r,s \vDash_{\mathcal{K}} \Box\phi$ iff $\forall s' \in S \cdot s \preceq s' \Rightarrow r,s' \vDash_{\mathcal{K}} \phi$*

The definition of the forcing relation is an extension of the BI forcing relation with the cases for $\Box$ and $\Diamond$. For instance $r,s \vDash_{\mathcal{K}} \Diamond\phi$ means that a resource r at state s satisfies $\Diamond\phi$ if a state s' can be reached from the state s ($s \preceq s'$) such that r in state s' satisfies ϕ ($r,s' \vDash_{\mathcal{K}} \phi$). Now we define the notion of validity.

Definition 4 (Validity). *A formula ϕ is* valid*, denoted $\vDash \phi$, if and only if $e,s \vDash_{\mathcal{K}} \phi$ for all dynamic resource models $\mathcal{K}$ (and all states $s \in S$).*

The notation $\phi \vDash \psi$ means that for all resources r and all states s of any dynamic resource model $\mathcal{K}$, if $r,s \vDash_{\mathcal{K}} \phi$ then $r,s \vDash_{\mathcal{K}} \psi$.

We give two lemmas that hold for all dynamic resource models $\mathcal{K}$, all formulae ϕ, all resources $r, r' \in R$ and all states $s' \in S$.

Lemma 1 (Monotonicity). *If $r,s \vDash_{\mathcal{K}} \phi$ and $r \sqsubseteq r'$ then $r',s \vDash_{\mathcal{K}} \phi$.*

Lemma 2 (Inconsistency). *We have $\pi,s \vDash_{\mathcal{K}} \phi$.*

3 Expressiveness of DBI

We have previously introduced a semantics for modelling resources having dynamic properties. In this section we emphasize the interest of this modal extension of BI by illustrating it through some simple examples.

The first example deals with the management of resources with dynamic properties. In BI logic the propositional symbols are considered as static descriptions/properties of resources. But, we know that resource properties are not always static. For example, if we consider the price of gold and silver, it is a dynamic property depending not only on the resource. Let us denote r_g the resource ”one ounce of gold” and r_s the resource ”one ounce of silver”. Propositional symbols P_{g_y} and P_{s_y} are prices of r_g and r_s on January 1st of the year y. Moreover, s_y denotes the state of the market on January 1st of the year y. With DBI we are able to express the evolution of the silver and gold price:

$$r_g \bullet r_s, s_{1970} \vDash_{\mathcal{K}} (P_{g_{1970}} * P_{s_{1970}}) \wedge \Diamond(P_{g_{2012}} * P_{s_{2012}})$$

It means that on January 1st of the year 1970 (s_{1970}), a resource composed by one ounce of gold and one ounce of silver ($r_g \bullet r_s$) has two properties: it could be decomposed into two resources respectively satisfying the properties $P_{g_{1970}}$ and $P_{s_{1970}}$ ($P_{g_{1970}} * P_{s_{1970}}$) and, in a future state, it could be decomposed into two resources respectively satisfying the properties $P_{g_{2012}}$ and $P_{s_{2012}}$ ($P_{g_{2012}} * P_{s_{2012}}$).

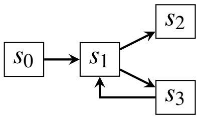
The second example illustrates how with DBI and a dynamic resource monoid we can deal with properties on interacting systems. A dynamic resource monoid can be viewed as two interacting systems. Indeed a resource monoid can model a first system, where resources are states of this system and the preoder on resources is the state reachability of this system [2]. Furthermore, the dynamic part of a dynamic resource monoid (set of states with a preorder), can be viewed as an automaton and easily models a second system. Moreover, the dynamic interpretation can be viewed as the result of the interaction of these systems. For example, $(r,s) \in [\![p]\!]$ can express that, if a first system is in state r and a second system is in state s then their interaction satisfies the property p. Here the word *interaction* does not mean that one of these systems influences the second one: the preorder on resources does not depend on states and the preorder on states does not depend on resources. Then the interaction $(r,s) \in [\![p]\!]$ means that there are two free (non influencing) systems which can perform together an action, which satisfies the property p if the first system is in state r and the second system is in state s.

Let us consider a message sent in a network and modelled with a resource monoid. We consider only five states (resources) $R = \{e, m_{sent}, m_{passing}, m_{delivered}, \pi\}$, where e is the state with no message, π is the state with an error that occurs in the system, m_{sent} is the state where the message is sent, $m_{passing}$ is the state where the message is passing in transit and $m_{delivered}$ is the state where the message is delivered. The relation $\sqsubseteq$, where reflexivity and transitivity are not represented, is:

$$\begin{tikzcd} & \boxed{\pi} & \boxed{e} \arrow[l] \\ \boxed{m_{delivered}} \arrow[r] \arrow[ur] & \boxed{m_{passing}} \arrow[r] \arrow[u] & \boxed{m_{sent}} \arrow[u] \arrow[ul] \end{tikzcd}$$

In a first step, there is no message (e). Then the message is created and sent (m_{sent}). In a third step, it is passing in transit ($m_{passing}$) and then, in a fourth step, it is delivered ($m_{delivered}$). As we can remark, $m_{passing} \sqsubseteq m_{sent}$, but $m_{passing}$ is the next state of m_{sent} and it is not a mistake. As m_{sent} can reach $m_{passing}$ then we aim the properties of $m_{passing}$ to be satisfied by the resource m_{sent}. In other words, if a resource r satisfies a property p, then all resources that can reach r satisfy p. This is the property (K) of Definition 2. In this example, we only consider one message and then we define $\bullet$ by $(e \bullet r = r)$ and $(r \bullet r' = \pi$ if $r \neq e$ and $r' \neq e)$, but it is possible to consider states composed by more than one message. We remark that π is the biggest resource (by definition of dynamic resource monoid), so when an error occurs (π), all states are reachable: it is considered that when an error occurs, it is impossible to predict the behavior of the system.

Now we define the following service as a second system, where reflexivity and transitivity of $\preceq$ are not represented. It contains four states $S = \{s_0, s_1, s_2, s_3\}$ with s_0 as initial state and in the state s_3 our service reads the delivered messages.



$$\begin{tikzcd} & & s_2 \\ s_0 \arrow[r] & s_1 \arrow[ur] \arrow[dr] & \\ & & s_3 \arrow[ul, bend left] \end{tikzcd}$$

Having defined a dynamic resource monoid we are able to express that when the message is sent, it is possible that our service read this message, that is: $m_{sent}, s_0 \vDash_{\mathcal{K}} \Diamond P_{m_{read}}$, where $P_{m_{read}}$ is the propositional symbol "message read" that occurs when m is delivered and the service is in state s_3: $[\![P_{m_{read}}]\!] = \{(r, s_3) \mid m_{delivered} \sqsubseteq r\}$.

We have $m_{delivered}, s_3 \vDash_{\mathcal{K}} P_{m_{read}}$. As $s_0 \preceq s_3$ then $m_{delivered}, s_0 \vDash_{\mathcal{K}} \Diamond P_{m_{read}}$ (the DBI modalities encode the reachability of states). As m_{sent} can reach $m_{delivered}$ ($m_{delivered} \sqsubseteq m_{sent}$) then $m_{sent}, s_0 \vDash_{\mathcal{K}} \Diamond P_{m_{read}}$ (DBI monotonicity encodes the resource reachability).

4 A proof System for DBI

In this section, we propose a proof system for DBI, in the spirit of previous works on labelled proof system for BI with resource graphs [4]. We introduce some rules to deal with modalities and also the notions of state labels and constraints, in order to capture some dynamic aspects.

4.1 Labels for Resources and States

In labelled tableaux method for BI [4], there are *labels* and *constraints* in order to capture some semantic information inside the proof system. Labels are related to the resource set (R), a label composition is related to the resource composition ($\bullet$) and relations on labels named *label constraints* are related to $\sqsubseteq$. In DBI, the resource monoids are dynamic and then there are two sets (for resources and states) and two relations (on resources and states). Thus we introduce a new kind of labels and constraints to deal with states. Let us now define labels and constraints for DBI.

Definition 5 (Resource labels). *L_r is a set of* resource labels *built from a constant* 1, *an infinite countable set of constants $\gamma_r = \{c_1, c_2, ...\}$ and a function denoted $\circ$,*

$$X ::= 1 \mid c_i \mid X \circ X$$

where $c_i \in \gamma_r$. Moreover $\circ$ is a function on L_r that is associative, commutative and 1 *is its unit. A* resource constraint *is an expression of the form $x \leq y$ where x and y are resource labels.*

For example the resource label $c_1 \circ 1 \circ c_2 \circ c_1$ is equal to the resource label $c_1 \circ c_1 \circ c_2$. We denote xy the resource label $x \circ y$. Moreover we say that x is a *resource sub-label* of y if and only if there exists z such that $x \circ z = y$. The set of resource sub-labels of x is denoted $\mathcal{E}(x)$.

Definition 6 (State labels). *L_s is an infinite countable set of* state labels *($L_s = \{l_1, l_2, ...\}$). A* state constraint *on such labels is an expression of the form $x \lhd y$, where x and y are state labels.*

Definition 7 (Domain). *Let $\mathcal{C}_r$ be a resource constraints set, the* domain *of $\mathcal{C}_r$, denoted $\mathcal{D}_r(\mathcal{C}_r)$, is the set of all resource sub-labels appearing in $\mathcal{C}_r$. In particular: $\mathcal{D}_r(\mathcal{C}_r) = \bigcup_{x \leq y \in \mathcal{C}_r} (\mathcal{E}(x) \cup \mathcal{E}(y))$.*

Definition 8 (Alphabet). *The* alphabet *of a set of resource / state constraints is the set of all label constants appearing in $\mathcal{C}_r$ / $\mathcal{C}_s$.*

In particular we have $\mathcal{A}_r(\mathcal{C}_r) = \gamma_r \cap \mathcal{D}_r(\mathcal{C}_r)$ and $\mathcal{A}_s(\mathcal{C}_s) = \bigcup_{u \lhd v \in \mathcal{C}_s} \{u, v\}$.

We can remark that $\sqsubseteq$ is reflexive, transitive and compatible. Moreover, $\preceq$ is reflexive and transitive. These properties have to be captured by the constraint sets. For that we introduce a notion of closure of constraints.

Definition 9 (Closure of resource constraints). *Let $\mathcal{C}_r$ be a set of resource constraints, the* closure *of $\mathcal{C}_r$ (denoted $\overline{\mathcal{C}_r}$) is the least relation closed under the following rules such that $\mathcal{C}_r \subseteq \overline{\mathcal{C}_r}$*

$$\frac{x \leq y \qquad y \leq z}{x \leq z}\langle t_r \rangle \quad \frac{xy \leq xy}{x \leq x}\langle d_r \rangle \quad \frac{ky \leq ky \qquad x \leq y}{kx \leq ky}\langle c_r \rangle \quad \frac{x \leq y}{x \leq x}\langle l_r \rangle \quad \frac{x \leq y}{y \leq y}\langle r_r \rangle$$

We can remark that as these rules do not introduce new resource label constants, then $\mathcal{A}_r(\mathcal{C}_r) = \mathcal{A}_r(\overline{\mathcal{C}_r})$.

Definition 10 (Closure of state constraints). *Let $\mathcal{C}_s$ be a set of state constraints, the* closure *of $\mathcal{C}_s$ (denoted $\overline{\mathcal{C}_s}$) is the least relation closed under the following rules such that $\mathcal{C}_s \subseteq \overline{\mathcal{C}_s}$:*

$$\frac{x \lhd y}{x \lhd x}\langle l_s \rangle \qquad \frac{x \lhd y}{y \lhd y}\langle r_s \rangle \qquad \frac{x \lhd y \qquad y \lhd z}{x \lhd z}\langle t_s \rangle$$

As illustration we consider $\mathcal{C}_s = \{l_1 \lhd l_2, l_2 \lhd l_3, l_3 \lhd l_4\}$. We have $l_1 \lhd l_2 \in \overline{\mathcal{C}_s}$ because $\mathcal{C}_s \subseteq \overline{\mathcal{C}_s}$ and we have $l_1 \lhd l_4 \in \overline{\mathcal{C}_s}$ because

$$\frac{\dfrac{l_1 \lhd l_2 \qquad l_2 \lhd l_3}{l_1 \lhd l_3}\langle t_s \rangle \qquad l_3 \lhd l_4}{l_1 \lhd l_4}\langle t_s \rangle$$

Proposition 1. *Let $\mathcal{C}_r$ be a set of resource constraints, the following properties hold:*

1. *If $kx \leq y \in \overline{\mathcal{C}_r}$ then $x \leq x \in \overline{\mathcal{C}_r}$*
2. *If $x \leq ky \in \overline{\mathcal{C}_r}$ then $y \leq y \in \overline{\mathcal{C}_r}$*

Corollary 1. *Let $\mathcal{C}_r$ be a set of resource constraints, $x \in \mathcal{D}_r(\overline{\mathcal{C}_r})$ iff $x \leq x \in \overline{\mathcal{C}_r}$.*

Lemma 3 (Compactness). *Let $\mathcal{C}_r$ (resp. $\mathcal{C}_s$) be a (possibly infinite) set of resource constraints (resp. state constraints). If $x \leq y \in \overline{\mathcal{C}_r}$ (resp. $u \lhd v \in \overline{\mathcal{C}_s}$) then there exists a finite set $\mathcal{C}_f$ such that $\mathcal{C}_f \subseteq \mathcal{C}_r$ (resp. $\mathcal{C}_f \subseteq \mathcal{C}_s$) and $x \leq y \in \overline{\mathcal{C}_f}$ (resp. $u \lhd v \in \overline{\mathcal{C}_f}$).*

4.2 A Labelled Tableaux Method for DBI

We now define a labelled tableaux method for DBI in the spirit of previous works for BI [4] and BBI [8].

Definition 11 (Labelled formula / CSS). *A* labelled formula *is a 4-uplet $(\mathbb{S}, \phi, x, u) \in \{\mathbb{T}, \mathbb{F}\} \times \mathcal{L} \times L_r \times L_s$ written $\mathbb{S}\phi : (x,u)$. A* constrained set of statements *(CSS) is a triple $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$, where $\mathcal{F}$ is a set of labelled formulae, $\mathcal{C}_r$ is a set of resource constraints and $\mathcal{C}_s$ is a set of state constraints, such that the following property, called (P_{css}), holds: if $\mathbb{S}\phi : (x,u) \in \mathcal{F}$ then $x \leq x \in \overline{\mathcal{C}_r}$ and $u \lhd u \in \overline{\mathcal{C}_s}$.*

A CSS $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ is a representation of a branch in which the formulae are the labelled formulae of $\mathcal{F}$ and the constraints on labels are the elements of $\mathcal{C}_r$ and $\mathcal{C}_s$. Our calculus extends some principles of BI calculus by adding a second kind of labels (state labels) and a set of constraints ($\mathcal{C}_s$) for state labels.

A CSS $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ is *finite* iff $\mathcal{F}$, $\mathcal{C}_r$ and $\mathcal{C}_s$ are finite. We define the relation $\preccurlyeq$ by: $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle \preccurlyeq \langle \mathcal{F}', \mathcal{C}_r', \mathcal{C}_s' \rangle$ iff $\mathcal{F} \subseteq \mathcal{F}'$ and $\mathcal{C}_r \subseteq \mathcal{C}_r'$ and $\mathcal{C}_s \subseteq \mathcal{C}_s'$. Moreover we denote $\langle \mathcal{F}_f, \mathcal{C}_{r_f}, \mathcal{C}_{s_f} \rangle \preccurlyeq_f \langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ when $\langle \mathcal{F}_f, \mathcal{C}_{r_f}, \mathcal{C}_{s_f} \rangle \preccurlyeq \langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ holds and $\langle \mathcal{F}_f, \mathcal{C}_{r_f}, \mathcal{C}_{s_f} \rangle$ is finite.

Definition 12 (Inconsistent label). *Let $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ be a CSS and x be a resource label. x is* inconsistent *if there exist two resource labels y and z such that $yz \leq x \in \overline{\mathcal{C}_r}$ and $\mathbb{T}\bot : (y,u) \in \mathcal{F}$. A label is* consistent *if it is not inconsistent.*

Proposition 2. *Let $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ be a CSS. The following properties hold:*

1. *If $y \leq x \in \overline{\mathcal{C}_r}$ and x is a consistent label then y is a consistent label.*
2. *If $xy \in \mathcal{D}_r(\overline{\mathcal{C}_r})$ is a consistent label then x and y are consistent labels.*

Figure 1 presents rules of labelled tableaux method for DBI. Let us remark that "c_i and c_j are new label constants" means $c_i \neq c_j \in \gamma_r \setminus \mathcal{A}_r(\mathcal{C}_r)$ and that "l_i is a new label constant" means $l_i \in L_s \setminus \mathcal{A}_s(\mathcal{C}_s)$. We note $\oplus$ the concatenation of lists. For example $[e_1; e_2; e_4] \oplus [e_4; e_3] = [e_1; e_2; e_4; e_4; e_3]$.

Definition 13 (DBI-tableau). *A* DBI-tableau *for a finite CSS $\langle \mathcal{F}_0, \mathcal{C}_{r_0}, \mathcal{C}_{s_0} \rangle$ is a list of CSS (*branches*), built inductively according the following rules:*

1. *The one branch list $[\langle \mathcal{F}_0, \mathcal{C}_{r_0}, \mathcal{C}_{s_0} \rangle]$ is a DBI-tableau for $\langle \mathcal{F}_0, \mathcal{C}_{r_0}, \mathcal{C}_{s_0} \rangle$*

$$\frac{\mathbb{T}\phi\wedge\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (x,u), \mathbb{T}\psi : (x,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{T}\wedge\rangle \qquad \frac{\mathbb{F}\phi\wedge\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{F}\phi : (x,u)\}, \emptyset, \emptyset\rangle \mid \langle\{\mathbb{F}\psi : (x,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{F}\wedge\rangle$$

$$\frac{\mathbb{T}\phi\vee\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (x,u)\}, \emptyset, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : (x,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{T}\vee\rangle \qquad \frac{\mathbb{F}\phi\vee\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{F}\phi : (x,u), \mathbb{F}\psi : (x,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{F}\vee\rangle$$

$$\frac{\mathbb{T}\mathrm{I} : (x,u)\in\mathcal{F}}{\langle\emptyset, \{1\leq x\}, \emptyset\rangle}\ \langle\mathbb{T}\mathrm{I}\rangle$$

$$\frac{\mathbb{T}\phi\to\psi : (x,u)\in\mathcal{F} \text{ and } x\leq y\in\overline{\mathcal{C}_r}}{\langle\{\mathbb{F}\phi : (y,u)\}, \emptyset, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : (y,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{T}\to\rangle \qquad \frac{\mathbb{F}\phi\to\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (c_i,u), \mathbb{F}\psi : (c_i,u)\}, \{x\leq c_i\}, \emptyset\rangle}\ \langle\mathbb{F}\to\rangle$$

$$\frac{\mathbb{T}\phi*\psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (c_i,u), \mathbb{T}\psi : (c_j,u)\}, \{c_ic_j\leq x\}, \emptyset\rangle}\ \langle\mathbb{T}*\rangle \qquad \frac{\mathbb{F}\phi*\psi : (x,u)\in\mathcal{F} \text{ and } yz\leq x\in\overline{\mathcal{C}_r}}{\langle\{\mathbb{F}\phi : (y,u)\}, \emptyset, \emptyset\rangle \mid \langle\{\mathbb{F}\psi : (z,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{F}*\rangle$$

$$\frac{\mathbb{T}\phi -\!\!* \psi : (x,u)\in\mathcal{F} \text{ and } xy\leq xy\in\overline{\mathcal{C}_r}}{\langle\{\mathbb{F}\phi : (y,u)\}, \emptyset, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : (xy,u)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{T}-\!\!*\rangle \qquad \frac{\mathbb{F}\phi -\!\!* \psi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (c_i,u), \mathbb{F}\psi : (xc_i,u)\}, \{xc_i\leq xc_i\}, \emptyset\rangle}\ \langle\mathbb{F}-\!\!*\rangle$$

$$\frac{\mathbb{T}\Diamond\phi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{T}\phi : (x,l_i)\}, \emptyset, \{u\lhd l_i\}\rangle}\ \langle\mathbb{T}\Diamond\rangle \qquad \frac{\mathbb{F}\Diamond\phi : (x,u)\in\mathcal{F} \text{ and } u\leq v\in\overline{\mathcal{C}_s}}{\langle\{\mathbb{F}\phi : (x,v)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{F}\Diamond\rangle$$

$$\frac{\mathbb{T}\Box\phi : (x,u)\in\mathcal{F} \text{ and } u\leq v\in\overline{\mathcal{C}_s}}{\langle\{\mathbb{T}\phi : (x,v)\}, \emptyset, \emptyset\rangle}\ \langle\mathbb{T}\Box\rangle \qquad \frac{\mathbb{F}\Box\phi : (x,u)\in\mathcal{F}}{\langle\{\mathbb{F}\phi : (x,l_i)\}, \emptyset, \{u\lhd l_i\}\rangle}\ \langle\mathbb{F}\Box\rangle$$

Note: c_i, c_j and l_i are new label constants.

Fig. 1. Tableaux rules for DBI

2. *If the list $\mathcal{T}_m \oplus [\langle\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle] \oplus \mathcal{T}_n$ is a DBI-tableau for $\langle\mathcal{F}_0, \mathcal{C}_{r_0}, \mathcal{C}_{s_0}\rangle$ and*

$$\frac{cond(\langle\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle)}{\langle\mathcal{F}_1, \mathcal{C}_{r_1}, \mathcal{C}_{s_1}\rangle \mid \ldots \mid \langle\mathcal{F}_k, \mathcal{C}_{r_k}, \mathcal{C}_{s_k}\rangle}$$

is an instance of a rule of Figure 1 for which $cond(\langle\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle)$ is fulfilled, then the list $\mathcal{T}_m \oplus [\langle\mathcal{F}\cup\mathcal{F}_1, \mathcal{C}_r\cup\mathcal{C}_{r_1}, \mathcal{C}_s\cup\mathcal{C}_{s_1}\rangle; \ldots; \langle\mathcal{F}\cup\mathcal{F}_k, \mathcal{C}_r\cup\mathcal{C}_{r_k}, \mathcal{C}_s\cup\mathcal{C}_{s_k}\rangle] \oplus \mathcal{T}_n$ is a DBI-tableau for $\langle\mathcal{F}_0, \mathcal{C}_{r_0}, \mathcal{C}_{s_0}\rangle$.

A DBI-tableau for a formula ϕ is a DBI-tableau for $\langle\{\mathbb{F}\phi : (1,l_1)\}, \{1\leq 1\}, \{l_1\lhd l_1\}\rangle$.

It is possible to prove, by observing rules of the tableaux method for DBI, that new CSS, obtained by applying a rule, respect the condition (P_{css}) of Definition 11. Then, for all branches $\langle\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle$ of a DBI-tableau for a formula ϕ, as $\mathbb{F}\phi : (1,l_1)\in\mathcal{F}$, then $1\leq 1\in\overline{\mathcal{C}_r}$ and $1\in\mathcal{D}_r(\overline{\mathcal{C}_r})$.

A first kind of rules concerns $\langle\mathbb{T}\mathrm{I}\rangle$, $\langle\mathbb{F}\to\rangle$, $\langle\mathbb{T}*\rangle$, $\langle\mathbb{F}-\!\!*\rangle$, $\langle\mathbb{T}\Diamond\rangle$ and $\langle\mathbb{F}\Box\rangle$. These rules introduce new constraints and also new label constants (c_i, c_j and l_i), except for $\langle\mathbb{T}\mathrm{I}\rangle$ that only introduces a new constraint. Let us illustrate the $\langle\mathbb{T}\Diamond\rangle$ rule. To apply this rule on a CSS $\langle\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle$ on a labelled formula $\mathbb{T}\Diamond\phi : (c_1,l_3)\in\mathcal{F}$, we choose a new label which does not appear in $\mathcal{C}_s$. For example, we say that $l_{10} \notin \mathcal{C}_s$. Thus, by choosing l_{10}, we can apply the rule, getting the new CSS $\langle \mathcal{F} \cup \{\mathbb{T}\phi : (c_1, l_{10})\}, \mathcal{C}_r, \mathcal{C}_s \cup \{l_3 \lhd l_{10}\}\rangle$. We notice the new state constraint $l_3 \lhd l_{10}$ added to the set of constraints. Let us observe that the $\langle \mathbb{T}*\rangle$ rule introduces two new resource labels. Concerning the rule $\langle \mathbb{F}\text{-}\!*\rangle$, as $\mathbb{F}\psi : (xc_i, u)$ is added to the set of labelled formulae, xc_i has to belong to $\overline{\mathcal{C}_r}$ in order to satisfy the condition (P_{css}) of Definition 11. By adding $xc_i \leq xc_i$ to $\mathcal{C}_r$, xc_i belongs to $\overline{\mathcal{C}_r}$ and so (P_{css}) is satisfied.

A second kind of rules concerns $\langle \mathbb{T} \rightarrow\rangle$, $\langle \mathbb{F}*\rangle$, $\langle \mathbb{T}\text{-}\!*\rangle$, $\langle \mathbb{F}\Diamond\rangle$ and $\langle \mathbb{T}\Box\rangle$. These rules have a condition on a closure of label constraints. In order to apply one of these rules we have to choose an existing label which satisfies the condition and then apply the rule using it. Otherwise, we cannot apply such rules. We illustrate the $\langle \mathbb{T}\Box\rangle$ rule: let a CSS $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle$ such that $\mathbb{T}\Box\phi : (c_1, l_1) \in \mathcal{F}$. To apply this rule, we have to choose a state label l such that $l_1 \lhd l \in \overline{\mathcal{C}_s}$. If we consider that $l_1 \leq l_2 \in \overline{\mathcal{C}_s}$ then we can decide to apply the rule using l_2, getting the CSS $\langle \mathcal{F} \cup \{\mathbb{T}\phi : (c_1, l_2)\}, \mathcal{C}_r, \mathcal{C}_s\rangle$. Let us observe that $\langle \mathbb{F}*\rangle$ rule needs to choose two labels y and z such that $yz \leq x \in \overline{\mathcal{C}_r}$.

Definition 14 (Closure condition). *A CSS $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s\rangle$ is* closed *if one of the following conditions holds:*

1. $\mathbb{T}\phi : (x,u) \in \mathcal{F}$, $\mathbb{F}\phi : (y,u) \in \mathcal{F}$ *and* $x \leq y \in \overline{\mathcal{C}_r}$
2. $\mathbb{F}\mathrm{I} : (x,u) \in \mathcal{F}$ *and* $1 \leq x \in \overline{\mathcal{C}_r}$
3. $\mathbb{F}\top : (x,u) \in \mathcal{F}$
4. $\mathbb{F}\phi : (x,u) \in \mathcal{F}$ *and x is inconsistent*

A CSS is open *if it is not closed. A DBI-tableau is closed if all its branches are closed.*

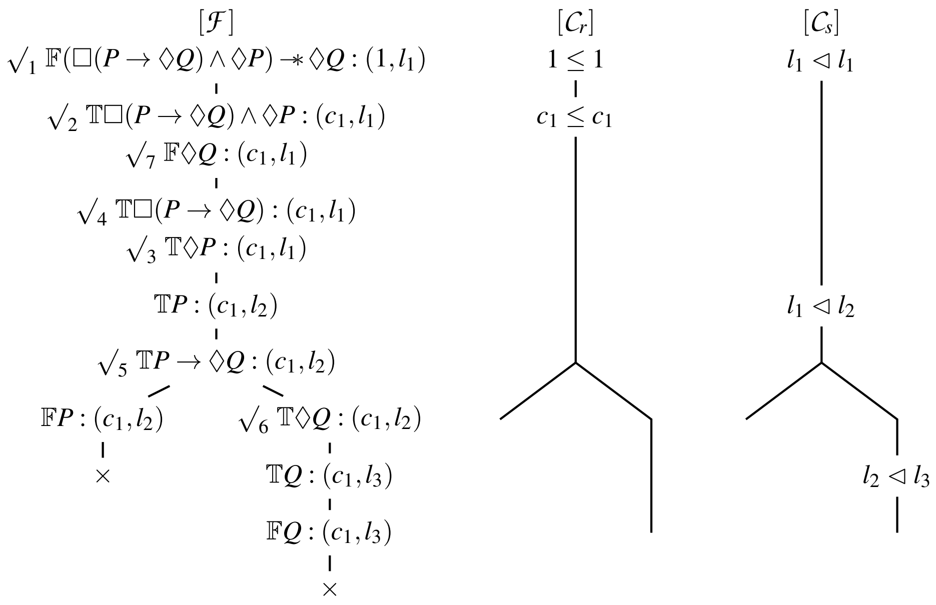
Definition 15 (DBI-proof). *A* DBI-proof *for a formula ϕ is a DBI-tableau for ϕ which is closed.*

Let us recall that we deal with labelled formulae with two kinds of labels: resource labels and state labels. Each CSS (branch) contains two sets of constraints, one for resources and another for states. Moreover the closure of such constraints can be represented by graphs. There are rules which modify constraint sets (graphs) and introduce new labels. Other rules have a set of conditions that must be satisfied, by finding labels satisfying it and then to solve constraints on the constraint graphs.

Let us now consider the formula $\phi \equiv (\Box(P \rightarrow \Diamond Q) \wedge \Diamond P) \text{-}\!* \Diamond Q$ and give a DBI-proof for it. By Definition 13, the following DBI-tableau $[\langle\{\mathbb{F}\phi : (1, l_1)\}, \{1 \leq 1\}, \{l_1 \lhd l_1\}\rangle]$ is a DBI-tableau for ϕ. We introduce a new representation for a DBI-tableau, which is

$[\mathcal{F}]$	$[\mathcal{C}_r]$	$[\mathcal{C}_s]$
$\mathbb{F}(\Box(P \rightarrow \Diamond Q) \wedge \Diamond P) \text{-}\!* \Diamond Q : (1, l_1)$	$1 \leq 1$	$l_1 \lhd l_1$

We can observe that there are three columns, one for the labelled formula sets of the CSS of the DBI-tableau ($[\mathcal{F}]$), one for the resource constraint sets of the CSS of the DBI-tableau ($[\mathcal{C}_r]$) and one for the state constraint sets of the CSS of the DBI-tableau ($[\mathcal{C}_s]$). By applying some rules, we obtain the following DBI-tableau:



$[\mathcal{F}]$
$\surd_1\ \mathbb{F}(\Box(P \to \Diamond Q) \wedge \Diamond P) \twoheadrightarrow \Diamond Q : (1, l_1)$
$\surd_2\ \mathbb{T}\Box(P \to \Diamond Q) \wedge \Diamond P : (c_1, l_1)$
$\surd_7\ \mathbb{F}\Diamond Q : (c_1, l_1)$
$\surd_4\ \mathbb{T}\Box(P \to \Diamond Q) : (c_1, l_1)$
$\surd_3\ \mathbb{T}\Diamond P : (c_1, l_1)$
$\mathbb{T}P : (c_1, l_2)$
$\surd_5\ \mathbb{T}P \to \Diamond Q : (c_1, l_2)$
$\mathbb{F}P : (c_1, l_2)$
$\times$
$\surd_6\ \mathbb{T}\Diamond Q : (c_1, l_2)$
$\mathbb{T}Q : (c_1, l_3)$
$\mathbb{F}Q : (c_1, l_3)$
$\times$
$[\mathcal{C}_r]$
$1 \leq 1$
$c_1 \leq c_1$
$[\mathcal{C}_s]$
$l_1 \lhd l_1$
$l_1 \lhd l_2$
$l_2 \lhd l_3$

We decorate a labelled formula with $\surd_i$ to show that we apply a rule on this formula at step i. We remark that columns ($[\mathcal{F}]$, $[\mathcal{C}_r]$ and $[\mathcal{C}_s]$) are trees that contain two branches. There are two branches because there are two CSS in the DBI-tableau. The branches on the left (resp. right) contain the elements of the first (resp. second) CSS. We also remark that all CSS are closed (denoted $\times$). The CSS of the left is closed because $\mathbb{T}P : (c_1, l_2) \in \mathcal{F}$, $\mathbb{F}P : (c_1, l_2) \in \mathcal{F}$ and $c_1 \leq c_1 \in \overline{\mathcal{C}_r}$. Thus, by definition, this DBI-tableau is a DBI-proof of $(\Box(P \to \Diamond Q) \wedge \Diamond P) \twoheadrightarrow \Diamond Q$.

5 Soundness and Completeness Results

The soundness proof uses similar techniques than the ones used in BI for a labelled tableaux method [4]. The key point is the notion of *realizability* of a CSS $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$, that means there exists a dynamic model $\mathcal{K}$ and embeddings from resource labels to the resource set ($\lfloor \cdot \rfloor$) and state labels to the state set ($\lceil \cdot \rceil$) of $\mathcal{K}$ such that if $\mathbb{T}\phi : (x, u) \in \mathcal{F}$ then $\lfloor x \rfloor, \lceil u \rceil \vDash_{\mathcal{K}} \phi$ and if $\mathbb{F}\phi : (x, u) \in \mathcal{F}$ then $\lfloor x \rfloor, \lceil u \rceil \nvDash_{\mathcal{K}} \phi$.

Definition 16 (Realization). *Let $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ be a CSS. A* realization *of it is a triple $(\mathcal{K}, \lfloor . \rfloor, \lceil . \rceil)$ such that $\mathcal{K} = (\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ is a dynamic resource model, $\mathcal{M} = (R, \bullet, e, \pi, \sqsubseteq, S, \preceq)$, $\lfloor . \rfloor : \mathcal{D}_r(\overline{\mathcal{C}_r}) \to R$ and $\lceil . \rceil : \mathcal{A}_s(\mathcal{C}_s) \to S$, such that:*

- $\lfloor 1 \rfloor = e$
- $\lfloor x \circ y \rfloor = \lfloor x \rfloor \bullet \lfloor y \rfloor$
- *If $\mathbb{T}\phi : (x, u) \in \mathcal{F}$ then $\lfloor x \rfloor, \lceil u \rceil \vDash_{\mathcal{K}} \phi$*
- *If $\mathbb{F}\phi : (x, u) \in \mathcal{F}$ then $\lfloor x \rfloor, \lceil u \rceil \nvDash_{\mathcal{K}} \phi$*

– *If* $x \le y \in \mathcal{C}_r$ *then* $\lfloor x \rfloor \sqsubseteq \lfloor y \rfloor$
– *If* $u \lhd v \in \mathcal{C}_s$ *then* $\lceil u \rceil \preceq \lceil v \rceil$

We say that a CSS/branch is *realizable* if there exists a realization of it. We say that a tableau is *realizable* if it contains a realizable CSS/branch.

Lemma 4. *Let* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *be a CSS and* $(\mathcal{K}, \lfloor . \rfloor, \lceil . \rceil)$ *a realization of it. For all* $x \le y \in \overline{\mathcal{C}_r}$, $\lfloor x \rfloor \sqsubseteq \lfloor y \rfloor$ *and for all* $u \lhd v \in \overline{\mathcal{C}_s}$, $\lceil u \rceil \preceq \lceil v \rceil$.

Lemma 5. *The closed DBI-tableaux are not realizable.*

Lemma 6. *The expansion rules preserve realizability, i.e., if a rule of the DBI-tableau method is applied on a labelled formula of a realizable CSS then one of the obtained CSS is realizable.*

Theorem 1 (Soundness). *Let* ϕ *be a formula, if there exists a DBI-proof of* ϕ *then* ϕ *is valid.*

Proof. Let $\mathcal{T}$ be a DBI-proof of ϕ. Let us assume that ϕ is not valid. Then there exits a dynamic resource model $\mathcal{K}$ such that $e, s \not\vDash_{\mathcal{K}} \phi$. If we consider $\lfloor 1 \rfloor = e$ and $\lceil l_1 \rceil = s$ we obtain a realisation $(\mathcal{K}, \lfloor . \rfloor, \lceil . \rceil)$ of the initial CSS $\langle \{\mathbb{F}\phi : (1, l_1)\}, \{1 \le 1\}, \{l_1 \lhd l_1\} \rangle$. Thus, by Lemma 6, one branch of $\mathcal{T}$ is realizable. But by Lemma 5 it is contradictory, because as $\mathcal{T}$ is a DBI-proof, then $\mathcal{T}$ is closed. Thus ϕ is valid.

Before to study completeness we consider the countermodel extraction for DBI tableaux method. The main idea consists in transforming resource and state constraints into a dynamic resource monoid, from a branch $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ which is not closed.

In order to obtain a countermodel, this transformation has to verify two properties: if $\mathbb{T}\phi : (x,u) \in \mathcal{F}$ then $x, u \vDash_{\mathcal{K}} \phi$ and if $\mathbb{F}\phi : (x,u) \in \mathcal{F}$ then $x, u \not\vDash_{\mathcal{K}} \phi$. In order to satisfy them, our method needs to *saturate* labelled formulae (to obtain a Hintikka CSS), that means, for instance, if $\mathbb{T}\Box\phi : (x,u) \in \mathcal{F}$ then we want that $x, u \vDash_{\mathcal{K}} \Box\phi$, so for all state labels v such that $u \lhd v \in \overline{\mathcal{C}_s}$, $\mathbb{T}\phi : (x,v) \in \mathcal{F}$ has to be verified.

Definition 17 (Hintikka CSS). *A CSS* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *is a* Hintikka CSS *if for any formula* $\phi, \psi \in \mathcal{L}$ *and any label* $x, y \in L_r$ *and* $u, v \in L_s$*:*

1. $\mathbb{T}\phi : (x,u) \notin \mathcal{F}$ *or* $\mathbb{F}\phi : (y,u) \notin \mathcal{F}$ *or* $x \le y \notin \overline{\mathcal{C}_r}$
2. $\mathbb{F}\mathrm{I} : (x,u) \notin \mathcal{F}$ *or* $1 \le x \notin \overline{\mathcal{C}_r}$
3. $\mathbb{F}\top : (x,u) \notin \mathcal{F}$
4. $\mathbb{F}\phi : (x,u) \notin \mathcal{F}$ *or x is consistent*
5. *If* $\mathbb{T}\mathrm{I} : (x,u) \in \mathcal{F}$ *then* $1 \le x \in \overline{\mathcal{C}_r}$
6. *If* $\mathbb{T}\phi \wedge \psi : (x,u) \in \mathcal{F}$ *then* $\mathbb{T}\phi : (x,u) \in \mathcal{F}$ *and* $\mathbb{T}\psi : (x,u) \in \mathcal{F}$
7. *If* $\mathbb{F}\phi \wedge \psi : (x,u) \in \mathcal{F}$ *then* $\mathbb{F}\phi : (x,u) \in \mathcal{F}$ *or* $\mathbb{F}\psi : (x,u) \in \mathcal{F}$
8. *If* $\mathbb{T}\phi \vee \psi : (x,u) \in \mathcal{F}$ *then* $\mathbb{T}\phi : (x,u) \in \mathcal{F}$ *or* $\mathbb{T}\psi : (x,u) \in \mathcal{F}$
9. *If* $\mathbb{F}\phi \vee \psi : (x,u) \in \mathcal{F}$ *then* $\mathbb{F}\phi : (x,u) \in \mathcal{F}$ *and* $\mathbb{F}\psi : (x,u) \in \mathcal{F}$
10. *If* $\mathbb{T}\phi \to \psi : (x,u) \in \mathcal{F}$ *then* $\forall y \in L_r$, $x \le y \in \overline{\mathcal{C}_r} \Rightarrow \mathbb{F}\phi : (y,u) \in \mathcal{F}$ *or* $\mathbb{T}\psi : (y,u) \in \mathcal{F}$

11. *If* $\mathbb{F}\phi \to \psi : (x,u) \in \mathcal{F}$ *then* $\exists y \in L_r$, $x \le y \in \overline{\mathcal{C}_r}$ *and* $\mathbb{T}\phi : (y,u) \in \mathcal{F}$ *and* $\mathbb{F}\psi : (y,u) \in \mathcal{F}$
12. *If* $\mathbb{T}\phi * \psi : (x,u) \in \mathcal{F}$ *then* $\exists y,z \in L_r$, $yz \le x \in \overline{\mathcal{C}_r}$ *and* $\mathbb{T}\phi : (y,u) \in \mathcal{F}$ *and* $\mathbb{T}\psi : (z,u) \in \mathcal{F}$
13. *If* $\mathbb{F}\phi * \psi : (x,u) \in \mathcal{F}$ *then* $\forall y,z \in L_r$, $yz \le x \in \overline{\mathcal{C}_r} \Rightarrow \mathbb{F}\phi : (y,u) \in \mathcal{F}$ *or* $\mathbb{F}\psi : (z,u) \in \mathcal{F}$
14. *If* $\mathbb{T}\phi -\!\!* \psi : (x,u) \in \mathcal{F}$ *then* $\forall y \in L_r$, $xy \in \mathcal{D}_r(\overline{\mathcal{C}_r}) \Rightarrow \mathbb{F}\phi : (y,u) \in \mathcal{F}$ *or* $\mathbb{T}\psi : (xy,u) \in \mathcal{F}$
15. *If* $\mathbb{F}\phi -\!\!* \psi : (x,u) \in \mathcal{F}$ *then* $\exists y \in L_r$, $xy \in \mathcal{D}_r(\overline{\mathcal{C}_r})$ *and* $\mathbb{T}\phi : (y,u) \in \mathcal{F}$ *and* $\mathbb{F}\psi : (xy,u) \in \mathcal{F}$
16. *If* $\mathbb{T}\Diamond\phi : (x,u) \in \mathcal{F}$ *then* $\exists v \in L_s$, $u \lhd v \in \overline{\mathcal{C}_s}$ *and* $\mathbb{T}\phi : (x,v) \in \mathcal{F}$
17. *If* $\mathbb{F}\Diamond\phi : (x,u) \in \mathcal{F}$ *then* $\forall v \in L_s$, $u \lhd v \in \overline{\mathcal{C}_s} \Rightarrow \mathbb{F}\phi : (x,v) \in \mathcal{F}$
18. *If* $\mathbb{T}\Box\phi : (x,u) \in \mathcal{F}$ *then* $\forall v \in L_s$, $u \lhd v \in \overline{\mathcal{C}_s} \Rightarrow \mathbb{T}\phi : (x,v) \in \mathcal{F}$
19. *If* $\mathbb{F}\Box\phi : (x,u) \in \mathcal{F}$ *then* $\exists v \in L_s$, $u \lhd v \in \overline{\mathcal{C}_s}$ *and* $\mathbb{F}\phi : (x,v) \in \mathcal{F}$

The conditions (1), (2), (3) and (4) of Definition 17 certify that a Hintikka CSS is not closed. Others conditions certify that all labelled formulae of a Hintikka CSS are saturated. Let us now define a function Ω that allows us to extract a countermodel from a Hintikka CSS.

Definition 18 (Function Ω). *Let* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *be a Hintikka CSS and* $\overline{\mathcal{C}_r}_\omega$ *be the restriction of* $\overline{\mathcal{C}_r}$ *to constraints including only consistent labels. The function* Ω *associates to* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *a triple* $\Omega(\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle) = (\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ *where* $\mathcal{M} = (R, \bullet, e, \pi, \sqsubseteq, S, \preceq)$*, such that:*

- $R = \mathcal{D}_r(\overline{\mathcal{C}_r}_\omega) \cup \{\pi\}$, *with* $\pi \notin \mathcal{D}_r(\overline{\mathcal{C}_r})$
- $S = \mathcal{A}_s(\mathcal{C}_s)$
- $e = 1$
- $\bullet$ *is defined by:* $\forall r_1, r_2 \in R \begin{cases} r_1 \bullet r_2 = r_1 \circ r_2 & \text{if } r_1 \circ r_2 \in \mathcal{D}_r(\overline{\mathcal{C}_r}_\omega) \\ r_1 \bullet r_2 = \pi & \text{otherwise} \end{cases}$
- $r_1 \sqsubseteq r_2$ *iff* $r_1 \le r_2 \in \overline{\mathcal{C}_r}_\omega$ *or* $r_2 = \pi$
- $s_1 \preceq s_2$ *iff* $s_1 \lhd s_2 \in \overline{\mathcal{C}_s}$
- $(r,s) \in [\![P]\!]$ *iff* $(r = \pi)$ *or* $(\exists r' \in R, r' \sqsubseteq r$ *and* $\mathbb{T}P : (r',s) \in \mathcal{F})$

Let $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ be a CSS and $x \in \mathcal{D}_r(\overline{\mathcal{C}_r})$. We remark that x is a consistent label resource if and only if $x \in \mathcal{D}_r(\overline{\mathcal{C}_r}_\omega)$. Indeed, if $x \in \mathcal{D}_r(\overline{\mathcal{C}_r})$ then by Corollary 1, $x \le x \in \overline{\mathcal{C}_r}$. Thus, as x is consistent, all resource labels and sub-labels of x are consistent by Proposition 2. Thus $x \le x \in \overline{\mathcal{C}_r}_\omega$ and $x \in \mathcal{D}_r(\overline{\mathcal{C}_r}_\omega)$. Now, if $x \in \mathcal{D}_r(\overline{\mathcal{C}_r}_\omega)$ then there exist $xy \le z \in \overline{\mathcal{C}_r}_\omega$ or $z \le xy \in \overline{\mathcal{C}_r}_\omega$. Therefore x is consistent otherwise $xy \le z \notin \overline{\mathcal{C}_r}_\omega$ or $z \le xy \notin \overline{\mathcal{C}_r}_\omega$.

Lemma 7. *Let* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *be a Hintikka CSS and* $\Omega(\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle) = (\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ *where* $\mathcal{M} = (R, \bullet, e, \pi, \sqsubseteq, S, \preceq)$. $(\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ *is a dynamic resource model.*

Lemma 8. *Let* $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ *be a Hintikka CSS. Let* $\Omega(\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle) = (\mathcal{M}, [\![\cdot]\!], \vDash_{\mathcal{K}})$ *where* $\mathcal{M} = (R, \bullet, e, \pi, \sqsubseteq, S, \preceq)$. *For any formula* ϕ *the following properties hold:*

1. $\pi, s \vDash_{\mathcal{K}} \phi$
2. *If* $\mathbb{F}\phi : (r,s) \in \mathcal{F}$ *and* r *consistent then* $r, s \nvDash_{\mathcal{K}} \phi$
3. *If* $\mathbb{T}\phi : (r,s) \in \mathcal{F}$ *and* r *consistent then* $r, s \vDash_{\mathcal{K}} \phi$

Lemma 9. *Let $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$ be a Hintikka CSS such that $\mathbb{F}\phi : (1, s) \in \mathcal{F}$. ϕ is not valid.*

Proof. If the resource label 1 is inconsistent, then it is contradictory because $\mathbb{F}\phi : (1, s) \in \mathcal{F}$ and by condition (4) of Definition 17. Thus 1 is consistent. By Lemma 7, $\Omega(\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle)$ is a dynamic resource model. By Lemma 8, $e, s \not\models_{\mathcal{K}} \phi$ in this model. Thus $\Omega(\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle)$ is a countermodel of ϕ and then ϕ is not valid.

The proof of completeness consists in building a Hintikka CSS from a CSS which cannot be closed, in the spirit of the proof developed for BBI [8]. Then we need a fair strategy and a oracle which contains all finite *consistent* (not closed but saturated) CSS.

Definition 19 (Fair strategy). *A* fair strategy *is a labelled formulae sequence $(\mathbb{S}_i F_i : (x_i, u_i))_{i \in \mathbb{N}}$ in $\{\mathbb{T}, \mathbb{F}\} \times \mathcal{L} \times L_r \times L_s$ such that any labelled formula occurs infinitely many times in this sequence, that is $\{i \in \mathbb{N} \mid \mathbb{S}_i F_i : (x_i, u_i) \equiv \mathbb{S}F : (x, u)\}$ is infinite for any $\mathbb{S}F : (x, u) \in \{\mathbb{T}, \mathbb{F}\} \times \mathcal{L} \times L_r \times L_s$.*

Proposition 3. *There exists a fair strategy.*

The main argument is that the set of labelled formulae is countable.

Definition 20. *Let $\mathcal{P}$ be a set of CSS.*

1. *$\mathcal{P}$ is* $\preccurlyeq$-closed *if $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle \in \mathcal{P}$ holds whenever $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle \preccurlyeq \langle \mathcal{F}', \mathcal{C}'_r, \mathcal{C}'_s \rangle$ and $\langle \mathcal{F}', \mathcal{C}'_r, \mathcal{C}'_s \rangle \in \mathcal{P}$ hold.*
2. *$\mathcal{P}$ is of* finite character *if $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle \in \mathcal{P}$ holds whenever $\langle \mathcal{F}_f, \mathcal{C}_{rf}, \mathcal{C}_{sf} \rangle \in \mathcal{P}$ holds for every $\langle \mathcal{F}_f, \mathcal{C}_{rf}, \mathcal{C}_{sf} \rangle \preccurlyeq_f \langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle$.*
3. *$\mathcal{P}$ is* saturated *if for any $\langle \mathcal{F}, \mathcal{C}_r, \mathcal{C}_s \rangle \in \mathcal{P}$ and any instance*

$$\frac{cond(\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s)}{\langle \mathcal{F}_1, \mathcal{C}_{r1}, \mathcal{C}_{s1} \rangle \ \mid \ \dots \ \mid \ \langle \mathcal{F}_k, \mathcal{C}_{rk}, \mathcal{C}_{sk} \rangle}$$

of a rule of Figure 7, if $cond(\mathcal{F}, \mathcal{C}_r, \mathcal{C}_s)$ is fulfilled then $\langle \mathcal{F} \cup \mathcal{F}_i, \mathcal{C}_r \cup \mathcal{C}_{ri}, \mathcal{C}_s \cup \mathcal{C}_{si} \rangle \in \mathcal{P}$ for at least one $i \in \{1, \dots, k\}$.

Definition 21 (Oracle). *An* oracle *is a set of non closed CSS which is $\preccurlyeq$-closed, of finite character and saturated.*

Lemma 10. *There exists an oracle which contains every finite CSS for which there exists no closed DBI-tableau.*

This oracle is the set of all CSS such that there exists no closed DBI-tableau for their finite sub-CSS ($\preccurlyeq$). Let us assume that there exists no DBI-proof of formula φ and show that φ is not valid by constructing a Hintikka CSS. Let us note that φ denotes the formula for which we are constructing a Hintikka CSS and ϕ denotes any formula. Let $\mathcal{T}_0$ a initial DBI-tableau for φ, we have

1. $\mathcal{T}_0 = [\langle \{\mathbb{F}\varphi : (1, l_1)\}, \{1 \leq 1\}, \{l_1 \lhd l_1\} \rangle]$
2. $\mathcal{T}_0$ cannot be closed

By Lemma 10, there exists an oracle which contains every finite CSS for which there exists no closed DBI-tableau. Let $\mathcal{P}$ be such an oracle. By hypothesis we have $\langle\{\mathbb{F}\varphi : (1,l_1)\},\{1 \leq 1\},\{l_1 \lhd l_1\}\rangle \in \mathcal{P}$. By Proposition 3, there exists a fair strategy. Let $\mathcal{S}$ be such a strategy. We denoted $\mathbb{S}_i F_i : (x_i,u_i)$ the i^{th} formula of $\mathcal{S}$. We built a sequence $\langle\mathcal{F}_i,\mathcal{C}_{ri},\mathcal{C}_{si}\rangle_{0\leqslant i}$ as follows:

– $\langle\mathcal{F}_0,\mathcal{C}_{r0},\mathcal{C}_{s0}\rangle = \langle\{\mathbb{F}\varphi : (1,l_1)\},\{1 \leq 1\},\{l_1 \lhd l_1\}\rangle$
– If $\langle\mathcal{F}_i \cup \{\mathbb{S}_i F_i : (x_i,u_i)\},\mathcal{C}_{ri},\mathcal{C}_{si}\rangle \notin \mathcal{P}$ then $\langle\mathcal{F}_{i+1},\mathcal{C}_{ri+1},\mathcal{C}_{si+1}\rangle = \langle\mathcal{F}_i,\mathcal{C}_{ri},\mathcal{C}_{si}\rangle$
– If $\langle\mathcal{F}_i \cup \{\mathbb{S}_i F_i : (x_i,u_i)\},\mathcal{C}_{ri},\mathcal{C}_{si}\rangle \in \mathcal{P}$ then $\langle\mathcal{F}_{i+1},\mathcal{C}_{ri+1},\mathcal{C}_{si+1}\rangle = \langle\mathcal{F}_i \cup \{\mathbb{S}_i F_i : (x_i,u_i)\} \cup F_e,\mathcal{C}_{ri} \cup \mathcal{C}_{re},\mathcal{C}_{si} \cup \mathcal{C}_{se}\rangle$ such that F_e, $\mathcal{C}_{re}$ and $\mathcal{C}_{se}$ are determined by:

S_i	F_i	F_e	$\mathcal{C}_{re}$	$\mathcal{C}_{se}$
$\mathbb{F}$	$\phi \to \psi$	$\{\mathbb{T}\phi : (a,u_i),\mathbb{F}\psi : (a,u_i)\}$	$\{x_i \leq a\}$	$\emptyset$
$\mathbb{T}$	$\phi * \psi$	$\{\mathbb{T}\phi : (a,u_i),\mathbb{T}\psi : (b,u_i)\}$	$\{ab \leq x_i\}$	$\emptyset$
$\mathbb{F}$	$\phi \mathrel{-\!\!*} \psi$	$\{\mathbb{T}\phi : (a,u_i),\mathbb{F}\psi : (x_i a,u_i)\}$	$\{x_i a \leq x_i a\}$	$\emptyset$
$\mathbb{T}$	I	$\emptyset$	$\{1 \leq x_i\}$	$\emptyset$
$\mathbb{T}$	$\Diamond\phi$	$\{\mathbb{T}\phi : (x_i,c)\}$	$\emptyset$	$\{u_i \lhd c\}$
$\mathbb{F}$	$\Box\phi$	$\{\mathbb{F}\phi : (x_i,c)\}$	$\emptyset$	$\{u_i \lhd c\}$
Otherwise		$\emptyset$	$\emptyset$	$\emptyset$

with $a = c_{2i+1}$, $b = c_{2i+2}$ and $c = l_{i+2}$.

Proposition 4. *For any $i \in \mathbb{N}$, the following properties hold:*

1. $\mathbb{F}\varphi : (1,l_1) \in \mathcal{F}_i$, $1 \leq 1 \in \mathcal{C}_{ri}$ *and* $l_1 \lhd l_1 \in \mathcal{C}_{si}$
2. $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$, $\mathcal{C}_{ri} \subseteq \mathcal{C}_{ri+1}$ *and* $\mathcal{C}_{si} \subseteq \mathcal{C}_{si+1}$
3. $\langle\mathcal{F}_i,\mathcal{C}_{ri},\mathcal{C}_{si}\rangle_{0\leqslant i} \in \mathcal{P}$
4. $\mathcal{A}_r(\mathcal{C}_{ri}) \subseteq \{1,c_1,c_2,...,c_{2i}\}$
5. $\mathcal{A}_s(\mathcal{C}_{si}) \subseteq \{l_1,l_2,...,l_{i+1}\}$

We now consider the limit CSS $\langle\mathcal{F}_\infty,\mathcal{C}_{r\infty},\mathcal{C}_{s\infty}\rangle$ of the sequence $\langle\mathcal{F}_i,\mathcal{C}_{ri},\mathcal{C}_{si}\rangle_{0\leqslant i}$ defined by:

$$\mathcal{F}_\infty = \bigcup_i \mathcal{F}_i \quad \text{and} \quad \mathcal{C}_{r\infty} = \bigcup_i \mathcal{C}_{ri} \quad \text{and} \quad \mathcal{C}_{s\infty} = \bigcup_i \mathcal{C}_{si}$$

Proposition 5. *We have $\langle\mathcal{F}_\infty,\mathcal{C}_{r\infty},\mathcal{C}_{s\infty}\rangle \in \mathcal{P}$ and for all labelled formulae $\mathbb{S}\phi : (x,u)$, if $\langle\mathcal{F}_\infty \cup \{\mathbb{S}\phi : (x,u)\},\mathcal{C}_{r\infty},\mathcal{C}_{s\infty}\rangle \in \mathcal{P}$ then $\mathbb{S}\phi : (x,u) \in \mathcal{F}_\infty$*

Lemma 11. *The limit CSS is a Hintikka CSS.*

Theorem 2 (Completeness). *Let φ be a formula, if φ is valid then there exists a DBI-proof for φ.*

Proof. We suppose that there is no DBI-proof of φ and show that φ is not valid. Our method allows us to build a limit CSS that is a Hintikka CSS, by Lemma 11. By property 1 of Proposition 4, $\mathbb{F}\varphi : (1,l_1) \in \mathcal{F}_i$. By Lemma 9, φ is not valid.

6 Conclusion

We have defined and studied a modal extension of BI, called DBI, that allows us to express dynamic properties about resources. We propose a Kripke semantics for DBI and a labelled tableaux method that is proved sound and complete w.r.t. this semantics. Compared to previous works on proof-theory in BI, the labelled tableaux method for DBI deals not only with a so-called resource graph but also with a state graph. Moreover we show how we can generate countermodels in case of non-validity.

Future works will be devoted to the study of other extensions of BI with other modalities such that fragments of SCRP/MBI [12], in order to mix dynamic resources and processes, and also of the semantics based on Petri nets for such extensions.

References

1. Biri, N., Galmiche, D.: A Separation Logic for Resource Distribution. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 23–37. Springer, Heidelberg (2003)
2. Engberg, U., Winskel, G.: Completeness results for Linear Logic on Petri nets. Annals of Pure and Applied Logic 86, 101–135 (1997)
3. Galmiche, D., Méry, D.: Tableaux and Resource Graphs for Separation Logic. Journal of Logic and Computation 20(1), 189–231 (2010)
4. Galmiche, D., Méry, D., Pym, D.: The semantics of BI and Resource Tableaux. Math. Struct. in Comp. Science 15(6), 1033–1088 (2005)
5. Girard, J.Y.: Linear logic. Theoretical Computer Science 50(1), 1–102 (1987)
6. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM 12(10), 576–580 (1969)
7. Ishtiaq, S., O'Hearn, P.W.: BI as an assertion language for mutable data structures. In: 28th ACM Symposium on Principles of Programming Languages, POPL 2001, London, UK, pp. 14–26 (2001)
8. Larchey-Wendling, D.: The Formal Proof of the Strong Completeness of Boolean BI (2012), http://www.loria.fr/~larchey/BBI
9. Larchey-Wendling, D., Galmiche, D.: The Undecidability of Boolean BI through Phase Semantics. In: 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, Edinburgh, UK, pp. 147–156 (July 2010)
10. Milner, R.: Communication and concurrency. Prentice-Hall, Inc., Upper Saddle River (1989)
11. O'Hearn, P.W., Pym, D.J.: The Logic of Bunched Implications. Bulletin of Symbolic Logic 5(2), 215–244 (1999)
12. Pym, D.J., Tofts, C.: Systems modelling via resources and processes: Philosophy, calculus, semantics, and logic. Electronic Notes in Theoretical Computer Science 172, 545–587 (2007)
13. Pym, D.J.: The Semantics and Proof Theory of the Logic of Bunched Implications. Applied Logic Series, vol. 26. Kluwer Academic Publishers (2002)
14. Reynolds, J.: Separation logic: A logic for shared mutable data structures. In: IEEE Symposium on Logic in Computer Science, Copenhagen, Danemark, pp. 55–74 (July 2002)

Stuttering for Abstract Probabilistic Automata

Benoît Delahaye[1], Kim G. Larsen[2], and Axel Legay[1]

[1] INRIA/IRISA, France
{benoit.delahaye,axel.legay}@inria.fr
[2] Aalborg University, Denmark
kgl@cs.aau.dk

Abstract. Probabilistic Automata (PAs) are a widely-recognized mathematical framework for the specification and analysis of systems with non-deterministic and stochastic behaviors. In a series of recent papers, we proposed Abstract Probabilistic Automata (APAs), a new abstraction framework for representing possibly infinite sets of PAs. We have developed a complete abstraction theory for APAs, and also proposed the first specification theory for them. APAs support both satisfaction and refinement operators, together with classical stepwise design operators.

One of the major drawbacks of APAs is that the formalism cannot capture PAs with hidden actions – such actions are however necessary to describe behaviors that shall not be visible to a third party. In this paper, we revisit and extend the theory of APAs to such context. Our first main result takes the form of proposal for a new probabilistic satisfaction relation that captures several definitions of PAs with hidden actions. Our second main contribution is to revisit all the operations and properties defined on APAs for such notions of PAs. Finally, we also establish the first link between stochastic modal logic and APAs, hence linking an automata-based specification theory to a logical one.

1 Introduction

Nowadays, systems are tremendously big and complex and mostly result from the assembling of several components. These components are usually designed by teams working *independently* but with a common agreement on what the interface of each component should be. These interfaces, also called specifications, precise the behaviors expected from each component as well as the environment in which they can be used, but do not impose any constraint on how the components are implemented.

Instead of relying on Word/Excel text documents or modeling languages such as UML/XML, as is usually done in practice, a series of recent works recommend relying most possibly on mathematically sound formalisms. Mathematical foundations that allow to reason at the abstract level of interfaces, in order to infer properties of the global implementation, and to design or to advisedly (re)use components is a very active research area, known as *compositional reasoning* [18]. Any good specification theory shall be equipped with a *satisfaction relation* (to decide whether an implementation satisfies a specification), a *refinement relation* (to compare sets of implementations), a *logical conjunction* (to compute intersection of sets of implementations), and

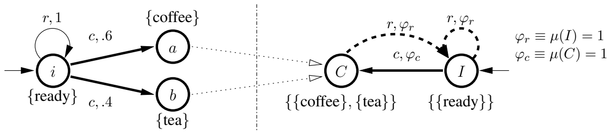


Fig. 1. Implementation PA (left) and specification APA (right) of a coffee machine

a *structural composition* (to combine specifications). Additionally, properties such as precongruence of composition with respect to refinement [18] shall also be satisfied.

Building good specification theories has been the subject of intensive studies among which one finds classical logical specifications, various process algebrae such as CSP, or Input/Output automata/interfaces (see [19,7,24]). Recently, a new series of works has concentrated on *modal specifications* [20], a language theoretic account of a fragment of the modal mu-calculus logic which is known to admit a more flexible and easy-to-use compositional refinement method than those carried out in CSP [20,29,3].

As soon as systems include randomized algorithms, probabilistic protocols, or interact with physical environment, probabilistic models are required to reason about them. This is exacerbated by requirements for fault tolerance, when systems need to be analyzed quantitatively for the amount of failure they can tolerate, or for the delays that may appear. As Henzinger and Sifakis [18] point out, introducing probabilities into design theories allows assessing dependability of IT systems in the same manner as commonly practiced in other engineering disciplines.

In recent works [5,10,6], we proposed Constraint Markov Chains (CMCs), a complete specification theory for pure stochastic systems, namely Markov Chains (MCs). Roughly speaking, a CMC is a MC equipped with a constraint on the next-state probabilities from any state. An implementation for a CMC is thus a MC, whose next-state probability distribution satisfies the constraint associated with each state. Contrary to Interval Markov Chains where sets of distributions are represented by intervals, CMCs are closed under both composition and conjunction. Later, in [8], the CMC approach was extended to handle those systems that combine both stochastic and non-deterministic behaviors, i.e., Probabilistic Automata (PA). APAs, whose theory is implemented in the APAC toolset [9], is the result of combining Modal Automata and CMCs – the abstractions for labelled transition systems and Markov Chains, respectively. Like other modal-based specification theories, our formalism can be used in various areas, including abstract model checking and compositional reasoning.

The specification theory induced by APAs is more expressive than any classical specification theories where both implementations and specifications are represented by the same object. As an example, Segala's theory assumes that both specifications and implementations are represented with PAs [31,26]. Such an approach does not permit to represent an infinite set of non-deterministic behaviors in a finite way. On the other hand, while satisfaction relation between PAs[25] can be expressed with classical notions of (stochastic) simulations [31], ours requires the use of a rather more complex definition of equivalence relation. Consider the implementation (left) and specification (right) of a coffee machine given in Figure 1. The specification specifies that there are two possible transitions from initial state I: a may transition labeled with action r (reset) and a must transition labeled with action c (coin). May transitions, which may not be implemented, are represented with dashed arrows. Must transitions, which shall be present in any implementation of the specification, are represented with plain arrows. The probability distributions associated with these actions are specified by the constraints φ_r and φ_c, respectively. One can see that the implementation gives a more precise behavior of the coffee machine: action r loops back to initial state i with probability 1, while coin leads to state a (coffee) with probability $.6$ and to state b (tea) with probability $.4$. Satisfaction between implementation and specification lifts the classical notion of simulation for PAs to APAs as follows: (1) all must transitions of the specification must be matched with transitions in the implementations, and (2) all transitions in the implementation must be matched with may transitions in the specification. Additionally, we have to check that the probability distributions in the implementation are matched with probability distributions in the specification that satisfy the given constraints.

Contribution. In the process of incremental design (as well as for other applications), it may be necessary to incrementally widen the scope of implementations. Usually, the latter is done by permitting the addition of hidden actions also called stutter steps [31,4] in the implementation. Introducing such actions is known to complicate the definition and the computation of operations such as bisimulation/simulation [31]. Moreover, it may break up some properties such as precongruence of refinement with respect to composition [31]. The objective of this paper is to extend the APA specification theory by considering implementations with stuttering steps. Our first contribution is the definition of a new stochastic satisfaction relation for APAs. This relation generalizes stochastic simulation to the APA level. We then study various notions of stuttering and compare their expressivity. We also study the impact of adding stuttering on various properties such as precongruence of refinement with respect to composition. Finally, we define and study ML-(A)PA that is a new modal logic for APAs and stuttering PAs. ML-(A)PA generalizes the PML logic [22,21] of Larsen et al. from PAs to APAs and stuttering PAs.

Related Work. A wide spectrum of different approaches study stuttering for non stochastic systems [32] and stochastic ones [28,1,2]. In [2], the authors define weak bisimulation for fully probabilistic processes. This is in contrast with our model that combines both probabilistic and non-deterministic aspects. In [28,1], weak bisimulation is extended to *strictly alternating* systems that combine both non-determinism and probabilities. Although such systems are similar to PAs, it is known that weak (branching) bisimulation for alternating systems is incomparable to weak bisimulation for non-alternating systems [30]. Moreover, it is worth mentioning that above mentioned works report on computing and checking weak bisimulation between probabilistic systems, while our aim is to propose a notion of weak simulation (satisfaction) between a probabilistic system and a probabilistic specification that represents a possibly infinite set of implementations.

In [14], the author defines a notion of constraints on states to represent sets of probability distributions. Although this formalism resembles the one of constraints used in APAs, the constraints in [14] are used in a different context. Indeed, while we use constraints to represent sets of probabilistic transitions, [14] uses them to replace the non-deterministic choice between internal transitions by probability distributions.

Finally we mention that the problem of defining compositionality in the probabilistic setting with hidden steps has also been addressed in various settings [30,27,23,15,14,16]. In particular [15] defines a general parallel composition operator for CSP that deals with hidden steps, and [16] suggests the removal of hidden steps through a transformation of CSP models. In both papers, the systems considered are strictly alternating and results are obtained with respect to a ready-trace notion of equivalence on processes, which makes it incomparable to our notion of stuttering satisfaction between specifications and implementations.

2 A Probabilistic Satisfaction for Abstract Probabilistic Automata

2.1 Abstract Probabilistic Automata

Let *Dist*(S) denote a set of all discrete probability distributions over a finite set S and $\mathbb{B}_2 = \{\top, \bot\}$.

Definition 1. *A PA[31] is a tuple* (S, A, L, AP, V, s_0)*, where* S *is a finite set of states with the initial state* $s_0 \in S$*,* A *is a finite set of actions,* L*:* $S \times A \times \mathit{Dist}(S) \to \mathbb{B}_2$ *is a (two-valued transition) function,* AP *is a finite set of atomic propositions and* V*:* $S \to 2^{AP}$ *is a state-labeling function.*

Consider a state s, an action a, and a probability distribution μ. The value of $L(s, a, \mu)$ is set to $\top$ in case there exists a transition from s under action a to a distribution μ on successor states. In other cases, we have $L(s, a, \mu) = \bot$.

We now switch to Abstract Probabilistic Automata (APA)[8], that is a specification theory for PAs. Let S be a finite set. We define $C(S)$ to be the set of constraints defined over discrete probability distributions on S. Each element $\varphi \in C(S)$ describes a set of distributions: *Sat*$(\varphi) \subseteq$ *Dist*(S). Let $\mathbb{B}_3 = \{\top, ?, \bot\}$. APAs are formally defined as follows.

Definition 2. *An APA[8] is a tuple* (S, A, L, AP, V, s_0)*, where* S *is a finite set of states,* $s_0 \in S$*,* A *is a finite set of actions, and* AP *is a finite set of atomic propositions.* $L : S \times A \times C(S) \to \mathbb{B}_3$ *is a* three-*valued distribution-constraint function, and* $V : S \to 2^{2^{AP}}$ *maps each state in* S *to a set of admissible labelings.*

APAs play the role of specifications in our framework. An APA transition abstracts transitions of a certain unknown PA, called its implementation. Given a state s, an action a, and a constraint φ, the value of $L(s, a, \varphi)$ gives the modality of the transition. More precisely the value $\top$ means that transitions under a must exist in the PA to some distribution in *Sat*(φ); ? means that these transitions are allowed to exist; $\bot$ means that such transitions must not exist. Again L may be partial. A lack of value for given argument is equivalent to the $\bot$ value, so we will sometimes avoid defining $\bot$-value rules in constructions to avoid clutter, and occasionally will say that something applies if L takes the value of $\bot$, meaning that it is either taking this value or it is undefined. The function V labels each state with a subset of the powerset of AP, which models a disjunctive choice of possible combinations of atomic propositions.

2.2 A Probabilistic Satisfaction for APA

We now study the notion of satisfaction that relates a probabilistic automata $P = (S_P, A, L_P, AP, V_P, s_0^P)$ to its corresponding APA specification $N = (S, A, L, AP, V, s_0)$, The notion of satisfaction proposed in [8] directly relates distributions in P to distributions in N. As in the notion of probabilistic forward simulation presented in [25], we now extend this notion to account for linear combinations of distributions in N, hence generalizing results in [8].

Definition 3. *Let S and S' be non-empty sets, and μ, μ' be distributions; $\mu \in Dist(S)$ and $\mu' \in Dist(S')$. We say that μ is* simulated *by μ' with respect to a relation $\mathcal{R} \subseteq S \times S'$ and a* correspondence function $\delta : S \to (S' \to [0, 1])$ *iff*

1. *for all $s \in S$, $\delta(s)$ is a distribution on S' if $\mu(s) > 0$,*
2. *for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and*
3. *whenever $\delta(s)(s') > 0$ then $(s, s') \in \mathcal{R}$.*

We write $\mu \Subset_{\mathcal{R}}^{\delta} \mu'$ meaning that μ is simulated by μ' with respect to $\mathcal{R}$ and δ, and we write $\mu \Subset_{\mathcal{R}} \mu'$ iff there exists a function δ such that $\mu \Subset_{\mathcal{R}}^{\delta} \mu'$.

We then define probabilistic satisfaction as follows.

Definition 4 (Probabilistic Satisfaction). *Let $P = (S_P, A, L_P, AP, V_P, s_0^P)$ be a PA and $N = (S, A, L, AP, V, s_0)$ be an APA. A binary relation $\mathcal{R} \subseteq S_P \times S$ is a probabilistic satisfaction relation iff, for any $(s, s') \in \mathcal{R}$, the following conditions hold:*

- *for all $a \in A$ and $\varphi' \in C(S)$ such that $L(s', a, \varphi') = \top$, there exists a distribution $\mu_P \in Dist(S_P)$ such that $L_P(s, a, \mu_P) = \top$ and there exists $\mu' \in Sat(\varphi')$ such that $\mu_P \Subset_{\mathcal{R}} \mu'$,*
- *for all $a \in A$ and $\mu_P \in Dist(S_P)$ such that $L_P(s, a, \mu_P) = \top$, there exists $\varphi_1, \ldots \varphi_n \in C(S)$ such that for all i, $L(s', a, \varphi_i) \neq \bot$ and there exists $\mu_i \in Sat(\varphi_i)$ and $\rho_i \in [0, 1]$ such that $\sum_i \rho_i = 1$ and $\mu_P \Subset_{\mathcal{R}} (\sum_i \rho_i \mu_i)$, and*
- *$V_P(s) \in V(s')$.*

We say that P probabilistically satisfies N, written $P \models_P N$ iff there exists a probabilistic satisfaction relation $\mathcal{R}$ such that $s_0^P \,\mathcal{R}\, s_0$. The set of probabilistic implementations of APA N is defined by $[\![N]\!]_P = \{P \mid P \models_P N\}$.

It is easy to see that this extension of satisfaction is conservative with respect to all the good properties presented in [8]. In the rest of this paper, we study the impact of adding stuttering to the specification theory.

3 Stuttering for Abstract Probabilistic Automata

We now study an extension of the APAs specification theory where implementations may have stutter steps. In the rest of this section, we first introduce various notions of stuttering for PAs and then we extend the satisfaction relation to them. Later, we shall study the impact of stuttering on refinement and structural/logical composition.

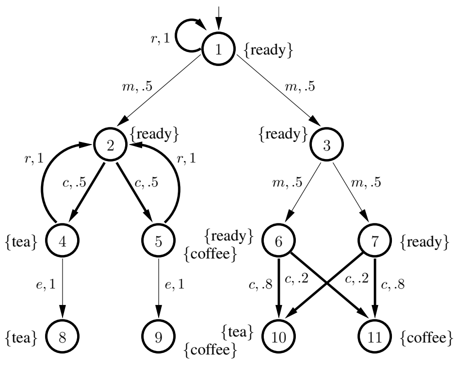


Fig. 2. Stuttering PA P with m and e as hidden actions

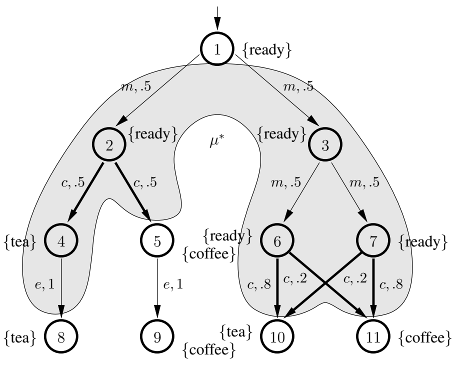


Fig. 3. Stuttering transition $1 \rightarrow^* \mu^*$ in P where stuttering happens both before and after visible action c.

3.1 Introducing Stutter Actions

Consider a PA $P = (S_P, A, L_P, AP, V_P, s_0^P)$. We must assume that any state s' that can be reached from a state s by following a sequence of hidden actions $\mathcal{H} \subseteq A_P$ cannot be distinguished from s, i.e., have the same valuation as s.

Definition 5 (Consistent set of hidden actions). *Let $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ and let $\mathcal{H} \subseteq A_P$. We say that $\mathcal{H}$ is a consistent set of hidden actions regarding P if $\forall s \in S_P$ and $\forall a \in \mathcal{H}$, if there exists $\mu \in Dist(S_P)$ such that $L_P(s, a, \mu) = \top$, then $\forall s' \in S$, we have $\mu(s') > 0 \Rightarrow V_P(s') = V_P(s)$.*

The following example shows that, as it is the case for other specifications theories (see e.g. [13]), there are various ways to formally define a stuttering transition.

Example 1. Consider the stuttering PA P given in Figure 2, and whose set of consistent hidden action is given by $\{m, e\}$. P represents a coffee machine that has two modes. Action m allows choosing between the two modes. In mode A, represented by state 2 and its successors, the action c leads to states labeled with tea and coffee with probability .5 each. From states 4 and 5, either the coffee machine can be reset with action r, but will stay in the same mode, or can suffer an error (action e) that leads to deadlock states 8 and 9. In mode B, one can again choose a sub-mode with action m, leading to states 6 and 7 that deliver tea and coffee with different probabilities.

Considering different notions of stuttering will lead to different sets of executions for the PA P. As an example, stuttering could be restricted to happen only before visible actions. The execution presented in Figure 4 represents a stuttering execution $1 \xrightarrow{c}{}^* \mu_0^*$ (informally, one can reach distribution μ_0^* from state 1 by following action c interleaved with hidden actions), where the internal action m happens before the visible action c, leading to a distribution μ_0^*. Remark that such an execution could not be considered if we restricted stuttering to happen only after a visible action. The unfolding of P given in Figure 5 presents two stuttering executions $1 \xrightarrow{r}{}^* \mu_1^*$ and $2 \xrightarrow{c}{}^* \mu_2^*$ where in both

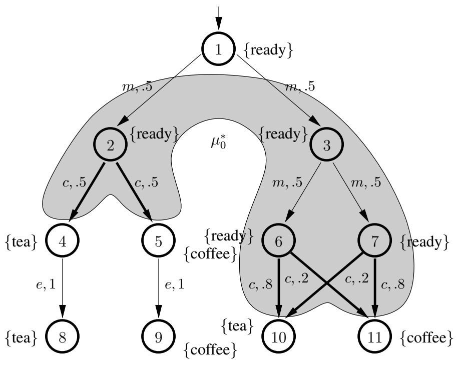


Fig. 4. Stuttering transition $1 \to^* \mu_0^*$ in PA P of Figure 2 where stuttering happens before visible action c

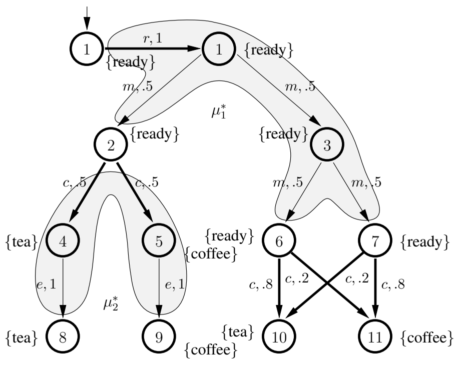


Fig. 5. Stuttering transitions $1 \to^* \mu_1^*$ and $2 \to^* \mu_2^*$ in PA P of Figure 2 where stuttering happens after visible actions r and c

cases stuttering only happens after the visible action. Again, such executions could not be considered if we restricted stuttering to happen only before a visible action. Finally, the execution presented in Figure 3 represents a stuttering transition $1 \xrightarrow{c}{}^* \mu^*$ in P where stuttering happens both before and after the visible action c.

As illustrated in the example above, the choice made in the definition of stuttering will have a strong impact on the executions allowed in PAs. In order to be as general as possible, we choose to allow stuttering to happen both before and after visible actions. The only restriction we make is that stuttering cannot happen independently of visible actions, that is, for each stuttering transition, a visible action must be taken. This leads to the following definition.

Definition 6 (Stuttering transitions for PAs). *Let $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ be a PA, and let $\mathcal{H} \subseteq A_P$ be a consistent set of hidden actions. We define the notion of $\mathcal{H}$-stuttering recursively as follows:*

Base case: *For all $s \in S_P$, $a \in A_P$ and $\mu \in Dist(S_P)$, we say that $s \xrightarrow[\mathcal{H}]{a}{}^1 \mu$ iff $L(s, a, \mu) = \top$. As a shortcut, we write $s \xrightarrow[\mathcal{H}]{\tau}{}^1 \mu$ if there exists $b \in \mathcal{H}$ such that $s \xrightarrow[\mathcal{H}]{b}{}^1 \mu$.*

Recursion: *For all $s \in S_P$, $k > 1$, $a \in A_P$ and $\mu^* \in Dist(S_P)$, we say that $s \xrightarrow[\mathcal{H}]{a}{}^k \mu^*$ iff*

1. *either $a \notin \mathcal{H}$ and there exists $\mu_1 \in Dist(S_P)$ and $b \in \mathcal{H}$ such that $L(s, b, \mu_1) = \top$ and the following conditions hold:*
   - *for all states $r \in S_P$ such that $\mu_1(r) > 0$, there exists $k' < k$ and $\mu_r \in Dist(S_P)$ such that $r \xrightarrow[\mathcal{H}]{a}{}^{k'} \mu_r$, and*
   - *for all $s' \in S_P$,*

$$\mu^*(s') = \sum_{r \in S_P} \mu_1(r)\mu_r(s')$$

2. *or there exists* $\mu_1 \in Dist(S_P)$ *such that* $L(s, a, \mu_1) = \top$ *and a subset* $R \subseteq S_P$ *such that the following conditions hold:*
   - *for all states* $r \in R$, *we have* $\mu_1(r) > 0$ *and there exists* $k' < k$ *and* $\mu_r \in Dist(S_P)$ *such that* $r \xrightarrow[\mathcal{H}]{\tau}{}^{k'} \mu_r$, *and*
   - *for all* $s' \in S_P$,

$$\mu^*(s') = \begin{cases} \sum_{r \in R} \mu_1(r)\mu_r(s') & \text{if } s' \in R \\ \mu_1(s') + \sum_{r \in R} \mu_1(r)\mu_r(s') & \text{otherwise.} \end{cases}$$

We say that $s \xrightarrow[\mathcal{H}]{a}{}^* \mu^*$ *if there exists* $k > 0$ *such that* $s \xrightarrow[\mathcal{H}]{a}{}^k \mu^*$.

Informally stuttering can happen either before (case 1) or after (case 2) taking the visible action a. Remark that both cases are not exclusive and can interleave. If stuttering occurs before action a, then all successor states r must admit a stuttering transition involving a. In such case, the overall probability of reaching a state s' is the sum through all stuttering paths. If stuttering occurs after action a, then we denote by R the set of successor states from which we stutter, and by $S_P \setminus R$ the set of states in which we stop. Remark that the set R is dynamic in the sense that a different set R may be chosen for each step of a stuttering transition. In this case the overall probability of going to a state $s' \in R$ is the sum through all stuttering paths, while the overall probability of going to a state $s' \notin R$ is the addition of the probabilities of going to s' directly (without stutter) with the the sum through all stuttering paths.

In the rest of the paper, we denote by $\xrightarrow[\mathcal{H}]{a}{}^*_A$ (resp. $\xrightarrow[\mathcal{H}]{a}{}^*_B$) stuttering transitions where stuttering only happens after (resp. before) the visible action a, obtained by removing item 1. (resp. 2.) from the recursive part of Definition 6.

Example 2. Consider the PA $P = (S_P, A_P, L_P, AP, V_P, 1)$ given in Figure 2, and a distribution μ^* such that $\mu^*(5) = \mu^*(8) = \mu^*(10) = \mu^*(11) = .25$.

The situation is represented in Figure 3. Let us see how to derive that $1 \xrightarrow[\{e,m\}]{c}{}^3 \mu^*$. We follow the following description.

1. for $[1 \xrightarrow[\{e,m\}]{c}{}^3 \mu^*]$., we have $c \notin \{e, m\}$ and $L_P(1, m, \mu_1) = \top$ with $m \in \{e, m\}$ (case 1). states 2 and 3 are the only states for which μ_1 gives a non-zero probability, and $2 \xrightarrow[\{e,m\}]{c}{}^2 \mu_2$ and $3 \xrightarrow[\{e,m\}]{c}{}^2 \mu_3$, with $\mu^*(s') = \mu_1(2)\mu_2(s') + \mu_1(3)\mu_3(s')$.
2. for $[2 \xrightarrow[\{e,m\}]{c}{}^2 \mu_2]$., we have $L_P(2, c, \mu_2') = \top$ (case 2) and there exists $R = \{4\} \subseteq S_P$ such that $\mu_2'(4) > 0$ and $4 \xrightarrow[\{e,m\}]{\tau}{}^1 \mu_4$. In addition, we obtain after simplifications:
   $$\mu_2 : \begin{cases} 8 \mapsto \mu_2'(4)\mu_4(8) = .5 \\ 5 \mapsto \mu_2'(5) + 0 = .5 \end{cases}$$
   We observe that $[4 \xrightarrow[\{e,m\}]{\tau}{}^1 \mu_4]$ is a base case.
3. $[3 \xrightarrow[\{e,m\}]{c}{}^2 \mu_3]$. We have $c \notin \{e, m\}$ and $L_P(3, m, \mu_3') = \top$ with $m \in \{e, m\}$ (case 1). states 6 and 7 are the only states for which μ_3' gives a non-zero probability, and $6 \xrightarrow[\{e,m\}]{c}{}^1 \mu_6$ and $7 \xrightarrow[\{e,m\}]{c}{}^1 \mu_7$ with $\mu_3(s') = \mu_3'(6)\mu_6(s') + \mu_3'(7)\mu_7(s')$. We observe that $[6 \xrightarrow[\{e,m\}]{c}{}^1 \mu_6]$ and $[7 \xrightarrow[\{e,m\}]{c}{}^1 \mu_7]$ are base cases.

Finally, we obtain the following result:

$$\boldsymbol{\mu^*(5)} = \mu_1(2)(\mu_2'(5)) = .25 \qquad \boldsymbol{\mu^*(10)} = \mu_1(3)(\mu_3'(6)\mu_6(10) + \mu_3'(7)\mu_7(10)) = .25$$

$$\boldsymbol{\mu^*(8)} = \mu_1(2)(\mu_2'(4)\mu_4(8)) = .25 \quad \boldsymbol{\mu^*(11)} = \mu_1(3)(\mu_3'(6)\mu_6(11) + \mu_3'(7)\mu_7(11)) = .25$$

3.2 On Stutter Satisfaction

We now introduce the notion of stutter satisfaction, that is an extension of Definition 4 for stuttering PAs.

Definition 7 (Stutter Satisfaction). *Let $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ be a PA, let $N = (S, A, L, AP, V, s_0)$ be an APA such that $A \subseteq A_P$ and $\mathcal{H} = A_P \setminus A$ is a consistent set of hidden actions for P. A binary relation $\mathcal{R} \subseteq S_P \times S$ is a* stutter satisfaction relation *iff, for any $(s, s') \in \mathcal{R}$, the following conditions hold:*

1. *for all $a \in A$ and $\varphi' \in C(S)$, if $L(s', a, \varphi') = \top$, then there exists a distribution $\mu^* \in Dist(S_P)$ such that $s \xrightarrow[\mathcal{H}]{a} {}^*\mu^*$ and there exists $\mu \in Sat(\varphi')$ such that $\mu^* \Subset_{\mathcal{R}} \mu$,*
2. *for all $\mu^* \in Dist(S_P)$ and $a \in A$ such that $s \xrightarrow[\mathcal{H}]{a} {}^*\mu^*$, there exist constraints $\varphi_1, \ldots \varphi_n \in C(S)$ such that for all i, $L(s', a, \varphi_i) \neq \bot$ and there exist $\rho_i \in [0, 1]$ and $\mu_i \in Sat(\varphi_i)$ such that $\sum_i \rho_i = 1$ and $\mu^* \Subset_{\mathcal{R}} (\sum_i \rho_i \mu_i)$, and*
3. *$V_P(s) \in V(s')$.*

We say that $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ stutter-satisfies $N = (S, A, L, AP, V, s_0)$, written $P \models^* N$, iff $A \subseteq A_P$, $\mathcal{H} = A_P \setminus A$ is a consistent set of hidden actions for P, and there exists a stutter satisfaction relation $\mathcal{R}$ such that $s_0^P \,\mathcal{R}\, s_0$. The set of stuttering implementations of APA N is given by $[\![N]\!]^* = \{P \mid P \models^* N\}$. Algorithms to decide such satisfaction relation can be obtained directly from those proposed in [10,11] for the case where there exists no stuttering loops. Otherwise, the problem is still open.

Example 3. The PA P given in Figure 2 satisfies the specification of the coffee machine of Figure 1 with the notion of stuttering satisfaction given above. The stuttering satisfaction relation $\mathcal{R}$ is as follows: $\mathcal{R} = \{(\{1, 2, 3, 6, 7\}, I), (\{4, 5, 8, 9, 10, 11\}, C)\}$. We show how state 1 of P satisfies state I of the specification and leave it to the reader to verify that the rest of the relation $\mathcal{R}$ satisfies the axioms of Definition 7 above.

- In the specification, we have $L(I, c, \varphi_c) = \top$. There exists a matching distribution in P: we have $1 \xrightarrow[\{e,\, m\}]{c} {}^3\mu^*$, with μ^* defined in Example 2, and $\mu^* \Subset_{\mathcal{R}} \mu_c$ with $\mu_c : C \mapsto 1 \in Sat(\varphi_c)$,
- in the implementation, we can verify that for all $a \in \{r, c\}$ and μ_P^* such that $1 \xrightarrow[\{e,\, m\}]{a} {}^*\mu_P^*$, we have a matching constraint and distribution in the specification: either φ_r or φ_c, and
- $V_P(1) = \{\text{ready}\} \in V(I) = \{\{\text{ready}\}\}$.

Remark that the choice we made on the definition of stutter transitions by allowing stuttering to happen both before and after the visible action strongly influences the notion of stuttering satisfaction. We denote by $\models^*_A$ (resp. $\models^*_B$) the notion of satisfaction obtained by replacing the general notion of stutter transition $\xrightarrow[\mathcal{H}]{a}{}^*$ with the restricted notion $\xrightarrow[\mathcal{H}]{a}{}^*_A$ (resp. $\xrightarrow[\mathcal{H}]{a}{}^*_B$). The following theorem states that the different notions of stutter satisfaction $\models^*$, $\models^*_A$ and $\models^*_B$ cannot be compared in general.

Theorem 1. *There exists PAs P, P_A and P_B and an APA N such that (1) $P \models^*_A N$, $P \models^*_B N$, and $P \models^* N$; (2) $P_A \models^*_A N$, $P_A \not\models^*_B N$, and $P_A \not\models^* N$; (3) $P_B \not\models^*_A N$, $P_B \models^*_B N$, and $P_B \models^* N$.*

Refinement. We now consider Refinement that is a relation that allows us to compare APAs in terms of sets of implementations. In Segala's theory, refinement boils down to (stochastic) simulation. In the context of APAs, refinement usually extends the definition of satisfaction. Extending Definition 7 would require to consider stuttering in the specification itself, which is not the topic of this paper. For this reason, we use the refinement relation proposed in [11].

Definition 8 (Refinement([11])). *Let $N = (S, A, L, AP, V, s_0)$ and $N' = (S', A, L', AP,$
$V', s'_0)$ be APAs. $\mathcal{R} \subseteq S \times S'$ is a* refinement *relation if and only if, for all $(s, s') \in \mathcal{R}$, the following conditions hold:*

1. *$\forall a \in A$, $\forall \varphi' \in C(S')$, if $L'(s', a, \varphi') = \top$, then $\exists \varphi \in C(S) : L(s, a, \varphi) = \top$ and $\forall \mu \in Sat(\varphi)$, $\exists \mu' \in Sat(\varphi')$ such that $\mu \Subset_{\mathcal{R}} \mu'$,*
2. *$\forall a \in A$, $\forall \varphi \in C(S)$, if $L(s, a, \varphi) \neq \bot$, then $\forall \mu \in Sat(\varphi)$, $\exists \varphi' \in C(S')$: $L'(s', a, \varphi') \neq \bot$ and $\exists \mu' \in Sat(\varphi')$ such that $\mu \Subset_{\mathcal{R}} \mu'$, and*
3. *$V(s) \subseteq V'(s')$.*

We say that N refines N', denoted $N \preceq_W N'$, if and only if there exists a refinement relation relating s_0 and s'_0. In [11], it is shown that for two given APAs N_1 and N_2, we have $N_1 \preceq_W N_2 \Rightarrow [\![N_1]\!] \subseteq [\![N_2]\!]$, where $[\![N_i]\!]$ represent PAs without stuttering steps. The following theorem extends this result to the case of PAs with stuttering steps.

Theorem 2. *Let P be a PA and let N and N' be APAs. If $P \models^* N$ and $N \preceq_W N'$, then $P \models^* N'$.*

Conjunction. We now turn our attention to the interaction between stuttering and conjunction. Due to space limitations, the definition of conjunction is given in [12]. As proven in [11], conjunction is the greatest lower bound with respect to refinement [11], i.e. for all APAs N_1, N_2 and N_3, $(N_1 \preceq_W N_2) \wedge (N_1 \preceq_W N_3) \iff N_1 \preceq_W (N_2 \owedge N_3)$. Furthermore, it coincides with the intersection of sets of (non-stuttering) implementations: for all N_1 and N_2, $[\![N_1]\!] \cap [\![N_2]\!] = [\![N_1 \owedge N_2]\!]$. In the following, we show that this result is preserved with the new notion of stuttering implementation.

Theorem 3. *Given two APAs N_1 and N_2, it holds that $[\![N_1]\!]^* \cap [\![N_2]\!]^* = [\![N_1 \owedge N_2]\!]^*$.*

	PA Semantics	**APA Semantics**
ψ	$s \models^* \psi \iff$	$s \models \psi \iff$
V_{val}	$V_P(s) \in V_{\text{val}}$	$V(s) \subseteq V_{\text{val}}$
$\psi_1 \wedge \psi_2$	$s \models^* \psi_1$ and $s \models^* \psi_2$	$s \models \psi_1$ and $s \models \psi_2$
$\langle a \rangle_{\rhd p} \psi'$	$\exists \mu^* \in Dist(S_P)$ s.t. $s \xrightarrow[\mathcal{H}]{a}{}^* \mu^*$ and $\left(\sum_{\{s' \mid s' \models^* \psi'\}} \mu^*(s') \rhd p\right)$	$\exists \varphi \in C(S)$ s.t. $L(s, a, \varphi) = \top$ and $\left(\forall \mu \in Sat(\varphi) : \sum_{\{s' \mid s' \models \psi'\}} \mu(s') \rhd p\right)$
$[a]_{\rhd p} \psi'$	$\forall \mu^* \in Dist(S_P)$, if $s \xrightarrow[\mathcal{H}]{a}{}^* \mu^*$, then $\left(\sum_{\{s' \mid s' \models^* \psi'\}} \mu^*(s') \rhd p\right)$	$\forall \varphi \in C(S)$, if $L(s, a, \varphi) \neq \bot$, then $\left(\forall \mu \in Sat(\varphi) : \sum_{\{s' \mid s' \models \psi'\}} \mu(s') \rhd p\right)$

Fig. 6. Semantics of ML-(A)PA for PAs and APAs

4 Logical Characterization

We now turn our attention to proposing a modal logic *ML-(A)PA* for PAs and APAs. This logic resembles the Probabilistic Modal Logic PML [22,21]. The main differences between PML and ML-(A)PA are that (1) ML-(A)PA is designed to specify properties for both PAs and APAs, while PML is restricted to PAs, (2) The semantics of ML-(A)PA for PAs considers stuttering transitions, while PML does not, and finally (3) unlike PML, ML-(A)PA is disjunction and negation-free. We first give the syntax of ML-(A)PA and semantics for PAs and APAs, then we study its soundness and completeness.

$$\psi ::= V_{\text{val}} \mid \psi_1 \wedge \psi_2 \mid \langle a \rangle_{\rhd p} \psi' \mid [a]_{\rhd p} \psi',$$

where $V_{\text{val}} \in 2^{2^{AP}}$, $a \in A$, $\rhd \in \{\geq, >\}$, and $p \in [0, 1]$. Let $F(A, AP)$ be the set of formulas over A and AP.

We define the semantics of ML-(A)PA for both PAs and APAs. Let $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ be a PA and let $N = (S, A, L, AP, V, s_0)$ be an APA. Assume that $A \subseteq A_P$ is a set of actions such that $\mathcal{H} = A_P \setminus A$ is a consistent set of hidden actions for P. We define the satisfaction relation between states of P (resp. N) and formulas in $F(A, AP)$ by induction as in Figure 6. We say that P satisfies ψ, written $P \models^* \psi$ iff $A_P \setminus A$ is a consistent set of hidden actions for P and $s_0^P \models^* \psi$. We say that N satisfies ψ, written $N \models \psi$ iff $s_0 \models \psi$. The logic ML-(A)PA and its relation to PAs/APAs is illustrated in the following example.

Example 4. Consider the specification of a coffee machine N given in Figure 1 and the implementation P of the coffee machine given in Figure 2. Let $A = \{r, c\}$ and $AP = \{\mathsf{ready}, \mathsf{tea}, \mathsf{coffee}\}$ and consider the following formulas in $F(A, AP)$:

$$\psi_1 ::= [c]_{\geq 1}\{\{\mathsf{coffee}\}, \{\mathsf{tea}\}\} \qquad \psi_3 ::= \langle c \rangle_{\geq .5}\{\{\mathsf{coffee}\}\}$$
$$\psi_2 ::= [r]_{\geq 1}\left([c]_{\geq 1}\{\{\mathsf{coffee}\}, \{\mathsf{tea}\}\}\right) \qquad \psi_4 ::= \{\{\mathsf{ready}\}\} \wedge \langle c \rangle_{\geq 1}\left([r]_{\geq 1}\{\{\mathsf{ready}\}\}\right)$$

One can verify that $N \models \psi_1$, $N \models \psi_2$ and $N \models \psi_4$ and that N does not satisfy ψ_3. Indeed, state C of N does not satisfy the formula $\{\{\mathsf{coffee}\}\}$. However, one can verify that $P \models^* \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$. In particular, the satisfaction of ψ_3 is ensured by

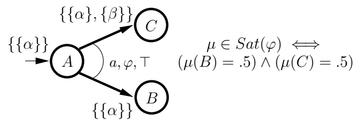

$\{\{\alpha\},\{\beta\}\}$
$\{\{\alpha\}\}$
$a, \varphi, \top$
$\{\{\alpha\}\}$
$\mu \in Sat(\varphi) \iff (\mu(B) = .5) \wedge (\mu(C) = .5)$

Fig. 7. APA $N_{\leq}$ such that $N_{\leq} \models \psi_5$

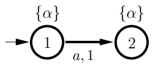


Fig. 8. PA $P_{\leq}$ such that $P_{\leq} \models^* N_{\leq}$ and $P_{\leq} \not\models^* \psi_5$

the existence of a distribution μ^* given in Figure 3 such that $1 \xrightarrow[\{e,\,m\}]{c} {}^*\mu^*$ in P and $\mu^*(\{\mathsf{coffee}\}) = .5$.

We now show that ML-(A)PA is sound and complete with respect to stutter satisfaction. We start with soundness.

Theorem 4 (Soundness). *Let $N = (S, A, L, AP, V, s_0)$ be an APA and $\psi \in F(A, AP)$ be a formula. If $N \models \psi$, then for all PA $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ such that $P \models^* N$, it holds that $P \models^* \psi$.*

It is worth mentioning that soundness would not hold if ML-(A)PA was equipped with negation or with the comparison operators $\{<, \leq\}$. This is illustrated in the following example.

Example 5. Assume that ML-(A)PA is equiped with the dual comparison operator $\leq$. Consider the formula $\psi_5 ::= [a]_{\leq .5}\{\{\alpha\}\}$.

Consider APA $N_{\leq}$ given in Figure 7. Since $\{\{\alpha\},\{\beta\}\} \not\subseteq \{\{\alpha\}\}$, we have that state C of $N_{\leq}$ does not satisfy $\{\{\alpha\}\}$. It thus follows that $N_{\leq} \models \psi_5$. Now consider PA $P_{\leq}$ given in Figure 8. One can verify that $P_{\leq} \models^* N_{\leq}$. However, since state 2 of $P_{\leq}$ satisfies $\{\{\alpha\}\}$, we have that $P_{\leq} \not\models^* \psi_5$. A similar example can be produced to prove that ML-(A)PA would not be sound if equiped with negation.

We now show that ML-(A)PA is complete with respect to stutter satisfaction.

Theorem 5 (Completeness). *Let $N = (S, A, L, AP, V, s_0)$ be a consistent APA and let $\psi \in F(A, AP)$. It holds that $(\forall P \in [\![N]\!]^*, P \models^* \psi) \implies N \models \psi$.*

This theorem is proved using an induction technique on the structure of the formula. Due to space limitations, the proof is reported to [12]. It is worth mentioning that completeness would not hold if ML-(A)PA was equiped with disjunction. This is illustrated in the following example, adapted from [3].

Example 6. Let $N_\vee = (\{A, B\}, \{a\}, L_\vee, \{\alpha, \beta\}, V_\vee, A)$ be an APA such that $V_\vee(A) = V_\vee(B) = \{\{\alpha\}\}$ and $L(A, a, \varphi) =?$ with $\mu \in Sat(\varphi)$ iff $\mu(B) = 1$. Assume that ML-(A)PA is extended with disjunction and consider the formula $\psi_6 ::= \langle a\rangle_{\geq 1}\{\{\alpha\}\} \vee [a]_{\geq 1}\{\{\beta\}\}$. Since state A does not have any must transition, we have that $N_\vee \not\models \langle a\rangle_{\geq 1}\{\{\alpha\}\}$. Moreover, since $V_\vee(B) \not\subseteq \{\{\beta\}\}$, we have that $N_\vee \not\models [a]_{\geq 1}\{\{\beta\}\}$. As a consequence, $N_\vee \not\models \psi_6$. However, any implementation of $N_\vee$ either contains no transition at all, thus satisfying $[a]_{\geq 1}\{\{\beta\}\}$, or it contains a transition leading to $\{\alpha\}$ with probability 1, thus satisfying $\langle a\rangle_{\geq 1}\{\{\alpha\}\}$. As a consequence, $\forall P \in [\![N_\vee]\!]^*, P \models^* \psi_6$.

In addition to being sound and complete with respect to stutter satisfaction, ML-(A)PA also matches the notion of conjunction of APAs, as shown in the following theorem.

Theorem 6. *Let N_1 and N_2 be two APAs and let ψ_1 and ψ_2 be two formulas. If $N_1 \Vdash \psi_1$ and $N_2 \Vdash \psi_2$ then $(N_1 \owedge N_2) \Vdash (\psi_1 \wedge \psi_2)$.*

5 On Composition of APAs and Stuttering

We now show that the notion of structural composition that allows to combine APAs does not preserve precongruence of refinement. Consider the classical notion of composition between PAs, originally proposed by Segala [31] and extended to the setting of APAs [8]. This notion of composition allows to synchronize on a common set of actions $\bar{A}$ while allowing independent progress on the complement of $\bar{A}$. When composing APAs, the resulting constraint represents products of distributions satisfying the original constraints. Due to space limitations, the formal definition is given in [12]. Unfortunately, the notion of stuttering satisfaction as presented in Section 3 is not compatible with composition. This is formalized in the following theorem. Due to space limitations, the detailed proof is given in [12].

Theorem 7. *There exists two compatible (in the sense of composition) PAs P_1 and P_2 and two compatible (in the sense of composition) APAs N_1 and N_2 such that $P_1 \models^* N_1$, $P_2 \models^* N_2$ and $P_1 \parallel_{\bar{A}} P_2 \not\models^* N_1 \parallel_{\bar{A}} N_2$.*

The reason for this setback is the well known problem of distributed scheduling [17]. When composing two stuttering PAs, one allows interleaving of atomic stuttering steps from both sides, which generates extra behaviors. Our solution is to transform a PA P with a consistent set of hidden actions $\mathcal{H}$ into a non-stuttering PA $\widehat{P}^{\mathcal{H}}$ that satisfies the same APA specifications as P. This transformation removes stuttering by computing all the distributions that can be reached with stuttering in P and inserting them in the transition function of $\widehat{P}^{\mathcal{H}}$.

Definition 9. *Let $P = (S_P, A_P, L_P, AP, V_P, s_0^P)$ be a PA and let $\mathcal{H}$ be a consistent set of hidden actions for P. Define the PA $\widehat{P}^{\mathcal{H}} = (S_P, A_P \setminus \mathcal{H}, \widehat{L}_P^{\mathcal{H}}, AP, V_P, s_0^P)$ such that*

$$\forall s \in S, a \in A_P \setminus \mathcal{H}, \mu \in Dist(S),\ \widehat{L}_P^{\mathcal{H}}(s, a, \mu) = \top \iff s \xrightarrow[\mathcal{H}]{a}{}^* \mu \text{ in } P.$$

By construction, $\widehat{P}^{\mathcal{H}}$ is such that for all APA $N = (S, A_P \setminus \mathcal{H}, L, AP, V, s_0)$, we have

$$P \models^* N \iff \widehat{P}^{\mathcal{H}} \models N.$$

We have the following theorem.

Theorem 8. *Let $P_1 = (S_P^1, A_P^1, L_P^1, AP^1, V_P^1, s_0^{P_1})$ and $P_2 = (S_P^2, A_P^2, L_P^2, AP^2, V_P^2, [4]s_0^{P_2})$ be two PAs such that $AP^1 \cap AP^2 = \emptyset$. Let $N_1 = (S^1, A^1, L^1, AP^1, V^1, s_0^1)$ and $N_2 = (S^2, A^2, L^2, AP^2, V^2, s_0^2)$ be APAs such that $\mathcal{H}_1 = A_P^1 \setminus A_1$ and $\mathcal{H}_2 = A_P^2 \setminus A_2$ are consistent sets of hidden actions for P_1 and P_2 respectively, with $\mathcal{H}_1 \cap A_2 = \mathcal{H}_2 \cap A_1 = \emptyset$. For all $\bar{A} \subseteq A_1 \cap A_2$, we have the following:*

$$\text{if } P_1 \models^* N_1 \text{ and } P_2 \models^* N_2 \text{ then } \widehat{P_1}^{\mathcal{H}_1} \parallel_{\bar{A}} \widehat{P_2}^{\mathcal{H}_2} \models N_1 \parallel_{\bar{A}} N_2.$$

6 Future Work

In the future, we will study specifications with stuttering. This is complex as one will have to define a notion of may/must stutter transition in the specification APAs. The main problem is the constraints on distributions: the recursive step in the stutter transitions will have to take into account and propagate that the stutter remains valid for any solution of the constraints. Finally, all the work should also be implemented in APAC.

References

1. Andova, S., Willemse, T.A.C.: Branching bisimulation for probabilistic systems: Characteristics and decidability. Theor. Comput. Sci. 356, 325–355 (2006)
2. Baier, C., Hermanns, H.: Weak Bisimulation for Fully Probabilistic Processes. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 119–130. Springer, Heidelberg (1997)
3. Bauer, S.S., Juhl, L., Larsen, K.G., Legay, A., Srba, J.: Extending modal transition systems with structured labels. MSCS 22, 1–37 (2012)
4. Bauer, S.S., Mayer, P., Schroeder, A., Hennicker, R.: On Weak Modal Compatibility, Refinement, and the MIO Workbench. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 175–189. Springer, Heidelberg (2010)
5. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: Compositional design methodology with constraint markov chains. In: QEST. IEEE Computer (2010)
6. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint markov chains. Theor. Comput. Sci. 412, 4373–4404 (2011)
7. de Alfaro, L., Henzinger, T.A.: Interface automata. In: FSE, pp. 109–120. ACM Press (2001)
8. Delahaye, B., Katoen, J.-P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wąsowski, A.: Abstract Probabilistic Automata. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 324–339. Springer, Heidelberg (2011)
9. Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: APAC: a tool for reasoning about Abstract Probabilistic Automata. In: QEST. IEEE Computer (2011)
10. Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wąsowski, A.: New Results on Constraint Markov Chains. Performance Evaluation (2011) (to appear)
11. Delahaye, B., Katoen, J.-P., Larsen, K.G., Legay, A., Pedersen, M.L., Sher, F., Wasowski, A.: New Results on Abstract Probabilistic Automata. In: ACSD. IEEE Computer (2011)
12. Delahaye, B., Larsen, K.G., Legay, A.: Stuttering in abstract probabilistic automata (long version). Technical report (2012), http://delahaye.benoit.free.fr
13. Fischbein, D., Braberman, V., Uchitel, S.: A Sound Observational Semantics for Modal Transition Systems. In: Leucker, M., Morgan, C. (eds.) ICTAC 2009. LNCS, vol. 5684, pp. 215–230. Springer, Heidelberg (2009)
14. Georgievska, S.: Probability and Hiding in Concurrent Processes. PhD thesis, Eindhoven University of Technology (2011)
15. Georgievska, S., Andova, S.: Composing Systems While Preserving Probabilities. In: Aldini, A., Bernardo, M., Bononi, L., Cortellessa, V. (eds.) EPEW 2010. LNCS, vol. 6342, pp. 268–283. Springer, Heidelberg (2010)
16. Georgievska, S., Andova, S.: Probabilistic CSP: Preserving the Laws via Restricted Schedulers. In: Schmitt, J.B. (ed.) MMB & DFT 2012. LNCS, vol. 7201, pp. 136–150. Springer, Heidelberg (2012)
17. Giro, S., D'Argenio, P.R., Ferrer Fioriti, L.M.: Partial Order Reduction for Probabilistic Systems: A Revision for Distributed Schedulers. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 338–353. Springer, Heidelberg (2009)

18. Henzinger, T.A., Sifakis, J.: The Embedded Systems Design Challenge. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 1–15. Springer, Heidelberg (2006)
19. Hermanns, H., Herzog, U., Katoen, J.: Process algebra for performance evaluation. TCS 274, 43–87 (2002)
20. Larsen, K.G.: Modal Specifications. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 232–246. Springer, Heidelberg (1990)
21. Larsen, K.G., Skou, A.: Compositional Verification of Probabilistic Processes. In: Cleaveland, W.R. (ed.) CONCUR 1992. LNCS, vol. 630, pp. 456–471. Springer, Heidelberg (1992)
22. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. In: POPL, pp. 344–352 (1989)
23. Lowe, G.: Representing nondeterministic and probabilistic behaviour in reactive processes. Formal Asp. Comput. 3, 1 (1993)
24. Lynch, N., Tuttle, M.R.: An introduction to Input/Output automata. CWI-Quarterly 2 (1989)
25. Lynch, N., Segala, R., Vaandrager, F.: Compositionality for Probabilistic Automata. In: Amadio, R., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 208–221. Springer, Heidelberg (2003)
26. Mitra, S., Lynch, N.A.: Proving approximate implementations for probabilistic i/o automata. Electr. Notes Theor. Comput. Sci. 174, 71–93 (2007)
27. Morgan, C., McIver, A., Seidel, K., Sanders, J.W.: Refinement-oriented probability for CSP. Formal Asp. Comput. 8 (1996)
28. Philippou, A., Lee, I., Sokolsky, O.: Weak Bisimulation for Probabilistic Systems. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 334–349. Springer, Heidelberg (2000)
29. Raclet, J.-B.: Quotient de spécifications pour la réutilisation de composants. PhD thesis, Université de Rennes I (2007) (in French)
30. Segala, R.: Modeling and Verification of Randomized Distributed Real-time Systems. PhD thesis, MIT (1995)
31. Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. NJC 2, 250–273 (1995)
32. van Glabbeek, R.J.: The Linear Time - Branching Time Spectrum II. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 66–81. Springer, Heidelberg (1993)

Call-by-Value Non-determinism in a Linear Logic Type Discipline

Alejandro Díaz-Caro[1,⋆], Giulio Manzonetto[1,2], and Michele Pagani[1,2]

[1] Université Paris 13, Sorbonne Paris Cité, LIPN, F-93430, Villetaneuse, France
[2] CNRS, UMR 7030, F-93430, Villetaneuse, France

Abstract. We consider the call-by-value λ-calculus extended with a may-convergent non-deterministic choice and a must-convergent parallel composition. Inspired by recent works on the relational semantics of linear logic and non-idempotent intersection types, we endow this calculus with a type system based on the so-called Girard's second translation of intuitionistic logic into linear logic. We prove that a term is typable if and only if it is converging, and that its typing tree carries enough information to give a bound on the length of its lazy call-by-value reduction. Moreover, when the typing tree is minimal, such a bound becomes the exact length of the reduction.

Keywords: λ-calculus, linear logic, non-determinism, call-by-value.

1 Introduction

The intersection type discipline provides logical characterisations of operational properties of λ-terms, namely of various notions of termination, like head-, weak- and strong-normalisation (see [10,22], and [16] as a reference). The basic idea is to look at types as the set of terms having a given computational property — the type $\alpha \cap \beta$ being the set of those terms enjoying both properties α and β. With this intuition in mind, the intersection is naturally idempotent ($\alpha \cap \alpha = \alpha$).

Another way to understand the intersection type discipline is as a deductive system for presenting the compact elements of a specific reflexive Scott domain (see e.g. [1, §3.3]). The set of types assigned to a closed term captures the interpretation of such a term in the associated domain. Intersection types are then a powerful tool for enlightening the relations between denotational semantics, syntactical types and computational properties of programs.

Intersection types have been recently revisited in the setting of the relational semantics **Rel** of Linear Logic (LL). **Rel** is a semantics providing a more *quantitative* interpretation of the λ-calculus than Scott domains. Loosely speaking, the relational interpretation of a λ-term M not only tells us whether M converges on an argument, but in case it does, it also provides information on the number of times M needs to call[1] its argument to converge. Just like the intersection

⋆ Partially supported by grants from DIGITEO and Région Île-de-France.

[1] The notion of *calling an argument* should be made precise by specifying an operational semantics, which is usually achieved through an evaluating machine.

type discipline captures Scott domains, *non-idempotent* intersection type systems represent relational models. In this framework the type $\alpha_1 \cap \cdots \cap \alpha_k$ may be more accurately represented as the finite multiset $[\alpha_1, \ldots, \alpha_k]$. The lack of idempotency is the key ingredient to model the resource sensitiveness of **Rel** — while in the usual systems $M : \alpha \cap \beta$ stands for "M can be used either as data of type α or as data of type β", when the intersection is not idempotent the meaning of $M : [\alpha, \beta]$ becomes "M will be called *once* as data of type α and *once* as data of type β". Hence, types should no longer be understood as *sets of terms*, but rather as *sets of calls* to terms.

The first intersection type system based on **Rel** has been presented in [11], where de Carvalho introduced system R, a type discipline capturing the relational version of Engeler's model. More precisely, he proved that system R, beyond characterising converging terms, carries information on the evaluation sequence as well — the size of a derivation tree typing a term is a bound on the number of steps needed to reach a normal form. Similar results are obtained in [6] for a variant of system R characterising strong normalisation and giving a bound to the longest β-reduction sequence. More recently, Ehrhard introduced a non-idempotent intersection type system characterising the convergence in the call-by-value λ-calculus [14]. Also in this case, the size of a derivation tree bounds the length of the lazy (i.e. no evaluation under λ's) call-by-value β-reduction sequence. Our goal is to extend Ehrhard's system with non-determinism.

Our starting point is [9], where it is shown that the relational model $\mathcal{D}$ of the call-by-name λ-calculus provides a natural interpretation of both may and must non-determinism. Since **Rel** interprets λ-terms as relations, the *may*-convergent non-deterministic choice can be expressed in the model as the set-theoretical union. The *must*-convergent parallel composition, instead, is interpreted by using the operation $\mathcal{D} \otimes \mathcal{D} \multimap \mathcal{D}$ obtained by combining the mix rule $\mathcal{D} \otimes \mathcal{D} \multimap \mathcal{D} \parr \mathcal{D}$ with the contraction rule $\mathcal{D} \parr \mathcal{D} \multimap \mathcal{D}$, this latter holding since the call-by-name model $\mathcal{D}$ has shape $?A$ for $A = \mathcal{D}^{\mathbb{N}} \multimap \bot$. We will show that the same principle (*may*-convergence as *union* of interpretations and *must*-convergence as *mix* rule plus contraction) still works in the call-by-value setting.

Ehrhard's call-by-value type system is based on the so-called "second Girard's translation" of intuitionistic logic into LL [15,19]. The translation of a type α is actually given by two mutually defined mappings ($\alpha \mapsto \alpha^v$ and $\alpha \mapsto \alpha^c$) reflecting the two sorts (*values* and *computations*) at the basis of the call-by-value λ-calculus:

$$\iota^v = \iota, \qquad (\alpha \to \beta)^v = \alpha^c \multimap \beta^c, \qquad \alpha^c = !\alpha^v,$$

where ι is an atom. Hence, the relational model described by Ehrhard's typing system yields a solution to the equation $\mathcal{V} \simeq !\mathcal{V} \multimap !\mathcal{V}$ in **Rel**. Since in this semantics $\multimap$ is interpreted by the cartesian product and ! by finite multisets, a functional type for a value in this system is a pair (p, q) of types for computations, and a type for a computation is a multiset $[\alpha_1, \ldots, \alpha_n]$ of value types (representing n calls to a single value that must behave as $\alpha_1, \ldots, \alpha_n$).

In order to deal with the must non-determinism, namely the parallel composition, we must add to the translation considered by Ehrhard a further exponential level, called here the *parallel sort*:

$$\iota^v = \iota, \qquad (\alpha \to \beta)^v = \alpha^c \multimap \beta^{\|}, \qquad \alpha^c = !\alpha^v, \qquad \alpha^{\|} = ?\alpha^c. \qquad (1)$$

This translation enjoys the nice property of mapping the call-by-value λ-calculus into the polarised fragment of LL, as described by Laurent in [17]. Then, our typing system is describing an object in **Rel** satisfying the equation $\mathcal{V} \simeq !\mathcal{V} \multimap ?!\mathcal{V}$, where the ? connective is interpreted by the finite multiset operator. In this setting a value type is a pair $(p, [q_1, \dots, q_n])$ of a computational type p and a parallel type, that is a multiset of computations $q_1, \dots, q_n$. Intuitively, a value of that type needs a computation of type p to create a parallel composition of n computations of types $q_1, \dots, q_n$, respectively. Notice that, following [9], the composition of the mix rule and the contraction one yields an operation $?!\mathcal{V} \otimes ?!\mathcal{V} \multimap ?!\mathcal{V}$ which is used to interpret the parallel composition.

To avoid a clumsy notation with multisets of multisets, we prefer to denote a !-multiset $[\alpha_1, \dots, \alpha_m]$ (the type of a computation) with the linear logic multiplicative conjunction $\alpha_1 \otimes \cdots \otimes \alpha_m$, a ?-multiset $[q_1, \dots, q_n]$ (the type of a parallel composition of computations) with the multiplicative disjunction $q_1 \parr \cdots \parr q_n$, and finally a pair $(p, [q_1, \dots, q_n])$ with the linear implication $p \multimap (q_1 \parr \cdots \parr q_n)$. Such a notation stresses the fact that the non-idempotent intersection type systems issued from **Rel** are essentially contained in the multiplicative fragment of LL (modulo the associativity, commutativity and neutrality equivalences).

Contents. Several non-deterministic extensions of the λ-calculus have been proposed in the literature, both in the call-by-name (e.g. [9,12]) and in the call-by-value setting (e.g. [7,13]). In the present paper we focus on the call-by-value λ-calculus, first introduced in [21], endowed with two binary operators $+$ and $\|$ representing non-deterministic choice and parallel composition, respectively. The resulting calculus, denoted here $\Lambda_{+\|}$, is quite standard and its operational semantics is given in Section 2 through a machine performing lazy call-by-value reduction. Following [9], we model non-deterministic choice as *may* non-determinism and parallel composition as *must*. This is reflected in our reduction and in our notion of convergence. Indeed, every time the machine encounters $M + N$ in active position it actually performs a choice, while encountering $M \parallel N$ it interleaves reductions in M and in N; finally a term M converges when there is a reduction of the machine from M to a normal form.

Section 3 is devoted to provide the type discipline for $\Lambda_{+\|}$, based on the multiplicative fragment of LL (as discussed above), and to define a measure $|\cdot|$ associating a number with every type derivation. Such a measure "extracts" from the information present in the typing tree of a term, a bound on the length of its evaluation. In Section 4 we show that our type system satisfies good properties like subject reduction and expansion. We also prove that the measure associated with the typing tree of a term decreases by 1 at every reduction step, giving thus a proof of weak normalisation in ω for typable terms. From these properties it ensues directly that a term is typable if and only if it converges. Moreover, thanks

β_v-reduction
$(\lambda x.M)V \to M[V/x]$

$+$-reductions
$M + N \to M$
$M + N \to N$

$\|$-reductions
$(M \parallel N)P \to MP \parallel NP$
$V(M \parallel N) \to VM \parallel VN$

Contextual rules

$$\frac{M \to M'}{M \parallel N \to M' \parallel N} \qquad \frac{N \to N'}{M \parallel N \to M \parallel N'} \qquad \frac{M \to M' \quad (*)}{MN \to M'N} \qquad \frac{M \to M' \quad (*)}{VM \to VM'}$$

Fig. 1. Reduction semantics for $\Lambda_{+\|}$. The condition $(*)$ stands for "$M \neq P \parallel Q$".

to the resource consciousness of our type system, we are able to strengthen such a result — we prove that, whenever M converges, there is a type derivation $\vdash M : \alpha$ (with α satisfying a suitable minimality condition) such that the associated measure provides the exact number of steps reducing M to a normal form.

Finally, in Section 5 we discuss the properties of the model in **Rel** underlying our system. As expected, the interpretation turns out to be adequate, i.e. a term converges if and only if its interpretation is non-empty. On the other hand such a model is not fully abstract — there are terms having different interpretations and that cannot be (semi-)separated using applicative contexts. Our counterexample does not rely on the presence of $+$ and $\|$.

2 The Call-by-Value Non-deterministic Machine

We consider the call-by-value λ-calculus [21], extended with non-deterministic and parallel operators in the spirit of [9]. The set $\Lambda_{+\|}$ of *terms* and the set $\mathrm{V}_{+\|}$ of *values* are defined by mutual induction as follows (where x ranges over a countable set Var of variables):

$$\begin{array}{lrcll} \textit{Terms:} & M, N, P, Q & ::= & V \mid MN \mid M + N \mid M \parallel N & \Lambda_{+\|} \\ \textit{Values:} & V & ::= & x \mid \lambda x.M & \mathrm{V}_{+\|} \end{array}$$

Intuitively, $M+N$ denotes the *non-deterministic choice* between M and N, while $M \parallel N$ stands for their parallel composition. Such operators are not required to be associative nor commutative. As usual, we suppose that application associates to the left and λ-abstraction to the right. Moreover, to lighten the notation, we assume that application and λ-abstraction take precedence over $+$ and $\|$.

The *α-conversion* and the set $\mathrm{FV}(M)$ of *free variables of* M are defined as usual in λ-calculus [5, §2.1]. A term M is *closed* whenever $\mathrm{FV}(M) = \emptyset$.

Given $M \in \Lambda_{+\|}$ and $V \in \mathrm{V}_{+\|}$, we denote by $M[V/x]$ the term obtained by simultaneously substituting the value V for all free occurrences of x in M, subject to the usual proviso about renaming bound variables in M to avoid capture of free variables in V. Hereafter terms are considered up to α-conversion.

Definition 1 (Operational semantics). *The operational semantics of $\Lambda_{+\|}$ is given in Figure 1. We denote by $\to^*$ the transitive and reflexive closure of $\to$.*

The side condition $(*)$ on the context rules for the application avoids critical pairs with the $\|$-rules: this is not actually needed but it simplifies some proofs. A term M is called a *normal form* if there is no $N \in \Lambda_{+\|}$ such that $M \to N$. In particular, all (parallel compositions of) values are normal forms. Note that when M is closed then either it is a parallel composition of values or it reduces.

Definition 2. *A closed term $M \in \Lambda_{+\|}$* converges *if and only if there exists a reduction $M \to^* V_1 \| \cdots \| V_n$ for some $V_i \in \mathrm{V}_{+\|}$.*

The intuitive idea underlying the above notion of convergence is the following:

- The non-deterministic choice $M + N$ is treated as *may*-convergent, either of the alternatives may be chosen during the reduction and the sum converges if either M or N does.
- The parallel composition $M \| N$ is modelled as *must*-convergent, the reduction forks and the parallel composition converges if both M and N do.

Let us provide some examples. We set $\mathbf{I} = \lambda x.x$, $\mathbf{\Delta} = \lambda x.xx$ and we denote by $\mathbf{\Omega}$ the paradigmatic non-converging term $\mathbf{\Delta\Delta}$, which reduces to itself as $\mathbf{\Delta}$ is a value. The reduction is *lazy*, i.e. it does not reduce under abstractions, so for example $\lambda y.\mathbf{\Omega}$ is a normal form. In fact, when considering closed terms, the parallel compositions of values are exactly the normal forms, thus justifying Definition 2. We would like to stress that our system is designed in such a way that a parallel composition of values is not a value. As a consequence, the term $P = \lambda k.\mathbf{\Delta} \| \mathbf{\Delta}$ is not a value, so the term $(\lambda x.x\mathbf{I}x)P$ is converging. Indeed, it reduces to $(\lambda x.x\mathbf{I}x)(\lambda k.\mathbf{\Delta}) \| (\lambda x.x\mathbf{I}x)\mathbf{\Delta} \to^* \mathbf{\Delta} \| \mathbf{\Delta}$. Notice that, if we consider P as a value, then $(\lambda x.x\mathbf{I}x)P$ would diverge since it would reduce to $P\mathbf{I}P \to^* (\mathbf{\Delta} \| \mathbf{I})P \to^* \mathbf{\Delta}P \| P$ and one can check easily that $\mathbf{\Delta}P$ diverges.

The presence of the non-deterministic choice $+$ enlightens a typical feature of the call-by-value λ-calculus: application is bilinear (i.e. it commutes with $+$) while abstraction is not linear. Indeed, one can prove that $(M+M')(N+N')$ and $MN+MN'+M'N+M'N'$ are operationally indistinguishable, while $\lambda x.(M+N)$ and $\lambda x.M + \lambda x.N$, in general, are not. For example, take $S = \lambda x.(x + \mathbf{I})$, $S' = \lambda x.x + \lambda x.\mathbf{I}$, $E_{\mathbf{I}} = \lambda x.\mathbf{I}$, $E_{\mathbf{\Omega}} = \lambda x.\mathbf{\Omega}$, and $F = \lambda b.bE_{\mathbf{\Omega}}(bE_{\mathbf{I}}E_{\mathbf{\Omega}})\mathbf{I}$. Now observe that FS is converging to the value $\mathbf{I}$, while FS' diverges. Indeed, remarking that $SE_{\mathbf{I}}E_{\mathbf{\Omega}}$ reduces non-deterministically to $\mathbf{I}$ and to $E_{\mathbf{\Omega}}$, we have:

$$\begin{tikzcd}
 & & & & E_{\mathbf{\Omega}}\mathbf{II} \arrow[dr] & \\
 & & & E_{\mathbf{\Omega}}(SE_{\mathbf{I}}E_{\mathbf{\Omega}})\mathbf{I} \arrow[ur, "*"] \arrow[dr, "*"'] & & \mathbf{\Omega I} \\
 & & & & E_{\mathbf{\Omega}}E_{\mathbf{\Omega}}\mathbf{I} \arrow[ur] & \\
FS \arrow[r] & SE_{\mathbf{\Omega}}(SE_{\mathbf{I}}E_{\mathbf{\Omega}})\mathbf{I} \arrow[r] & (E_{\mathbf{\Omega}} + \mathbf{I})(SE_{\mathbf{I}}E_{\mathbf{\Omega}})\mathbf{I} \arrow[uur] \arrow[ddr] & & & \\
 & & & & \mathbf{III} \arrow[r, "*"] & \mathbf{I} \\
 & & & \mathbf{I}(SE_{\mathbf{I}}E_{\mathbf{\Omega}})\mathbf{I} \arrow[ur, "*"] \arrow[dr, "*"'] & & \\
 & & & & \mathbf{I}E_{\mathbf{\Omega}}\mathbf{I} \arrow[r, "*"] & \mathbf{\Omega}
\end{tikzcd}$$

while FS' has two reducts, either $F\mathbf{I}$ reducing to $\mathbf{\Omega I}$, or $FE_{\mathbf{I}}$ reducing to $\mathbf{\Omega}$.

$$\frac{}{x:\tau\vdash x:\tau}\,ax \qquad \frac{\Delta_i, x:\tau_i\vdash M:\alpha_i \qquad 1\le i\le n}{\bigotimes_{i=1}^{n}\Delta_i\vdash \lambda x.M:\bigotimes_{i=1}^{n}(\tau_i\multimap\alpha_i)}\multimap_I \quad n\ge 0$$

$$\frac{\Delta\vdash M:\mathop{\Large\parr}_{i=1}^{k}\bigotimes_{j=1}^{n_i}(\tau_{ij}\multimap\alpha_{ij}) \qquad \Gamma_i\vdash N:\mathop{\Large\parr}_{j=1}^{n_i}\tau_{ij} \quad 1\le i\le k}{\Delta\otimes\bigotimes_{i=1}^{k}\Gamma_i\vdash MN:\mathop{\Large\parr}_{i=1}^{k}\mathop{\Large\parr}_{j=1}^{n_i}\alpha_{ij}}\multimap_E \quad \begin{matrix}k\ge 1\\ n_i\ge 1\end{matrix}$$

$$\frac{\Delta\vdash M:\alpha}{\Delta\vdash M+N:\alpha}+_\ell \qquad \frac{\Delta\vdash N:\alpha}{\Delta\vdash M+N:\alpha}+_r \qquad \frac{\Delta\vdash M:\alpha_1 \qquad \Gamma\vdash N:\alpha_2}{\Delta\otimes\Gamma\vdash M\parallel N:\alpha_1\parr\alpha_2}\parallel_I$$

Fig. 2. Type system: the inference rules

Finally, we give two examples mixing $+$ and $\parallel$. The term $(\lambda x.(x\parallel x))(V+V')$ converges either to $V\parallel V$ or to $V'\parallel V'$, while the term $(\lambda x.(x+x))(V\parallel V')$ converges to $V\parallel V'$, only.

3 Linear Logic Based Type System

In this section we introduce our type system based on linear logic. The set $\mathbb{T}$ of *(parallel) types* and the set $\mathbb{C}$ of *computational types* are generated by the following grammar:

parallel-types:	$\alpha,\beta ::= \alpha\parr\beta \mid \tau$	$\mathbb{T}$
computational-types:	$\tau,\rho ::= \mathbf{1}\mid\tau\otimes\rho\mid\tau\multimap\alpha$	$\mathbb{C}$

For the sake of simplicity, types are considered up to associativity and commutativity of the tensor $\otimes$ and the par $\parr$. The type $\mathbf{1}$, which is the only atomic type, represents the empty tensor and is therefore its neutral element (i.e. $\tau\otimes\mathbf{1}=\tau$). Accordingly, we write $\otimes_{i=1}^{n}\tau_i$ for $\tau_1\otimes\cdots\otimes\tau_n$ when $n\ge 1$, and for $\mathbf{1}$ when $n=0$. Similarly, when $n\ge 1$, $\parr_{i=1}^{n}\alpha_i$ stands for $\alpha_1\parr\cdots\parr\alpha_n$.

As mentioned in the introduction, $\tau_1\otimes\cdots\otimes\tau_n$ and $\alpha_1\parr\cdots\parr\alpha_k$ are actually notations representing two different kinds of multisets, namely the !- and ?-multisets (respectively). Under this correspondence, $\mathbf{1}$ represent the empty !-multiset. We do not allow the empty par as it would correspond to an empty sum of terms, that would be delicate to treat operationally (cf. [4]).

Note that neither $\otimes$ nor $\parr$ are supposed idempotent.

Definition 3. *A* context Γ *is a total map from* Var *to* $\mathbb{C}$*, such that* $\mathrm{dom}(\Gamma)=\{x\mid\Gamma(x)\ne\mathbf{1}\}$ *is finite. The tensor of two contexts* Γ *and* Δ*, written* $\Gamma\otimes\Delta$*, is defined pointwise.*

As a matter of notation, we write $x_1 : \tau_1, \dots, x_n : \tau_n$ for the context Γ such that $\Gamma(x_i) = \tau_i$ and $\Gamma(y) = \mathbf{1}$ for all $y \notin \vec{x}$. The context mapping all variables to $\mathbf{1}$ is denoted by $\emptyset$; note that $\Gamma \otimes \emptyset = \Gamma$.

Definition 4. – *The* type system *for $\Lambda_{+\|}$ is defined in Figure 2.* Typing judgements *are of the form $\Gamma \vdash M : \alpha$; when $\Gamma = \emptyset$ we simply write $\vdash M : \alpha$.* Derivation trees *will be denoted by π.*
– *A term $M \in \Lambda_{+\|}$ is* typable *if there exist $\alpha \in \mathbb{T}$ and a context Γ such that $\Gamma \vdash M : \alpha$.*

The rules for typing non-deterministic choice and parallel composition reflect their operational behaviour. Non-deterministic choice is may-convergent, thus it is enough to ask that one of the terms in a sum is typable; on the other hand parallel composition is must-convergent, we therefore require that all its components are typable. Intuitively, when dealing with closed terms, the $\parr$ operator can be only introduced to type a parallel composition, and gives an account of the number of its components. In fact, for closed regular λ-terms, the type system looses the $\parr$-level and collapses to the one presented in [14].

The $\multimap_E$ rule reflects the distribution of the parallel operator over the application. For example, take $M = x \parallel x'$ and $N = y \parallel y'$ in the premises of $\multimap_E$, then we have $k = 2$ and $n_1 = n_2 = 2$ so that the type of the term MN is a $\parr$ of four types, which is in accordance with $(x \parallel x')(y \parallel y') \to^* (xy \parallel xy') \parallel (x'y \parallel x'y')$.

Remark 1. *For every $V \in \mathrm{V}_{+\|}$ we can derive $\vdash V : \mathbf{1}$. Indeed, if V is a variable, then the derivation follows by ax; if V is an abstraction, then it follows by $\multimap_I$ using $n = 0$. As a simple consequence we get $\vdash V_1 \parallel \cdots \parallel V_k : \mathbf{1} \parr \cdots \parr \mathbf{1}$ (k times) for all $V_1, \dots, V_k \in \mathrm{V}_{+\|}$.*

Concerning the possible types of values, the next more general lemma holds.

Lemma 1. *Let $V \in \mathrm{V}_{+\|}$. If $\Delta \vdash V : \alpha$ then $\alpha \in \mathbb{C}$.*

Proof. A proof of $\Delta \vdash V : \alpha$ ends in either a *ax* or a $\multimap_I$ rule. In both cases α is a computational-type. □

To help the reader to get familiar with the type system, we provide some examples of typable and untypable terms.

Example 1. Recall that $\mathbf{I} = \lambda x.x$, $\mathbf{\Delta} = \lambda x.xx$ and $\mathbf{\Omega} = \mathbf{\Delta}\mathbf{\Delta}$.

1. $\vdash \mathbf{I} : \bigotimes_{i=1}^{n}(\tau_i \multimap \tau_i)$ and $\vdash \lambda x.\mathbf{I} : \bigotimes_{i=1}^{n}(\mathbf{1} \multimap \bigotimes_{j=1}^{k_i}(\tau_{ij} \multimap \tau_{ij}))$.
2. $\vdash \mathbf{\Delta} : \bigotimes_{i=1}^{n}((\tau_i \multimap \alpha_i) \otimes \tau_i) \multimap \alpha_i$.
3. $\mathbf{\Omega}$ is not typable. By contradiction, suppose $\vdash \mathbf{\Omega} : \alpha$. By ($\multimap_E$) and (2) there is a type τ such that $\vdash \mathbf{\Delta} : \tau \multimap \alpha$ and $\vdash \mathbf{\Delta} : \tau$. Let us choose such a τ with minimal size. Applying (2) to $\vdash \mathbf{\Delta} : \tau \multimap \alpha$, we get $\tau = (\tau' \multimap \alpha) \otimes \tau'$, from which one can deduce (see Lemma 2, below) that $\vdash \mathbf{\Delta} : \tau' \multimap \alpha$ and $\vdash \mathbf{\Delta} : \tau'$, thus contradicting the minimality of τ.
4. However, $\vdash \lambda x.\mathbf{\Omega} : \mathbf{1}$, so $\vdash \lambda x.\mathbf{\Omega} + \mathbf{\Omega} : \mathbf{1}$, but $\lambda x.\mathbf{\Omega} \parallel \mathbf{\Omega}$ is not typable.
5. From (1) and (4) we get: $\vdash \mathbf{I} \parallel \lambda x.\mathbf{\Omega} : (\bigotimes_{i=1}^{n}(\tau_i \multimap \tau_i)) \parr \mathbf{1}$.

We now define a measure associating a natural number with every derivation tree. In Section 4.1 we prove that such a measure decreases along the reduction. In the next definition we follow the notation of Figure 2, in particular in the $\multimap_E$-case the parameter n_i refers to the arity of the $⅋$ in the conclusion of π_i.

Definition 5. *The* measure $|\pi|$ *of a derivation tree* π *is defined inductively as:*

$$\pi = \frac{}{S}\, ax \qquad |\pi| = 0$$

$$\pi = \frac{\pi_1 \quad \cdots \quad \pi_n}{S} \multimap_I \qquad |\pi| = \textstyle\sum_{i=1}^{n} |\pi_i|$$

$$\pi = \frac{\pi_0 \quad \pi_1 \ldots \pi_k}{S} \multimap_E \quad \begin{matrix} k \geq 1 \\ n_i \geq 1 \end{matrix} \qquad |\pi| = \textstyle\sum_{i=0}^{k} |\pi_i| + (\sum_{i=1}^{k} 2n_i) - 1$$

$$\pi = \frac{\pi'}{S} +_\ell \quad \mathit{or} \quad \pi = \frac{\pi'}{S} +_r \qquad |\pi| = |\pi'| + 1$$

$$\pi = \frac{\pi_1 \qquad \pi_2}{S} \parallel_I \qquad |\pi| = |\pi_1| + |\pi_2|$$

Hereafter, we may slightly abuse the notation and write $\pi = \Gamma \vdash M : \alpha$ to refer to a derivation tree π ending by the sequent $\Gamma \vdash M : \alpha$.

The measure of a derivation only depends on its rules of type $\multimap_E$, $+_\ell$ and $+_r$. These are in fact the kinds of rules that can type a redex (β_v and $\parallel$ redexes are typed by $\multimap_E$ rules, $+$ redexes by $+_\ell$, $+_r$ rules). Each occurrence of a $+_\ell$ or $+_r$ rule counts for one, because a $+$-reduction does not create new rules in the derivation typing the contractum (see the proof of Theorem 2 for more details). An occurrence of a $\multimap_E$ counts for the number of "active" connectives appearing in the principal premise, i.e. the number of the connectives that are underlined in the left-most premise of the $\multimap_E$ rule in Figure 2, indeed

$$\underbrace{\sum_{i=1}^{k} n_i}_{\multimap\text{'s}} + \underbrace{\sum_{i=1}^{k} (n_i - 1)}_{\otimes\text{'s}} + \underbrace{(k-1)}_{⅋\text{'s}} = (\sum_{i=1}^{k} 2n_i) - 1.$$

Such a weight is needed since the $\parallel$-reduction creates two new $\multimap_E$ rules in the derivation typing the contractum. The measure decreases however, since the sum of the weight of the two new rules is less than the weight of the eliminated rule.

For example, let us consider the derivation tree π in Figure 3, which types the $\parallel$-redex $\mathbf{\Delta}(\mathbf{I} \parallel \lambda xy.\mathbf{\Omega})$ with $\mathbf{1} ⅋ \mathbf{1}$, and has three $\multimap_E$ rules — one of weight 1 in each subtree π_1, π_2, and one of weight 3 giving the conclusion, so that $|\pi| = 5$. Now, the $\multimap_E$-rule ending π splits into two $\multimap_E$-rules in the derivation tree π' typing the contractum of $\mathbf{\Delta}(\mathbf{I} \parallel \lambda xy.\mathbf{\Omega})$, namely $\pi' = \ \vdash \mathbf{\Delta I} \parallel \mathbf{\Delta}(\lambda xy.\mathbf{\Omega}) : \mathbf{1} ⅋ \mathbf{1}$. However, $|\pi'| = |\pi| - 1$ since the number of the active connectives of the $\multimap_E$-rule concluding π is greater than the sum of the number of the active connectives of its "residuals" in π'.

Finally, note that the term $\mathbf{\Delta}(\mathbf{I} \parallel \lambda xy.\mathbf{\Omega})$ reduces to the value $\mathbf{I} \parallel \lambda y.\mathbf{\Omega}$ in $5 = |\pi|$ steps. As we will show in Theorem 4 this does not happen by chance.

$$\pi = \dfrac{\dfrac{\pi_1 = x:\tau \vdash xx:\mathbf{1} \quad \pi_2 = x:\tau \vdash xx:\mathbf{1}}{\vdash \mathbf{\Delta} : (\tau \multimap \mathbf{1}) \otimes (\tau \multimap \mathbf{1})}\multimap_I \qquad \dfrac{\vdash \mathbf{I}:\tau \quad \vdash \lambda xy.\mathbf{\Omega}:\tau}{\vdash \mathbf{I} \parallel \lambda xy.\mathbf{\Omega} : \tau \parr \tau}\parallel_I}{\vdash \mathbf{\Delta}(\mathbf{I} \parallel \lambda xy.\mathbf{\Omega}) : \mathbf{1} \parr \mathbf{1}}\multimap_E$$

$$\pi' = \dfrac{\dfrac{\dfrac{\pi_1 = x:\tau \vdash xx:\mathbf{1}}{\vdash \mathbf{\Delta} : \tau \multimap \mathbf{1}}\multimap_I \quad \vdash \mathbf{I}:\tau}{\vdash \mathbf{\Delta I} : \mathbf{1}}\multimap_E \qquad \dfrac{\dfrac{\pi_2 = x:\tau \vdash xx:\mathbf{1}}{\vdash \mathbf{\Delta} : \tau \multimap \mathbf{1}}\multimap_I \quad \vdash \lambda xy.\mathbf{\Omega}:\tau}{\vdash \mathbf{\Delta}(\lambda xy.\mathbf{\Omega}) : \mathbf{1}}\multimap_E}{\mathbf{\Delta I} \parallel \mathbf{\Delta}(\lambda xy.\mathbf{\Omega}) : \mathbf{1} \parr \mathbf{1}}\parallel_I$$

Fig. 3. Derivation trees typing, respectively, the $\parallel$-redex $\mathbf{\Delta}(\mathbf{I} \parallel \lambda xy.\mathbf{\Omega})$ and its contractum $\mathbf{\Delta I} \parallel \mathbf{\Delta}(\lambda xy.\mathbf{\Omega})$, taking $\tau = (\mathbf{1} \multimap \mathbf{1}) = (\mathbf{1} \multimap \mathbf{1}) \otimes \mathbf{1}$

4 Properties of the Type System

We prove that the set of types assigned to a term is invariant under $\to$, in a non-deterministic setting. More precisely, Theorem 2 states that if N is the contractum of a $\{\beta_v, \|\}$-redex in M, then any type of M is a type of N, and if N and N' are the two possible contracta of a $+$-redex in M, then any type of M is either a type of N or of N' (*subject reduction*). On the other hand Theorem 3 shows the converse, namely that whenever $M \to N$, any type of N is a type of M (*subject expansion*).

Moreover, the two theorems combined prove that the measure associated with the typing tree of a term decreases (resp. increases) of exactly one unit at each typed step of reduction (resp. expansion). This is typical of non-idempotent intersection type systems, as discussed in the introduction. As a consequence, any typable term M is normalising and the measure of specific derivation trees of M gives the length of a converging reduction sequence.

4.1 Subject Reduction

In order to prove subject reduction we first need some preliminary lemmas. Their proofs are lengthy but not difficult, therefore we write explicitly only the most interesting cases.

Lemma 2. *We have that* $\pi = \Delta \vdash V : \bigotimes_{i=1}^{n} \tau_i$ *if and only if* $\Delta = \bigotimes_{i=1}^{n} \Delta_i$ *and* $\pi_i = \Delta_i \vdash V : \tau_i$ *for all* $i = 1, \ldots, n$. *Moreover,* $|\pi| = \sum_{i=1}^{n} |\pi_i|$.

Proof. We only prove ($\Rightarrow$), the other direction being similar. Since V is a value, the last rule of π is either *ax* or $\multimap_I$. The first case is trivial. In the second case, $V = \lambda x.M$ and the premises of the $\multimap_I$-rule are $m \geq n$, say $\pi'_j = \Delta_j, x :$

$\rho_j \vdash M : \alpha_j$ for $j \leq m$, and $\tau_1 = \bigotimes_{j=1}^{m_1} \rho_j \multimap \alpha_j$ and $\Delta_1 = \bigotimes_{j=1}^{m_1} \Delta_j, \ldots,$ $\tau_n = \bigotimes_{j=m_{n-1}+1}^{m_n} \rho_j \multimap \alpha_j$ and $\Delta_n = \bigotimes_{j=m_{n-1}+1}^{m_n} \Delta_j$, with $m_1 + \cdots + m_n = m$.

Notice $|\pi| = \sum_{j=1}^{m} |\pi'_j|$. Then, for every $i \leq n$, a $\multimap_I$-rule with premises $\pi'_{m_{i-1}+1}, \ldots, \pi'_{m_i}$ yields $\pi_i = \Delta_i \vdash \lambda x.M : \tau_i$, with $|\pi_i| = \sum_{j=m_{i-1}+1}^{m_i} |\pi'_i|$, therefore $|\pi| = \sum_{i=1}^{n} |\pi_i|$. □

Lemma 3 (Substitution lemma). *If $\pi_1 = \Delta, x : \tau \vdash M : \alpha$ and $\pi_2 = \Gamma \vdash V : \tau$, then there is $\pi_3 = \Delta \otimes \Gamma \vdash M[V/x] : \alpha$. Moreover $|\pi_3| = |\pi_1| + |\pi_2|$.*

Proof. By structural induction on M. We only treat the most interesting case, namely $M = NP$. In this case, the last rule of π_1 is a $\multimap_E$-rule with $k+1$ premises, say $\pi_1^0 = \Delta_0, x : \tau_0 \vdash N : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\rho_{ij} \multimap \alpha_{ij})$, and for $i = 1, \ldots, k$, $\pi_1^i = \Delta_i, x : \tau_i \vdash P : \bigparr_{j=1}^{n_i} \rho_{ij}$, where $\Delta = \bigotimes_{i=0}^{k} \Delta_i$, $\tau = \bigotimes_{i=0}^{k} \tau_i$, $\alpha = \bigparr_{i=1}^{k} \bigparr_{j=1}^{n_i} \alpha_{ij}$ and $|\pi_1| = \sum_{i=0}^{k} |\pi_1^i| + (\sum_{i=1}^{k} 2n_i) - 1$. By Lemma 2, we can split π_2 into $k+1$ derivations $\pi_2^i = \Gamma_i \vdash V : \tau_i$, for $i = 0, \ldots, k$, such that $\Gamma = \bigotimes_{i=0}^{k} \Gamma_i$ and $|\pi_2| = \sum_{i=0}^{k} |\pi_2^i|$. By the induction hypothesis, there are $\pi_3^0 = \Delta_0 \otimes \Gamma_0 \vdash N[V/x] : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\rho_{ij} \multimap \alpha_{ij})$, with $|\pi_3^0| = |\pi_1^0| + |\pi_2^0|$, and for $i = 1, \ldots, k$, $\pi_3^i = \Delta_i \otimes \Gamma_i \vdash P[V/x] : \bigparr_{j=1}^{n_i} \rho_{ij}$, with $|\pi_3^i| = |\pi_1^i| + |\pi_2^i|$. Hence, by rule $\multimap_E$, we have

$$\pi_3 = (\Delta_0 \otimes \Gamma_0) \otimes \bigotimes_{i=1}^{k} (\Delta_i \otimes \Gamma_i) \vdash N[V/x]P[V/x] : \bigparr_{i=1}^{k} \bigparr_{j=1}^{n_i} \alpha_{ij}$$

Notice that $(\Delta_0 \otimes \Gamma_0) \otimes \bigotimes_{i=1}^{k} (\Delta_i \otimes \Gamma_i) = \Delta \otimes \Gamma$ and $N[V/x]P[V/x] = (NP)[V/x]$. Moreover, $|\pi_3| = \sum_{i=0}^{k} |\pi_3^i| + (\sum_{i=1}^{k} 2n_i) - 1 = \sum_{i=0}^{k} (|\pi_1^i| + |\pi_2^i|) + (\sum_{i=1}^{k} 2n_i) - 1 = (\sum_{i=0}^{k} |\pi_1^i| + (\sum_{i=1}^{k} 2n_i) - 1) + \sum_{i=0}^{k} |\pi_2^i| = |\pi_1| + |\pi_2|$. □

We now prove the subject reduction property, which ensures that the type is preserved during reduction, while the measure of the typing is strictly decreasing.

As a matter of terminology, we say that a term M *reduces to a term N using $+$-reductions*, if $M \to N$ is derivable as a direct consequence of a $+$-reduction and (possibly) some contextual rules. In the following proof, given a set S, we denote by $\sharp S$ its cardinality.

Theorem 2 (Subject reduction). *Let $\pi = \Delta \vdash M : \alpha$.*

– *If $M \to N$ without using $+$-reductions, then there is $\pi' = \Delta \vdash N : \alpha$.*
– *If $M \to N_1$ and $M \to N_2$ using $+$-reductions, then there is π' such as either $\pi' = \Delta \vdash N_1 : \alpha$ or $\pi' = \Delta \vdash N_2 : \alpha$.*

Moreover, in both cases we have $|\pi'| = |\pi| - 1$.

Proof. We proceed by induction on the length of the derivation of $M \to N$. We only treat the most interesting cases.

– $(\lambda x.M')V \to M'[V/x]$. Then, the last rule of π is a $\multimap_E$-rule with $k+1$ premises, say $\pi_0 = \Delta' \vdash \lambda x.M' : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\rho_{ij} \multimap \alpha_{ij})$ and for every

$i = 1, \dots, k$, $\pi_i = \Gamma_i \vdash V : \parr_{j=1}^{n_i} \rho_{ij}$, with moreover $\Delta = \Delta' \otimes \bigotimes_{i=1}^{k} \Gamma_i$, $\alpha = \parr_{i=1}^{k} \parr_{j=1}^{n_i} \alpha_{ij}$, and $|\pi| = \sum_{i=0}^{k} |\pi_i| + (\sum_{i=1}^{k} 2n_i) - 1$. However, since Lemma 1 entails that $k = n_1 = 1$ we get $|\pi| = |\pi_0| + |\pi_1| + 1$. In addition, the only possibility for π_0 is to come from $\pi'_0 = \Delta', x : \rho \vdash M' : \alpha$, where $|\pi_0| = |\pi'_0|$. By Lemma 3, $\pi' = \Delta' \otimes \Gamma \vdash M'[V/x] : \alpha$, where $|\pi'| = |\pi'_0| + |\pi_1| = |\pi_0| + |\pi_1| = |\pi| - 1$. We conclude since $\Delta' \otimes \Gamma = \Delta$.

– Let $V(M \parallel N) \to VM \parallel VN$. Then $\pi = \Delta \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash V(M \parallel N) : \parr_{i=1}^{k} \parr_{j=1}^{n_i} \alpha_{ij}$ ends in a $\multimap_E$ rule having as premises $\pi_0 = \Delta \vdash V : \parr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\rho_{ij} \multimap \alpha_{ij})$ and, for $i = 1, \dots, k$, $\pi_i = \Gamma_j \vdash M \parallel N : \parr_{j=1}^{n_i} \rho_{ij}$. Thus, we have $|\pi| = \sum_{j=0}^{k} |\pi_i| + (\sum_{i=1}^{k} 2n_i) - 1$. However, by Lemma 1, $k = 1$, so we omit the index i where it is not needed, and $|\pi| = |\pi_0| + |\pi_1| + 2n - 1$. Then $\pi_1^1 = \Gamma_1 \vdash M : \parr_{j \in S} \rho_j$ and $\pi_1^2 = \Gamma_2 \vdash N : \parr_{j \in \bar{S}} \rho_j$, where $\Gamma = \Gamma_1 \otimes \Gamma_2, \emptyset \neq S \subsetneq \{1, \dots, k\}$ and $\bar{S} = \{1, \dots, k\} \backslash S$ with $|\pi_1| = |\pi_1^1| + |\pi_1^2|$. By Lemma 2, we can split π_0 into two derivations, $\pi_0^S = \bigotimes_{j \in S} \Delta_j \vdash V : \bigotimes_{j \in S} (\rho_j \multimap \alpha_j)$ and $\pi_0^{\bar{S}} = \bigotimes_{j \in \bar{S}} \Delta_j \vdash V : \bigotimes_{j \in S} (\rho_j \multimap \alpha_j)$, with $|\pi_0^S| + |\pi_0^{\bar{S}}| = |\pi_0|$. By rule $\multimap_E$, we have $\pi^1 = \bigotimes_{j \in S} \Delta_j \otimes \Gamma_1 \vdash VM : \parr_{j \in S} \alpha_j$ and $\pi^2 = \bigotimes_{j \in \bar{S}} \Delta_j \otimes \Gamma_2 \vdash VN : \parr_{j \in \bar{S}} \alpha_j$, where $|\pi^1| = |\pi_0^S| + |\pi_1^1| + 2\sharp S - 1$, and $|\pi^2| = |\pi_0^{\bar{S}}| + |\pi_1^2| + 2\sharp\bar{S} - 1$. By rule $\parallel_I$, $\pi' = \bigotimes_{j=1}^{n} \Delta_i \otimes \Gamma_1 \otimes \Gamma_2 \vdash VM \parallel VN : \parr_{j=1}^{n} \alpha_j$, where $|\pi'| = |\pi^1| + |\pi^2| = (|\pi_0^S| + |\pi_1^1| + 2\sharp S - 1) + (|\pi_0^{\bar{S}}| + |\pi_1^2| + 2\sharp\bar{S} - 1) = |\pi_0| + |\pi_1| + 2\sharp S + 2\sharp\bar{S} - 2 = |\pi_0| + |\pi_1| + 2n - 2 = |\pi| - 1$. □

4.2 Subject Expansion

The proof of the fact that our system enjoys subject expansion follows by straightforward induction, once one has proved the commutation of abstraction with abstraction, application, non-deterministic choice and parallel composition.

Theorem 3 (Subject expansion). *If $M \to N$ and $\pi = \Delta \vdash N : \alpha$, then there is $\pi' = \Delta \vdash M : \alpha$, such that $|\pi'| = |\pi| + 1$.*

Proof. By induction on the length of the derivation of $M \to N$, splitting into cases depending on its last rule. We only consider the most interesting case, i.e. $(\lambda x.M')V \to M'[V/x]$ where $M' = PQ$. One first needs to establish, by induction on π, a claim about the commutation of abstraction with application.

Claim. If $\pi = \Delta \vdash ((\lambda x.P)V)((\lambda x.Q)V) : \alpha$, where the last rule of π is a $\multimap_E$ rule having $k+1$ premises, then there exists $\pi' = \Delta \vdash (\lambda x.PQ)V : \alpha$ such that $|\pi'| = |\pi| - k$.

By definition we have $N = (PQ)[V/x] = P[V/x]Q[V/x]$. So, $\pi = \Delta \vdash N : \alpha$ ends in a $\multimap_E$-rule with $k+1$ premises $\pi_0 = \Delta' \vdash P[V/x] : \parr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\tau_{ij} \multimap \alpha_{ij})$ and $\pi_i = \Gamma_i \vdash Q[V/x] : \parr_{j=1}^{n_i} \tau_{ij}$ for $i = 1, \dots, k$, with $\Delta = \Delta' \otimes \bigotimes_{i=1}^{k} \Gamma_i$, $\alpha = \parr_{i=1}^{k} \parr_{j=1}^{n_i} \alpha_{ij}$ and $|\pi| = \sum_{i=0}^{k} \pi_i + (\sum_{i=1}^{k} 2n_i) - 1$. Then, by the induction hypothesis, we get $\pi'_0 = \Delta' \vdash (\lambda x.P)V : \parr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\tau_{ij} \multimap \alpha_{ij})$, and

$\pi'_i = \Gamma_i \vdash (\lambda x.Q)V : \invamp_{j=1}^{n_i} \tau_{ij}$, with $|\pi'_i| = |\pi_i| + 1$. Hence by rule $\multimap_E$ we obtain $\pi'' = \Delta' \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash ((\lambda x.P)V)((\lambda x.Q)V) : \invamp_{i=1}^{k} \invamp_{j=1}^{n_i} \alpha_{ij}$, with $|\pi''| = \sum_{i=0}^{k} |\pi'_i| + (\sum_{i=1}^{k} 2n_i) - 1$. By the above claim, we get $\pi' = \Delta' \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash (\lambda x.PQ)V : \invamp_{i=1}^{k} \invamp_{j=1}^{n_i} \alpha_{ij}$ such that $|\pi'| = |\pi''| - k = |\pi| + 1$. $\square$

4.3 Convergence

From our "quantitative" versions of subject reduction and subject expansion one easily obtains that our type system captures exactly the weakly normalising terms, and that the size $|\pi|$ of a derivation tree $\pi = \vdash M : \alpha$ decreases along the reduction of M. However, when α satisfies in addition a suitable minimality condition (namely the fact that α is of shape $\mathbf{1} \invamp \cdots \invamp \mathbf{1}$), then we can be more precise and say that there exists a reduction from M to a normal form, having length *exactly* $|\pi|$.

In the following $\invamp^k \mathbf{1}$, with $k > 0$, stands for $\mathbf{1} \invamp \cdots \invamp \mathbf{1}$ (k times).

Theorem 4. *Let M be a closed term, and $k > 0$. There is a typing tree π for $\vdash M : \invamp^k \mathbf{1}$ iff there are values $V_1, \ldots, V_k$ and a reduction $M \to^* V_1 \parallel \cdots \parallel V_k$ of length $|\pi|$.*

Proof. ($\Rightarrow$) Suppose $\pi = \vdash M : \invamp^k \mathbf{1}$. We proceed by induction on $|\pi|$. If $M = V_1 \parallel \cdots \parallel V_{k'}$, then π must start with a tree of $k' - 1$ rules $\parallel$, and then k' rules $\multimap_I$ with conclusion, respectively, $\vdash V_1 : \mathbf{1}, \ldots, \vdash V_{k'} : \mathbf{1}$. We then have $k = k'$, and M trivially converges to $V_1 \parallel \cdots \parallel V_{k'}$ in $|\pi| = 0$ steps.

Otherwise, since M is closed, there exists N such that $M \to N$. By Theorem 2, such an N can be chosen in such a way $\pi' = \vdash N : \invamp^k \mathbf{1}$, with $|\pi'| = |\pi| - 1$. From the induction hypothesis we know that N converges in $|\pi'|$ steps to $V_1 \parallel \cdots \parallel V_k$. Therefore, M converges in $|\pi'| + 1 = |\pi|$ steps to $V_1 \parallel \cdots \parallel V_k$.

($\Leftarrow$) Suppose that $M \to^* V_1 \parallel \cdots \parallel V_k$. By Remark 1, there is $\pi = \vdash V_1 \parallel \cdots \parallel V_k : \invamp^k \mathbf{1}$ and $|\pi| = 0$. Therefore, by the subject expansion (Theorem 3) there is $\pi' = \vdash M : \invamp^k \mathbf{1}$ and $|\pi'|$ is equal to the length of the reduction $M \to^* V_1 \parallel \cdots \parallel V_k$. $\square$

Corollary 1. *Let M be closed, then M is typable if and only if M converges.*

5 Adequacy and (Lack of) Full Abstraction

The choice of presenting a model through a type discipline or a reflexive object is more a matter of taste rather than a technical decision. (Compare for instance the type system of [20] and the interpretation of [9]). The model $\mathcal{V}$ associated with our type system lives in the category **Rel** of sets and relations (refer to [14] for more details) and is defined by $\mathcal{V} = \bigcup_{n \in \mathbb{N}} \mathcal{V}_n$, with

$$\mathcal{V}_0 = \emptyset, \qquad \mathcal{V}_{n+1} = \mathcal{M}_{\mathrm{f}}(\mathcal{V}_n) \times \mathcal{M}_{\mathrm{f}}(\mathcal{M}_{\mathrm{f}}(\mathcal{V}_n)),$$

where $\mathcal{M}_{\mathrm{f}}(X)$ denotes the set of finite multisets over a set X. In fact, $\mathcal{M}_{\mathrm{f}}(X)$ interprets in **Rel** the exponentials $!X$ and $?X$, whilst the cartesian product is the linear implication $\multimap$, so that $\mathcal{V}$ is the minimal solution of the equation $\mathcal{V} \simeq\ !\mathcal{V} \multimap ?!\mathcal{V}$. Recalling Equation 1 in the introduction, this means that the object $\mathcal{V}$ represents "value types", while computational types $\mathbb{C}$ will be represented by elements of $\mathcal{C} = !\mathcal{V} = \mathcal{M}_{\mathrm{f}}(\mathcal{V})$ and parallel-types $\mathbb{T}$ as elements of $\mathcal{T} = ?\mathcal{C} = \mathcal{M}_{\mathrm{f}}(\mathcal{C})$. This intuition can be formalized by defining two injections $(\cdot)^\circ : \mathbb{T} \to \mathcal{T}$ and $(\cdot)^\bullet : \mathbb{C} \to \mathcal{C}$ by mutual induction, as follows: $\tau^\circ = [\tau^\bullet]$, $(\alpha \parr \beta)^\circ = \alpha^\circ \uplus \beta^\circ$, $\mathbf{1}^\bullet = []$, $(\tau \otimes \rho)^\bullet = \tau^\bullet \uplus \rho^\bullet$ and $(\tau \multimap \alpha)^\bullet = [(\tau^\bullet, \alpha^\circ)]$.

It is beyond the scope of the present paper to give the explicit inductive definition of the interpretation of terms. For our purpose it is enough to know that such an interpretation can be characterised (up to isomorphism) as follows.

Definition 6. *The* interpretation of a closed term M *is defined by* $[\![M]\!] = \{\alpha \mid\ \vdash M : \alpha\} \subseteq \mathbb{T}$.

The interpretations of terms are naturally ordered by set-theoretical inclusion; an interesting problem is to determine whether there is a relationship between this ordering and the following observational preorder on terms.

Definition 7 (Observational preorder). *Let $M, N \in \Lambda_{+\|}$ be closed. We set $M \sqsubseteq N$ iff for all closed terms $\vec{P}$, $M\vec{P}$ converges implies that $N\vec{P}$ converges.*

A model is called *adequate* if $[\![M]\!] \subseteq [\![N]\!]$ entails $M \sqsubseteq N$; it is called *fully abstract* if in addition the converse holds.

The adequacy of the model $\mathcal{V}$ follows easily from Theorem 4 and the monotonicity of the interpretation.

Corollary 2 (Adequacy). *For all M, N closed, if $[\![M]\!] \subseteq [\![N]\!]$ then $M \sqsubseteq N$.*

On the contrary, $\mathcal{V}$ is not fully abstract. This is due to the fact that the call-by-value λ-calculus admits the creation of an 'ogre' that is able to 'eat' any finite sequence of arguments and converge, constituting then a top of the call-by-value observational preorder. Following [7], we define the ogre as $\mathrm{Y}^\star = \mathbf{\Delta}^\star\mathbf{\Delta}^\star$ where $\mathbf{\Delta}^\star = \lambda xy.xx$. The ogre $\mathrm{Y}^\star$ converges since $\mathrm{Y}^\star \to \lambda y.\mathrm{Y}^\star$ and remains convergent when applied to every sequence of values, by discarding them one at time.

Lemma 4. *For all closed terms M we have $M \sqsubseteq \mathrm{Y}^\star$.*

Proof. Given a term M and a sequence $\vec{P} = P_1 \cdots P_k$ of closed terms it is easy to check that $M\vec{P}$ can converge only when $\vec{P}$ converges. In that case we have $\mathrm{Y}^\star\vec{P} \to^* (\lambda y.\mathrm{Y}^\star)(V_1 \| \cdots \| V_n)P_2 \cdots P_k \to^* \mathrm{Y}^\star P_2 \cdots P_k \| \cdots \| \mathrm{Y}^\star P_2 \cdots P_k \to^* \lambda y.\mathrm{Y}^\star \| \cdots \| \lambda y.\mathrm{Y}^\star$. Therefore $\mathrm{Y}^\star$ is maximal with respect to $\sqsubseteq$. $\square$

It is easy to check that $\mathbf{1}$ and $(\mathbf{1} \multimap \mathbf{1}) \otimes (\mathbf{1} \multimap (\mathbf{1} \multimap \mathbf{1}))$ are valid types for $\mathrm{Y}^\star$, and thus belong to its interpretation. The following lemma gives a precise characterisation of $[\![\mathrm{Y}^\star]\!]$.

Lemma 5. $\alpha \in [\![\mathrm{Y}^\star]\!]$ *iff* $\alpha = \bigotimes_{i=0}^{n}(\mathbf{1} \multimap \alpha_i)$ *with* $n \geq 0$ *and* $\alpha_i \in [\![\mathrm{Y}^\star]\!]$ *for all* $i \leq n$. *In particular, we have that* $[\![\mathbf{I}]\!] \not\subseteq [\![\mathrm{Y}^\star]\!]$.

Proof. The crucial point is to remark that $\mathrm{Y}^\star \to \lambda y.\mathrm{Y}^\star$, so by Theorem 2 and 3, we get $[\![\mathrm{Y}^\star]\!] = [\![\lambda y.\mathrm{Y}^\star]\!]$. Therefore we have the following chain of equivalences:

$$\begin{aligned}
&\alpha \in [\![\mathrm{Y}^\star]\!] \text{ iff } \alpha \in [\![\lambda y.\mathrm{Y}^\star]\!] \\
&\quad\text{iff } \alpha = \otimes_{i=0}^{n}(\tau_i \multimap \alpha_i) \in [\![\lambda y.\mathrm{Y}^\star]\!] && \text{by Lemma 1, } n \geq 0 \\
&\quad\text{iff } \alpha = \otimes_{i=0}^{n}(\tau_i \multimap \alpha_i) \text{ and } \forall i, \tau_i \multimap \alpha_i \in [\![\lambda y.\mathrm{Y}^\star]\!] && \text{by Lemma 2} \\
&\quad\text{iff } \alpha = \otimes_{i=0}^{n}(\tau_i \multimap \alpha_i) \text{ and } \forall i, \tau_i = \mathbf{1} \text{ and } \alpha_i \in [\![\mathrm{Y}^\star]\!] && \text{since } y \notin \mathrm{FV}(\mathrm{Y}^\star).
\end{aligned}$$

We have that $[\![\mathbf{I}]\!] \not\subseteq [\![\mathrm{Y}^\star]\!]$ as, for instance, $(\mathbf{1} \multimap \mathbf{1}) \multimap (\mathbf{1} \multimap \mathbf{1}) \in [\![\mathbf{I}]\!] \setminus [\![\mathrm{Y}^\star]\!]$. □

Summing up, get that $\mathbf{I} \sqsubseteq \mathrm{Y}^\star$, while $[\![\mathbf{I}]\!] \not\subseteq [\![\mathrm{Y}^\star]\!]$.

6 Conclusion and Future Work

We introduced a call-by-value non-deterministic λ-calculus with a type system ensuring convergence. We proved that such a type system gives a bound on the length of the lazy call-by-value reduction sequences, which is the exact length when the typing is minimal. Finally, we show that the relational model $\mathcal{V}$ capturing our type system is adequate, but not fully abstract.

As our counterexample to full abstraction contains no non-deterministic operators, it also holds for the standard call-by-value λ-calculus and the relational model described in [14]. This is a notable difference with the call-by-name case, where the relational model is proven to be fully abstract for the pure call-by-name λ-calculus [18], while other counterexamples (see [9,8]) break full abstraction in presence of may or must non-deterministic operators. An open problem is to find a relational model fully abstract for the call-by-value λ-calculus.

Various fully abstract models of may and must non-determinism are known in the setting of Scott domain based semantics and idempotent intersection types. In particular, for the call-by-value case we mention [7,13]. Comparing these models and type systems with the ones issued from the relational semantics is a research direction started in [14] with some notable results. It would be interesting to reach a better understanding of the role played by intersection idempotency in the question of full abstraction.

Another axis of research is to generalize our approach to study the convergence in (call-by-name and call-by-value) λ-calculi with richer algebraic structures than simply may/must non-deterministic operators, such as [23,4]. In these calculi the choice operator is enriched with a weight, i.e. sums of terms are of the form $\alpha.M + \beta.N$, where α, β are scalars from a given semiring, pondering the choice. We would like to design type systems characterizing convergence properties in these systems. First steps have been done in [2,3].

Acknowledgements. We wish to thank Thomas Ehrhard and Simona Ronchi Della Rocca for interesting discussions, and the anonymous reviewers for their careful reading.

References

1. Amadio, R., Curien, P.L.: Domains and Lambda-Calculi. Cambridge Tracts in Theoretical Computer Science, vol. 46. Cambridge University Press (1998)
2. Arrighi, P., Díaz-Caro, A.: A System F accounting for scalars. Logical Methods in Computer Science 8(1:11) (2012)
3. Arrighi, P., Díaz-Caro, A., Valiron, B.: A type system for the vectorial aspects of the linear-algebraic λ-calculus. In: DCM 2011. EPTCS, vol. 88, pp. 1–15 (2012)
4. Arrighi, P., Dowek, G.: Linear-Algebraic λ-Calculus: Higher-Order, Encodings, and Confluence. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 17–31. Springer, Heidelberg (2008)
5. Barendregt, H.: The lambda calculus: its syntax and semantics. North-Holland, Amsterdam (1984)
6. Bernadet, A., Lengrand, S.: Complexity of Strongly Normalising λ-Terms via Non-idempotent Intersection Types. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 88–107. Springer, Heidelberg (2011)
7. Boudol, G.: Lambda-calculi for (strict) parallel functions. Information and Computation 108(1), 51–127 (1994)
8. Breuvart, F.: On the discriminating power of tests in the resource λ-calculus (submitted), Draft available at http://hal.archives-ouvertes.fr/hal-00698609
9. Bucciarelli, A., Ehrhard, T., Manzonetto, G.: A relational semantics for parallelism and non-determinism in a functional setting. APAL 163(7), 918–934 (2012)
10. Coppo, M., Dezani-Ciancaglini, M.: A new type-assignment for λ-terms. Archiv für Math. Logik 19, 139–156 (1978)
11. de Carvalho, D.: Execution time of lambda-terms via denotational semantics and intersection types. INRIA Report RR-6638, http://hal.inria.fr/inria-00319822/PDF/RR-6638.pdf, to appear in Math. Struct. in Comp. Sci. (2008)
12. Dezani-Ciancaglini, M., de'Liguoro, U., Piperno, A.: Filter models for conjunctive-disjunctive lambda-calculi. Theor. Comp. Sci. 170(1-2), 83–128 (1996)
13. Dezani-Ciancaglini, M., de'Liguoro, U., Piperno, A.: A filter model for concurrent lambda-calculus. SIAM J. Comput. 27(5), 1376–1419 (1998)
14. Ehrhard, T.: Collapsing non-idempotent intersection types. In: CSL 2012. LIPIcs, vol. 16, pp. 259–273 (2012)
15. Girard, J.Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
16. Krivine, J.L.: Lambda-calcul: types et modèles. Études et recherches en informatique, Masson (1990)
17. Laurent, O.: Étude de la polarisation en logique. PhD thesis, Université de Aix-Marseille II, France (2002)
18. Manzonetto, G.: A General Class of Models of $\mathcal{H}^*$. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 574–586. Springer, Heidelberg (2009)
19. Maraist, J., Odersky, M., Turner, D.N., Wadler, P.: Call-by-name, call-by-value, call-by-need and the linear λ-calculus. Theor. Comp. Sci. 228(1-2), 175–210 (1999)
20. Pagani, M., Ronchi Della Rocca, S.: Linearity, non-determinism and solvability. Fundam. Inform. 103(1-4), 173–202 (2010)
21. Plotkin, G.D.: Call-by-name, call-by-value and the λ-calculus. Theor. Comp. Sci. 1(2), 125–159 (1975)
22. Sallé, P.: Une généralisation de la théorie de types en λ-calcul. RAIRO: Informatique Théorique 14(2), 143–167 (1980)
23. Vaux, L.: The algebraic lambda calculus. Math. Struct. in Comp. Sci. 19(5), 1029–1059 (2009)

The Wadge Hierarchy of Petri Nets ω-Languages

Jacques Duparc[1], Olivier Finkel[2], and Jean-Pierre Ressayre[2]

[1] Faculty of Business and Economics, University of Lausanne, CH-1015 - Lausanne
[2] Equipe de Logique Mathématique, Institut de Mathématiques de Jussieu, CNRS et Université Paris 7, U.F.R. de Mathématiques, case 7012, site Chevaleret 75205 Paris Cedex 13, France
jacques.duparc@unil.ch, {finkel,ressayre}@math.univ-paris-diderot.fr

Abstract. We describe the Wadge hierarchy of the ω-languages recognized by deterministic Petri nets. This is an extension of the celebrated Wagner hierarchy which turned out to be the Wadge hierarchy of the ω-regular languages. Petri nets are an improvement of automata. They may be defined as partially blind multi-counter automata. We show that the whole hierarchy has height ω^{ω^2}, and give a description of the restrictions of this hierarchy to every fixed number of partially blind counters.

1 Introduction

The languages of infinite words – also called ω-languages – that are accepted by finite automata were first studied by Büchi in order to prove the decidability of the monadic second order theory of one successor over the integers. Since then, the regular ω-languages have been intensively studied, mostly for applications to specification and verification of non-terminating systems. See [29, 40, 41] for many results and references. Following this trend, the acceptance of infinite words by other types of finite machines, such as pushdown automata, multi-counter automata, Petri nets, or even Turing machines, were later considered [4, 9, 20, 32, 40].

Since the set of infinite words over a finite alphabet becomes a topological space once equipped with the Cantor topology, a way to study the complexity of the languages of infinite words accepted by finite machines is to study their topological complexity. This consists in providing their precise localization inside the projective hierarchy, the Borel hierarchy, or even the Wadge hierarchy (a great refinement of the Borel hierarchy). This work was conducted through [9, 25, 33, 35, 37, 38, 39, 40, 41].

It is well known that every ω-language accepted by a deterministic Büchi automaton is a $\mathbf{\Pi}^0_2$-set, and that an ω-language accepted by a non-deterministic Büchi (or Muller) automaton is a $\mathbf{\Delta}^0_3$-set. The Borel hierarchy of regular ω-languages is then determined. Moreover, Landweber proved that one can effectively determine the Borel complexity of a regular ω-language accepted by a given Muller or Büchi automaton, see [24, 29, 40, 41]. Elaborating on this result, Klaus Wagner completely described the Wadge hierarchy of the ω-regular languages [44]. It is nowadays called the Wagner hierarchy, and its length is the ordinal ω^ω. Wagner gave an automaton-like characterization of this hierarchy, based on the notions of chain and superchain, together with an algorithm to compute the Wadge (Wagner) degree of any given ω-regular language. Later, Wilke and Yoo proved that the Wadge degree of an ω-regular language may be computed in polynomial time [45]. This hierarchy was thouroughly studied by Carton and Perrin in [2, 3], and by Victor Selivanov in [31, 34].

Since there are various classes of finite machines recognizing ω-languages, each of them yields a countable sub-hierarchy of the Wadge hierarchy. Since the 1980's it has been an endeavor to describe these sub-hierarchies. It started with the work of Klaus Wagner on the ω-regular languages – although Wagner was unaware at the time of the connections between the Wadge hierarchy and his own work. The Wadge hierarchy of deterministic context-free ω-languages was determined, together with its length: $\omega^{(\omega^2)}$ [6, 7]. The problem whether this hierarchy is decidable remains open. The Wadge hierarchy induced by the subclass of deterministic one blind counter automata was determined in an effective way [11], and other partial decidability results were obtained [12]. It was then proved that the Wadge hierarchy of context-free ω-languages is the same as the one of effective analytic sets[1] [15, 20]. Intriguingly, the only Wadge class for which one can decide whether a given context-free ω-language belongs to or not, is the rudimentary singleton $\{\emptyset\}$, see [12, 13, 14]. In particular, one cannot decide whether a non-deterministic pushdown automaton is universal or not. This latter decision problem is actually Π^1_2-complete, hence located at the second level of the analytical hierarchy and "highly undecidable", [18]. Moreover the second author proved that the topological complexity of some context-free ω-languages may be subject to change from one model of set theory to another [17]. (Similar results hold for ω-languages accepted by 2-tape Büchi automata [16, 17].) Finally, the Wadge hierarchy of ω-languages of deterministic Turing machines was determined by Victor Selivanov, [32].

Petri nets are among the many accepting devices that are more powerful than finite automata in that they recognize more ω-languages that finite automata. They apply to the description of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent transitions and places. The distributions of tokens over the places define the configurations of the net. Petri nets work as an improvement of automata, since they may be defined as *partially blind multicounter automata* [21]. Petri nets have been extensively examined, particularly in concurrency theory (see for instance [10, 30]). The infinite behavior of Petri nets was first studied by Valk [42], and the one of deterministic Petri nets, by Carstensen [1].

In this paper, we first consider deterministic blind multicounter automata (corresponding to deterministic Petri nets) and the ω-languages that they accept when they are equipped with a Muller acceptance condition. This forms the class of deterministic Petri net ω-languages denoted $\mathcal{L}^3_{\omega dt}$ in [1].

We describe the Wadge hierarchy of the ω-languages recognized by deterministic Petri nets. This is an extension of the celebrated Wagner hierarchy of

[1] The class of all effective analytic sets (denoted Σ^1_1) is the class of all the ω-languages recognized by (non-deterministic) Turing machines.

the ω-regular languages. We show that the whole hierarchy has height ω^{ω^2}, and give a description of the restrictions of this hierarchy to some fixed number of partially blind counters.

2 Recalls on ω-Languages, Automata and Petri Nets

We assume the reader to be familiar with the theories of formal languages and ω-regular languages (see [22, 29, 41]).

Through along the paper, we assume Σ to be any finite set, called the alphabet. A finite word (string) over Σ is any sequence of the form $u = a_1 \ldots a_k$, where $k \in \mathbb{N}$ and $a_i \in \Sigma$ holds for each $i \leq k$. Notice that when $k = 0$, u is the empty word denoted by ε. We denote by $|u|$ the length of the word u (here $|u| = k$). We write $u(i) = a_i$ and $u[i] = u(1) \ldots u(i)$ for $i \leq k$ and $u[0] = \varepsilon$. The set of all finite words over Σ is denoted Σ^*.

An infinite word over Σ is some sequence of the form $x = a_1 a_2 \ldots a_n \ldots$ where $a_i \in \Sigma$ holds for all non-zero integers i. These infinite words are called ω-words for their length corresponds to ω: the first infinite ordinal. An infinite word x over Σ can be viewed as a mapping $x : \mathbb{N} \longrightarrow \Sigma$, so we write $x = x(1)x(2)\ldots$ and $x[n] = x(1)x(2)\ldots x(n)$ for its prefix of length n[2]. We write Σ^ω for the set of all ω-words over the alphabet Σ, so that an ω-language over the alphabet Σ is nothing but a subset of Σ^ω.

As usual, the concatenation of two finite words u and v is denoted uv. It naturally extends to the concatenation of a finite word u and an ω-word x to give the ω-words $y = ux$ defined by: $y(k) = u(k)$ if $k \leq |u|$, and $y(k) = x(k-|u|)$ if $k > |u|$. Given any finite word u, and any finite or infinite word x, u is a prefix of x (denoted $u \sqsubseteq x$) if $u(i) = x(i)$ holds for every non-zero integer $i \leq |u|$. Finally, for $V \subseteq \Sigma^*$, $V^\omega = \{\sigma = u_1 \ldots u_n \ldots \in \Sigma^\omega \mid u_i \in V, \forall i \geq 1\}$.

A *finite state machine (FSM)* is a quadruple $M = (Q, \Sigma, \delta, q_0)$, where Q is a finite set of states, Σ is a finite input alphabet, $q_0 \in Q$ is the initial state and δ is a mapping from $Q \times \Sigma$ into 2^Q . It is *deterministic (DFSM)* if $\delta : Q \times \Sigma \longrightarrow Q$.

Given an infinite word x, the infinite sequence of states $\rho = q_1 q_2 q_3 \ldots$ is called an (infinite) run of M on x starting in state p, if both $q_1 = p$ and $q_{i+1} \in \delta(q_i, a_i)$ $(\forall i \geq 1)$ hold. In case p is the initial state of M $(p = q_0)$, then ρ is simply called an infinite run of M on x.

We denote by $In(\rho) = \{q \in Q \mid \forall m \, \exists n > m \; q_n = q\}$ the set of states that appear infinitely often in ρ.

Equipped with an acceptance condition F, a finite state machine becomes a finite state automaton $M = (Q, \Sigma, \delta, q_0, F)$. It is a *Büchi automaton (BA)* when $F \subseteq Q$, and a *Muller automaton (MA)* when $F \subseteq 2^Q$. A Büchi automaton (respectively a Muller automaton) accepts x if for some infinite run of M on x, $In(\rho) \cap F$ is not empty (respectively $In(\rho) \in F$ holds). The ω-language accepted by an automaton is the set of all the infinite words it accepts. The classical result of R. Mc Naughton [28] establishes that non-deterministic Büchi automata, and

[2] Note that the enumeration $x = x(1)x(2)\ldots$ does not start at 0 so that we recover the empty word as $x[0]$.

both deterministic and non-deterministic Muller automata recognize the exact same ω-languages known as the ω-regular languages[3].

A partially blind multicounter automaton is a finite automaton equipped with a finite number (k) of partially blind counters. The content of any such counter is a non-negative integer. A counter is said to be partially blind when the multicounter automaton cannot test whether the content of the counter is zero. This means that if a transition of the machine is enabled when the content of a counter is zero then the same transition is also enabled when the content of the same counter is a non-zero integer. In order to get a partially blind multicounter automaton – simply called a *blind multicounter automaton* – which accepts the same language as a given Petri net, one can distinguish between the places of a Petri net by dividing them into the bounded ones (the number of tokens in such a place at any time is uniformly bounded) and the unbounded ones. Then each unbounded place may be seen as a blind counter, and the tokens in the bounded places determine the state of the blind multicounter automaton. The transitions of the Petri net may then be seen as the finite control of the blind multicounter automaton and the labels of these transitions are then the input symbols.

Contrary to what happens with non-deterministic Petri nets, allowing ε-transitions does not increase the expressive power of deterministic Petri nets which read ω-words [1]. For this reason, we restrict ourselves to the sole real time – *i.e.*, ε-transition free – blind multicounter automata. Also, without loss of generality we may assume that every transition, for every counter, either increases or decreases its content by 1 or leaves it untouched.

Definition 1. *For k any non-zero integer, A (real time) deterministic k-blind-counter machine (k-BCM) is of the form $\mathcal{M} = (Q, \Sigma, \delta, q_0)$ where Q is a finite set of states, Σ is a finite input alphabet, $q_0 \in Q$ is the initial state, and the transition relation δ is a partial mapping from $Q \times \Sigma \times \{0,1\}^k$ into $Q \times \{0,1,-1\}^k$.*

If the machine $\mathcal{M}$ is in state q, and for each i, $c_i \in \mathbb{N}$ is the content of the counter $\mathcal{C}_i$, then the configuration (or global state) of $\mathcal{M}$ is the $(k+1)$-tuple $(q, c_1, \ldots, c_k)$.

Given any $a \in \Sigma$, $q, q' \in Q$, and $(c_1, \ldots, c_k) \in \mathbb{N}^k$, if both $\delta(q, a, i_1, \ldots, i_k) = (q', j_1, \ldots, j_k)$, and $j_l \in E = \{l \in \{1, \ldots, k\} \mid c_l = 0\} \Rightarrow j_l \in \{0,1\}$ hold, then we write $a : (q, c_1, \ldots, c_k) \mapsto_{\mathcal{M}} (q', c_1 + j_1, \ldots, c_k + j_k)$. Thus the transition relation must verify: if $\delta(q, a, i_1, \ldots, i_k) = (q', j_1, \ldots, j_k)$, and $i_m = 0$ holds for some $m \in \{1, \ldots, k\}$, then we must have $j_m = 0$ or $j_m = 1$ (but $j_m = -1$ is prohibited). Moreover the k counters of $\mathcal{M}$ are blind, i.e., *if $\delta(q, a, i_1, \ldots, i_k) = (q', j_1, \ldots, j_k)$ holds, and $i_m = 0$ for $m \in E \subseteq \{1, \ldots, k\}$, then $\delta(q, a, i'_1, \ldots, i'_k) = (q', j_1, \ldots, j_k)$ holds also whenever $i_m = i'_m$ for $m \notin E$, and $i'_m = 0$ or $i'_m = 1$ for $m \in E$.*

For any finite word $u = a_1 a_2 \ldots a_n$ over Σ, a sequence of configurations $\rho = (q_i, c^i_1, \ldots c^i_k)_{1 \leq i \leq n+1}$ is a run of $\mathcal{M}$ on u, starting in configuration $(p, c_1, \ldots, c_k)$ iff $(q_1, c^1_1, \ldots, c^1_k) = (p, c_1, \ldots, c_k)$, and $a_i : (q_i, c^i_1, \ldots c^i_k) \mapsto_{\mathcal{M}}$

[3] The class of all the ω-regular languages is also characterized as the "ω-Kleene closure" of the class *REG* of all the (finitary) regular languages. Where given any class of finitary languages $\mathcal{L}$, the ω-Kleene closure of $\mathcal{L}$ is the class of ω-languages $\{\bigcup_{1 \leq i \leq n} U_i.V_i^{\omega} \mid U_i, V_i \in \mathcal{L}\}$.

$(q_{i+1}, c_1^{i+1}, \ldots c_k^{i+1})$ *(all* $1 \leq i \leq n$*). This notion extends naturally to infinite words: for* $x = a_1a_2 \ldots a_n \ldots$ *any* ω*-word over* Σ*, an* ω*-sequence of configurations* $(q_i, c_1^i, \ldots c_k^i)_{i\geq 1}$ *is called a complete run of* $\mathcal{M}$ *on* x*, starting in configuration* $(p, c_1, \ldots, c_k)$ *iff* $(q_1, c_1^1, \ldots c_k^1) = (p, c_1, \ldots, c_k)$*, and* $a_i : (q_i, c_1^i, \ldots c_k^i) \mapsto_{\mathcal{M}} (q_{i+1}, c_1^{i+1}, \ldots c_k^{i+1})$ *(for all* $1 \leq i$*).*

A complete run ρ *of* $\mathcal{M}$ *on* x*, starting in configuration* $(q_0, 0, \ldots, 0)$*, is simply called "a run of* $\mathcal{M}$ *on* x*".*

Definition 2. *A Büchi (resp. Muller) deterministic k-blind-counter automaton is some k-BCM* $\mathcal{M}' = (Q, \Sigma, \delta, q_0)$*, equipped with an acceptance condition* F*:* $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$*. It is a Büchi (resp. Muller[4]) k-blind-counter automaton when* $F \subseteq Q$ *(resp.* $F \subseteq 2^Q$*), and it accepts* x *if the infinite run of* $\mathcal{M}'$ *on* x *verifies* $In(\rho) \cap F \neq \emptyset$ *(respectively* $In(\rho) \in F$*).*

We write $L(\mathcal{M})$ *for the* ω*-language accepted by* $\mathcal{M}$*, and* $\mathbf{BC(k)}$ *for the class of* ω*-languages accepted by Muller deterministic k-blind-counter automata.*

3 Borel and Wadge Hierarchies

We assume the reader to be familiar with basic notions of topology that may be found in [23, 25, 27], and of ordinals (in particular the operations of multiplication and exponentiation) that may be found in [36].

For any given finite alphabet X – that contains at least two letters – we consider X^ω as the topological space equipped with the Cantor topology[5]. The open sets of X^ω are those of the form WX^ω, for some $W \subseteq X^*$. The closed sets are the complements of the open sets. The class that contains both the open sets and the closed sets, and is closed under countable union and intersection is the class of Borel sets. It is nicely set up in a hierarchy but counting how many times these latter operations are needed.

This defines the Borel Hierarchy: $\mathbf{\Sigma}_1^0$ is the class of open sets , and $\mathbf{\Pi}_1^0$ is the class of closed sets. For any non-zero integer n, $\mathbf{\Sigma}_{n+1}^0$ is the class of countable unions of sets inside $\mathbf{\Pi}_n^0$, while $\mathbf{\Pi}_{n+1}^0$ is the class of countable intersections of sets inside $\mathbf{\Sigma}_n^0$. More generally, for any non-zero countable ordinal α, $\mathbf{\Sigma}_\alpha^0$ is the class of countable unions of sets in $\cup_{\gamma<\alpha}\mathbf{\Pi}_\gamma^0$, and $\mathbf{\Pi}_\alpha^0$ is the class of countable intersections of sets in $\cup_{\gamma<\alpha}\mathbf{\Sigma}_\gamma^0$.

The Borel rank of a subset A of X^ω is the least ordinal $\alpha \geq 1$ such that A belongs to $\mathbf{\Sigma}_\alpha^0 \cup \mathbf{\Pi}_\alpha^0$. By ways of continuous pre-image, the Borel hierarchy turns into the refined Wadge Hierarchy.

Definition 3 ($\leq_w, \equiv_w, <_w$). *We let* X, Y *be two finite alphabets, and* $A \subseteq X^\omega, B \subseteq Y^\omega$*,* A *is said Wadge reducible to* B *(denoted* $A \leq_W B$*) iff there exists some continuous function* $f : X^\omega \longrightarrow Y^\omega$ *that satisfies* $\forall x \in X^\omega$ $(x \in A \Leftrightarrow f(x) \in B)$*.*

[4] The Muller acceptance condition was denoted 3-acceptance in [24, 1], and $(inf, =)$ in [40].

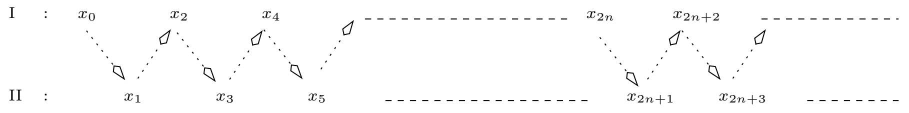
[5] The product topology of the discrete topology on X.

We write $A \equiv_w B$ for $A \leq_w B \leq_w A$, and $A <_w B$ for $A \leq_w B \not\leq_w A$. A set $A \subseteq X^\omega$ is *self dual* if $A \equiv_w X^\omega \smallsetminus A$ (denoted $A^\complement$) is verified. It is *non-self dual* otherwise [6].

It is easy to verify that the relation $\leq_w$ is both reflexive and transitive, and that $\equiv_w$ is an equivalence relation. Given any set A, the class of all its continuous pre-images forms a topological[7] class $\mathbf{\Gamma}$ called a Wadge class. A set is $\mathbf{\Gamma}$-*complete* if it both belongs to $\mathbf{\Gamma}$, and (Wadge) reduces every element in it[8]. It turns out that $\mathbf{\Sigma}^0_\alpha$ (resp. $\mathbf{\Pi}^0_\alpha$) is a Wadge class and any set in $\mathbf{\Sigma}^0_\alpha \smallsetminus \mathbf{\Pi}^0_\alpha$ (resp. $\mathbf{\Pi}^0_\alpha \smallsetminus \mathbf{\Sigma}^0_\alpha$) is $\mathbf{\Sigma}^0_\alpha$-complete (resp. $\mathbf{\Pi}^0_\alpha$-complete). Both $\mathbf{\Sigma}^0_n$-complete and $\mathbf{\Pi}^0_n$-complete sets (any $0 < n < \omega$) are examined in [38].

Wadge reducibility participates in game theory for continuous functions may be regarded as strategies for a player in a two-player game of perfect information and infinite length:

Definition 4. *Given any mapping $f : X^\omega \longrightarrow Y^\omega$, the game $\mathbf{G}(f)$ is the two-player game where players take turn picking letters in X for I and Y for II, player I starting the game, and player II being allowed in addition to pass her turn, while player I is not.*



After ω-many moves, player I and player II have respectively constructed $x \in X^\omega$ and $y \in Y^* \cup Y^\omega$. Player II wins the game if $y = f(x)$, otherwise player I wins.

So, in the game $\mathbf{G}(f)$, a strategy for player I is a mapping $\sigma : (Y \cup \{s\})^\star \longrightarrow X$, where s is a new letter not in Y that stands for II's moves when she passes her turn[9]. A strategy for player II is a mapping $f : X^+ \longrightarrow Y \cup \{s\}$. A strategy is called winning if it ensures a win whatever the opponent does.

This game was designed to characterize the continuous functions. Wadge found out that given $f : X^\omega \longrightarrow Y^\omega$, f is continuous $\iff$ II has a winning strategy in $\mathbf{G}(f)$. This is an easy exercise (see [23, 27]).

Definition 5. *For $A \subseteq X^\omega$ and $B \subseteq Y^\omega$, the Wadge game $\mathbf{W}(A, B)$ is the same as $\mathbf{G}(f)$, except that II wins iff $y \in Y^\omega$ and $(x \in A \iff y \in B)$ hold.*[10]

[6] Non-self dual sets are precisely those that verify $A \not\leq_w A^\complement$.

[7] A topological class is a class that is closed under continuous pre-images.

[8] It follows that two sets are complete for the same topological class iff they are Wadge equivalent.

[9] "s" stands for "skips".

[10] One sees immediately that a winning strategy for II in $\mathbf{W}(A, B)$ yields a continuous mapping $f : X^\omega \longrightarrow Y^\omega$ that guaranties that $A \leq_w B$ holds, whereas any continuous function f that witnesses the reduction relation $A \leq_w B$ gives rise to some winning strategy for II in $\mathbf{G}(f)$ which is also winning for II in $\mathbf{W}(A, B)$. This shows that for $A \subseteq X^\omega$ and $B \subseteq Y^\omega$, $A \leq_w B \iff$ *II has a winning strategy in* $\mathbf{W}(A, B)$.

In 1975, Martin proved Borel determinacy [23, 26], whose consequence is that for every Wadge game $\mathbf{W}(A, B)$, either player I or II has a winning strategy as long as both A and B are Borel. As immediate consequences, Wadge obtained that for any Borel $A, B \subseteq X^\omega$, there are no three $\leq_w$-incomparable Borel sets. Moreover, if $A \not\leq_w B$ and $B \not\leq_w A$, then $A \equiv_w B^{\complement}$. Later on, Martin and Monk proved that there is no sequence $(A_i)_{i\in\omega}$ of Borel subsets of X^ω such that $A_0 >_w A_1 >_w A_2 >_w \ldots A_n >_w A_{n+1} >_w \ldots$ holds [23, 43]. We recall that a set S is well ordered by the binary relation $<$ on S iff $<$ is a linear order on S such that there is no strictly infinite $<$-decreasing sequence of elements from S.

It follows that up to complementation and $\equiv_w$, the class of Borel subsets of X^ω, is well-ordered by $<_w$. Therefore, there is a unique ordinal $|WH|$ isomorphic to this well-ordering, together with a mapping d^0_W from the Borel subsets of X^ω onto $|WH|$, such that for all Borel subsets A, B: $d^0_W A < d^0_W B \Leftrightarrow A <_w B$, and $d^0_W A = d^0_W B \Leftrightarrow (A \equiv_w B$ or $A \equiv_w B^{\complement})$.

This well-ordering restricted to the Borel sets of finite ranks[11] has length the first ordinal that is a fixpoint of the operation $\alpha \longrightarrow {\omega_1}^\alpha$ [5, 43], where ω_1 is the first uncountable ordinal.

In order to study the Wadge hierarchy of the class $\mathbf{BC(k)}$ of ω-languages accepted by Muller deterministic k-blind-counter automata, we concentrate on the non-self dual sets as in [5], and slightly modify the definition of the Wadge degree. For $A \subseteq X^\omega$, such that $A >_w \emptyset$, we set $d_w(\emptyset) = d_w(\emptyset^{\complement}) = 1$, $d_w(A) = sup\{d_w(B) + 1 \mid B$ non-self dual and $B <_W A\}$.

Every ω-language which is accepted by a deterministic Petri net – more generally by a deterministic $\mathbf{X}$-automaton in the sense of [9] or by a deterministic Turing machine – is a boolean combination of $\mathbf{\Sigma^0_2}$-sets thus its Wadge degree inside the whole Wadge hierarchy of Borel sets is located below ω_1^ω. Moreover, every ordinal $0 < \alpha < \omega_1^\omega$ admits a unique Cantor normal form of base ω_1 [36], *i.e.*, it can be written as $\alpha = \omega_1^{n_j}.\delta_j + \omega_1^{n_{j-1}}.\delta_{j-1} + \cdots + \omega_1^{n_1}.\delta_1$ where $0 < j < \omega$, $0 \leq n_1 < \ldots < n_j < \omega$, and $\delta_j, \delta_{j-1}, \ldots, \delta_1$ are non-zero countable ordinals.

From Wagner's study, such an ordinal is the Wadge degree of an ω-regular language iff $\delta_j, \delta_{j-1}, \ldots, \delta_1$ are all integers. It is also known that such an ordinal is the Wadge degree of a deterministic context-free ω-language if and only if these multiplicative coefficients are all below ω^ω [6]. We add to this picture the following results that exhibits the Wadge hierarchy of $\mathbf{BC(k)}$:

1. for every non-null ordinal α whose Cantor normal form of base ω_1 is
$$\alpha = \omega_1^{n_j}.\delta_j + \omega_1^{n_{j-1}}.\delta_{j-1} + \cdots + \omega_1^{n_1}.\delta_1$$
where, for some integer $k \geq 1$, $\delta_1, \ldots, \delta_j$ are (non-null) ordinals $< \omega^{k+1}$, there exists some ω-language $L \in \mathbf{BC(k)}$ whose Wadge degree is α.
2. Non-self dual ω-languages in $\mathbf{BC(k)}$ have Wadge degrees of the above form.

Next section is dedicated to operations that will be needed in the proof.

[11] The Borel sets of finite ranks are those in $\bigcup_{n\in\mathbb{N}} \mathbf{\Sigma}^0_n = \bigcup_{n\in\mathbb{N}} \mathbf{\Pi}^0_n$.

4 Operations over Sets of ω-Words

4.1 The Sum

Definition 6. *For $\{X_+, X_-\}$ a partition in non-empty sets of $X_B \setminus X_A$ with $X_A \subseteq X_B$, $A \subseteq X_A^\omega$, and $B \subseteq X_B^\omega$, $B + A = A \cup X_A^* X_+ B \cup X_A^* X_- B^\complement$.*

A player in charge of $B + A$ in a Wadge game is like a player who begins the play in charge of A, and at any moment may also decide to start anew but being in charge this time of either B or of $B^\complement$ [12].

Proposition 7 (Wadge). *For non-self dual Borel sets A and B,*

$$d_w(B + A) = d_w(B) + d_w(A).$$

Notice that for any non-self dual Borel sets A, B, C, we have both $A+(B+C) \equiv_w (A + B) + C$, and $(B + A)^\complement \equiv_w B + A^\complement$. Although the class $\mathbf{BC}(\mathbf{k})$ is not closed under complementation, and $B + A$ was defined as $A \cup X_A^* X_+ B \cup X_A^* X_- B^\complement$, we may however use of the formulation $B + A \in \mathbf{BC}(\mathbf{k})$ for $A, B \in \mathbf{BC}(\mathbf{k})$ if some $C \in \mathbf{BC}(\mathbf{k})$ verifies $C \equiv_w B^\complement$.

4.2 The Countable Multiplication

We first need to define the supremum of a countable family of sets.

Definition 8. *For any bijection $f : \mathbb{N} \longrightarrow I$, any family $(A_i)_{i \in I}$ of non-self dual Borel subsets of X^ω, we fix some letter $e \in X$ to define*

$$\sup_{i \in I} A_i = \bigcup_{n \in \mathbb{N}} (X \setminus \{e\})^n e A_{f(n)}.$$

Proposition 9. *(See [5, 6].) For $(A_i)_{i \in I}$ any countable family of non-self dual Borel subsets of X^ω such that $\forall i \in I \quad \exists j \in I \quad A_i <_w A_j$, then*

1. *$\sup_{i \in I} A_i$ is a non-self dual Borel subset of X^ω, and*
2. *$d_w(\sup_{i \in I} A_i) = \sup\{d_w(A_i) \mid i \in I\}$.*

By combining sum and supremum, we get multiplication by countable ordinals.

Definition 10. *For $A \subseteq X^\omega$, and $0 < \alpha < \omega_1$, $A \bullet \alpha$ is inductively defined by $A \bullet 1 = A$, $A \bullet (\nu + 1) = (A \bullet \nu) + A$, and $A \bullet \beta = \sup_{\delta \in \beta} A \bullet \delta$, for β limit.*

[12] The first letter in $X_B \setminus X_A$ that is played decides the choice of B or $B^\complement$. Notice that given any finite alphabets X, Y which contain at least two letters, and any $B \subseteq X^\omega$, there exists $B' \subseteq Y^\omega$ such that $B \equiv_w B'$. Moreover, if for some integer $k \geq 0$ we have $B \in \mathbf{BC}(\mathbf{k})$, then B' can be taken in $\mathbf{BC}(\mathbf{k})$. So that we may write $B + A$ whatever space B is a subset of, simply meaning $B' + A$ where B' is any set that satisfies both $B' \equiv_w B$ and $B' \subseteq X^\omega$ for some X that contains the alphabet from which A is taken from, and strictly extends it with at least two new letters.

By Propositions 7 and 9, this operation verifies the following.

Proposition 11. *Let $A \subseteq X^\omega$ be some non-self dual Borel set, and $0 < \alpha < \omega_1$,*

$$d_w(A \bullet \alpha) = d_w(A) \cdot \alpha.$$

For a player in charge of $A \bullet \alpha$ in a Wadge game, everything goes as if (s)he could switch again and again between being in charge of A or $A^\complement$ – starting anew every time (s)he does so – but restrained from doing so infinitely often by having to construct a decreasing sequence of ordinals $< \alpha$ on the side every time (s)he switches.

4.3 The Multiplication by ω_1

Definition 12. *For $A \subseteq X^\omega$, and $a, b \notin X$ two different letters, $Y = X \cup \{a, b\}$, $A \bullet \omega_1 \subseteq (X \cup \{a, b\})^\omega$ is defined*[13] *by $A \bullet \omega_1 = A \ \cup \ Y^* aA \ \cup \ Y^* bA^\complement$.*

Inside a Wadge game, a player in charge of $A \bullet \omega_1$ may switch indefinitely between being in charge of A or its complement, deleting what (s)he has already played each time.

Proposition 13. *(See [5].) For any non-self dual Borel $A \subseteq X^\omega$, $A \bullet \omega_1$ is non-self dual Borel, and $d_w(A \bullet \omega_1) = d_w(A) \cdot \omega_1$.*

The following property will be very useful.

Proposition 14. *If $A \subseteq X^\omega$ is regular, then $A \bullet \omega_1$ is also regular.*

Proof. Immediate from the closure of the class REG_ω under finite union, complementation, and left concatenation by finitary regular languages [7]. □

4.4 Canonical Non-self Dual Sets

The empty set, considered as an ω-language over a finite alphabet is a Borel set of Wadge degree 1, *i.e.*, $d_w(\emptyset) = 1$. It is a non-self dual set and its complement has the same Wadge degree[14]. On the basis of the emptyset or its complement, the operations defined above provide non-self dual Borel sets for every Wadge degree $< \omega_1^\omega$. For notational purposes, given any $A \subseteq X^\omega$ we define $A \bullet \omega_1^n$ by induction on $n \in \mathbb{N}$ by: $A \bullet \omega_1^0 = A$, and $A \bullet \omega_1^{n+1} = (A \bullet \omega_1^n) \bullet \omega_1$.

Clearly, by Proposition 13, $d_w(A \bullet \omega_1^n) = d_w(A) \cdot \omega_1^n$ holds for every non-self dual Borel $A \subseteq X^\omega$. It follows that the ω-language $\emptyset \bullet \omega_1^n$ is a non-self dual Borel set whose Wadge degree is precisely ω_1^n.

Every non-null ordinal $\alpha < \omega_1^\omega$ admits a unique Cantor normal form of base ω_1:

$$\alpha = \omega_1^{n_j} \cdot \delta_j + \omega_1^{n_{j-1}} \cdot \delta_{j-1} + \cdots + \omega_1^{n_1} \cdot \delta_1.$$

where $\omega > j > 0$, $\omega > n_j > n_{j-1} > \ldots > n_1 \geq 0$, and $\delta_j, \delta_{j-1}, \ldots, \delta_1$ are non-zero countable ordinals [36].

As in [5, 6], we set $\Omega(\alpha) := (\emptyset \bullet \omega_1^{n_j}) \bullet \delta_j + (\emptyset \bullet \omega_1^{n_{j-1}}) \bullet \delta_{j-1} + \cdots + (\emptyset \bullet \omega_1^{n_1}) \bullet \delta_1$.

By Propositions 7, 11, and 13 $d_w(\Omega(\alpha)) = \alpha$ holds.

[13] This operation was denoted $A \longrightarrow A\hat{\ }\infty$ in [7], and $A \longrightarrow A^\natural$ in [6].

[14] *i.e.*, $d_w(\emptyset) = d_w(X^\omega) = 1$.

5 A Hierarchy of $BC(k)$

From now on, we restrain ourselves to the sole ordinals $\alpha < \omega_1^\omega$ whose Cantor normal form of base ω_1 contains only multiplicative coefficients strictly below ω^{k+1}, and we construct for every such α some Muller deterministic k-blind-counter automata $\mathcal{M}_\alpha$ and $\mathcal{M}_\alpha^-$ such that both $L(\mathcal{M}_\alpha) \equiv_w \Omega(\alpha)$ and $L(\mathcal{M}_\alpha^-) \equiv_w \Omega(\alpha)^{\complement}$ hold.

To start with, notice that for every integer n since $\emptyset \bullet \omega^n \in REG_\omega$ is verified, there exist deterministic Muller automata $\mathcal{O}_n = (Q_n, X_n, \delta_n, q_n^0, \mathcal{F}_n)$, where $\mathcal{F}_n \subseteq 2^{Q_n}$ is the collection of designated state sets, such that $L(\mathcal{O}_n) = \emptyset \bullet \omega^n$. We prove the following results:

Proposition 15. *For any ω-regular language A, any integer $j \geq 1$ there exist ω-languages $B, C \in \mathbf{BC(j)}$ such that $B \equiv_w (A \bullet \omega^j)$ and $C \equiv_w (A \bullet \omega^j)^{\complement}$.*

A careful generalization of the ideas of the proofs of Proposition 15 leads to:

Proposition 16. *For any ω-regular A, integer k, and ordinal $\omega^k \leq \alpha < \omega^{k+1}$, there exist $B, C \in \mathbf{BC(k)}$ such that both $B \equiv_w (A \bullet \alpha)$ and $C \equiv_w (A \bullet \alpha)^{\complement}$ hold.*

Theorem 17. *Let $\alpha < \omega_1^\omega$ be any ordinal of the form*

$$\alpha = \omega_1^{n_j} \cdot \delta_j + \omega_1^{n_{j-1}} \cdot \delta_{j-1} + \cdots + \omega_1^{n_0} \cdot \delta_0$$

where $\omega > j \geq 0$, $\omega > n_j > n_{j-1} > \ldots > n_0 \geq 0$, and $\omega^\omega > \delta_j, \delta_{j-1}, \ldots, \delta_0 > 0$.

Let k be the least integer such that $\forall i \leq j \quad \delta_i < \omega^{k+1}$. Then there exist ω-languages $B, C \in \mathbf{BC(k)}$ such that $B \equiv_w \Omega(\alpha)$ and $C \equiv_w \Omega(\alpha)^{\complement}$.

We recall that $\Omega(\alpha) := (\emptyset \bullet \omega_1^{n_j}) \bullet \delta_j + (\emptyset \bullet \omega_1^{n_{j-1}}) \bullet \delta_{j-1} + \cdots + (\emptyset \bullet \omega_1^{n_0}) \bullet \delta_0$.

6 Localisation of $BC(k)$

This section is dedicated to proving that there is no other Wadge class generated by some non-self dual ω-language in $BC(k)$ than the ones described in Theorem 17. Prior to this we need a technical result about the Wadge hierarchy together with a few others on ordinal combinatorics, and notations.

For some $A \subseteq X^\omega$ and $u \in X^*$, we write $u^{-1}A$ for the set $\{x \in X^\omega \mid ux \in A\}$. We say that A is *initializable* if player II has a w.s. in the Wadge game $\mathbf{W}(A, A)$ even though she is restricted to positions $u \in X^*$ that verify $u^{-1}A \equiv_w A$.

Lemma 18. *For $A \subseteq X^\omega$ any initializable set, $B \subseteq Y^\omega$, and δ, θ any countable ordinals,*

$$A \bullet (\theta + 1) \leq_w B \leq_w A \bullet \delta \implies \exists u \in Y^* \begin{cases} u^{-1}B \equiv_w A \bullet (\theta + 1) \\ \quad or \\ u^{-1}B \equiv_w (A \bullet (\theta + 1))^{\complement}. \end{cases}$$

Lemma 19. *We let $B \subseteq Y^\omega$, $A \subseteq X^\omega$ be any initializable set, and δ, θ be any countable ordinals. We consider any set of the form*

$$C = A \bullet \omega_1^n \bullet \nu_n + \cdots + A \bullet \omega_1^{n-1} \bullet \nu_{n-1} + \cdots + A \bullet \omega_1 \bullet \nu_1$$

for any non-zero integer n, and countable coefficients $\nu_n, \nu_{n-1}, \ldots, \nu_1$ with at least one of them being non-null.

$$C + A \bullet (\theta + 1) \leq_w B \leq_w C + A \bullet \delta \implies \exists u \in Y^* \begin{cases} u^{-1}B \equiv_w C + A \bullet (\theta + 1) \\ \text{or} \\ u^{-1}B \equiv_w (C + A \bullet (\theta + 1))^{\complement}. \end{cases}$$

We recall that for any set of ordinals $\mathcal{O}$, its order type – denoted $ot(\mathcal{O})$ – is the unique ordinal that is isomorphic to $\mathcal{O}$ ordered by membership.

Definition 20. *The function $\mathcal{H} : \omega^\omega \times \omega^\omega \longrightarrow On$ is defined by*

$$\mathcal{H}(\alpha, \beta) \ = \ \omega^k \cdot (l_k + m_k) + \omega^{k-1} \cdot (l_{k-1} + m_{k-1}) + \cdots + \omega^0 \cdot (l_0 + m_0).$$

Where (a variation of the) the Cantor normal form of base ω of α (resp. β) is $\alpha = \omega^k \cdot l_k + \omega^{k-1} \cdot l_{k-1} + \cdots + \omega^0 \cdot l_0$, $\beta = \omega^k \cdot m_k + \omega^{k-1} \cdot m_{k-1} + \cdots + \omega^0 \cdot m_0$, with $l_k, m_k, l_{k-1}, m_{k-1}, \ldots, l_0, m_0 \in \mathbb{N}$. (Some of these integers may be null[15].)

Lemma 21. *Let $\mathcal{H} : \omega^\omega \times \omega^\omega \longrightarrow On$, $0 < \alpha', \alpha, \beta' \beta < \omega^\omega$ with $\alpha' \leq \alpha$, $\beta' \leq \beta$ but either $\alpha' < \alpha$ or $\beta' < \beta$, then $\mathcal{H}(\alpha', \beta') < \mathcal{H}(\alpha, \beta)$.*

We make use of the mapping $\mathcal{H}$ to prove the following combinatorial result.

Lemma 22. *Let α, β, γ be non-null ordinals with $\alpha, \beta < \omega^\omega$, and $f : \gamma \longrightarrow \{0, 1\}$.* If both $\alpha = ot(f^{-1}[0])$ and $\beta = ot(f^{-1}[1])$ hold, then $\gamma \leq \mathcal{H}(\alpha, \beta)$.

Corollary 23. *Let k, n be non-null integers, γ be any ordinal, $0 \leq \alpha_0, \ldots, \alpha_k < \omega^n$, and $f : \gamma \longrightarrow \{0, \ldots, k\}$.* If $\forall i \leq k \ \ \alpha_i = ot(f^{-1}[i])$ holds, then $\gamma < \omega^{\mathrm{n}}$.

Lemma 24. *Let k be some non-null integer, $(\mathbb{N}^k, \lesssim)$ be a well-ordering such that for every k-tuples $(a_0, \ldots, a_{k-1}), (b_0, \ldots, b_{k-1}) \in \mathbb{N}^k$ the following holds:*

$$(a_0, \ldots, a_{k-1}) \lesssim (b_0, \ldots, b_{k-1}) \Longrightarrow \begin{cases} \forall i < k \quad a_i \leq b_i \\ \text{or} \\ \exists i, j < k \quad \text{such that } a_i < b_i \text{ and } a_j > b_j. \end{cases}$$

Then, the order type of $(\mathbb{N}^k, \lesssim)$ is at most ω^k.

Lemma 25. *We let k be any non-null integer, $B \in \mathbf{BC(k)}$, $A \subseteq X^\omega$ be any initializable set, and δ any countable ordinal.*

$$B \leq_w A \bullet \delta \implies B \leq_w A \bullet \alpha \ \text{ for some } \alpha < \omega^{k+1}.$$

[15] In particular, $l_k, l_{k-1}, \ldots \ m_k, m_{k-1}, \ldots$ might be null, but since $\alpha, \beta > 0$ holds, at least one of the l_i's, and one of the m_i's are different from zero.

An immediate consequence is that $B \equiv_w A \bullet \delta$ holds only for ordinals $\delta < \omega^{k+1}$.

Proof. First notice that for every $B \subseteq X^\omega$, and every $u \in X^*$, if $B \in \mathbf{BC(k)}$ holds, then $u^{-1}B \in \mathbf{BC(k)}$ holds too.

Towards a contradiction, we assume that $A \bullet \alpha <_w B \leq_w A \bullet \delta$ holds for all $\alpha < \omega^{k+1}$. We let $\mathcal{B}$ be a k-blind counter automaton that recognizes B. By Lemma 18, for each successor ordinal $\alpha < \omega^{k+1}$ there exists some $u_\alpha \in X^*$ such that $u_\alpha^{-1}B \equiv_w A \bullet \alpha$ or $u_\alpha^{-1}B \equiv_w (A \bullet \alpha)^\complement$. For each such u_α, we form $(q_\alpha, c_{\alpha,0}, c_{\alpha,1}, \ldots, c_{\alpha,k-1})$ where q_α denotes the control state that $\mathcal{B}$ is in after having read u_α, and $c_{\alpha,i}$ the height of its counter number i (any $i < k$).

Now there exists necessarily some control state q such that the order type of the set $S = \{\alpha < \omega^{k+1} \mid \alpha \text{ successor and } q_\alpha = q\}$ is ω^{k+1}. By Lemma 24 there exist $\alpha, \alpha' \in S$ such that $\alpha' < \alpha$ holds together with $c_{\alpha,i} \leq c'_{\alpha,i}$ (any $i < k$). (Without loss of generality, we may even assume that $\omega \leq \alpha' < \alpha$ holds.) Let us denote $\mathcal{B}_{\alpha'}$ the k-blind counter automaton $\mathcal{B}$ that starts in state $(q_{\alpha'}, c_{\alpha',0}, c_{\alpha',1}, \ldots, c_{\alpha',k-1})$, and $\mathcal{B}_\alpha$ the one that starts in state $(q_\alpha, c_{\alpha,0}, c_{\alpha,1}, \ldots, c_{\alpha,k-1})$. Notice that since $c_{\alpha,i} \leq c'_{\alpha,i}$ holds for all $i < k$, $\mathcal{B}_{\alpha'}$ performs exactly the same as $\mathcal{B}_\alpha$ except when the latter crashes for it tries to decrease a counter that is already empty. But it is then not difficult to see that given the above assumption – that $\omega \leq \alpha' < \alpha$ holds – $u_\alpha^{-1}B \leq_w u_{\alpha'}^{-1}B$ holds which leads to either $A \bullet \alpha \leq_w A \bullet \alpha'$ or $(A \bullet \alpha)^\complement \leq_w A \bullet \alpha'$. In both cases, it contradicts $\alpha' < \alpha$. □

Notice that $\emptyset \bullet \omega_1^n$ being initializable, we have in particular the following result.

Lemma 26. *For k, n any integers, A any non-self dual ω-language in $\mathbf{BC(k)}$, and any non-zero countable ordinal α, A or $A^\complement \equiv_w (\emptyset \bullet \omega_1^n) \bullet \alpha \Longrightarrow \alpha < \omega^{k+1}$.*

In a similar way, we may now state the following lemma.

Lemma 27. *We let k be any non-null integer, $B \in \mathbf{BC(k)}$, $A \subseteq X^\omega$ be any initializable set, δ be any countable ordinal, and C be any set of the form*

$$C = A \bullet \omega_1^n \bullet \nu_n + \cdots + A \bullet \omega_1^{n-1} \bullet \nu_{n-1} + \cdots + A \bullet \omega_1 \bullet \nu_1$$

for any non-zero integer n, and countable multiplicative coefficients $\nu_n, \nu_{n-1}, \ldots, \nu_1$ with at least one of them being non-null. Then we have

$$B \leq_w C + A \bullet \delta \quad \Longrightarrow \quad B \leq_w C + A \bullet \alpha \text{ for some } \alpha < \omega^{k+1}.$$

Theorem 28. *Let k be any non-null integer, $B \subseteq X^\omega$ be non-self dual. If $B \in \mathbf{BC(k)}$, then either B or $B^\complement$ is Wadge equivalent to some*

$$\Omega(\alpha) = (\emptyset \bullet \omega_1^{n_j}) \bullet \delta_j + (\emptyset \bullet \omega_1^{n_{j-1}}) \bullet \delta_{j-1} + \cdots + (\emptyset \bullet \omega_1^{n_0}) \bullet \delta_0.$$

where $j \in \mathbb{N}$, $n_j > n_{j-1} > \ldots > n_0$ and $\omega^{k+1} > \delta_j, \delta_{j-1}, \ldots, \delta_0 > 0$.

Proof. This is an almost immediate consequence of Lemmas 25 and 27. □

This settles the case of the non-self dual ω-languages in $\mathbf{BC}(\mathbf{k})$. For the self-dual ones, it is enough to notice the easy following:

1. Given any $A \subseteq X^\omega$, if $A \in \mathbf{BC}(\mathbf{k})$ is self dual, then there exists two non-self dual sets $B, C \subseteq X^\omega$ such that both B and C belong to $\mathbf{BC}(\mathbf{k})$, $B \equiv_w C^{\complement}$, and $A \equiv_w X_0 B \cup X_1 C$, where $\{X_0, X_1\}$ is any partition of X in two non-empty sets.
2. If $A \subseteq X^\omega$ and $B \subseteq X^\omega$ are non-self dual, verify $A \equiv_w B^{\complement}$, and both belong to $\mathbf{BC}(\mathbf{k})$, then, given any partition of X in two non-empty sets $\{X_0, X_1\}$, $X_0 A \cup X_1 B$ is self-dual, and also belongs to $\mathbf{BC}(\mathbf{k})$.

If we set $d^\circ(A) = sup\{d^\circ(B) + 1 \mid B <_W A\}$(any $A \subseteq X^\omega$), then we obtain that there exists an ω-language $B \subseteq X^\omega$ recognized by some deterministic Petri net, such that $A \equiv_w B$ holds iff $d^\circ A$ is of the form $\alpha = \omega_1^n \cdot \delta_n + \cdots + \omega_1^0 \cdot \delta_0$ for some $n \in \mathbb{N}$, and $\omega^\omega > \delta_n, \ldots, \delta_0 \geq 0$. Finally, an easy computation provides $(\omega^\omega)^\omega = \omega^{\omega^2}$ as the height of the Wadge hierarchy of ω-languages recognized by deterministic Petri nets.

7 Conclusions

We provided a description of the extension of the Wagner hierarchy from automata to deterministic Petri Nets with Muller acceptance conditions. The results are rigorously the same if we replace Muller acceptance conditions with parity acceptance conditions. But with Büchi acceptance conditions instead, it becomes even simpler since the ω-languages are no more boolean combinations of $\mathbf{\Sigma}_2^0$-sets, but $\mathbf{\Pi}_2^0$-sets. So, the whole hierarchy comes down to the following:

Corollary 29. *For any $A \subseteq X^\omega$, there exists an ω-language $B \subseteq X^\omega$ recognized by some deterministic Petri net with* Büchi acceptance conditions*, such that $A \equiv_w B$ iff either $d^\circ A = \omega_1$, and A is $\mathbf{\Pi}_2^0$-complete, or $d^\circ A < \omega^\omega$.*

Deciding the degree of a given ω-language in $\mathbf{BC}(\mathbf{k})$, for $k \geq 2$, recognized by some deterministic Petri net – either with *Büchi* or *Muller* acceptance conditions, remains an open question. Notice that for $k = 1$ this decision problem has been shown to be decidable by the second author in [11].

Another rather interesting open direction of research is to go from deterministic to non-deterministic Petri nets. It is clear that this step forward brings new Wadge classes – for instance there exist ω-languages recognized by non-deterministic Petri nets with *Büchi acceptance conditions* that are not $\mathbf{\Delta}_3^0$ [19] – but the description of this whole hierarchy still requires more investigations.

References

[1] Carstensen, H.: Infinite Behaviour of Deterministic Petri Nets. In: Koubek, V., Janiga, L., Chytil, M.P. (eds.) MFCS 1988. LNCS, vol. 324, pp. 210–219. Springer, Heidelberg (1988)

[2] Carton, O., Perrin, D.: Chains and superchains for ω-rational sets, automata and semigroups. Internat. J. Algebra Comput. 7(7), 673–695 (1997)
[3] Carton, O., Perrin, D.: The Wagner hierarchy of ω-rational sets. Internat. J. Algebra Comput. 9(5), 597–620 (1999)
[4] Cohen, R., Gold, A.: ω-computations on Turing machines. Theoretical Computer Science 6, 1–23 (1978)
[5] Duparc, J.: Wadge hierarchy and Veblen hierarchy: Part 1: Borel sets of finite rank. Journal of Symbolic Logic 66(1), 56–86 (2001)
[6] Duparc, J.: A hierarchy of deterministic context free ω-languages. Theoretical Computer Science 290(3), 1253–1300 (2003)
[7] Duparc, J., Finkel, O., Ressayre, J.-P.: Computer science and the fine structure of Borel sets. Theoretical Computer Science 257(1-2), 85–105 (2001)
[8] Duparc, J., Finkel, O., Ressayre, J.-P.: The Wadge hierarchy of Petri nets omega-languages. Special Volume in Honor of Victor Selivanov at the Occasion of his Sixtieth Birthday, Pedagogical University of Novosibirsk (September 2012), http://hal.archives-ouvertes.fr/hal-00743510
[9] Engelfriet, J., Hoogeboom, H.J.: X-automata on ω-words. Theoretical Computer Science 110(1), 1–51 (1993)
[10] Esparza, J.: Decidability and Complexity of Petri Net Problems, an Introduction. In: Reisig, W., Rozenberg, G. (eds.) APN 1998. LNCS, vol. 1491, pp. 374–428. Springer, Heidelberg (1998)
[11] Finkel, O.: An Effective Extension of the Wagner Hierarchy to Blind Counter Automata. In: Fribourg, L. (ed.) CSL 2001. LNCS, vol. 2142, pp. 369–383. Springer, Heidelberg (2001)
[12] Finkel, O.: Topological properties of omega context free languages. Theoretical Computer Science 262(1-2), 669–697 (2001)
[13] Finkel, O.: Wadge hierarchy of omega context free languages. Theoretical Computer Science 269(1-2), 283–315 (2001)
[14] Finkel, O.: Borel hierarchy and omega context free languages. Theoretical Computer Science 290(3), 1385–1405 (2003)
[15] Finkel, O.: Borel ranks and Wadge degrees of context free omega languages. Mathematical Structures in Computer Science 16(5), 813–840 (2006)
[16] Finkel, O.: Wadge degrees of infinitary rational relations. Special Issue on Intensional Programming and Semantics in Honour of Bill Wadge on the Occasion of his 60th Cycle, Mathematics in Computer Science 2(1), 85–102 (2008)
[17] Finkel, O.: The complexity of infinite computations in models of set theory. Logical Methods in Computer Science 5(4:4), 1–19 (2009)
[18] Finkel, O.: Highly undecidable problems for infinite computations. Theoretical Informatics and Applications 43(2), 339–364 (2009)
[19] Finkel, O.: On the topological complexity of ω-languages of non-deterministic Petri nets, p. 1 (preprint, 2012)
[20] Finkel, O.: Topological complexity of context free ω-languages: A survey. In: Language, Culture, Computation: Studies in Honor of Yaacov Choueka. LNCS. Springer (to appear, 2013)
[21] Greibach, S.: Remarks on blind and partially blind one way multicounter machines. Theoretical Computer Science 7, 311–324 (1978)
[22] Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley Publishing Co. (2001)
[23] Kechris, A.S.: Classical descriptive set theory. Springer, New York (1995)
[24] Landweber, L.: Decision problems for ω-automata. Mathematical Systems Theory 3(4), 376–384 (1969)

[25] Lescow, H., Thomas, W.: Logical Specifications of Infinite Computations. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1993. LNCS, vol. 803, pp. 583–621. Springer, Heidelberg (1994)

[26] Martin, D.A.: Borel determinacy. Ann. Math. 102(2), 363–371 (1975)

[27] Moschovakis, Y.N.: Descriptive set theory, vol. 155. Am. Math. Soc. (2009)

[28] Naughton, R.M.: Testing and generating infinite sequences by a finite automaton. Information and Control 9, 521–530 (1966)

[29] Perrin, D., Pin, J.-E.: Infinite words, automata, semigroups, logic and games. Pure and Applied Mathematics, vol. 141. Elsevier (2004)

[30] Desel, J., Reisig, W., Rozenberg, G. (eds.): ACPN 2003. LNCS, vol. 3098. Springer, Heidelberg (2004)

[31] Selivanov, V.: Fine hierarchy of regular ω-languages. Theoretical Computer Science 191, 37–59 (1998)

[32] Selivanov, V.: Wadge degrees of ω-languages of deterministic Turing machines. RAIRO-Theoretical Informatics and Applications 37(1), 67–83 (2003)

[33] Selivanov, V.: Fine hierarchies and m-reducibilities in theoretical computer science. Theoretical Computer Science 405(1-2), 116–163 (2008)

[34] Selivanov, V.: Fine hierarchy of regular aperiodic omega-languages. International Journal of Foundations of Computer Science 19(3), 649–675 (2008)

[35] Selivanov, V.: Wadge reducibility and infinite computations. Special Issue on Intensional Programming and Semantics in Honour of Bill Wadge on the Occasion of his 60th Cycle, Mathematics in Computer Science 2(1), 5–36 (2008)

[36] Sierpiǹski, W.: Cardinal and ordinal numbers. PWN, Warszawa (1965)

[37] Simonnet, P.: Automates et théorie descriptive. PhD thesis, Univ. Paris VII (1992)

[38] Staiger, L.: Hierarchies of recursive ω-languages. Elektronische Informationsverarbeitung und Kybernetik 22(5-6), 219–241 (1986)

[39] Staiger, L.: Research in the theory of ω-languages. Journal of Information Processing and Cybernetics 23(8-9), 415–439 (1987); Mathematical aspects of informatics

[40] Staiger, L.: ω-languages. In: Handbook of Formal Languages, vol. 3, pp. 339–387. Springer, Berlin (1997)

[41] Thomas, W.: Automata on infinite objects. In: Handbook of Theoretical Computer Science. Formal models and semantics, vol. B, pp. 135–191. Elsevier (1990)

[42] Valk, R.: Infinite behaviour of Petri nets. Theor. Comp. Sc. 25(3), 311–341 (1983)

[43] Wadge, W.: Reducibility and determinateness in the Baire space. PhD thesis, University of California, Berkeley (1983)

[44] Wagner, K.: On ω-regular sets. Information and Control 43(2), 123–177 (1979)

[45] Wilke, T., Yoo, H.: Computing the Wadge degree, the Lifschitz Degree, and the Rabin Index of a Regular Language of Infinite Words in Polynomial Time. In: Mosses, P.D., Nielsen, M. (eds.) CAAP 1995, FASE 1995, and TAPSOFT 1995. LNCS, vol. 915, pp. 288–302. Springer, Heidelberg (1995)

Iterated Contraction Based on Indistinguishability

Konstantinos Georgatos[1,2]

[1] Department of Mathematics and Computer Science
John Jay College
City University of New York
524 West 59th Street
New York, New York 10019

[2] Doctoral Program in Computer Science
Graduate Center
City University of New York
365 Fifth Avenue
New York, New York 10016

kgeorgatos@jjay.cuny.edu

Abstract. We introduce a class of set-theoretic operators on a tolerance space that models the process of minimal belief contraction, and therefore a natural process of iterated contraction can be defined. We characterize the class of contraction operators and study the properties of the associated iterated belief contraction.

1 Introduction

Vagueness can be generated in several ways, but among them, distinguishability — i.e., our observational power of telling whether two objects are distinct — is the most cited. Its complement, the relation of indistinguishability has long been used in logic to model a variety of systems. Most notably, the accessibility relation of Kripke models is often interpreted as an indistinguishability. Indistinguishability is often assumed to be an equivalence, like the accessibility relation of the modal logic S5. However, a more interesting view arises when we drop transitivity and assume that indistinguishability is only reflexive and symmetric, that is, a tolerance [1,2].

When indistinguishability is an equivalence, then the objects form a partition. When indistinguishability is a tolerance, then a more refined situation appears where the objects form a graph. In particular, the graph comes equipped with a natural notion of distance based on the shortest path called geodesic. We have argued that we can use the geodesic distance to measure similarity. Our idea ([3]) rests on the following maxim: two objects are similar when there is a context within which they are indistinguishable. Therefore, similarity can be *measured* with degrees of indistinguishability.

For example, when two houses appear indistinguishable from a certain distance x, then it is safe to say that those houses are similar. For, if we get closer, an inspection might reveal differences but their similarity will persist. Thus, indistinguishability at distance x implies similarity. The smaller the distance x, the more similar the objects are.

Now consider the following problem: assume that our view of the world — that is, the set of states we consider possible — is A and we are asked to add to A at least one element from a subset B. In other words, we are asked to change our view to allow for the possibility that B holds. It seems natural to pick the elements of B that are most similar to those of A. In our framework, similarity is measured by the geodesic distance; that is, it is measured by degrees of indistinguishability, and therefore we would pick those elements of B that are closest to A. Making B possible is equivalent to *contract* its complement B^c so this process will be called contraction.

Contraction by minimization over a geodesic is straightforward. We illustrate the process with the following example (edges represent the reflexive symmetric tolerance relation).

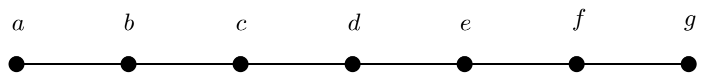


Fig. 1.

In Figure 1, let $A = \{d\}$ and $B = \{a, b, f, g\}$ then $A \oplus B$ the contraction of A with B equals the subset $\{d, b, f\}$. This is because the distance of d from b and f is 2 while the distance of d from a and g is 3. So b and f are closest to A and we choose to augment A with both of them.

Let $C = \{a, f, g\}$. Then $A \oplus C = \{d, f\}$.

Now, let $D = \{a, f\}$ and $G = \{b, g\}$. We have $A \oplus D = \{d, f\}$ and $A \oplus G = \{d, b\}$. The global character of the geodesic metric allows us to iterate the contraction operator. So, we have $(A \oplus D) \oplus G = \{d, f, g\}$ and $(A \oplus G) \oplus D = \{d, b, a\}$. Therefore this example is also a counterexample to commutativity.

The main result of this paper is a characterization of the contraction operator $\oplus$ on subsets of the elements of the tolerance space. This result appears in Section 4 (Proposition 4). However the geodesic revision postulates can be translated to propositional language and we will conclude by discussing this translation while comparing it with earlier literature.

2 Tolerance Spaces and Their Geodesic

We will use a reflexive and symmetric relation to model indistinguishability. A set equipped with such a relation is frequently called a tolerance space. In addition, we will assume that the space is connected:

Definition 1. *Let X be a set and $R \subseteq X \times X$ a relation on X. Then (X, R) is called a* (connected) tolerance space *when R is reflexive, symmetric, and*

(X is) connected, i.e., for all $x, y \in X$ there is a non negative integer n such that $xR^n y$.

In the above definition, we assume $R^0 = 1_X$, $R^n = R^{n-1} \circ R$ for $n > 0$.

Given a tolerance space (X, R) we can define a metric called *geodesic* with a map d from $X \times X$ to Z^+ (the set of non-negative integers) where

$$d(x, y) = \min\{n \mid xR^n y\}.$$

Note that a geodesic metric is not any integer metric. The values of the geodesic metric are determined by adjacency. The results of this paper depend heavily on this property which can be described with: for all $x, y \in X$ such that $d(x, y) = n$ with $1 < n < \infty$ there is $z \in V$ with $z \neq x, y$ such that $d(x, y) = d(x, z) + d(z, y)$. In particular we can choose z so that $d(x, z) = 1$. Note here that a geodesic metric is a topological metric, that is, it satisfies identity, symmetry and triangle inequality.

The geodesic distance extends to distance between non-empty subsets with

$$d(A, B) = \min\{d(x, y) \mid x \in A, y \in B\}.$$

We shall also write $d(x, A)$ for $d(\{x\}, A)$. Similarly for $d(A, x)$. We will write A^c for the complement of A.

Lemma 1. *If A and A^c are non-empty, we have $d(A, A^c) = 1$.*

Proof. Suppose $d_G(A, A^c) = n$ with $n > 1$ (it cannot be 0). Then there exists a minimal path of length n between A and A^c: $x_1 \in A, x_2, \ldots, x_{n+1} \in A^c$ where $x_i R x_{i+1}$. However, either $x_2 \in A$ or $x_2 \in A^c$ and a shorter path arises in both cases, which is a contradiction.

3 Contraction Based on a Geodesic

Contraction with a means to augment our belief state with the *possibility* of a because removal of beliefs translates to the addition of possible states. Apart from this semantic view, such addition could be achieved syntactically in a language where possibility can be expressed (e.g. modal logic).

Definition 2. *Given a tolerance space (X, R) and $A, B \subseteq X$ with $A \neq \emptyset$ then the (induced) contraction of A with B is defined with*

$$A \oplus_R B = \begin{cases} A \cup \{y \in B \mid d(A, y) = d(A, B)\} & \textit{if } B \neq \emptyset \\ A & \text{otherwise.} \end{cases}$$

Contraction is just one of the interesting operators one can define on tolerance spaces. In [4,5], we studied conditioning and revision operators. The following lemma holds.

Lemma 2. *1. $d(A, x) = 1$ if and only if $x \in (A \oplus_R A^c) \cap A^c$*

Table 1. Geodesic Contraction Rules

1. $A \subseteq A \oplus B$
2. If $A \cap B \neq \emptyset$, then $A \oplus B = A$
3. If $B \neq \emptyset$, then $A \oplus B \cap B \neq \emptyset$.
4. If $B = \emptyset$ then $A \oplus B = A$
5. If $A \subseteq B^c$ then $A \oplus B = ((A \oplus A^c) \oplus B) \cap (A \cup B)$
6. If $A \oplus C \cap B \neq \emptyset$ and $B \subseteq C$, then $A \oplus B = (A \oplus C) \cap (A \cup B)$
7. If $A \subseteq B$ then $A \oplus B^c \subseteq B \oplus B^c$
8. If $A \oplus A^c \subseteq B^c$ then $B \oplus B^c \subseteq A^c$

2. $d(A,x) = n$, for $n > 1$, if and only if, $d(A \oplus_R A^c, x) = n - 1$.

Proof. Part 1 is straightforward.

For 2, $d(A,x) = n$ implies that there exists $y \in A$ such that $d(y,x) = n$. Since $n > 1$, this implies that there exists $z \in V$ such that $d(y,z) + d(z,x) = (y,x)$ with $d(y,z) = 1$. Therefore, $d(z,x) = n - 1$ which implies that $z \notin A$ (for if not the distance of x from A would be less than n). Therefore $d(A,z) = 1$ and from above $z \in A \oplus_R A^c$. This implies that $d(A \oplus_R A^c, x) \leq n - 1$. Now suppose that $d(A \oplus_R A^c, x) = k < n - 1$. Then there would be $z' \in A \oplus_R A^c$ with $d(z',x) = k$. By Lemma above $d(A,z') = 1$. This implies that there is $y' \in A$ such that $d(y',z') = 1$. Hence, $d(y',x) < d(y',z') + d(z',x) = k+1 \leq n-1$ which contradicts $d(A,x) = n$.

Definition 3. *An operator that satisfies 1–8 of the Table 1 will be called geodesic contraction.*

A few words about the rules appearing in Table 1: Rule 1 implies reflexivity for the underlying relation of indistinguishability and corresponds to AGM postulate of Inclusion (see Table 4), Rule 2 to Vacuity, Rule 3 to Success. Rule 4 determines how the operator acts when we contract with the empty set (there is no effect). Rule 5 is the inductive step for contraction. Rule 6 is a weak form of monotonicity in the first argument and similarly Rule 7 for the second argument. Finally, Rule 8 implies symmetry for the underlying relation of indistinguishability.

Proposition 1. *The contraction operator defined in a tolerance space (see Definition 2) is geodesic.*

Proof. We show soundness for selected rules.

Rule 1 holds by definition.

For 2, observe that $A \cap B \neq \emptyset$ implies that $d(A,B) = 0$ and therefore $A \oplus B = A \cup \{x \in B \mid d(A,x) = 0\} = A \cup (A \cap B) = A$.

For Rule 5, assume $A \subseteq B^c$. If $B = \emptyset$ then $A \oplus B = A = (A \oplus A^c) \cap A$. If not assume $d(A,B) = n > 0$. By Lemma 2, we have $d(A \oplus A^c, B) = n - 1$ and therefore

$$\{x \in B \mid d(A,x) = n\} = \{x \in B \mid d(A \oplus A^c, x) = n - 1\},$$

Table 2. Geodesic Contraction Properties

1. $A \oplus B \subseteq A \cup B$
2. $(A \oplus B) \cap B^c = A$
3. If $(A \oplus C) \cap B \neq \emptyset$ and $B \subseteq C$, then $A \oplus B \subseteq A \oplus C$
4. If $(A \oplus B \cup C) \cap B \neq \emptyset$, then $A \oplus B \subseteq A \oplus (B \cup C)$
5. If $A \subseteq B$ then $A \oplus A^c \subseteq B \oplus B^c$
6. $A \oplus (B \cup C) \subseteq A \oplus B \cup A \oplus C$
7. If $A \oplus B \cap C \neq \emptyset$ then $A \oplus C \subseteq A \oplus B$
8. If $B \subseteq A^c$, then $(A \oplus A^c) \cap B \neq \emptyset$ iff $(B \oplus B^c) \cap A \neq \emptyset$
9. If $A \oplus B \cap C \neq \emptyset$ then $A \oplus B \oplus C \subseteq A \oplus B$

again using Lemma 2. We have

$$((A \oplus A^c) \oplus B) \cap (A \cup B) = ((A \oplus A^c) \cup \{x \in B \mid d(A \oplus A^c, x) = n-1\}) \cap (A \cup B).$$

If $d(A, B) = 1$ then

$$(A \oplus A^c) \cap (A \cup B) = A \cup (A^c \cap B) = A \cup \{x \in B \mid d(A, x) = 1\}$$

and

$$\{x \in B \mid d(A \oplus A^c, x) = 0\} \cap (A \cup B) = \{x \in B \mid d(A, x) = 1\}.$$

If $d(A, B) > 1$ then

$$(A \oplus A^c) \cap (A \cup B) = A$$

and

$$\{x \in B \mid d(A \oplus A^c, x) = n-1\} \cap (A \cup B) = \{x \in B \mid d(A, x) = n\}$$

In both cases we have $((A \oplus A^c) \oplus B) \cap (A \cup B) = A \oplus B$.

Proposition 2. *A geodesic contraction operator satisfies the properties of Table 2.*

Proof. (selected) Property 1 follows from Rules 2 and 5.
Property 2 follows from Property 1.
Property 3 follows from Rule 6.
Property 4 follows from Property 3 because $B \subseteq B \cup C$.
Property 5 follows from Rules 6 and 7.

Proposition 3. *The properties of Table 3 do not hold for the class of geodesic contraction operators.*

Proof. (selected) We employ Proposition 1. All counterexamples are based on the tolerance space of Figure 1.

Table 3. Non-valid Properties

1. If $A \oplus C \subseteq B^c$, then $(A \oplus C) \oplus B \subseteq (A \oplus B) \cup (A \oplus C)$
2. If $A \oplus C \subseteq B^c$ and $B \subseteq C$, then $(A \oplus C) \oplus B \subseteq (A \oplus B) \cup (A \oplus C)$
3. $(A \oplus B) \oplus C = (A \oplus C) \oplus B$
4. If $A \oplus C \subseteq B^c$ and $A \oplus B \subseteq C^c$, then $(A \oplus B) \oplus C = (A \oplus C) \oplus B$
5. If $(A \oplus B) \cap C \neq \emptyset$ and $(A \oplus C) \cap B \neq \emptyset$ then $A \oplus B = A \oplus C$
6. If $A \subseteq B^c$ then $C \oplus B \subseteq (A \oplus B) \cup C$
7. If $A \cup B \subseteq C$ then $(A \oplus C) \cup B = (B \oplus C) \cup A$
8. If $A \subseteq C \subseteq B^c$ then $A \oplus B \subseteq C \oplus B$
9. If $A \subseteq B$ then $A \oplus C \subseteq B \oplus C$
10. If $A \cup C \subseteq B^c$, then $A \subseteq (A \oplus B) \cap D$ iff $C \subseteq (C \oplus B) \cap D$

For Property 1, let $A = \{d\}$, $B = \{a, f\}$ and $C = \{b, g\}$. We have $A \oplus B = \{d, f\}$ and $A \oplus C = \{d, b\}$. So, we have $(A \oplus B) \cup (A \oplus C) = \{d, b, f\}$ and $(A \oplus C) \oplus B = \{d, b, a\}$.

For Property 4, let $A = \{d\}$, $B = \{a, f\}$ and $C = \{b, g\}$ as above. We have $(A \oplus B) \oplus C = \{d, f, g\}$ and $(A \oplus C) \oplus B = \{d, b, a\}$.

For Property 6, let $A = \{d\}$, $B = \{a, f\}$ and $C = \{b, g\}$ again as above. We have $C \oplus B = \{a, b, f, g\}$ and $(A \oplus B) \cup C = \{b, d, f, g\}$.

4 The Tolerance Space Generated by Contraction

Next, we study the conditions under which a geodesic contraction operator defines a tolerance. We shall show that this correspondence is bijective (Proposition 4). To this end, suppose X is a set and $\oplus$ is an operation on its subsets. Define a relation $R_\oplus$ on X with $(x, y) \in R_\oplus$ if and only if $y \in \{x\} \oplus \{x\}^c$.

Lemma 3. *Suppose $\oplus$ is a geodesic contraction. Then, $R_\oplus$ is reflexive and symmetric.*

Proof. Reflexivity holds because, by Rule 1, $\{x\} \subseteq \{x\} \oplus \{x\}^c$ so $x \in \{x\} \oplus \{x\}^c$. For symmetry, suppose $(x, y) \in R_\oplus$ we have $y \in \{x\} \oplus \{x\}^c$ therefore $\{y\} \cap \{x\} \oplus \{x\}^c \neq \emptyset$ which implies by Rule 8 that $\{x\} \cap \{y\} \oplus \{y\}^c \neq \emptyset$. The latter implies that $x \in \{y\} \oplus \{y\}^c$, i.e. $(y, x) \in R_\oplus$.

We will write d for the geodesic distance generated by $R_\oplus$.

Lemma 4. *Suppose $\oplus$ is geodesic. Then,*

$$\{x \mid d(A, x) \leq 1\} = A \oplus A^c.$$

Proof. For the left to right inclusion, let $A \subseteq X$ such that $x \in A$. Since $\{x\} \subseteq A$, we have $\{x\} \oplus \{x\}^c \subseteq A \oplus A^c$ by Property 5.

For the other direction, suppose $x \in A \oplus A^c$. If $x \in A$ then $d(A, x) = 0$. Assume $x \notin A$, Since $x \in A \oplus A^c$ implies $A \oplus A^c \cap \{x\} \neq \emptyset$, Rule 8 applies and we have $\{x\} \oplus \{x\}^c \cap A \neq \emptyset$. So, there exists $y \in A$ with $y \in \{x\} \oplus \{x\}^c$ which implies $(x, y) \in R_\oplus$, and by symmetry $(y, x) \in R_\oplus$ i.e. $d(y, x) = 1$ which implies $d(A, x) = 1$.

Now we can show that a geodesic generates a connected tolerance space.

Lemma 5. *If $\oplus$ is a geodesic contraction then $(X, R_\oplus)$ is connected.*

Proof. Suppose $(X, R_\oplus)$ is not connected. Then place the (at least two) connected components of X in two subsets A and A^c. Since $A^c \neq \emptyset$ then, by Rule 3, $A \oplus A^c \cap A^c \neq \emptyset$. Let $x \in A \oplus A^c \cap A^c$. By Lemma 4, $d(A, x) = 1$ which implies that there exits $y \in A$ such that $yR_\oplus x$, a contradiction.

Lemma 6. *Suppose $\oplus$ is a geodesic contraction. Then we have, for all $n > 1$, $d(A, x) = n$ iff $d(A \oplus A^c, x) = n - 1$.*

Proof. Using Lemma 4 the proof is similar to Lemma 2.2.

Next we show that all geodesic contraction operators are induced by a tolerance space (using Definition 2) .

Proposition 4. *Given a geodesic contraction $\oplus$, we have*

$$\oplus = \oplus_{R_\oplus}.$$

Proof. Pick $A, B \subseteq X$. We shall show that $A \oplus B = A \oplus_{R_\oplus} B$ by induction on the geodesic distance $d(A, B)$ of A and B in the space $(X, R_\oplus)$.

Assume $A, B \neq \emptyset$. Let $d(A, B) = 0$ then there exists $x \in A \cap B$ therefore $A \cap B \neq \emptyset$ so by Rule 2 (satisfied by both $\oplus$ and $\oplus_{R_\oplus}$) we have $A \oplus B = A \cap B = A \oplus_{R_\oplus} B$.

Let $d(A, B) = 1$. We have $A \oplus_{R_\oplus} B = A \cup \{x \in B \mid d(A, x) = 1\}$. By Lemma 4, the latter set is exactly $A \oplus A^c \cap (A \cup B)$. We have $A \oplus B = A \oplus A^c \cap (A \cup B)$ by Rules 2 and 5.

Assume that is true for all k where $1 \leq k < n$ and let $d(A, B) = n$. By definition,

$$A \oplus_{R_\oplus} B = A \cup \{x \in B \mid d(A, x) = n\}.$$

We have,

$$A \oplus_{R_\oplus} B = ((A \oplus_{R_\oplus} A^c) \oplus_{R_\oplus} B) \cap (A \cup B)$$

by Rule 5. Because of the same rule,

$$A \oplus B = ((A \oplus A^c) \oplus B) \cap (A \cup B).$$

So it suffices to prove

$$(A \oplus_{R_\oplus} A^c) \oplus_{R_\oplus} B = (A \oplus A^c) \oplus B.$$

By the induction hypothesis, we have

$$A \oplus_{R_\oplus} A^c = A \oplus A^c,$$

so it remains to show

$$(A \oplus A^c) \oplus_{R_\oplus} B = (A \oplus A^c) \oplus B$$

which follows from Induction hypothesis because $d(A \oplus A^c, B) = n - 1$ using Lemma 6.

Finally assume $B = \emptyset$. Then $A \oplus B = A \oplus_{R_\oplus} B = A$ using Rule 4.

The above proposition shows that the set of rules of Table 1 characterizes the class of geodesic metrics. Note that it also implies that the given tolerance space (X, R) equals the tolerance space generated by $\oplus_R$.

5 Comparison with Previous Approaches

Our notion of similarity based on the geodesic distance is novel (introduced in [3]). We do not regard similarity as a basic notion but rather one that is derived from the more fundamental concept of indistinguishability. Our models should be thought of as graphs or Kripke models with an accessibility relation. The geodesic distance that leads to similarity is a byproduct of this basic structure.

Nevertheless, our approach ultimately employs minimization of a similarity relation and, therefore, it is part of a traditional approach of performing belief change using similarity. One of the first uses of similarity was Lewis's evaluation of counterfactuals ([6]): $a > b$ holds when b is true at the most similar a-world(s) to the current one. Lewis used a reflexive and symmetric relation of similarity indexed with propositions.

Non-indexed global similarity relations have been mostly identified with metric distances. A non-metric approach satisfying Williamson's axioms ([7]) for a quaternary comparative similarity relation appears in [8] where correspondence results are shown between properties of selection functions (that can be used to define contraction) and global similarity relations based on the subset difference between theories. This study is syntactic because the similarity relation is imposed on theories rather than models, and is confined to properties of the selection function rather than the properties of the associated contraction operation.

The metric approach to belief change has been studied in detail in [9]. Although the characterization results of [9] are confined to belief revision operators, it is certain that corresponding postulates for distance-based contraction will all be valid in our framework. However, as we have noted in [4], this correspondence is not bijective, as two distance metrics may generate the same belief revision operator; that is, logical operations are too coarse to tell the difference between two distance metrics. Other results relating metrics and geodesic metrics to non-prioritized belief revision (or merge) operators (as in [10,11]) will appear in a future paper.

Now, we will attempt to place geodesic contraction among the numerous contraction proposals. Using geodesic contraction, one can define the traditional contraction operation $\ominus$ with

$$A \ominus B = A \oplus B^c.$$

Table 4. AGM Contraction Postulates

$K \div p = \mathrm{Cn}(K \div p)$	Closure
$K \div \subseteq K$	Inclusion
If $p \notin K$, then $K \div p = K$	Vacuity
If $p \notin \mathrm{Cn}(\emptyset)$, then $p \notin \mathrm{Cn}(K \div p)$	Success
If $p \leftrightarrow q \in \mathrm{Cn}(\emptyset)$, then $K \div p = K \div q$	Extensionality
$K \subseteq \mathrm{Cn}((K \div p) \cup \{p\})$	Recovery
$K \div p \cap K \div q \subseteq K \div (p \wedge q)$	Intersection
If $p \notin K \div (p \wedge q)$, then $K \div (p \wedge q) \subseteq K \div p$	Conjunction

As our approach is semantic and our rules are set theoretic, we will assume that there is a way to turn the rules using subsets and set operations to rules about theories. This happens when the operator Mod, turning a theory to the set of its models, is surjective (every subset is represented by a theory), and the operator Th, turning a set of models to the corresponding closed theory are inverses of each other (on closed theories). For example, such a correspondence appears when we use a finite language, classical propositional logic for syntax, and binary valuations for models. Assuming such a translation, we are able to compare our rules and framework to those appearing in the majority of contraction literature. Our presentation is informal.

We start with the basic AGM contraction postulates ([12]). Closure and Extensionality are assured by our framework. As we noted earlier, Inclusion corresponds to reflexivity with Rule 1. Vacuity corresponds to Rule 2. Success corresponds to connectedness with Rule 3. Recovery is satisfied by geodesic contraction and corresponds to Property 2. Many authors have identified cases where recovery is not appropriate ([13,14,15]). A desirable feature of our framework is that it allows us to define a variety of contraction operators, including some where recovery does not hold. The semantics, or rather, the underlying knowledge representation structure is what guides the formation of the contraction rules.

As an example, suppose that the speed limit is 65 miles per hour and our tachometer reads 64 miles per hour. If we allow the possibility of speeding, then, using contraction, we reach the state 64,66. However, we might want, as usually the case is, to consider intervals, or more general, convex sets as epistemic states. So an alternative would be to include 65 and instead reach the state $\{64, 65, 66\}$. This is achieved by defining a withdrawal (see [13]) operator with:

$$A \oplus_R B = \begin{cases} \{y \mid d(A, y) \leq d(A, B)\} & \text{if } B \neq \emptyset \\ A & \text{otherwise.} \end{cases}$$

If now we receive information that we did not speed after all, then the final state according to our framework should be $\{64, 65\}$ rather than the initial $\{64\}$. Therefore, recovery fails.

We will now turn to verifying contraction postulates governing successive contraction. First, consider the following rule

$$\text{If } q \notin K \div p, \text{ then } K \div p \div q = K \div p$$

which follows from the AGM postulates and corresponds to Property 9. The following postulate is called Insertion in [16]

$$\text{If } q \in K \div p, \text{ then } K \div p \div q = (K \div p) \cap (K \div q)$$

and is not valid in our framework. It corresponds to the invalid property 1. Insertion is a property of Hansson's *Unified Global Specified Meet* contraction ([17]) so geodesic contraction does not belong in this class of operators. Moreover, there is no rule that reduces geodesic contraction to a *single-step* contraction. The slightly weaker property

$$\text{If } q \in K \div p \text{ and } p \vdash q, \text{ then } K \div p \div q = (K \div p) \cap (K \div q)$$

is not valid either (see non-valid Property 2). This property was introduced in [16] and is satisfied by the class of *Principled Iterated* contraction operators, so geodesic contraction does not belong in this class. *Lexicographic* contraction ([18]) is a Principled Iterated contraction, so geodesic contraction is not lexicographic.

Geodesic operators are not commutative; that is the following property

$$K \div p \div q = K \div q \div p$$

is not valid in the geodesic class. Its counterpart is non-valid Property 3. In contrast, commutativity is valid in the *Finite State* contraction operators of [19]. Even the following weaker form of commutativity

$$\text{If } q \in K \div p \text{ and } p \in K \div q, \text{ then } K \div p \div q = K \div q \div p$$

fails (non-valid Property 4).

Geodesic contractions are global in the sense of Hansson, as they are defined for every non-empty subset. However they violate almost all rules suggested in [20]. In particular, the following four:

If $p \in K_1$, then $(K_1 \div p) \cap K_2 \subseteq K_2 \div p$	Non-addition
If $p \in K_1 \cap K_2$, then $(K_1 \div p) \cap K_2 = (K_2 \div p) \cap K_1$	Symmetric inclusion
If $p \in K_1 \subseteq K_2$, then $K_1 \div p \subseteq K_2 \div p$	Contractive monotonicity
For all z and all $p \in K_1 \cap K_2$:	
$K_1 \subseteq K_1 \div p + z$ iff $K_2 \subseteq K_2 \div p + z$	Recovery equivalence

are all not valid (they correspond to invalid Properties 6, 7, 8 and 10, respectively).

6 Conclusion

We have introduced a semantic framework based on a view that similarity can be measured with degrees of indistinguishability. The framework consists of tolerance spaces and their associated geodesic metrics. Then, we defined a contraction operator through the minimization of the geodesic metric. We presented a simple set of properties that characterize the class of geodesic contraction and properties that are valid and not valid for this class. We translated those properties into the usual postulates of contraction and iterated contraction. Although geodesic contraction satisfies all AGM postulates, it violates the majority of postulates in the iterated contraction literature.

The fact that geodesic contraction fails to satisfy the suggested logical postulates can only be regarded as negative, both for geodesic contraction and said postulates. Geodesic contraction appears to be weak. Although geodesic contraction has firm, easily comprehensible semantics, those semantics can easily produce counterexamples. Similarly, it seems that suggested iterated contraction operators fail to include a restricted but useful source of contraction: error. In addition, geodesic metrics are easily generated as thresholds of distance metrics, so we believe that further research towards applications may be useful. Theoretical directions can include the study of other useful operators such as the withdrawal operator of the previous section as well as formal logical systems whose semantics include minimization of a metric on a tolerance space.

Acknowledgments. I am grateful to the anonymous referees whose comments helped improve this paper. This work was supported by the PSC-CUNY award 64599-0042.

References

1. Zeeman, E.C.: The topology of the brain and visual perception. In: Fort, M.K. (ed.) The Topology of 3-Manifolds, pp. 240–256. Prentice Hall, Englewood Cliffs (1962)
2. Bell, J.L.: A new approach to quantum logic. The British Journal for the Philosophy of Science 37, 83–99 (1986)
3. Georgatos, K.: On indistinguishability and prototypes. Logic Journal of the IGPL 11(5), 531–545 (2003)
4. Georgatos, K.: Geodesic revision. Journal of Logic and Computation 19(3), 447–459 (2009)
5. Georgatos, K.: Conditioning by Minimizing Accessibility. In: Bonanno, G., Löwe, B., van der Hoek, W. (eds.) LOFT 2008. LNCS (LNAI), vol. 6006, pp. 20–33. Springer, Heidelberg (2010)
6. Lewis, D.: Counterfactuals. Harvard University Press, Cambridge (1973)
7. Williamson, T.: First-order logics for comparative similarity. Notre Dame Journal of Formal Logic 29, 457–481 (1988)
8. Hansson, S.O.: Similarity semantics and minimal changes of belief. Erkenntnis 37, 401–429 (1992)

9. Lehmann, D.J., Magidor, M., Schlechta, K.: Distance semantics for belief revision. J. Symb. Log. 66(1), 295–317 (2001)
10. Rabinowicz, W.: Global belief revision based on similarities between worlds. In: Hansson, S.O., Rabinowicz, W. (eds.) Logic for a Change: Essays Dedicated to Sten Lindström on the Occasion of His Fiftieth Birthday. Uppsala prints and preprints in philosophy, vol. 9, pp. 80–105. Department of Philosophy, Uppsala University (1995)
11. Schlechta, K.: Non-prioritized belief revision based on distances between models. Theoria 63(1-2), 34–53 (1997)
12. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. Journal of Symbolic Logic 50, 510–530 (1985)
13. Makinson, D.: On the status of the postulate of recovery in the logic of theory change. Journal of Philosophical Logic 16, 383–394 (1987)
14. Hansson, S.O.: Belief contraction without recovery. Studia Logica 50(2), 251–260 (1991)
15. Fermé, E.L.: On the logic of theory change: Contraction without recovery. Journal of Logic, Language and Information 7(2), 127–137 (1998)
16. Nayak, A.C., Goebel, R., Orgun, M.A.: Iterated belief contraction from first principles. In: Veloso, M.M. (ed.) IJCAI, pp. 2568–2573 (2007)
17. Hansson, S.O.: Multiple and iterated contraction reduced to single-step single-sentence contraction. Synthese 173(2), 153–177 (2010)
18. Nayak, A.C., Goebel, R., Orgun, M.A., Pham, T.: Taking LEVI IDENTITY Seriously: A Plea for Iterated Belief Contraction. In: Lang, J., Lin, F., Wang, J. (eds.) KSEM 2006. LNCS (LNAI), vol. 4092, pp. 305–317. Springer, Heidelberg (2006)
19. Tamminga, A.M.: Expansion and contraction of finite states. Studia Logica 76(3), 427–442 (2004)
20. Hansson, S.O.: Global and iterated contraction and revision: An exploration of uniform and semi-uniform approaches. J. Philosophical Logic 41(1), 143–172 (2012)

A Note on Extensions: Admissible Rules via Semantics

Jeroen Goudsmit[⋆]

Utrecht University,
Janskerkhof 13a 3512BL Utrecht, The Netherlands
J.P.Goudsmit@uu.nl
http://jeroengoudsmit.com

Abstract. Any intermediate logic with the disjunction property admits the Visser rules if and only if it has the extension property. This equivalence restricts nicely to the extension property up to n. In this paper we demonstrate that the same goes even when omitting the rule *ex falso quod libet*, that is, working over minimal rather than intuitionistic logic. We lay the groundwork for providing a basis of admissibility for minimal logic, and tie the admissibility of the Mints–Skura rule to the extension property in a stratified manner.

Keywords: admissible rules, minimal logic, disjunction property, extensions of Kripke models.

The admissible rules of a theory are those rules under which the theory is closed. Derivable rules are admissible. For classical propositional logic, this is the whole story. For intuitionistic propositional logic (IPC) — and minimal logic — it is not.

Friedman [7, Problem 40] conjectured admissibility for IPC to be decidable, as has been confirmed by Rybakov [25]. De Jongh and Visser conjectured that the *Visser rules* form a basis of admissibility for IPC, that is to say, all admissible rules of IPC become derivable after adjoining the Visser rules. Rozière [24] and Iemhoff [13] independently confirmed this. Again independently, Skura [27] demonstrated that IPC is the sole intermediate logic that admits a restricted form of the Visser rules.

At the Pisa Proof Theory workshop of 2012 George Metcalfe gave a tutorial on admissible rules. As has become standard practice, Metcalfe mentioned Lorenzen [20] as the first place where admissible rules where studied *an sich*. Jan von Plato objected that Johansson [18] already discussed them. Odintsov and Rybakov [23] proved admissibility for minimal logic to be decidable. In this paper we lay the groundwork for studying all admissible rules of Johansson's minimal logic, with the eventual goal of providing an explicit basis of admissibility.

This paper aims to provide uniformity to some of the literature regarding admissible rules for logics above minimal logic. We make several observations,

[⋆] Support by the Netherlands Organisation for Scientific Research under grant 639.032.918 is gratefully acknowledged.

many of which not elsewhere available in the generality stated here. Although this paper contains novel results, most notably the semantic characterization of admissibility for an adaptation of the rules studied by Skura, its main purpose is to provide a unified approach to the study of admissible rules over minimal logic.

1 Preliminaries

We first fix some basic notation. Many definitions are fairly straightforward adaptations of their well-known intuitionistic counterparts.

Definition 1 (Propositional Language). *The language of* propositional formulae *is defined as follows, starting from a fixed countably infinite set of* propositional variables Var*. A formula is said to be* atomic *if it consists solely of a variable or falsum, the set of atomic formulae is denoted as* Atom*.*

$$\mathcal{L} ::= \mathsf{Var} \mid \bot \mid \mathcal{L} \wedge \mathcal{L} \mid \mathcal{L} \vee \mathcal{L} \mid \mathcal{L} \to \mathcal{L}.$$

We will denote formulae by captital Latin letters at the beginning of the alphabet, and use greek capitals to refer exclusively to finite sets of formulae. For greater convenience we write $\Gamma \Rightarrow \Delta$ to mean $\bigwedge \Gamma \to \bigvee \Delta$, that is to say, the conjunction of all formulae in Γ implies the disjunction of all formulae in Δ. We will only use this notation when both Γ and Δ are non-empty. This definition of a Kripke model differs in one important regard from the standard definition as given for instance by Troelstra and van Dalen [29]. The difference is in that a valuation can determine whether $\bot$ is to hold, analogous to the definition of Došen [5], whereas in a Kripke model of IPC this is fixed. This to ensure completeness for minimal logic, in which $\bot$ does not derive everything.

Definition 2 (Kripke Model). *A* Kripke model *is a pair* $K = \langle K, v\rangle$ *where* K *is a partial order and* v *(the valuation) is a monotone map* $v : K \to \mathcal{P}(\mathsf{Atom})$*. We define a relation* $\Vdash$ *(forces) between* K *and* $\mathcal{L}$ *inductively as follows*

$$\begin{array}{lll}
k \Vdash A & := & k \in v(A) \textit{ for atomic } A \\
k \Vdash A \wedge B & := & k \Vdash A \textit{ and } k \Vdash B \\
k \Vdash A \vee B & := & k \Vdash A \textit{ or } k \Vdash B \\
k \Vdash A \to B & := & l \Vdash B \textit{ whenever } l \Vdash A \textit{ for all } k \leq l
\end{array}$$

The model K is said to be *rooted* if K has a least element, and K is *strict* when $k \Vdash \bot$ holds for no $k \in K$. We say that K is *finite* when K is finite and v maps but finitely many atoms to a non-empty upset. As usual, we write $K \Vdash A$ to mean that $k \Vdash A$ for all $k \in K$. For convenience we also write $K \Vdash x$ when $x \subseteq \mathcal{L}$ to mean that $K \Vdash A$ for all $A \in x$. The *theory* of K, written $\mathrm{Th}\, K$, equals the set of formulae A such that $K \Vdash A$.

Kripke models can be endowed with a topology, the Alexandroff topology, where opens are exactly upsets. Using the thus inherited topology we can define a sensible notion of maps as follows. These kinds of maps have been called p-morphisms and bounded morphisms, we will simply call them maps (of Kripke models). From the definition it naturally follows that for any map $f : K \to L$ we have $\mathrm{Th}\, L \subseteq \mathrm{Th}\, K$.

Definition 3 (Maps of Kripke Models). *A* map $f : \langle K, v \rangle \to \langle L, w \rangle$ *between Kripke models is an open and continuous function such that* $w \circ f = v$*.*

We often place models next to one another, below a formal definition.

Definition 4. *Given a set of Kripke models* $\mathcal{K} = \{K = \langle K, v_K \rangle \mid K \in \mathcal{K}\}$ *we define their disjoint union as*

$$\coprod \mathcal{K} := \left\langle \coprod \mathcal{K}, \bigcup_{K \in \mathcal{K}} v_K \right\rangle .$$

In order to be as generic as possible, and to not get involved with the intricacies of axiomatizations of the logics at hand, we use the notion of a consequence relation. Rybakov [26] already used consequence relations in the context of admissible rules, we shall do the same. We will use the formulation of Cintula and Metcalfe [3], where a consequence relation is concerned with multi-conclusion rules.[1] Multi-conclusion rules allow for a more succinct notation of the rule schemes we will use later on, and help us steer clear of some obstacles concerning the disjunction property, see Citkin [4].

First, a *rule* is an ordered pair of finite sets of formulae, written Γ/Δ. A consequence relation is simply a set of rules satisfying certain sensible properties. When reading the following definition, think of the relation $\vdash$ defined as $\Gamma \vdash \Delta$ iff there is a (minimal logic) proof of some $A \in \Delta$ with assumptions in Γ.

Definition 5 (Consequence Relation). *A* consequence relation *(or logic) consists of a relation called derivability, denoted* $\vdash$*, between finite sets of formulae subject to the following axioms, where* A *is a formula and* $\Gamma, \Theta, \Delta, \Pi$ *are finite sets of formulae.*

reflexivity	$A \vdash A$;
monotonicity	*if* $\Gamma \vdash \Delta$*, then* $\Gamma, \Theta \vdash \Delta, \Pi$;
transitivity	*if* $\Gamma \vdash \Delta, A$ *and* $A, \Theta \vdash \Pi$*, then* $\Gamma, \Theta \vdash \Delta, \Pi$;
structurality	*if* $\Gamma \vdash \Delta$*, then* $\sigma(\Gamma) \vdash \sigma(\Delta)$.

We extend the notation to infinite sets on the left by defining $x \vdash \Delta$ to mean that there exists a finite $\Gamma \subseteq x$ such that $\Gamma \vdash \Delta$. When one of the sets is a singleton we omit braces, and if it is empty we omit it entirely. A formulae A is said to be a *theorem* of this consequence relation if $\vdash A$. A logic is *consistent* when not all formulae are theorems.

Given any set of formulae x and any consequence relation $\vdash$ one can form a new consequence relation $\vdash_x$ where $\Gamma \vdash_x \Delta$ holds if and only if $x + \Gamma \vdash \Delta$.

[1] Note that they call the definition below a "finitary structural multi-conclusion consequence relation". We call this simply a consequence relation or logic.

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad \frac{\Gamma \vdash A \vee B \qquad A, \Gamma \vdash C \qquad B, \Gamma \vdash C}{\Gamma \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

Fig. 1. Closure properties of $\vdash$

We say that x is a *theory* if it contains all theorems of $\vdash_x$. Moreover, given a consequence relation $\vdash$ one can form the consequence relation of admissibility. Roughly said, a rule is admissible when for each substitution we know that if its assumptions are theorems under the substitution, then one of the conclusions must be a theorem under the same substitution.

Definition 6 (Admissible Rule). *Let $\vdash$ be a consequence relation and Γ/Δ a rule. We say that Γ/Δ is* admissible*, denoted $\Gamma \mathrel{|\!\sim} \Delta$ when for each substitution σ, if $\vdash \sigma(A)$ for all $A \in \Gamma$, then $\vdash \sigma(A)$ for some $A \in \Delta$.*

The thus defined relation $\mathrel{|\!\sim}$ of admissibility is a consequence relation. By structurality, $\mathrel{|\!\sim}$ contains $\vdash$, so reflexivity is clear. All other properties are a simple matter of verification. A rule is said to be *admissible for x* whenever it is admissible for $\vdash_x$.

It is important to keep in mind that a logic need not satisfy the deduction theorem, that is, $\Gamma + A \vdash B$ need not be equivalent to $\Gamma \vdash A \to B$, even when the logic is an extension of minimal logic. Likewise, when a rule is admissible for a given logic, it need not be admissible for an extension. In the following, we will let $\vdash$ stand for any logic which contains minimal logic, satisfying the close properties of Fig. 1, satisfying the deduction theorem.

2 Extensions of Models

An extension of a Kripke model is that same model, adjoined with a least element and a choice of valuation there. In this section we investigate when a given model of a theory has an extension satisfying the same theory. This is interesting in and of itself, but it also has applications for admissible rules. The characterization that is to be given at the end of this section suggests a particularly nice schema of admissible rules, namely the de Jongh rules. With some additional bookkeeping one can use this characterization to prove, for instance, that the de Jongh rules form a basis of the Gabbay–de Jongh logics of Gabbay and de Jongh [8], as has been done in Goudsmit and Iemhoff [11]. Here we show how these results are actually more general, in that they help towards providing a basis of admissibility of minimal logic. As we do not attain this goal here, we omit the bookkeeping to make the material more digestible.

Let us now first introduce notation for extensions. What the following definition comes down to is that the model K/x is K with a node (named $\underline{x}$) placed below it, which forces precisely the atomics x contains. For greater convenience we will often write K/x even when x contains non-atomic formulae, this is understood to denote $K/(x \cap \mathsf{Atom})$.

Definition 7 (Extension). *Let $K = \langle K, v\rangle$ be a Kripke model and $x \subset \mathrm{Th}\, K$ be a set of atoms. We define the* extension of K over x, *denoted K/x, as follows.*

$$K/x = \Big\langle K_x, \big(k \in K_x \mapsto v(A) \textit{ if } k \in K \textit{ and } x \textit{ otherwise}\big)\Big\rangle$$

Here K_x is the partial order with underlying set $K + \{\underline{x}\}$ ordered by $k \leq l$ if and only if $k \leq l$ holds in K or $k = \underline{x}$.

The following characterization is fairly straightforward and can be proven with almost no effort at all. It does look fairly familiar to the inductive characterization of the Aczel Slash as given in Smoryński [28, Theorem 5.1.18], and this is no coincidence. When we take K to be the canonical model of some theory x as in Definition 9, then this characterization is identical, as is can be readily seen in the presence of Lemma 2. Another observation: if $K \Vdash A \to B$ but $K \nVdash A$ then $K/x \Vdash A \to B$, which clarifies the importance of Definition 11 below. Later on we will formulate some constraints on theories x given models K under which $\mathrm{Th}\, K/x$ actually equals x. We will give an exact characterization when given a model K and a theory x there exists some extension K/y such that $\mathrm{Th}\, K/y \supseteq x$.

Lemma 1 (Forcing of Extensions). *Let K be any Kripke model and let $x \subseteq \mathrm{Th}\, K$ be arbitrary. The following hold:*

$$\begin{array}{lll}
K/x \Vdash C & \textit{iff} & x \ni C \textit{ for atomic } C \\
K/x \Vdash A \wedge B & \textit{iff} & K/x \Vdash A \textit{ and } K/x \Vdash B \\
K/x \Vdash A \vee B & \textit{iff} & K/x \Vdash A \textit{ or } K/x \Vdash B \\
K/x \Vdash A \to B & \textit{iff} & K \Vdash A \to B \textit{ and if } K/x \Vdash A \textit{ then } K/x \Vdash B
\end{array}$$

A logic is said to have the disjunction property when each derivable disjunction has a derivable disjunct. See [2] for a wonderful and comprehensive survey of intermediate logics and the disjunction property. The above characterization shows that the theory of each extension satisfies this disjunction property. So when we seek theories x such that $\mathrm{Th}\, K/x = x$ holds, x had better satisfy the disjunction property too.

To smoothen proofs we use a generalized form of the disjunction property, the idea of being saturated in something else. Note that a set of formulae has the disjunction property if it is saturated in itself, in which case we call it *saturated.* This is one of the many places where one could introduce further bookkeeping by restricting the formulae considered to some set, for instance the set of atomic formulae. For details regarding this one can consult [11], this is the last we speak of it.

Definition 8 (Saturated Set). *Let $x \subseteq y$ be sets or formulae. We say that x is saturated in y, written $x \preccurlyeq y$, whenever*

$$x \vdash \bigvee \Delta \quad \textit{entails} \quad y \cap \Delta \neq \emptyset \quad \textit{for all non-empty finite } \Delta.$$

Each theory can be extended to a saturated set, avoiding a chosen formula outside of this set. This is akin to a basic fact of lattices: given a filter and an ideal with empty intersection, there exists an extension of the filter to a prime filter, which does *not* intersect the ideal (see e.g. [19, section 2.3]). Let us mention three small results. The first is an immediate corollary of Lemma 3 and 4 of [11], the other two follow from the first. The final corollary comes in handy in proving that our canonical model works well, and it is also crucial in Lemma 4.

Corollary 1. *Let x be saturated in z. There exists a saturated set y such that $x \subseteq y \subseteq z$.*

Corollary 2 (Negative Saturation Lemma). *Let x and Δ be sets of formulae such that $x \nvdash \bigvee \Delta$. Now there exists a saturated set extending x not intersecting Δ.*

Corollary 3. *Let x be a set of formulae and let A and B be formulae such that $x \nvdash A \to B$. There exists a saturated set extending x such that $z \ni A$ and $z \not\ni B$.*

Let us now define the canonical model. Not to prove completeness, although it is a natural byproduct, but to link the disjunction property to a semantic property. Our canonical model is analogous to that of Došen [5, Definition 9]. It is also similar to the model of Aczel [1], but it differs in several regards, most importantly in that his model is *strict*.

A theory is said to be consistent when its associated logic is. This notion of consistency is sufficient to ensure the existence of (consistent) saturated sets above a consistent theory due to the Negative Saturation Corollary 2, which we need to make the model a bona fide model. When the logic at hand would be some intermediate logic, then any theory must contain $\bot \to A$ for any A. This ensures that a consistent theory does *not* contain $\bot$, so the canonical model under intermediate logics would always be strict, as desired.

Definition 9 (Canonical Model). *Let x be a consistent theory. The canonical model of x, denoted by $\operatorname{can} x$, is defined as the Kripke model*

$$\operatorname{can} x := \left\langle \, \{ \, y \supseteq x \mid y \ \textit{saturated} \, \} \, , y \mapsto y \cap \mathsf{Atom} \right\rangle \quad .$$

Lemma 2. *For any theory x we have $\operatorname{Th} \operatorname{can} x = x$.*

Proof. If $x \vdash C$ then every saturated extension contains C. Conversely, if $x \nvdash C$ there is a saturated extension which does not contain C. By structural induction along C we prove that for any $y \in \operatorname{can} x$ we have $y \Vdash C$ if and only if $y \ni C$. For atomic formulae this holds by definition. The conjunction and disjunction cases follow readily from induction and saturation. We are left with the implication case, where $C = A \to B$.

From right to left, suppose that $y \ni A \to B$ and let $z \supseteq y$ be a saturated set. If $z \Vdash A$ then $z \ni A$ by induction, whence $y \subseteq z \ni B$ by saturation, induction now finished the job. To prove the converse, suppose that $y \not\ni A \to B$. This yields a saturated $z \supseteq y$ such that $z \ni A$ but $z \not\ni B$ by Corollary 3. The desired follows by induction.

Corollary 4 (Completeness). *For any theory x, if $K \Vdash A$ for all models K with $x \subseteq \operatorname{Th} K$ then $x \vdash A$.*

We can characterize the disjunction property via the following semantic property. This has already been proven by Maksimova [21, Theorem 1 and 2]. The proof below is quite similar, in that it uses the same core idea, and it nicely illustrates how little changes when moving to minimal logic.

Theorem 1. *A theory x has the disjunction property if and only if for every model $K \Vdash x$ there is a rooted model $L \Vdash x$ and a map $K \to L$.*

Proof. The implication from right to left is fairly straightforward. Suppose x does not have the disjunction property. This gives some Δ such that $x \vdash \bigvee \Delta$ but for no $A \in \Delta$ we have $x \vdash A$. Via completeness this ensures us models $K_A \Vdash x$ such that $K_A \nVdash A$. Consider then $K := \coprod_{A \in \Delta} K_A$, by assumption we have a rooted model $L \Vdash x$ and a map $K \to L$. This entails that $\operatorname{Th} L \subseteq \operatorname{Th} K$. Observe that $L \Vdash x \vdash \bigvee \Delta$, whence $L \Vdash A$ for some $A \in \Delta$ because L is rooted. This in turn entails that $K \Vdash A$ and so $K_A \Vdash A$, a clear contradiction.

Let us now focus on the other implication. Let K be a Kripke model of x. Now consider the model $L := (K + \operatorname{can} x)/x$, and see that the inclusion from K to L is a map. We will prove that $x \ni C$ if and only if $L \Vdash C$. We proceed via Lemma 1, and there is only some work to be done in the implicative case. Note that $L \Vdash A \to B$ iff $K + \operatorname{can} x \Vdash A \to B$ and if $L \Vdash A$ then $L \Vdash B$. It is clear that $\operatorname{Th} \operatorname{can} x + K = \operatorname{Th} \operatorname{can} x = x$. By induction we know that $L \Vdash A$ and $L \Vdash B$ to be equivalent to $x \vdash A$ and $x \vdash B$. The burden of proof has completely dissolved, keeping in mind *modes ponens.*

Consider again a model K of a particular theory x. When we can find a theory $y \supseteq x$ such that $\operatorname{Th} K/y = y$ holds we know that an extension of K forcing x exists. Such a theory y is, in a way, a saturated approximation of the model K containing x. We are interested in the "best" such approximation, that is to say, a saturated extension containing x such that every larger saturated extension overshoots K. This idea is captured by the notion of a tight predecessor. The definition first took form in Iemhoff [13, Section 2.1.1] where it pertained to models, and was later adopted by Jeřábek [17, Definition 3.2] to suit modal logic. Goudsmit and Iemhoff [11, Definition 11] adapted the idea to theories, and it is this definition we use here.

Definition 10 (Tight Predecessor). *Let x and z and be sets of formulae. We say that x is a tight predecessor of z when x is saturated, $x \subseteq z$ and for each saturated set $y \supset x$ we have $z \subseteq y$.*

Definition 11 (Vacuous Implications and Assumptions). *Let x be a set of formulae. Define the following:*

$$\begin{aligned} \mathrm{I}(x) &:= \left\{ A \to B \,\middle|\, \text{for formulae } A \text{ and } B \text{ such that } x \ni A \to B \text{ but } x \not\ni A \right\} \\ x^{\mathrm{a}} &:= \left\{ A \,\middle|\, \text{for some } B \text{ we have } x \ni A \to B \right\} \end{aligned}$$

Lemma 3. *Let K be a Kripke structure, let $x \subseteq \mathrm{Th}\,K$ be arbitrary and let y be a set of implications. If y^{a} does not intersect $\mathrm{Th}\,K$ and $y \subseteq \mathrm{Th}\,K$ then $K/x \Vdash y$. In particular, $\mathrm{I}\,(\mathrm{Th}\,K) \subseteq \mathrm{Th}\,K/x$.*

We can now show that a tight predecessor of a model contains all information of the theory of the extension. In [11, Lemma 9] the intermediate case was treated, here we consider any extension of minimal logic. The proof below uses the characterization of extensions, which makes it a little smoother than the original. Note that to prove the equivalence for a formulae C, one needs only knowledge of structurally smaller formulae. The additional constraint about containing $\mathrm{I}\,(\mathrm{Th}\,K)$ may be dropped, as it can be shown to always hold.

Lemma 4 (Extension Lemma). *Let K be any model, and let x be a tight predecessor of $\mathrm{Th}\,K$ containing $\mathrm{I}\,(\mathrm{Th}\,K)$. Now $\mathrm{Th}\,K/x = x$ holds.*

Proof. We prove that $K/x \Vdash C$ iff $x \ni C$ by structural induction along C. Only the implication case is interesting, the other cases are either immediate or follow from induction and the rules under which we assumed $\vdash$ to be closed. We know that $K/x \Vdash A \to$ if and only if $K/x \Vdash B$ whenever both $K \Vdash A$ and $K/x \Vdash A$.

Let us first go from right to left. If $x \ni A \to B$ and $K/x \Vdash A$ then $x \ni A$ by induction, so $x \ni B$ by assumption. Induction yields $K/x \Vdash B$ as desired.

Now for the other direction, suppose that $K \Vdash A \to B$ and that if $K/x \Vdash A$ then $K/x \Vdash B$. Furthermore assume that $x \not\ni A \to B$. We will derive a contradiction from these assumptions. By Corollary 3, there is a saturated extension $y \supseteq x$ such that $y \ni A$ and $y \not\ni B$. There are two cases, either $y = x$ or $y \supset x$.

In the former case we know $x \ni A$ and $x \not\ni B$, so induction yields $K/x \Vdash A$ and $K/x \nVdash B$.

In the latter case first observe that $\mathrm{Th}\,K \subseteq y$. If $K \Vdash A$ then $K \Vdash B$ whence $y \ni B$. On the other hand, if $K \nVdash A$ then $A \to B \in \mathrm{I}\,(\mathrm{Th}\,K) \subseteq x$. Both cases thus yield a contradiction, proving the desired.

Below we cite one of the main results of [11, Theorem 1]. The main point is that this perfectly characterizes when a model has an extension satisfying a given theory, and that this characterization works for logics extending minimal logic.

Theorem 2. *Let K be a Kripke model, let x be a subset of $\mathrm{Th}\,K$. The following are equivalent:*

1. *the set $x \cup \mathrm{I}\,(\mathrm{Th}\,K)$ is saturated in $\mathrm{Th}\,K$;*
2. *there is a tight predecessor of $\mathrm{Th}\,K$ containing x and $\mathrm{I}\,(\mathrm{Th}\,K)$;*
3. *there is a $y \subseteq \mathrm{Th}\,K$ such that $x \subseteq y = \mathrm{Th}\,K/y$.*

3 Extension Properties and Projectivity

Every model of minimal logic has an extension, and the same goes for models of IPC (in both cases, forcing nothing at all will do). In the nth Gabby–de Jongh logic one can give an extension to any juxtaposition of $n + 1$ many models, but more might fail. This property is of particular interest to us.

Definition 12 (Extension Properties). *A theory x is said to have the n^{th}* extension property *when for each set $\mathcal{K}$ of n rooted models of x there exists an extension of $\coprod \mathcal{K}$ forcing x. Given another theory y, we say that y* has extensions over x *if for each model $K \Vdash y$ such that there is an extension forcing x, there is an extension forcing y.*

The property of having extensions over a given base theory is quite interesting. First, there is a nice correspondence with projectivity in the sense of Ghilardi [10]. Projectivity and projective formulae in particular play a role in unification theory and in characterizing admissible rules, see for instance [9] and [6]. Secondly, when y has extensions over x and x has the n^{th} extension property, so does y.

The definition below is a modest generalization of projectivity in the sense of that paper. Note that a formula would be called projective there precisely if it were projective over the theorems of IPC.

Definition 13 (Projectivity). *A set of formulae y is said to be* projective over x *when there is a substitution σ (called the* projective unifier*) such that*

$$x \vdash \sigma(A) \text{ for all } y \ni A \text{ and } y \vdash \sigma(A) \leftrightarrow A \text{ for all } A$$

We can readily prove that projectivity over x entails extensions over x. This specializes to the intermediate case in that projectivity over IPC entails the n^{th} extension property for all n.

Lemma 5. *Let $x \subseteq y$ be theories. Assume that y is projective over x. Then y has extensions over x.*

Proof. Take some Kripke model $K \Vdash y$ and assume that there exists an extension forcing x. By Theorem 2, this means that $x + \mathrm{I}(K) \preccurlyeq \mathrm{Th}\, K$. Per the same theorem, it suffices to prove that $y + \mathrm{I}(K) \preccurlyeq \mathrm{Th}\, K$. So suppose $y + \mathrm{I}(K) \vdash \bigvee \Delta$ for some finite non-empty $\Delta \subseteq \mathcal{L}$. This gives a finite $\Gamma \subseteq \mathrm{I}(K)$, which we can assume to be non-empty, such that $y \vdash \Gamma \Rightarrow \Delta$. Because y is projective over x we have a σ such that $x \vdash \sigma(A)$ for all $A \in y$ and $y \vdash \sigma(A) \equiv A$ for all A, fix this σ. Transitivity ensures that $x \vdash \sigma(\Gamma) \Rightarrow \sigma(\Delta)$.

Observe that $\sigma(\Gamma) \subseteq \mathrm{I}(K)$, because if $A \to B \in \mathrm{I}(K)$ then $K \nVdash A$ and $K \Vdash A \to B$. Now as $y \subseteq \mathrm{Th}\, K$ it follows that $K \nVdash \sigma(A)$ and $K \Vdash \sigma(A \to B)$.

We now know $x \vdash \sigma(\Gamma) \Rightarrow \sigma(\Delta)$ and $x + \mathrm{I}(K) \vdash \sigma(C)$ for all $C \in \Gamma$, so the deduction theorem and transitivity ensures $x + \mathrm{I}(K) \vdash \bigvee \sigma(\Delta)$. The assumption $x + \mathrm{I}(K) \preccurlyeq \mathrm{Th}\, K$ now proves $K \Vdash \sigma(A)$ for some $A \in \Delta$, whence $K \Vdash A$ is readily derived. This proves the desired.

Do note that the above proof is basically the same as that of [10, Theorem 5, (ii) entails (iii)], it is slightly rephrased in terms of our characterization of extensions, and isolates the bare necessities. The proof in the other direction unfortunately does not generalize as readily, this is still work in progress. We solely cite the easily generalizable part without proof.

Lemma 6. *Assume that the logic at hand is an intermediate logic. Let x be a theory with the finite model property and let A be a formula such that $x + A$ has extensions over x. Now $x + A$ is projective over x.*

We end this section with one observation on the use of projectivity towards providing bases of admissible rules. The nice property of projective formulae A is that $A \mathrel{|\!\!\sim} B$ if and only if $A \vdash B$.[2] When $x + A$ admits all admissible rules of x, that is to say, when each rule is admissible with respect to $\vdash_{x+A}$ whenever it is admissible with respect to $\vdash_x$, we get the same nice property. Below we prove this, and moreover show that when y is projective over x, it follows that y admits all rules of x. This is interesting, as y also "inherits n^{th} extension properties" from x as stated above.

Lemma 7. *Let x be closed under substitution. Suppose that A is such that all rules that are admissible for x are admissible for $x + A$ too. Now $A \mathrel{|\!\!\sim}_x B$ if and only if $x + A \vdash B$.*

Proof. From left to right, assume that $A \mathrel{|\!\!\sim}_x B$. Now $A \mathrel{|\!\!\sim}_{x+A} B$ follows from assumption, whence the desired is immediate by monotonicity. Conversely, suppose that $x + A \vdash B$. Let σ be arbitrary and additionally assume that $x \vdash \sigma(A)$. We now see that $x + \sigma(A) \supseteq \sigma(x) + \sigma(A) \vdash \sigma(B)$, so transitivity yields $x \vdash \sigma(B)$ as desired.

Lemma 8. *Let $x \subseteq y$ be sets of formulae. If y is projective over x then y admits all rules that x admits.*

Proof. Let ρ be the projective unifier of y over x. Assume that $\Gamma \mathrel{|\!\!\sim}_x \Delta$, and suppose that σ is such that $y \vdash \sigma(A)$ for all $\Gamma \ni A$. Because ρ unifies y under x we get $x \vdash \rho(A)$ for all $y \ni A$, so by transitivity, $x \vdash \rho\sigma(A)$ for all $A \in \Gamma$. Now note that as $\Gamma \mathrel{|\!\!\sim}_x \Delta$, we obtain a $A \in \Delta$ such that $x \vdash \rho\sigma(A)$. As $x \subseteq y$ we obtain $y \vdash \sigma(A)$ as desired.

4 Admissible Rules

The interesting scheme of admissible rules of choice are the Visser rules. They have been shown to be a basis of admissibility for IPC, and they correspond nicely to the extension property. To neatly restrict the above result to the n^{th} extension property the de Jongh rules were introduced in [11]. With the machinery available here we can quite smoothly prove that the de Jongh rules are admissible in any of the logics at hand that satisfy the disjunction property. Let us first define these rules. As an auxiliary definition, say that a non-empty set $\mathcal{U}$ is an n-cover of another set X if $\bigcup \mathcal{U} = X$, $\emptyset \notin \mathcal{U}$ and $|\mathcal{U}| \leq n$. Per natural

[2] This must be very well-known as it is re-proven quite often, see for instance Cintula and Metcalfe [3, Lemma 2.3], Iemhoff [14, Section 2.6], Iemhoff and Metcalfe [15, Lemma 1.a], Iemhoff and Metcalfe [16, Lemma 6], Jeřábek [17, Theorem 4.1] and Dzik [6, Corollary 6].

number n, finite non-empty set of implications Γ, non-empty finite set Δ and n-cover $\mathcal{U}$ of Γ^{a}, the n^{th} *de Jongh rule* determined by Γ, Δ and $\mathcal{U}$ is defined as below.

$$\frac{\Gamma \Rightarrow \Delta}{\{\Gamma \Rightarrow \Theta \mid \Theta \in \mathcal{U} \cup \Delta\}} \, \mathcal{U} \text{ is a } n\text{-cover of } \Gamma^{\mathrm{a}}$$

Lemma 9. *Assume that x has the disjunction property and the n^{th} extension property. Now x admits the n^{th} de Jongh rule.*

Proof. Suppose we have non-empty finite Γ (with only implications) and Δ (arbitrary) and an n-cover $\mathcal{U}$ of Γ^{a} such that $x \vdash \Gamma \Rightarrow \Delta$ but $x \nvdash \Gamma \Rightarrow \Theta$ for any $\Theta \in \mathcal{U} \cup \Delta$. Completeness provides us with rooted models K_U for each $U \in \mathcal{U}$ such that $K_U \Vdash x + \Gamma$ but $K_U \nVdash \bigvee U$. Furthermore, we have models K_A for each $A \in \Delta$ such that $K_A \Vdash x + \Gamma$ but $K_A \nVdash A$.

By the n^{th} extension property we have a model $K_{\mathcal{U}}$ extending $\coprod_{u \in \mathcal{U}} K_U$ such that $K_{\mathcal{U}} \Vdash x$. The disjunction property of x ensures us a rooted model $K \Vdash x$ of which $K_{\mathcal{U}}$ and all of K_A for $A \in \Delta$ are open subsets.

Suppose that $A \to B \in \Gamma$. If $K \Vdash A$ then pick some $A \in U \in \mathcal{U}$ and note that $K_U \Vdash A \vdash \bigvee U$, a contradiction. This entails that $K \Vdash x + \Gamma$. As $x + \Gamma \vdash \Delta$ we know that $K \Vdash \bigvee \Delta$. This gives a $A \in \Delta$ such that $K \Vdash A$, which entails $K_A \Vdash A$, *quod non*. Hence all n^{th} de Jongh rules are admissible.

The de Jongh rules have quite a lot of parameters, it would be nice if these could be restricted in number. A nice source of inspiration can be found in the rule below, which has been studied in several incarnations before. Its admissibility for singleton covers with $n = 2$ in IPC was discussed by Mints [22]. Skura [27] considered this rule, also with only singleton covers but for arbitrary n, and proved that IPC is the sole intermediate logic which admits them all. Per natural number n, non-empty finite set of implications Γ and n-cover $\mathcal{U}$ of Γ^{a} we define the n^{th} *Mints–Skura rule* as below.

$$\frac{\Gamma \Rightarrow \Gamma^{\mathrm{a}}}{\{\Gamma \Rightarrow \Theta \mid \Theta \in \mathcal{U}\}} \, \mathcal{U} \text{ is a } n\text{-cover of } \Gamma^{\mathrm{a}}$$

The rule clearly is a special case of the de Jongh rule, so it holds in the presence of the n^{th} extension property due to Lemma 9. We prove the the converse below.

Lemma 10. *Assume that x has the disjunction property and assume that x admits the n^{th} Mints–Skura rule. Now x has the n^{th} extension property.*

Proof. Consider a set $\mathcal{K}$ of n rooted models of x. Due to Theorem 2, the model $L := \coprod \mathcal{K}$ has an extension precisely if $x + \mathrm{I}(\mathrm{Th}\, L) \preccurlyeq \mathrm{Th}\, L$. We reason by contradiction, so suppose that $x + \mathrm{I}(\mathrm{Th}\, L) \vdash \bigvee \Delta$ yet $\mathrm{Th}\, L \cap \Delta$ is empty for some finite non-empty Δ. The latter ensures that $\Delta \subseteq \mathrm{I}(\mathrm{Th}\, L)^{\mathrm{a}}$, the former ensures some finite non-empty $\Gamma \subseteq \mathrm{I}(\mathrm{Th}\, L)$ such that $x + \Gamma \vdash \bigvee \Delta$. Monotonicity ensures that we can assume that $\Delta \subseteq \Gamma^{\mathrm{a}}$ without loss of any generality. We thus know that $x + \Gamma \vdash \bigvee \Gamma^{\mathrm{a}}$. Construct the cover $\mathcal{U}$ as below, clearly of size at most

n. It does not contain the empty set by construction, and its union equals Γ^{a}, so $\mathcal{U}$ is not empty as well.

$$\mathcal{U} := \{\Theta_K = \Gamma^{\mathrm{a}} - \mathrm{Th}\, K \mid K \in \mathcal{K}\} - \{\emptyset\}$$

Due to the n^{th} Mints–Skura rule at $\mathcal{U}$ we now know that $x \vdash \Gamma \Rightarrow \Theta_K$ for some $K \in \mathcal{K}$. Now see that as $L \Vdash x$ and $L \Vdash \mathrm{I}\,(\mathrm{Th}\, L) \supseteq \Gamma$ we know that $L \Vdash \bigvee \Theta_K$. This ensures that $K \Vdash \bigvee \Theta_K$, whence $\mathrm{Th}\, K$ must have a non-empty intersection with $\Theta_K = \Gamma^{\mathrm{a}} - \mathrm{Th}\, K$, utter nonsense of course.

The above two lemmas immediately entail the theorem below. Iemhoff [12, Lemma 3.3] argued that any intermediate logic with the n^{th} extension property for all n must be IPC. So the theorem below re-proves the main result of Skura [27], namely that IPC is the sole intermediate logic which admits all Mints–Skura rules.

Theorem 3. *Each logic above minimal logic with the disjunction property admits the n^{th} Mints–Skura rule if and only if it has the n^{th} extension property.*

References

[1] Aczel, P.H.: Saturated Intuitionistic Theories. In: Arnold Schmidt, H., Schutte, K., Thiele, H.J. (eds.) Contributions to Mathematical Logic Proceedings of the Logic Colloquium, Hannover 1966. Studies in Logic and the Foundations of Mathematics, vol. 50, pp. 1–11. Elsevier (1968)

[2] Chagrov, A., Zakharyashchev, M.: The Disjunction Property of Intermediate Propositional Logics. Studia Logica 50(2), 189–216 (1991)

[3] Cintula, P., Metcalfe, G.: Admissible rules in the implication-negation fragment of intuitionistic logic. Annals of Pure and Applied Logic 162(2), 162–171 (2010)

[4] Citkin, A.: A note on admissible rules and the disjunction property in intermediate logics. Archive for Mathematical Logic 51, 1–14 (2012)

[5] Došen, K.: Sequent-systems and groupoid models. II. Studia Logica 48(1), 41–65 (1989)

[6] Dzik, W.: Remarks on projective unifiers. Bulletin of the Section of Logic 40(1), 37–45 (2011)

[7] Friedman, H.: One Hundred and Two Problems in Mathematical Logic. The Journal of Symbolic Logic 40(2), 113–129 (1975)

[8] Gabbay, D.M., de Jongh, D.H.: A Sequence of Decidable Finitely Axiomatizable Intermediate Logics with the Disjunction Property. The Journal of Symbolic Logic 39(1), 67–78 (1974)

[9] Ghilardi, S.: Unification through Projectivity. Journal of Logic and Computation 7(6), 733–752 (1997)

[10] Ghilardi, S.: Unification in Intuitionistic Logic. The Journal of Symbolic Logic 64(2), 859–880 (1999)

[11] Goudsmit, J.P., Iemhoff, R.: On unification and admissible rules in Gabbay-de Jongh logics. Logic Group Preprint Series 297, 1–18 (2012)

[12] Iemhoff, R.: A(nother) characterization of intuitionistic propositional logic. Annals of Pure and Applied Logic 113(1-3), 161–173 (2001); First St. Petersburg Conference on Days of Logic and Computability

[13] Iemhoff, R.: On the Admissible Rules of Intuitionistic Propositional Logic. The Journal of Symbolic Logic 66(1), 281–294 (2001)
[14] Iemhoff, R.: Intermediate Logics and Visser's Rules. Notre Dame Journal of Formal Logic 46(1), 65–81 (2005)
[15] Iemhoff, R., Metcalfe, G.: Hypersequent Systems for the Admissible Rules of Modal and Intermediate Logics. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 230–245. Springer, Heidelberg (2009)
[16] Iemhoff, R., Metcalfe, G.: Proof theory for admissible rules. Annals of Pure and Applied Logic 159(1-2), 171–186 (2009)
[17] Jeřábek, E.: Admissible Rules of Modal Logics. Journal of Logic and Computation 15(4), 411–431 (2005)
[18] Johansson, I.: Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. Compositio Mathematica 4, 119–136 (1937)
[19] Johnstone, P.T.: Stone Spaces. Cambridge studies in advanced mathematics, vol. 3. Cambridge University Press (1982)
[20] Lorenzen, P.: Einführung in die operative Logik und Mathematik. In: Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, vol. 78. Springer (1955)
[21] Maksimova, L.L.: On Maximal Intermediate Logics with the Disjunction Property. Studia Logica: An International Journal for Symbolic Logic 45(1), 69–75 (1986)
[22] Mints, G.E.: Derivability of admissible rules. Journal of Mathematical Sciences 6, 417–421 (1976)
[23] Odintsov, S., Rybakov, V.V.: Unification and Admissible rules for paraconsistent minimal Johanssons' logic J and positive intuitionistic logic IPC+. Submitted to Annals of Pure and Applied Logic (February 2012)
[24] Rozière, P.: Règles admissibles en calcul propositionnel intuitionniste. Ph.D. thesis, Université de Paris VII (1992)
[25] Rybakov, V.V.: A criterion for admissibility of rules in the model system S4 and the intuitionistic logic. Algebra and Logic 23, 369–384 (1984)
[26] Rybakov, V.V.: Admissibility of Logical Inference Rules. Studies in Logic and the Foundations of Mathematics, vol. 136. Elsevier (1997)
[27] Skura, T.: A complete syntactical characterization of the intuitionistic logic. Reports on Mathematical Logic 23, 75–80 (1989)
[28] Smoryński, C.: Applications of Kripke models. In: Troelstra, A.S. (ed.) Metamathematical Investigation of Intuitionistic Arithmetic and Analysis. Lecture Notes in Mathematics, vol. 344, pp. 324–391. Springer, Heidelberg (1973)
[29] Troelstra, A.S., van Dalen, D.: Logic. In: Constructivism in Mathematics - An Introduction. Studies in Logic and the Foundations of Mathematics, vol. 121, pp. 35–111. Elsevier (1988)

Subset Space vs Relational Semantics of Bimodal Logic: Bringing Out the Difference

Bernhard Heinemann

Faculty of Mathematics and Computer Science,
University of Hagen, 58084 Hagen, Germany
bernhard.heinemann@fernuni-hagen.de

Abstract. Correspondence theory regarding the bimodal language for subset spaces can be based on a certain pseudo-monadic second-order language arising from the relevant semantics. Since the latter language is reducible to a two-sorted language of first-order predicate logic, one can apply well-established model-theoretic techniques to studying expressivity issues. In this way, a subset space analogue to a popular definability result of ordinary modal logic is proved first in this paper. On the other hand, subset spaces can easily be related to usual Kripke models, for which we have a (one-sorted) relational first-order correspondence language. Both of the concurrent correspondents are then used in the main part of the paper, where, among other things, some Goldblatt-Thomason style results as related to subset frames are proved.

Keywords: subset spaces, topology and knowledge, correspondence theory, subset frames, definability.

1 Introduction

Modal languages count as *specification languages* in many respects, e.g., in the case of distributed or multi-agent scenarios. The *scope* of formal description of such kind is usually measured with respect to a well-known *correspondence language* encompassing properly the semantics involved. At least two facets are inherent in this setting. First, *which properties of data structures can be specified by corresponding formulas?* In basic modal logic, for example, those properties (of Kripke models, there) are, with reference to a suitable first-order language L, identified as the bisimulation-invariant fragment of L. This is the content of the famous *van Benthem Characterization Theorem;* see [4], Theorem 2.68. Second, and equally important, *which classes of data structures can be determined by formulas of the specification language?* This question as well has been answered in the case of basic modal logic; see [4], Theorem 2.75, and Theorem 3.19 (the *Goldblatt-Thomason Theorem*). The aim of this paper is to study the second topic in connection with Moss and Parikh's bimodal language for subset spaces, $\mathcal{L}$, originating from the paper [15]. (The first aspect has been treated in [14].)

One may become acquainted with Moss and Parikh's approach connecting knowledge and topology by taking a first look at the semantics of $\mathcal{L}$. The basic semantic units are composed of two ingredients, to wit, the actual state x of the world and a 'neighborhood' U of x. In knowledge-theoretic contexts, which serve us as an example here, U may be viewed as the current *epistemic state* of an agent under discussion, i.e., the set of those states that cannot be distinguished by what the agent topically knows. The two modalities of the language, K and $\Box$, quantify across all elements of U and 'downward' over all neighborhoods contained in U, respectively. Thus K captures the idea of *knowledge* as usual (see [8]), and $\Box$ reflects *effort to acquire knowledge* since gaining knowledge goes hand in hand with a shrinkage of the epistemic state. In fact, knowledge acquisition is reminiscent of a topological procedure in this way. The appropriate logic for 'real' topological spaces was first determined by Georgatos in his thesis [10]. Meanwhile, a considerable amount of work was involved in the development of the $\mathcal{L}$-based theory of subset spaces and, in particular, topological spaces; see Ch. 6 of the handbook [2] for a guide to the literature up to 2006. Summing all this and the more recent achievements up, one may state that the emerging system embodies a solid toolkit for reasoning about knowledge as well as topological reasoning, with the two-valued semantics involving both states and ambient sets making up its distinctive feature.

Modal correspondence theory should take its course via the relevant semantics in any case. The exposition in the last paragraph indicates that the right correspondent to $\mathcal{L}$ coincides with the language L_2 for so-called *weak structures* that has been introduced by Flum and Ziegler in the monograph [9]. Seeing that L_2 is a first-order language in essence, the machinery of first-order model theory (cf., e.g., [6]) should be applicable to our problem. In fact, we shall make significant use of this in due course.

The outcome of this paper can be summarized as follows. First, the $\mathcal{L}$-definable classes of subset *spaces* are characterized in model-theoretic terms. And second, some important $\mathcal{L}$-definable classes of subset *frames* are determined by certain *invariances,* i.e., validity-preserving operations on frames. Both results will, in particular, indicate the need for more expressive power.

It should be mentioned that the subjects covered here (and in [14]) have turned out to several active branches of applied logic of importance. Concerning some hints to the relevant literature, we refer the reader to the notes to Chap. 2 and 3 of the textbook [4] instead of giving a more or less exhaustive listing here. But the recent paper [5] should be quoted separately, which proves a certain sublanguage of L_2 a perfect match for the classical topological interpretation of (mono-)modal logic (see [1]) in the sense of the above questioning.

The subsequent technical part of the paper is organized as follows. In the next section, we supply the basic definitions from [7] needed later on, and we define a *standard translation* of the set of all $\mathcal{L}$-formulas which is induced by the semantics of $\mathcal{L}$. The main topic of Section 3 is definability for subset spaces. Some first-order model theory comes into play here, and the ubiquitous notion of *bisimulation,* tuned to our context appropriately, plays a key role at this stage. In Section 4, four validity-preserving operations on subset spaces emerge, which turn out to be relevant to the definability problem for subset frames. Two corresponding characterization results for classes of subset frames are obtained in Section 5. What's more, the title of this paper will become really clear there. Finally, we sum up and add some further comments. – All proofs had to be omitted here due to the restrictions regarding space.

2 The Various Languages We Consider

In this section, we first fix the language $\mathcal{L}$ for subset spaces and extract a translation into first-order logic from that. Afterwards, we recall the language L_2 before connecting $\mathcal{L}$ to it. Finally, we assign Kripke models to subset spaces and mention the appropriate correspondence language L_1.

To begin with, we define the syntax of $\mathcal{L}$. Let $\mathsf{Prop} = \{p, q, \ldots\}$ be a denumerably infinite set of symbols called *proposition variables* (which should represent the basic facts about the states of the world). Then, the set Form of all $\mathcal{L}$-*formulas* over Prop is defined by the rule $\alpha \ ::= \ p \mid \neg\alpha \mid \alpha \wedge \alpha \mid \mathsf{K}\alpha \mid \Box\alpha$. Later on, the boolean connectives that are missing here are treated as abbreviations, as needed. The dual operators of K and $\Box$ are denoted by L and $\Diamond$, respectively; K is called the *knowledge operator* and $\Box$ the *effort operator*.

We now turn to the semantics of $\mathcal{L}$. For a start, we define the relevant domains. Let $\mathcal{P}(X)$ designate the powerset of a given set X.

Definition 1 (Semantic Domains)

1. *Let X be a non-empty set (of* states*) and $\mathcal{O} \subseteq \mathcal{P}(X)$ a set of subsets of X. Then, the pair $\mathcal{S} = (X, \mathcal{O})$ is called a* (subset) frame.
2. *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame. Then the set $\mathcal{N}_\mathcal{S} := \{(x, U) \mid x \in U \text{ and } U \in \mathcal{O}\}$ is called the* set of neighborhood situations *of $\mathcal{S}$.*
3. *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame. An $\mathcal{S}$-*valuation *is a mapping $V : \mathsf{Prop} \to \mathcal{P}(X)$.*
4. *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame and V an $\mathcal{S}$-valuation. Then, $\mathcal{M} := (X, \mathcal{O}, V)$ is called a* subset space (based on $\mathcal{S}$).

The term 'neighborhood situation' has been introduced just to denominate the semantic atoms of our modal language. Note that the first component of such a situation indicates the actual state of the world while the second reflects the uncertainty of the respective agent about it. Note also that $\mathcal{S}$-valuations only depend on states (and are independent of sets thus). This is in accordance with the common practice of modelling; cf. [8].

For a given subset space $\mathcal{M}$, we now define the relation of *satisfaction,* $\models_\mathcal{M}$, between neighborhood situations of the underlying frame and formulas from Form. Based on that, we define the notion of *validity* of $\mathcal{L}$-formulas in subset spaces and in subset frames. In the following, neighborhood situations are often written without parentheses.

Definition 2 (Satisfaction and Validity). *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame.*

1. *Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space based on $\mathcal{S}$, and let $x, U \in \mathcal{N}_\mathcal{S}$ be a neighborhood situation. Then*

$$\begin{array}{lcl} x, U \models_\mathcal{M} p & :\Longleftrightarrow & x \in V(p) \\ x, U \models_\mathcal{M} \neg\alpha & :\Longleftrightarrow & x, U \not\models_\mathcal{M} \alpha \\ x, U \models_\mathcal{M} \alpha \wedge \beta & :\Longleftrightarrow & x, U \models_\mathcal{M} \alpha \text{ and } x, U \models_\mathcal{M} \beta \\ x, U \models_\mathcal{M} \mathsf{K}\alpha & :\Longleftrightarrow & \forall\, y \in U : y, U \models_\mathcal{M} \alpha \\ x, U \models_\mathcal{M} \Box\alpha & :\Longleftrightarrow & \forall\, U' \in \mathcal{O} : [x \in U' \subseteq U \Rightarrow x, U' \models_\mathcal{M} \alpha]\,, \end{array}$$

where $p \in$ Prop and $\alpha, \beta \in$ Form. In case $x, U \models_\mathcal{M} \alpha$ is true we say that α holds in $\mathcal{M}$ at *the neighborhood situation x, U.*
2. *Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space based on $\mathcal{S}$. An $\mathcal{L}$-formula α is called* valid in $\mathcal{M}$ *iff it holds in $\mathcal{M}$ at every neighborhood situation of $\mathcal{S}$.*
3. *An $\mathcal{L}$-formula α is called* valid in $\mathcal{S}$ *iff it is valid in every subset space $\mathcal{M}$ based on $\mathcal{S}$; in this case, we write $\mathcal{S} \models \alpha$.*

Note that the idea of knowledge and effort described in the introduction is made precise by Item 1 of this definition. In particular, knowledge *is defined* as validity at all states that are indistinguishable to the agent; cf. [8]. – Obviously, subset spaces are on the same level of language as are Kripke models in normal modal logic (while subset frames correspond to Kripke frames).

The defining clauses on the right-hand side of the double arrows in Definition 2.1, give rise to a translation of $\mathcal{L}$-formulas into a two-sorted language of predicate logic. The target language should obviously contain state variables $v, v', \ldots$, set variables $\Upsilon, \Upsilon', \ldots$, and a binary relation symbol ε of the sort $(\mathit{state}, \mathit{set})$. (A binary relation symbol of the sort $(\mathit{set}, \mathit{set})$ is superfluous since inclusion of sets is definable.) Moreover, a unary predicate symbol P_p of the sort (state) is associated with every proposition variable $p \in$ Prop. Then, after fixing a state variable v, p is translated to $P_p(v)$, and this assignment is extended to a function on the set Form of all $\mathcal{L}$-formulas in a way described in a minute.

Before doing so, the language L_2 from [9], Part I, § 1 is recapitulated. The set At of all *atomic L_2-formulas* consists of all equations $v = v'$, where v, v' are state variables, all predicate expressions $P_p(v)$, where p is a proposition variable and v a state variable, and all membership expressions $v \,\varepsilon\, \Upsilon$, where v is as above and Υ a set variable. More complex formulas are obtained from At through (iterated) negation, conjunction, and universal quantification '$\forall\, v.$' and '$\forall\, \Upsilon.$' over state and set variables, respectively. (All the other common connectives and quantifiers are again treated as abbreviations.) Let Fml_2 be the resulting set of formulas. With that, the standard translation announced above is given in the next definition.

Definition 3 (Standard Translation). *Let v be a state variable and Υ a set variable. Then, a mapping $ST_{v,\Upsilon} :$ Form $\to \mathsf{Fml}_2$ is defined recursively by*

$$\begin{array}{lcl} ST_{v,\Upsilon}(p) & := & P_p(v) \\ ST_{v,\Upsilon}(\neg\alpha) & := & \neg ST_{v,\Upsilon}(\alpha) \\ ST_{v,\Upsilon}(\alpha \wedge \beta) & := & ST_{v,\Upsilon}(\alpha) \wedge ST_{v,\Upsilon}(\beta) \\ ST_{v,\Upsilon}(\mathsf{K}\alpha) & := & \forall\, v'.\,(v' \,\varepsilon\, \Upsilon \to ST_{v',\Upsilon}(\alpha)) \\ ST_{v,\Upsilon}(\Box\alpha) & := & \forall\, \Upsilon'.\,(v \,\varepsilon\, \Upsilon' \wedge \forall\, v.\,(v \,\varepsilon\, \Upsilon' \to v \,\varepsilon\, \Upsilon) \to ST_{v,\Upsilon'}(\alpha))\,, \end{array}$$

where $p \in \mathsf{Prop}$, $\alpha, \beta \in \mathsf{Form}$, and v', Υ' are variables that have not been used so far in applying ST.

In the following lemma, the intimate correlation between modal and first-order satisfaction is delineated. For that, note that any subset space $\mathcal{M} = (X, \mathcal{O}, V)$ induces an L_2-structure $\mathcal{M}' = \big(X, \mathcal{O}, \{V_{P_p} \mid p \in \mathsf{Prop}\}\big)$, where P_p is interpreted with $V(p)$: $V_{P_p} = V(p)$, for every $p \in \mathsf{Prop}$. (L_2-structures are called *weak structures* in [9].) Therefore, we may identify $\mathcal{M}$ and $\mathcal{M}'$, and we shall do so after the lemma. – A sort-respecting function mapping variables to values is called a *look-up table.*

Lemma 1. *Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space based on $\mathcal{S} = (X, \mathcal{O})$. Then we have, for all formulas $\alpha \in \mathsf{Form}$, neighborhood situations $x, U \in \mathcal{N}_\mathcal{S}$, and look-up tables $\mathfrak{b}$ evaluating v to x and Υ to U, $[x, U \models_\mathcal{M} \alpha \iff \mathcal{M}' \models ST_{v,\Upsilon}(\alpha)\,(\mathfrak{b})]$.*

One might want to look at subset spaces from a different point of view. Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame and $\mathcal{M} = (X, \mathcal{O}, V)$ a subset space based on it. Take $S := \mathcal{N}_\mathcal{S}$ as a set of worlds, and define two accessibility relations R (belonging to K) and R' (belonging to $\Box$) on S by $[(x, U)\, R\, (x', U') :\iff U = U']$ and $[(x, U)\, R'\, (x', U') :\iff (x = x' \wedge U' \subseteq U)]$, for all $(x, U), (x', U') \in S$. The emerging Kripke frame $F_\mathcal{S} := (S, \{R, R'\})$ is called *the frame induced by $\mathcal{S}$*. For every $p \in \mathsf{Prop}$, let $V'(p) := \{x, U \in S \mid x \in V(p)\}$. Then, a bimodal Kripke model $M_\mathcal{S} := (S, \{R, R'\}, V')$ results, which is equivalent to $\mathcal{M}$ in the following sense. For all $\alpha \in \mathsf{Form}$ and $x, U \in \mathcal{N}_\mathcal{S} = S$, formula α holds in $\mathcal{M}$ at the neighborhood situation x, U iff α is valid in $M_\mathcal{S}$ at the world (x, U). Note that the first-order correspondence language arising in this case, L_1, is evidently different from L_2 since it is one-sorted and contains two additional binary relation symbols (representing R and R'). Thus, although we shall have a notion of invariance of formulas in each case (see Section 4 below) and can relate the respective classes of models in a sense, we cannot switch the particular correspondents offhand. This is the first point where the present paper meets up with its title. – We consider L_2 the right language into which formulas from Form should be translated, due to the subset space semantics. Nevertheless, the traditional bimodal point of view will turn out to be helpful here and there.

3 Definability for Subset Spaces

One of the crucial notions we need for the definability result claimed in this section is that of *bisimulation.* As is known, the idea of *invariance of formulas* is captured by this concept in many cases, not least with respect to the most common semantics of modal logic; cf. [4], Ch. 2.2. Bisimulations for subset spaces were introduced in the paper [3]. For convenience, we repeat the definition and state a related lemma. Then, after introducing some notation, we get on to *saturated models,* which as well are crucially applied here.

Definition 4 (Subset Space Bisimulation). *For $i = 1, 2$, let $\mathcal{S}_i = (X_i, \mathcal{O}_i)$ be subset frames and $\mathcal{M}_i = (X_i, \mathcal{O}_i, V_i)$ subset spaces based on $\mathcal{S}_i$. Moreover, let*

$R \subseteq \mathcal{N}_{\mathcal{S}_1} \times \mathcal{N}_{\mathcal{S}_2}$ *be a non-empty binary relation. Then* R *is called a* subset space bisimulation (between $\mathcal{M}_1$ and $\mathcal{M}_2$) *iff the following five conditions are satisfied whenever* $(x_1, U_1)\, R\, (x_2, U_2)$ *is valid for any* $(x_1, U_1) \in \mathcal{N}_{\mathcal{S}_1}$ *and* $(x_2, U_2) \in \mathcal{N}_{\mathcal{S}_2}$*:*

1. *For all* $p \in \mathsf{Prop}$, $\big[x_1 \in V_1(p) \iff x_2 \in V_2(p)\big]$.
2. *For all* $x_1' \in U_1$ *there exists* $x_2' \in U_2$ *such that* $(x_1', U_1)\, R\, (x_2', U_2)$.
3. *For all* $x_2' \in U_2$ *there exists* $x_1' \in U_1$ *such that* $(x_1', U_1)\, R\, (x_2', U_2)$.
4. *If* $x_1 \in U_1'$ *for any* $U_1' \in \mathcal{O}_1$ *such that* $U_1' \subseteq U_1$*, then there exists* $U_2' \in \mathcal{O}_2$ *satisfying* $x_2 \in U_2' \subseteq U_2$ *and* $(x_1, U_1')\, R\, (x_2, U_2')$.
5. *If* $x_2 \in U_2'$ *for any* $U_2' \in \mathcal{O}_2$ *such that* $U_2' \subseteq U_2$*, then there exists* $U_1' \in \mathcal{O}_1$ *satisfying* $x_1 \in U_1' \subseteq U_1$ *and* $(x_1, U_1')\, R\, (x_2, U_2')$.

Due to the presence of two modalities, K and $\Box$, we have two so-called *forth conditions* as well as two *back conditions* here, given by 4.2, 4.4 and 4.3, 4.5, respectively. Note that the notion of bisimulation considered in Sect. 2.3 of the paper [7] is translated to subset spaces by Definition 4. There, a class of bimodal Kripke models called *cross axiom models* is concerned, which includes those considered at the end of the previous section. – The next lemma shows that subset space bisimulations really entail the invariance of $\mathcal{L}$-formulas.

Lemma 2 (Subset Space Invariance of Formulas). *For* $i = 1, 2$*, let* $\mathcal{M}_i = (X_i, \mathcal{O}_i, V_i)$ *be subset spaces based on* $\mathcal{S}_i = (X_i, \mathcal{O}_i)$*, and let* $R \subseteq \mathcal{N}_{\mathcal{S}_1} \times \mathcal{N}_{\mathcal{S}_2}$ *be a subset space bisimulation. Furthermore, assume that* $(x_1, U_1) \in \mathcal{N}_{\mathcal{S}_1}$ *and* $(x_2, U_2) \in \mathcal{N}_{\mathcal{S}_2}$ *satisfy* $(x_1, U_1)\, R\, (x_2, U_2)$*. Then, for all* $\mathcal{L}$*-formulas* $\alpha \in \mathsf{Form}$*, we have* $[x_1, U_1 \models_{\mathcal{M}_1} \alpha \iff x_2, U_2 \models_{\mathcal{M}_2} \alpha]$.

Let i, $\mathcal{S}_i$, $\mathcal{M}_i$, and (x_i, U_i), be as above. We stipulate some notations that follow analogous ones from ordinary modal logic. A triple like $(\mathcal{M}_1, x_1, U_1)$ is called a *situated subset space.* Two situated subset spaces $(\mathcal{M}_1, x_1, U_1)$ and $(\mathcal{M}_2, x_2, U_2)$ are called *modally equivalent* iff, for all $\alpha \in \mathsf{Form}$, it is true that $[x_1, U_1 \models_{\mathcal{M}_1} \alpha \iff x_2, U_2 \models_{\mathcal{M}_2} \alpha]$; we write $(\mathcal{M}_1, x_1, U_1) \leftrightsquigarrow (\mathcal{M}_2, x_2, U_2)$ in this case. Moreover, if there exists a subset space bisimulation R between $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $(x_1, U_1)\, R\, (x_2, U_2)$, then we write $(\mathcal{M}_1, x_1, U_1) \underline{\leftrightarrow} (\mathcal{M}_2, x_2, U_2)$. With that, Lemma 2 can be reformulated as follows.

Lemma 3. *For* $i = 1, 2$*, let* $\mathcal{M}_i = (X_i, \mathcal{O}_i, V_i)$ *be situated subset spaces. Then we have that* $(\mathcal{M}_1, x_1, U_1) \leftrightsquigarrow (\mathcal{M}_2, x_2, U_2)$ *whenever* $(\mathcal{M}_1, x_1, U_1) \underline{\leftrightarrow} (\mathcal{M}_2, x_2, U_2)$.

A class of models satisfying the opposite assertion, i.e., that (modal) equivalence implies bisimilarity, is usually called a *Hennessy-Milner class;* see [4], 2.52. Identifying a suitable Hennessy-Milner class of subset spaces, which will be supported by first-order model theory, is a key intermediate step towards the main theorem of this section.

For this purpose, note again that the L_2-logic of subset spaces is reducible to a two-sorted first-order logic L'' based on the same signature. This is to be made a little more precise now. First, we observe that a subset space $\mathcal{M} = (X, \mathcal{O}, V)$ is equivalent to a two-sorted first-order structure $\mathcal{M}' = ((X, \mathcal{O}), V_\varepsilon, V)$, where $(X, \mathcal{O})$ is the pair of carrier sets of the respective sorts, and V_ε indicates the canonical interpretation of the relation symbol 'ε'; moreover, 'equivalent' means that for every formula $\phi \in \mathsf{Fml}_2$ having, for some $n \in \mathbb{N}$, its free (state or set) variables among those from the set $\{\chi_1, \ldots, \chi_n\}$, and for all look-up tables $\mathfrak{b}$, it is true that $[\mathcal{M} \models \phi[\chi_1, \ldots, \chi_n]\,(\mathfrak{b}) \iff \mathcal{M}' \models \phi[\chi_1, \ldots, \chi_n]\,(\mathfrak{b})]$. And vice versa, a two-sorted structure $\mathcal{M} = ((X, Y), V_\varepsilon, V)$ with V_ε interpreting 'ε' somehow, gives rise to an equivalent subset space $\mathcal{M}' = (X, \mathcal{O}', V)$ by letting $\mathcal{O}' := \{U' \mid U \in Y\}$, where, for given state and set variables v respectively Υ, $U' := \{\mathfrak{b}(v) \in X \mid \mathfrak{b}$ a look-up table satisfying $\Upsilon \mapsto U$ and $\mathcal{M} \models v\,\varepsilon\,\Upsilon\,(\mathfrak{b})\}$.

Now, *ultraproducts* (and *ultrapowers*) of subset spaces can be obtained by making a detour via the two-sorted first-order models associated with subset spaces according to this reduction, in the following way. Let $I \neq \emptyset$ be an index set and $(\mathcal{M}_\iota)_{\iota \in I} = (X_\iota, \mathcal{O}_\iota, V_\iota)_{\iota \in I}$ a family of subset spaces. Moreover, let $(\mathcal{M}'_\iota)_{\iota \in I} = ((X_\iota, \mathcal{O}_\iota), (V_\varepsilon)_\iota, V_\iota)_{\iota \in I}$ be the family of induced L''-equivalents. Finally, let $\prod_{\iota \in I}(X_\iota, \mathcal{O}_\iota) := \{f : I \to \bigcup_{\iota \in I}(X_\iota \cup \mathcal{O}_\iota) \mid f(\iota) \in X_\iota \cup \mathcal{O}_\iota$ for all $\iota \in I\}$, and let $\mathfrak{U}$ be an ultrafilter over I. Take the usual notion of $\mathfrak{U}$-equivalence of functions from $\prod_{\iota \in I}(X_\iota, \mathcal{O}_\iota)$ (i.e., $f \sim_{\mathfrak{U}} g :\iff \{\iota \in I \mid f(\iota) = g(\iota)\} \in \mathfrak{U}$), and let $\prod_{\mathfrak{U}}(X_\iota, \mathcal{O}_\iota) := \{f_{\mathfrak{U}} \mid f \in \prod_{\iota \in I}(X_\iota, \mathcal{O}_\iota)\}$ be the set of all corresponding equivalence classes. $\prod_{\mathfrak{U}}(X_\iota, \mathcal{O}_\iota)$ should serve us as the carrier set of an appropriate L''-structure. Thus, we must identify X^* and $\mathcal{O}^*$, the intended sets of states and 'encoded opens' of a subset space $\mathcal{M}^*$, within $\prod_{\mathfrak{U}}(X_\iota, \mathcal{O}_\iota)$. Clearly, we define $\left[f_{\mathfrak{U}} \in X^* :\iff \{\iota \mid f(\iota) \in X_\iota\} \in \mathfrak{U}\right]$ and $\left[f_{\mathfrak{U}} \in \mathcal{O}^* :\iff \{\iota \mid f(\iota) \in \mathcal{O}_\iota\} \in \mathfrak{U}\right]$, for all $f_{\mathfrak{U}} \in \prod_{\mathfrak{U}}(X_\iota, \mathcal{O}_\iota)$. This assignment makes perfect sense, since we have $\left[f_{\mathfrak{U}} \notin X^* \iff \{\iota \mid f(\iota) \in X_\iota\} \notin \mathfrak{U} \iff \{\iota \mid f(\iota) \notin X_\iota\} \in \mathfrak{U} \iff \{\iota \mid f(\iota) \in \mathcal{O}_\iota\} \in \mathfrak{U} \iff f_{\mathfrak{U}} \in \mathcal{O}^*\right]$ due to the closure of ultrafilters under complements; in other words, variables can be interpreted in a sort-respecting way. As the definitions of the relation V_ε^* and the mapping V^* are as usual, we finally arrive at a subset space $(\mathcal{M}^*)' = (X^*, (\mathcal{O}^*)', V^*)$ via the structure $\mathcal{M}^* = ((X^*, \mathcal{O}^*), V_\varepsilon^*, V^*)$. We call $(\mathcal{M}^*)'$ the *ultraproduct* (*modulo* $\mathfrak{U}$) of the family $(\mathcal{M}_\iota)_{\iota \in I} = (X_\iota, \mathcal{O}_\iota, V_\iota)_{\iota \in I}$. In case the factors $\mathcal{M}_\iota$ coincide for all $\iota \in I$, we speak of an *ultrapower*. Note that these notions are straightforwardly extendable to situated subset spaces. – Below, we omit inverted commas when designating ultraproducts of given subset spaces.

We need a number of facts about ultraproducts and ultrapowers of subset spaces. All these are straightforward adjustments of basic model-theoretic results of (one-sorted) first-order logic to the present context. Their validity can easily be seen from the correspondences just established. First, the well-known *Theorem of Łoś* holds for subset spaces as well. Second, subset spaces are *elementarily embeddable* into ultrapowers of themselves by means of the diagonal mapping d; we write $d : \mathcal{M} \preccurlyeq \mathcal{M}^*$ in this case, as usual.

At this point, we should say a few words about related neighborhood situations in case of ultrapowers. Let $\mathcal{M} = (X, \mathcal{O}, V)$ be given. Then, $d : \mathcal{M} \preccurlyeq \mathcal{M}^* = (X^*, \mathcal{O}^*, V^*)$ means, in particular, that $\mathcal{O}^*$ contains the set $\{U^* \mid U \in \mathcal{O}\}$, where each U^* is obtained from the original U via the detour described above, thus satisfies $d\,[U] = U^* \cap d\,[X]$. Regarding satisfaction of formulas, the neighborhood situation x^*, U^* is the right image of a given situation x, U under the elementary embedding therefore, where $x^* = d(x)$. This applies, in particular, to formulas from Form so that we get $(\mathcal{M}, x, U) \leftrightsquigarrow (\mathcal{M}^*, x^*, U^*)$.

In order to state the third of the issues needed subsequently, we now introduce ω-saturated models. We follow the usual manner of speaking in doing so. Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space. Furthermore, let $A \subseteq X$ and $\mathcal{Q} \subseteq \mathcal{O}$ be any subsets of X and $\mathcal{O}$, respectively, and let $L_2[A, \mathcal{Q}]$ be the language obtained by extending L_2 with new constants $\underline{a}$ for all elements $a \in A$ and $\underline{U}$ for all members $U \in \mathcal{Q}$. The correspondingly expanded structure derived from $\mathcal{M}$ is denoted by $\mathcal{M}_{A,\mathcal{Q}}$. Finally, a set $\Phi(\chi) \subseteq \mathsf{Fml}_2$ of L_2-formulas having at most the (state or set) variable χ occurring free, is called a *type*. Then, $\mathcal{M}$ is called *ω-saturated* iff, for every pair $(A, \mathcal{Q})$ of *finite* sets $A \subseteq X$ and $\mathcal{Q} \subseteq \mathcal{O}$, and every type $\Phi(\chi)$ that is consistent with the L_2-theory of $\mathcal{M}_{A,\mathcal{Q}}$, there exists a *realization* of $\Phi(\chi)$ *in* $\mathcal{M}_{A,\mathcal{Q}}$, i.e., an element $x \in X$ or a subset $U \in \mathcal{O}$ (depending on the sort of χ) such that $\mathcal{M}_{A,\mathcal{Q}} \models \phi(\mathfrak{b})$ for all $\phi \in \Phi$ and all look-up tables $\mathfrak{b}$ mapping χ to x respectively U. – It can be shown that ω-saturated ultrapowers of subset spaces really exist.

Lemma 4. *Let $\mathcal{M}$ be a subset space. Then there exists an ω-saturated ultrapower $\mathcal{M}^*$ of $\mathcal{M}$.*

For a proof in the case of ordinary first-order logic (which uses Łoś's Theorem, among other things), see [6], Theorem 6.1.1. – Finally, we obtain the following analogue to [4], Theorem 2.74.

Theorem 1. *Two situated subset spaces $(\mathcal{M}_1, x_1, U_1)$ and $(\mathcal{M}_2, x_2, U_2)$ are modally equivalent iff they have bisimilar ultrapowers.*

So far, we have provided all the facts that are needed for proving the first of the main results of this paper, which is stated immediately after the next definition.

Definition 5 (Definable Classes of Subset Spaces). *Let $\mathfrak{K}$ be a class of situated subset spaces.*

1. *$\mathfrak{K}$ is called* closed under bisimulations *iff $(\mathcal{M}_1, x_1, U_1) \underline{\leftrightarrow} (\mathcal{M}_2, x_2, U_2)$ and $(\mathcal{M}_1, x_1, U_1) \in \mathfrak{K}$ implies $(\mathcal{M}_2, x_2, U_2) \in \mathfrak{K}$.*
2. *$\mathfrak{K}$ is called* closed under ultraproducts *iff, for every family $(\mathcal{M}_\iota, x_\iota, U_\iota)_{\iota \in I}$ of situated subset spaces such that $(\mathcal{M}_\iota, x_\iota, U_\iota) \in \mathfrak{K}$ for all $\iota \in I$, it follows that any ultraproduct $\prod_{\mathfrak{U}} (\mathcal{M}_\iota, x_\iota, U_\iota)$ of $(\mathcal{M}_\iota, x_\iota, U_\iota)_{\iota \in I}$ is in $\mathfrak{K}$.*
3. *$\mathfrak{K}$ is called* $\mathcal{L}$-definable *iff there exists a set $\Sigma \subseteq \mathsf{Form}$ of $\mathcal{L}$-formulas such that, for any situated subset space $(\mathcal{M}, x, U)$, the following is true:*

$$(\mathcal{M}, x, U) \in \mathfrak{K} \iff x, U \models_{\mathcal{M}} \alpha \text{ for all } \alpha \in \Sigma.$$

The following result characterizing $\mathcal{L}$-definable classes of subset spaces should be compared with Theorem 2.75 of the textbook [4].

Theorem 2. *A class $\mathfrak{K}$ of situated subset spaces is $\mathcal{L}$-definable, iff it is closed under both bisimulations and ultraproducts, and the complementary class $\overline{\mathfrak{K}}$ is closed under ultrapowers.*

Clearly, we have a result on definability by single formulas, too, as a counterpart to [4], Theorem 2.76 (not formulated here). We shall get back to these findings in the final section of this paper.

4 Definability for Subset Frames: Necessary Conditions

Turning now to definability questions for classes of subset *frames,* we consider the following validity preserving operations: *adding isolated points,* and *taking special subframes, disjoint unions,* and *images under bounded morphisms.* We first exemplify how these principles work. Then, we introduce *modally definable classes of subset frames* and prove that all such classes are closed under applying any of these operations to arbitrary members. Finally in this section, we give some examples of definable and non-definable classes of subset frames. – For a start, we define *subframes* of subset frames.

Definition 6 (Subframes). *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame. Then, a subset frame $\mathcal{S}' = (X', \mathcal{O}')$ is called a* subframe *of $\mathcal{S}$, iff $X' \subseteq X$ and $\mathcal{O}' = \{U' \in \mathcal{O} \mid U' \subseteq X'\}$.*

Note that $\mathcal{S}$ is a subframe of itself. – We have a very natural lemma.

Lemma 5. *Let $\mathcal{M} = (X, \mathcal{O}, V)$ be a subset space based on $\mathcal{S} = (X, \mathcal{O})$, $\mathcal{S}' = (X', \mathcal{O}')$ a subframe of $\mathcal{S}$, $V' : \mathsf{Prop} \to \mathcal{P}(X')$ defined by $V'(p) := V(p) \cap X'$ for all $p \in \mathsf{Prop}$, and $\mathcal{M}' := (X', \mathcal{O}', V')$. Then we have, for all formulas $\alpha \in \mathsf{Form}$ and neighborhood situations $x, U \in \mathcal{N}_{\mathcal{S}'}$, $[x, U \models_{\mathcal{M}'} \alpha \iff x, U \models_{\mathcal{M}} \alpha]$.*

A point of a subset frame $\mathcal{S} = (X, \mathcal{O})$ is called *isolated,* iff it is not contained in any $U \in \mathcal{O}$. Letting $\mathcal{I}_{\mathcal{S}}$ be the set of all isolated points of $\mathcal{S}$, Lemma 5 applies, in particular, to the subset frame $\mathcal{S}' := (X \setminus \mathcal{I}_{\mathcal{S}}, \mathcal{O})$. If, on the other hand, Y is a set of points that is disjoint from X, then the frame $\mathcal{S}'' = (X \cup Y, \mathcal{O})$ is said to be obtained from $\mathcal{S}$ by *adding isolated points.* – The counterparts to the *generated subframes* of ordinary modal logic (see, e.g., [4], Def. 3.13) are called *special subframes* here.

Definition 7 (Special Subframes). *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame. A subframe $\mathcal{S}' = (X', \mathcal{O}')$ of $\mathcal{S}$ is called* special, *iff $X' = U$ for some non-empty set $U \in \mathcal{O}$.*

Obviously, $\mathcal{S}$ (considered as a subframe of itself) is special iff $X \in \mathcal{O}$. In this case, we call the frame $\mathcal{S}$ itself *special.* Note that special subframes are, in the bimodal Kripke view introduced in Section 2, exactly those which are generated by a situation (x, U) in the usual sense.

We now address disjoint unions. The proceeding is similar in this case, as the definition is given first and a relevant lemma afterwards.

Definition 8 (Disjoint Unions). *Let I be an index set, and let $(\mathcal{S}_\iota)_{\iota \in I} = (X_\iota, \mathcal{O}_\iota)_{\iota \in I}$ be a* family of disjoint subset frames; *i.e., for all $\kappa, \lambda \in I$ we have $X_\kappa \cap X_\lambda = \emptyset$ whenever $\kappa \neq \lambda$. Then, the frame $\biguplus_{\iota \in I} \mathcal{S}_\iota := \left(\bigcup_{\iota \in I} X_\iota, \bigcup_{\iota \in I} \mathcal{O}_\iota\right)$ is called the* disjoint union *of the family $(\mathcal{S}_\iota)_{\iota \in I}$.*

Lemma 6. *Let $(\mathcal{S}_\iota)_{\iota \in I} = (X_\iota, \mathcal{O}_\iota)_{\iota \in I}$ be a family of disjoint subset frames. Furthermore, let $\mathcal{M}_\iota = (X_\iota, \mathcal{O}_\iota, V_\iota)$ be a subset space based on $\mathcal{S}_\iota$, for every $\iota \in I$, and let $V : \mathsf{Prop} \to \mathcal{P}\left(\bigcup_{\iota \in I} X_\iota\right)$ be defined by $V(p) := \bigcup_{\iota \in I} V_\iota(p)$ for all $p \in \mathsf{Prop}$. Finally, let $\biguplus_{\iota \in I} \mathcal{M}_\iota := \left(\bigcup_{\iota \in I} X_\iota, \bigcup_{\iota \in I} \mathcal{O}_\iota, V\right)$. Then we have, for all formulas $\alpha \in \mathsf{Form}$, indices $\iota \in I$, and neighborhood situations $x, U \in \mathcal{N}_{\mathcal{S}_\iota}$, $\left[x, U \models_{\mathcal{M}_\iota} \alpha \iff x, U \models_{\biguplus_{\iota \in I} \mathcal{M}_\iota} \alpha\right]$.*

The fourth topic to be investigated is images under bounded morphisms. The standard notion (see [4], 2.10) is adapted to subset frames in the following way.

Definition 9 (Bounded (Subset Frame) Morphisms). *For $i = 1, 2$, let $\mathcal{S}_i = (X_i, \mathcal{O}_i)$ be subset frames, and let $f : X_1 \to X_2$ be a mapping. Then f is called a* bounded (subset frame) morphism, *iff, for all $x \in X_1$ and $U_1 \in \mathcal{O}_1$ such that $x \in U_1$, and for every $U_2 \in \mathcal{O}_2$, we have*

1. *$f[U_1] \in \mathcal{O}_2$, and*
2. *if $f(x) \in U_2 \subseteq f[U_1]$, then there exists $U' \in \mathcal{O}_1$ such that $x \in U' \subseteq U_1$ and $f[U'] = U_2$.*

In this case, we write $f : \mathcal{S}_1 \to \mathcal{S}_2$; and we write $f : \mathcal{S}_1 \twoheadrightarrow \mathcal{S}_2$ if f is additionally surjective.

Note that the first condition of Definition 9 defines the *open* mappings between $\mathcal{S}_1$ and $\mathcal{S}_2$ in case of topological spaces with $\mathcal{O}_i$ being the respective sets of opens ($i = 1, 2$). While the *continuity* of f is required instead of 9.2 in that case, see [5], this must be modified here since we do not generally have the subset systems closed under intersections. – The semantic lemma associated with bounded subset frame morphisms reads as follows.

Lemma 7. *Let $\mathcal{S}_i = (X_i, \mathcal{O}_i)$ be subset frames and $\mathcal{M}_i = (X_i, \mathcal{O}_i, V_i)$ subset spaces based on $\mathcal{S}_i$, where $i = 1, 2$. Moreover, let $f : \mathcal{S}_1 \to \mathcal{S}_2$ be a bounded morphism satisfying $[x \in V_1(p) \iff f(x) \in V_2(p)]$ for all $p \in \mathsf{Prop}$. Then we have, for all formulas $\alpha \in \mathsf{Form}$ and neighborhood situations $x, U \in \mathcal{N}_{\mathcal{S}_1}$, $[x, U \models_{\mathcal{M}_1} \alpha \iff f(x), f[U] \models_{\mathcal{M}_2} \alpha]$.*

Connecting the notions fixed so far in this section, we show that every subset frame without isolated points is the image of the disjoint union of (isomorphic copies of) all its special subframes under a bounded morphism.

Lemma 8. *Let $\mathcal{S} = (X, \mathcal{O})$ be a subset frame without isolated points. For every $U \in I := \mathcal{O} \setminus \{\emptyset\}$, let $\mathcal{S}_U := (X_U, \mathcal{O}_U)$, where $X_U := \{(x, U) \mid x \in U\}$ and $\mathcal{O}_U := \{\{(x, U') \mid x \in U'\} \mid U' \in \mathcal{O}, U' \subseteq U\}$. Then, $(\mathcal{S}_U)_{U \in I}$ is a family of disjoint subset frames. Moreover, the mapping $f : \bigcup_{U \in I} X_U \to X$ defined by projecting onto the first component is a surjective bounded morphism, $f : \biguplus_{U \in I} \mathcal{S}_U \twoheadrightarrow \mathcal{S}$.*

Another simple application of bounded morphisms is that we may *forget the empty subset* without semantic loss, which is to be made precise as follows. Let a subset frame $\mathcal{S} = (X, \mathcal{O})$ be given. Consider the frame $\mathcal{S}' := (X, \mathcal{O} \setminus \{\emptyset\})$.

Then $\mathcal{S}'$ is a bounded morphic image of $\mathcal{S}$ (and vice versa), namely by means of the identical mapping. Now, apply Lemma 7.

In the following, we consider *$\mathcal{L}$-definable classes of subset frames.* The corresponding definition is similar to Definition 5.3.

Definition 10 (Definable Classes of Subset Frames). *Let $\mathfrak{K}$ be a class of subset frames. $\mathfrak{K}$ is called $\mathcal{L}$-*definable *iff there exists a set of $\mathcal{L}$-formulas $\Sigma \subseteq \mathsf{Form}$ such that, for any subset frame $\mathcal{S}$, $[\mathcal{S} \in \mathfrak{K} \iff \mathcal{S} \models \alpha$ for all $\alpha \in \Sigma]$.*

Having Definition 5.1 and 5.2 in mind, it should be clear what is meant when we say that $\mathfrak{K}$ is *closed under adding isolated points* or *under taking special subframes, disjoint unions,* or *bounded morphic images,* so that it is no more necessary to give an explicit definition here. Instead, we may proceed to the following theorem directly.

Theorem 3. *If a class $\mathfrak{K}$ of subset frames is $\mathcal{L}$-definable, then it is closed under adding isolated points, and under taking special subframes, disjoint unions, and bounded morphic images.*

It follows from Theorem 3 that, respectively, various classes and many L_2-definable properties of subset frames $(X, \mathcal{O})$ are not modally definable. We give some examples.

- The *non-existence of isolated points* $(\forall\, x \in X \,.\, \exists\, U \in \mathcal{O} \,.\, x \in U)$. Clearly, this property cannot be $\mathcal{L}$-definable because of the first closure property of a definable class of subset frames.
- The class of all *topological spaces.* This class is not closed under forgetting the empty subset (which implies the non-closure under taking bounded morphic images; see above).
- The class of all *formally connected* subset frames $(\exists\, U, U' \in \mathcal{O} \,.\, (U \cap U' = \emptyset$ and $U \cup U' = X))$. One can easily construct a formally connected subset frame having special subframes which fail to share this property.
- *Dichotomy* $(\forall\, U, U' \in \mathcal{O} \,.\, (U \subseteq U'$ or $U' \subseteq U))$. This property is obviously violated by forming disjoint unions.
- The class of all *weakly separated* subset frames $(\forall\, x, y \in X \,.\, (x \neq y \Rightarrow \exists\, U \in \mathcal{O} \,.\, (x \in U$ and $y \notin U)))$. A counterexample is yielded by the three-element frame $(\{x, y, z\}, \{\{x, z\}, \{y, z\}, \{x\}, \{y\}\})$ and its bounded morphic image $(\{u, v\}, \{\{u, v\}, \{u\}\})$, where the bounded morphism is mediated by $x, y \mapsto u$ and $z \mapsto v$.

But there are positive examples as well. Trivially, the class of *all* subset frames is definable, viz by its logic. We add two further simple examples. Call a subset frame $(X, \mathcal{O})$ *quasi-discrete* iff every $U \in \mathcal{O}$ is a singleton. The class of all quasi-discrete subset frames is defined by the formula schema $\alpha \to \mathsf{K}\alpha$. And call a subset frame $(X, \mathcal{O})$ *flat* iff any two distinct non-empty $U, U' \in \mathcal{O}$ are incomparable with respect to inclusion. This class is defined by the schema $\alpha \to \Box\alpha$. – All this indicates that there is substantial need for extending the expressive power of $\mathcal{L}$; see the discussion in Section 6 below.

5 Definability for Subset Frames: Sufficient Conditions

The aim of this section is to show that, altogether, the four closure properties of classes of frames considered in the previous section are not only necessary, but also sufficient for modal definability in some cases. First, we consider *finite* frames. To this end, we should be able to find appropriate *Jankov-Fine formulas;* cf. [4], Sect. 3.4.

Let a finite special subset frame $\mathcal{S} = (X, \mathcal{O})$ satisfying $\emptyset \notin \mathcal{O}$ be given. Suppose that $X = \{x_0, \ldots, x_n\}$ and $\mathcal{O} = \{U_0, \ldots, U_m\}$, where $U_0 = X$. Then, the Jankov-Fine formula $\alpha_{\mathcal{S}}$ *associated with* $\mathcal{S}$ is the conjunction of the following five formulas respectively schemata involving pairwise distinct proposition variables $p_0, \ldots, p_n \in \mathsf{Prop}$.

1. $\mathsf{K}\,(p_0 \vee \cdots \vee p_n)$
2. $\mathsf{K}\,(p_i \to \neg p_j)$, for all i, j satisfying $0 \leq i, j \leq n$ and $i \neq j$

The reader will realize in a minute that this is all we must express about the set of states of $\mathcal{S}$ seen individually.

We now turn to $\mathcal{O}$. Let $\{U_0, \ldots, U_m, U_{m+1}, \ldots, U_{2\cdot(2^n-1)}\}$ be the set of *all* non-empty subsets of X. If, for any $j \in \{1, \ldots, 2\cdot(2^n-1)\}$, the set U_j equals $\{x_{j_0}, \ldots, x_{j_{k_j}}\}$, then let $\alpha_{U_j} := \left(\bigwedge\limits_{l \in \{j_0, \ldots, j_{k_j}\}} \mathsf{L}p_l\right) \wedge \left(\bigwedge\limits_{l \in \{0, \ldots, n\} \setminus \{j_0, \ldots, j_{k_j}\}} \mathsf{K}\neg p_l\right)$; in addition, let $\alpha_{U_0} := \mathsf{L}p_0 \wedge \cdots \wedge \mathsf{L}p_n$. Evidently, every α_{U_j} corresponds to the subset U_j in the given model. With that, we get the subsequent description of the structure of $\mathcal{O}$, among other things, in terms of permitted and forbidden inclusions.

3. α_{U_0}
4. $\Box\left(\alpha_{U_i} \to \mathsf{L}\Diamond\alpha_{U_j}\right)$, for all i, j satisfying $0 \leq i \neq j \leq m$ and $U_j \subseteq U_i$
5. $\Box\left(\alpha_{U_i} \to \mathsf{K}\Box\neg\alpha_{U_j}\right)$, for all i, j satisfying $0 \leq i \neq j \leq 2\cdot(2^n-1)$ and $U_j \not\subseteq U_i$

Let V be any $\mathcal{S}$-valuation satisfying $V(p_i) = \{x_i\}$ for every $i = 0, \ldots, n$, and let $\mathcal{M} := (X, \mathcal{O}, V)$. Then, we clearly have that $x_0, U_0 \models_{\mathcal{M}} \alpha_{\mathcal{S}}$. We now show that $\alpha_{\mathcal{S}}$ constitutes a unique characteristic of the class of all finite subset frames of which a special subframe is mapped onto $\mathcal{S}$ by a bounded morphism, in the following way.

Lemma 9. *Let $\mathcal{S} = (X, \mathcal{O})$ be a finite special subset frame satisfying $\emptyset \notin \mathcal{O}$, and let $\alpha_{\mathcal{S}}$ be the Jankov-Fine formula associated with $\mathcal{S}$. Then, for any finite subset frame $\tilde{\mathcal{S}} = \left(\tilde{X}, \tilde{\mathcal{O}}\right)$, the following two conditions are equivalent.*

1. *There exist a special subframe $\mathcal{S}' = (X', \mathcal{O}')$ of $\tilde{\mathcal{S}}$, an $\mathcal{S}'$-valuation V', and a point $x' \in X'$, such that $x', X' \models_{\mathcal{M}'} \alpha_{\mathcal{S}}$, where $\mathcal{M}' := (X', \mathcal{O}', V')$.*
2. *For some special subframe $\mathcal{S}'$ of $\tilde{\mathcal{S}}$, there exists a bounded morphism $f : \mathcal{S}' \twoheadrightarrow \mathcal{S}$.*

The previous lemma puts us in a position to get on to the desired characterization of definable classes of finite subset frames.

Theorem 4. *Let $\mathfrak{K}$ be a class of finite subset frames. Then, $\mathfrak{K}$ is $\mathcal{L}$-definable iff it is closed under adding isolated points, and under taking special subframes, disjoint unions, and bounded morphic images.*

Our considerations in the second part of this section revolve around the Goldblatt-Thomason Theorem (see Section 1 above). As is known, *ultrafilter extensions* of Kripke models are involved herein in the case of basic modal logic. So, striving for a corresponding subset space analogue seems to require an appropriate idea of ultrafilter extension. But it is by no means clear how to define this notion in a way that sufficiently saturated structures result, to which an adjusted proof could make recourse. Thus, we try to circumvent this difficulty by taking the bimodal Kripke point of view.

Given a class $\mathfrak{K}$ of subset frames, let $\mathfrak{K}_1$ denote the class of all bimodal Kripke frames that are induced by the members of $\mathfrak{K}$ as desribed in Section 2. Every element of $\mathfrak{K}_1$ is an L_1-structure pretty much as every $\mathcal{S} \in \mathfrak{K}$ is an L_2-structure. We actually obtain that $\mathfrak{K}_1$ is L_1-definable in case $\mathfrak{K}$ is the class of *all* subset frames.

Proposition 1. *Let $\mathfrak{K}^a$ be the class of all subset frames. Then (the closure of) $\mathfrak{K}_1^a$ (under taking isomorphic copies) is an L_1-definable class of bimodal Kripke frames.*

Let $\mathfrak{K}_1^{ca}$ be the class of all *cross axiom frames;* see the corresponding definition in [7], Sect. 2.3, and cf. the remark right before Lemma 2 above. We get as a consequence of Proposition 1 that L_1 can distinguish between $\mathfrak{K}_1^a$ and $\mathfrak{K}_1^{ca}$ whereas bimodal logic cannot.

Corollary 1. *The L_1-logics of $\mathfrak{K}_1^a$ and $\mathfrak{K}_1^{ca}$ are different, while the respective bimodal logics are equal. In particular, the class $\mathfrak{K}_1^a$ is not modally definable.*

The Goldblatt-Thomason Theorem states under which circumstances first-order definable classes of frames *are* modally definable. As we now have the choice between the languages L_1 and L_2, we must decide which one to take. It has already been indicated by the introductory text above that we should consider classes $\mathfrak{K}$ of subset frames for which $\mathfrak{K}_1$ is L_1-definable here. In any case, we have that *modal* definability is transferred from Kripke to subset frames.

Lemma 10. *Let $\mathfrak{K}$ be a class of subset frames such that the associated class $\mathfrak{K}_1$ of bimodal Kripke frames is modally definable. Then $\mathfrak{K}$ itself is modally definable.*

Note that Corollary 1 shows that the converse of Lemma 10 is not true. – We now turn to some transfer results concerning the separate conditions of the Goldblatt-Thomason Theorem.

Lemma 11. *Let $\mathfrak{K}$ be any class of subset frames and $\mathfrak{K}_1$ its associated class of bimodal Kripke frames.*

1. *If $\mathfrak{K}$ is closed under taking special subframes, then $\mathfrak{K}_1$ is closed under taking generated subframes.*
2. *If $\mathfrak{K}$ is closed under taking disjoint unions, then the same is true of $\mathfrak{K}_1$.*

Note that a corresponding property is missing for taking images under bounded morphisms since applying this operation may destroy some of the L_1-characteristics of the members of $\mathfrak{K}_1^a$.

A class $\mathfrak{K}'$ of Kripke frames is said to *reflect ultrafilter extensions* iff, whenever the ultrafilter extension of any frame F is contained in $\mathfrak{K}'$, then F itself is in $\mathfrak{K}'$. As opposed to bounded morphisms, this property seems to be satisfied by most classes of induced bimodal Kripke frames because of the particular form of the accessibility relations on ultrafilter extensions. Anyway, *applying* the classical Goldblatt-Thomason Theorem yields the following lemma.

Lemma 12. *Let $\mathfrak{K}$ be any class of subset frames and $\mathfrak{K}_1$ its associated class of bimodal Kripke frames. Assume that $\mathfrak{K}_1$ is L_1-definable and closed under taking generated subframes, disjoint unions, and bounded morphic images, and reflects ultrafilter extensions. Then $\mathfrak{K}_1$ is $\mathcal{L}$-definable.*

Putting the outcomes of Lemma 10, Lemma 11, and Lemma 12, together, we obtain the subsequent Goldblatt-Thomason style result for subset frames.

Theorem 5. *Let $\mathfrak{K}$ be a class of subset frames such that the associated class $\mathfrak{K}_1$ is first-order definable, closed under taking bounded morphic images, and reflects ultrafilter extensions. Then, $\mathfrak{K}$ is $\mathcal{L}$-definable iff it is closed under adding isolated points, and under taking special subframes and disjoint unions.*

6 Concluding Remarks

We have investigated the expressive power of Moss and Parikh's bimodal language for knowledge and topology with regard to definability matters. It turned out that we could obtain fully satisfying results (i.e., complete analogues to the case of basic modal logic) on the level of subset *spaces,* which correspond to usual Kripke models. Following the philosophy voiced at the end of Sect. 2.6 of the textbook [4], the reason for this success lies in the fact that we have a *standard translation* of $\mathcal{L}$ into a language which is intrinsically first-order. The relevant methods, however, had to be adapted to the present context due to the different semantics, which, in particular, entails the presence of more than one sort.

Concerning *frame* definability, we have identified four necessary closure properties of classes of subset frames, which in case of *finite* frames have proved to be also sufficient. Furthermore, the question whether we can do without the finiteness assumption was answered affirmatively in case the class of associated bimodal Kripke frames is first-order definable and closed under some operations on frames that are studied in common modal logic: bounded morphisms and ultrafilter extensions. And while the closure under taking bounded morphic images restricts the possible classes drastically, this seems to be hardly true of reflecting ultrafilter extensions.

Going beyond that, the proof of a Goldblatt-Thomason analogue for subset frames referring to L_2*-definable classes* must be postponed to future research. Maybe an essential difference of the correspondence languages L_1 and L_2 regarding this will appear, though.

The expressivity results obtained in this paper underpin the early statement that $\mathcal{L}$ is a rather weak language; concerning this, see the discussion in the paper [7]. Therefore, it comes as no surprise in retrospect that expressive power has been added to $\mathcal{L}$ in this or that way. For example, hybridizations of $\mathcal{L}$ have been considered (see, e.g., [12]), or extra relations have been added (see [11] and [13]). As a consequence, more complex properties of subset frames can be captured then, in particular, some of those mentioned as negative examples in Section 4 (and even 'real' connectedness can be defined when interpretation is restricted to the class of all topological spaces). – It is an interesting task of future research to establish corresponding definability results in these cases as well; cf. [5], Sect. 5.

References

1. Aiello, M., van Benthem, J., Bezhanishvili, G.: Reasoning about space: The modal way. Journal of Logic and Computation 13(6), 889–920 (2003)
2. Aiello, M., Pratt-Hartmann, I.E., van Benthem, J.F.A.K.: Handbook of Spatial Logics. Springer (2007)
3. Başkent, C.: Topics in Subset Space Logic. Master's thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam (July 2007)
4. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
5. ten Cate, B., Gabelaia, D., Sustretov, D.: Modal languages for topology: Expressivity and definability. Annals of Pure and Applied Logic 159, 146–170 (2009)
6. Chang, C.C., Keisler, H.J.: Model Theory, 3rd edn. Studies in Logic and the Foundations of Mathematics, vol. 73. North-Holland, Amsterdam (1990)
7. Dabrowski, A., Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. Annals of Pure and Applied Logic 78, 73–110 (1996)
8. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
9. Flum, J., Ziegler, M.: Topological Model Theory. Lecture Notes in Mathematics, vol. 769. Springer, Berlin (1980)
10. Georgatos, K.: Modal Logics of Topological Spaces. Ph.D. thesis, City University of New York (May 1993)
11. Georgatos, K.: Updating knowledge using subsets. Journal of Applied Non-Classical Logics 21(3-4), 427–441 (2011)
12. Heinemann, B.: A hybrid logic for reasoning about knowledge and topology. Journal of Logic, Language and Information 17(1), 19–41 (2008)
13. Heinemann, B.: 'Topologic' can be more expressive. In: Proc. 8th Panhellenic Logic Symposium, Ioannina, Greece, pp. 58–62 (2011)
14. Heinemann, B.: Characterizing Certain Topological Specifications. In: Hirsch, E.A., Karhumäki, J., Lepistö, A., Prilutskii, M. (eds.) CSR 2012. LNCS, vol. 7353, pp. 184–195. Springer, Heidelberg (2012)
15. Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. In: Moses, Y. (ed.) Theoretical Aspects of Reasoning about Knowledge (TARK 1992), pp. 95–105. Morgan Kaufmann, Los Altos (1992)

Quantified Differential Temporal Dynamic Logic for Verifying Properties of Distributed Hybrid Systems*

Ping Hou and Hao Zheng

Department of Computer Science and Engineering, University of South Florida

Abstract. We combine quantified differential dynamic logic (Qd$\mathcal{L}$) for reasoning about the possible behavior of distributed hybrid systems with temporal logic for reasoning about the temporal behavior during their operation. Our logic supports verification of temporal and non-temporal properties of distributed hybrid systems and provides a uniform treatment of discrete transitions, continuous evolution, and dynamic dimensionality-changes. For our combined logic, we generalize the semantics of dynamic modalities to refer to hybrid traces instead of final states. Further, we prove that this gives a conservative extension of Qd$\mathcal{L}$ for distributed hybrid systems. On this basis, we provide a modular verification calculus that reduces correctness of temporal behavior of distributed hybrid systems to non-temporal reasoning, and prove that we obtain a complete axiomatization relative to the non-temporal base logic Qd$\mathcal{L}$. Using this calculus, we analyze temporal safety properties in a distributed air traffic control system where aircraft can appear dynamically.

1 Introduction

Ensuring correct functioning of cyber-physical systems is among the most challenging and most important problems in computer science, mathematics, and engineering. Hybrid systems are common mathematical models for cyber-physical systems with interacting discrete and continuous behavior [6,13]. Their behavior combines continuous evolution (called *flow*) characterized by differential equations and discrete jumps. However, not all relevant cyber-physical systems can be modeled as hybrid systems. Hybrid systems cannot represent physical control systems that are distributed or form a multi-agent system, e.g., distributed car control systems [15] and distributed air traffic control systems [8]. Such systems form *distributed hybrid systems* [7,16,21,22] with discrete, continuous, structural, and dimension-changing dynamics. Distributed hybrid systems combine the challenges of hybrid systems and distributed systems. Correctness of safety-critical

* This material is based upon work supported by the National Science Foundation under Grant No. 0930510. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

real-time and distributed hybrid systems depends on a safe operation throughout *all* states of all possible trajectories, and the behavior at intermediate states is highly relevant [1,4,6,11,13].

Temporal logics (TL) use temporal operators to talk about intermediate states [1,9,10,23]. They have been used successfully in model checking [1,3,13,14,18] of finite-state system abstractions. State spaces of distributed hybrid systems, however, often do not admit equivalent finite-state abstractions [13,18]. Instead of model checking, TL can also be used deductively to prove validity of formulas in calculi [5,6]. Valid TL formulas, however, only express very generic facts that are true for all systems, regardless of their actual behavior. Hence, the behavior of a specific system first needs to be axiomatized declaratively to obtain meaningful results. Then, however, the correspondence between actual system operations and a declarative temporal representation may be questioned.

Very recently, a dynamic logic, called *quantified differential dynamic logic* (Q$\mathsf{d}\mathcal{L}$) has been introduced as a successful tool for deductively verifying distributed hybrid systems [21,22]. Q$\mathsf{d}\mathcal{L}$ can analyze the behavior of actual distributed hybrid system models, which are specified operationally. Yet, operational distributed hybrid system models are *internalized* within Q$\mathsf{d}\mathcal{L}$ formulas, and Q$\mathsf{d}\mathcal{L}$ is closed under logical operators. However, Q$\mathsf{d}\mathcal{L}$ only considers the behavior of distributed hybrid systems at final states, which is insufficient for verifying safety properties that have to hold all the time.

We close this gap of expressivity by combining Q$\mathsf{d}\mathcal{L}$ with temporal logic [9,10,23]. In this paper, we introduce a logic, called *quantified differential temporal dynamic logic* (QdTL), which provides modalities for quantifying over traces of distributed hybrid systems based on Q$\mathsf{d}\mathcal{L}$. We equip QdTL with temporal operators to state what is true all along a trace or at some point during a trace. In this paper, we modify the semantics of the dynamic modality $[\alpha]$ to refer to all *traces* of α instead of all final states reachable with α (similarly for $\langle\alpha\rangle$). For instance, the formula $[\alpha]\Box\phi$ expresses that ϕ is true at each state during all traces of the distributed hybrid system α. With this, QdTL can also be used to verify temporal statements about the behavior of α at intermediate states during system runs. As in our non-temporal dynamic logic Q$\mathsf{d}\mathcal{L}$ [21,22], we use *quantified hybrid programs* as an operational model for distributed hybrid systems, since they admit a uniform compositional treatment of interacting discrete transitions, continuous evolutions, and structural/dimension changes in logic.

As a semantical foundation for combined temporal dynamic formulas, we introduce a hybrid trace semantics for QdTL. We prove that QdTL is a conservative extension of Q$\mathsf{d}\mathcal{L}$: for non-temporal specifications, trace semantics is equivalent to the non-temporal transition semantics of Q$\mathsf{d}\mathcal{L}$ [21,22].

As a means for verification, we introduce a sequent calculus for QdTL that successively reduces temporal statements about traces of quantified hybrid programs to non-temporal Q$\mathsf{d}\mathcal{L}$ formulas. In this way, we make the intuition formally precise that temporal safety properties can be checked by augmenting proofs with appropriate assertions about intermediate states. Like in [21,22], our calculus works compositionally. It decomposes correctness statements about quantified hybrid programs structurally into corresponding statements about its parts by symbolic transformation.

Our approach combines the advantages of Qd$\mathcal{L}$ in reasoning about the behaviour of operational distributed hybrid system models with those of TL to verify temporal statements about traces. We show that QdTL is sound and relatively complete. We argue that QdTL can verify practical systems and demonstrate this by studying temporal safety properties in distributed air traffic control. Our primary contributions are as follows:

- We introduce a logic for specifying and verifying temporal properties of distributed hybrid systems.
- We present a proof calculus for this logic, which, to the best of our knowledge, is the first verification approach that can handle temporal statements about distributed hybrid systems.
- We prove that this compositional calculus is a sound and complete axiomatization relative to differential equations.
- We verify temporal safety properties in a collision avoidance maneuver in distributed air traffic control, where aircraft can appear dynamically.

2 Related Work

Multi-party distributed control has been suggested for car control [15] and air traffic control [8]. Due to limits in verification technology, no formal analysis of temporal statements about the distributed hybrid dynamics has been possible for these systems yet. Analysis results include discrete message handling [15] or collision avoidance for two participants [8].

The importance of understanding dynamic/reconfigurable distributed hybrid systems was recognized in modeling languages SHIFT [7] and R-Charon [16]. They focused on simulation/compilation [7] or the development of a semantics [16], so that no verification is possible yet.

Other process-algebraic approaches, like χ [27], have been developed for modeling and simulation. Verification is still limited to small fragments that can be translated directly to other verification tools like PHAVer or UPPAAL, which do not support distributed hybrid systems.

Our approach is completely different. It is based on first-order structures and dynamic logic. We focus on developing a logic that supports temporal and non-temporal statements about distributed hybrid dynamics and is amenable to automated theorem proving in the logic itself.

Our previous work and other verification approaches for static hybrid systems cannot verify distributed hybrid systems. Distributed hybrid systems may have an unbounded and changing number of components/participants, which cannot be represented with any fixed number of dimensions of the state space.

Based on [24], Beckert and Schlager [2] added separate trace modalities to dynamic logic and presented a relatively complete calculus. Their approach only handles discrete state spaces. In contrast, QdTL works for hybrid programs with continuous and structural/dimensional dynamics.

Davoren and Nerode [6] extended the propositional modal μ-calculus with a semantics in hybrid systems and examine topological aspects. In [5], Davoren et al. gave a semantics in terms of general flow systems for a generalisation of CTL* [10]. In both cases, the authors of [6] and [5] provided Hilbert-style calculi to prove formulas that are valid for all systems simultaneously using abstract actions.

The strength of our logic primarily is that it is a first-order dynamic logic: it handles actual hybrid programs rather than only abstract actions of unknown effect. Our calculus directly supports verification of quantified hybrid programs with continuous evolution and structural/dimensional changes. First-order dynamic logic is more expressive and calculi are deductively stronger than other approaches [2,17].

3 Syntax of Quantified Differential Temporal Dynamic Logic

As a formal logic for verifying temporal specifications of distributed hybrid systems, we introduce *quantified differential temporal dynamic logic* (QdTL). QdTL extends dynamic logic for reasoning about system runs [12] with many-sorted first-order logic for reasoning about all ($\forall i : A\ \phi$) or some ($\exists i : A\ \phi$) objects of a sort A, e.g., the sort of all aircraft, and three other concepts:

Quantified hybrid programs. The behavior of distributed hybrid systems can be described by quantified hybrid programs [21,22], which generalize regular programs from dynamic logic [12] to distributed hybrid changes. The distinguishing feature of quantified hybrid programs is that they provide uniform discrete transitions, continuous evolutions, and structural/dimension changes along quantified assignments and quantified differential equations, which can be combined by regular control operations.

Modal operators. Modalities of dynamic logic express statements about all possible behavior ($[\alpha]\pi$) of a system α, or about the existence of a trace ($\langle\alpha\rangle\pi$), satisfying condition π. Unlike in standard dynamic logic, α is a model of a distributed hybrid system. We use quantified hybrid programs to describe α as in [21,22]. Yet, unlike in standard dynamic logic [12] or quantified differential dynamic logic (Qd$\mathcal{L}$) [21,22], π is a *trace formula* in QdTL, and π can refer to all states that occur *during* a trace using temporal operators.

Temporal operators. For QdTL, the temporal trace formula $\Box\phi$ expresses that the formula ϕ holds all along a trace selected by $[\alpha]$ or $\langle\alpha\rangle$. For instance, the state formula $\langle\alpha\rangle\Box\phi$ says that the state formula ϕ holds at every state along at least one trace of α. Dually, the trace formula $\Diamond\phi$ expresses that ϕ holds at some point during such a trace. It can occur in a state formula $\langle\alpha\rangle\Diamond\phi$ to express that there is such a state in some trace of α, or as $[\alpha]\Diamond\phi$ to say that, along each trace, there is a state satisfying ϕ. In this paper, the primary focus of attention is on homogeneous combinations of path and trace quantifiers like $[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$.

3.1 Quantified Hybrid Programs

QdTL supports a (finite) number of object *sorts*, e.g., the sort of all aircraft, or the sort of all cars. For continuous quantities of distributed hybrid systems like positions or velocities, we add the sort $\mathbb{R}$ for real numbers. *Terms* of QdTL are built from a set of (sorted) function/variable symbols as in many-sorted first-order logic. For representing appearance and disappearance of objects while running QHPs, we use an existence function symbol $\mathsf{E}(\cdot)$ that has value $\mathsf{E}(o) = 1$ if object o exists, and has value $\mathsf{E}(o) = 0$ when object o disappears or has not been created yet. We use $0, 1, +, -, \cdot$ with the usual notation and fixed semantics for real arithmetic. For $n \geq 0$ we abbreviate $f(s_1, \ldots, s_n)$ by $f(\boldsymbol{s})$ using vectorial notation and we use $\boldsymbol{s} = \boldsymbol{t}$ for element-wise equality.

As a system model for distributed hybrid systems, QdTL uses *quantified hybrid programs* (QHP) [21,22]. The quantified hybrid programs occurring in dynamic modalities of QdTL are regular programs from dynamic logic [12] to which quantified assignments and quantified differential equation systems for distributed hybrid dynamics are added. QHPs are defined by the following grammar (α, β are QHPs, θ a term, i a variable of sort A, f is a function symbol, $\boldsymbol{s}$ is a vector of terms with sorts compatible to f, and χ is a formula of first-order logic):

$$\alpha, \beta ::= \forall i : A\ f(\boldsymbol{s}) := \theta \mid \forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi \mid ?\chi \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

The effect of *quantified assignment* $\forall i : A\ f(\boldsymbol{s}) := \theta$ is an instantaneous discrete jump assigning θ to $f(\boldsymbol{s})$ simultaneously for all objects i of sort A. The QHP $\forall i : C\ a(i) := a(i) + 1$, for example, expresses that all cars i of sort C simultaneously increase their acceleration $a(i)$. The effect of *quantified differential equation* $\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi$ is a continuous evolution where, for all objects i of sort A, all differential equations $f(\boldsymbol{s})' = \theta$ hold and formula χ holds throughout the evolution (the state remains in the region described by χ). The dynamics of QHPs changes the interpretation of terms over time: $f(\boldsymbol{s})'$ is intended to denote the derivative of the interpretation of the term $f(\boldsymbol{s})$ over time during continuous evolution, not the derivative of $f(\boldsymbol{s})$ by its argument $\boldsymbol{s}$. For $f(\boldsymbol{s})'$ to be defined, we assume f is an $\mathbb{R}$-valued function symbol. For simplicity, we assume that f does not occur in $\boldsymbol{s}$. In most quantified assignments/differential equations $\boldsymbol{s}$ is just i. For instance, the following QHP expresses that all cars i of sort C drive by $\forall i : C\ x(i)'' = a(i)$ such that their position $x(i)$ changes continuously according to their respective acceleration $a(i)$.

The effect of test $?\chi$ is a *skip* (i.e., no change) if formula χ is true in the current state and *abort* (blocking the system run by a failed assertion), otherwise. *Nondeterministic choice* $\alpha \cup \beta$ is for alternatives in the behavior of the distributed hybrid system. In the *sequential composition* $\alpha; \beta$, QHP β starts after α finishes (β never starts if α continues indefinitely). *Nondeterministic repetition* α^* repeats α an arbitrary number of times, possibly zero times.

Structural dynamics of distributed hybrid systems corresponds to quantified assignments to function terms and we model the appearance of new participants in the distributed hybrid system, e.g., new aircraft appearing into the local flight scenario, by a program $n := \mathsf{new}\ A$ (see [21,22] for details).

3.2 State and Trace Formulas

The formulas of QdTL are defined similarly to first-order dynamic logic plus many-sorted first-order logic. However, the modalities $[\alpha]$ and $\langle\alpha\rangle$ accept trace formulas that refer to the temporal behavior of *all* states along a trace. Inspired by CTL and CTL* [9,10], we distinguish between state formulas, which are true or false in states, and trace formulas, which are true or false for system traces.

The *state formulas* of QdTL are defined by the following grammar (ϕ, ψ are state formulas, π is a trace formula, θ_1, θ_2 are terms of the same sort, i is a variable of sort A, and α is a QHP):

$$\phi, \psi ::= \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall i : A\ \phi \mid \exists i : A\ \phi \mid [\alpha]\pi \mid \langle\alpha\rangle\pi$$

We use standard abbreviations to define $\leq, >, <, \vee, \rightarrow$. Sorts $A \neq \mathbb{R}$ have no ordering and only $\theta_1 = \theta_2$ is allowed. For sort $\mathbb{R}$, we abbreviate $\forall x : \mathbb{R}\ \phi$ by $\forall x \phi$.

The *trace formulas* of QdTL are defined by the following grammar (π is a trace formula and ϕ is a state formula):

$$\pi ::= \phi \mid \Box\phi \mid \Diamond\phi$$

Formulas without $\Box$ and $\Diamond$ are *non-temporal* $\mathsf{Qd}\mathcal{L}$ *formulas.* Unlike in CTL, state formulas are true on a trace if they hold for the *last* state of a trace, not for the first. Thus, $[\alpha]\phi$ expresses that ϕ is true at the end of each trace of α. In contrast, $[\alpha]\Box\phi$ expresses that ϕ is true all along all states of every trace of α. This combination gives a smooth embedding of non-temporal $\mathsf{Qd}\mathcal{L}$ into QdTL and makes it possible to define a compositional calculus. Like CTL, QdTL allows nesting with a branching time semantics [9], e.g., $[\alpha]\Box((\forall i : C\ x(i) \geq 2) \rightarrow \langle\beta\rangle\Diamond(\forall i : C\ x(i) \leq 0))$. In the following, all formulas and terms have to be well-typed. For short notation, we allow conditional terms of the form $\mathsf{if}\ \phi\ \mathsf{then}\ \theta_1\ \mathsf{else}\ \theta_2\ \mathsf{fi}$ (where θ_1 and θ_2 have the same sort). This term evaluates to θ_1 if the formula ϕ is true and to θ_2 otherwise. We consider formulas with conditional terms as abbreviations, e.g., $\psi(\mathsf{if}\ \phi\ \mathsf{then}\ \theta_1\ \mathsf{else}\ \theta_2)$ for $(\phi \rightarrow \psi(\theta_1)) \wedge (\neg\phi \rightarrow \psi(\theta_2))$.

Example 1. Let C be the sort of all cars. By $x(i)$, we denote the position of car i, by $v(i)$ its velocity and by $a(i)$ its acceleration. Then the QdTL formula

$$(\forall i : C\ x(i) \geq 0) \rightarrow [\forall i : C\ x(i)' = v(i), v(i)' = a(i)\ \&\ v(i) \geq 0]\Box(\forall i : C\ x(i) \geq 0)$$

says that, if all cars start at a point to the right of the origin and we only allow them to evolve as long as all of them have nonnegative velocity, then they *always* stay up to the right of the origin. In this case, the QHP just consists of a quantified differential equation expressing that the position $x(i)$ of car i evolves over time according to the velocity $v(i)$, which evolves according to its acceleration $a(i)$. The constraint $v(i) \geq 0$ expresses that the cars never move backwards, which otherwise would happen eventually in the case of braking $a(i) < 0$. This formula is indeed valid, and we would be able to use the techniques developed in this paper to prove it.

4 Semantics of Quantified Differential Temporal Dynamic Logic

In standard dynamic logic [12] and the logic $\mathsf{Qd}\mathcal{L}$ [21,22], modalities only refer to the final states of system runs and the semantics is a reachability relation on states: State τ is reachable from state σ using α if there is a run of α which terminates in τ when started in σ. For QdTL, however, formulas can refer to intermediate states of runs as well. Thus, the semantics of a distributed hybrid system α is the set of its possible *traces*, i.e., successions of states that occur during the evolution of α.

4.1 Trace Semantics of Quantified Hybrid Programs

A *state* σ associates an infinite set $\sigma(A)$ of objects with each sort A, and it associates a function $\sigma(f)$ of appropriate type with each function symbol f, including $\mathsf{E}(\cdot)$. For simplicity, σ also associates a value $\sigma(i)$ of appropriate type with each variable i. The domain of $\mathbb{R}$ and the interpretation of $0, 1, +, -, \cdot$ is that of real arithmetic. We assume constant domain for each sort A: all states σ, τ share the same infinite domains $\sigma(A) = \tau(A)$. Sorts $A \neq C$ are disjoint: $\sigma(A) \cap \sigma(C) = \emptyset$. The set of all states is denoted by $\mathcal{S}$. The state σ_i^e agrees with σ except for the interpretation of variable i, which is changed to e. In addition, we distinguish a state Λ to denote the failure of a system run when it is *aborted* due to a test $?\chi$ that yields *false*. In particular, Λ can only occur at the end of an aborted system run and marks that there is no further extension.

Distributed hybrid systems evolve along piecewise continuous traces in multi-dimensional space, structural changes, and appearance or disappearance of agents as time passes. Continuous phases are governed by differential equations, whereas discontinuities are caused by discrete jumps. Unlike in discrete cases [2,24], traces are not just sequences of states, since distributed hybrid systems pass through uncountably many states even in bounded time. Beyond that, continuous changes are more involved than in pure real-time [1,14], because all variables can evolve along different differential equations. Generalizing the real-time traces of [14], the following definition captures hybrid behavior by splitting the uncountable succession of states into periods ν_i that are regulated by the same control law. For discrete jumps, some periods are point flows of duration 0.

The (trace) semantics of quantified hybrid programs is compositional, that is, the semantics of a complex program is defined as a simple function of the trace semantics of its parts.

Definition 1 (Hybrid Trace). *A* trace *is a (non-empty) finite or infinite sequence $\nu = (\nu_0, \nu_1, \nu_2, \ldots)$ of functions $\nu_k : [0, r_k] \to \mathcal{S}$ with respective durations*

$r_k \in \mathbb{R}$ *(for* $k \in \mathbb{N}$*). A* position *of* ν *is a pair* (k, ζ) *with* $k \in \mathbb{N}$ *and* ζ *in the interval* $[0, r_k]$*; the state of* ν *at* (k, ζ) *is* $\nu_k(\zeta)$*. Positions of* ν *are ordered lexicographically by* $(k, \zeta) \prec (m, \xi)$ *iff either* $k < m$*, or* $k = m$ *and* $\zeta < \xi$*. Further, for a state* $\sigma \in \mathcal{S}$*,* $\hat{\sigma} : 0 \mapsto \sigma$ *is the* point flow *at* σ *with duration* 0*. A trace* terminates *if it is a finite sequence* $(\nu_0, \nu_1, \ldots, \nu_n)$ *and* $\nu_n(r_n) \neq \Lambda$*. In that case, the last state* $\nu_n(r_n)$ *is denoted by* last ν*. The first state* $\nu_0(0)$ *is denoted by* first ν*.*

Unlike in [1,14], the definition of traces also admits finite traces of bounded duration, which is necessary for compositionality of traces in $\alpha; \beta$. The semantics of quantified hybrid programs α as the set $\mu(\alpha)$ of its possible traces depends on valuations $\sigma[\![\cdot]\!]$ of formulas and terms at intermediate states σ. The valuation of formulas will be defined in Definition 3. Especially, we use $\sigma_i^e[\![\cdot]\!]$ to denote the valuations of terms and formulas in state σ_i^e, i.e., in state σ with i interpreted as e.

Definition 2 (Trace Semantics of Quantified Hybrid Programs). *The* trace semantics, $\mu(\alpha)$, *of a quantified hybrid program* α*, is the set of all its possible hybrid traces and is defined inductively as follows:*

1. $\mu(\forall i : A\ f(\boldsymbol{s}) := \theta) = \{(\hat{\sigma}, \hat{\tau}) : \sigma \in \mathcal{S}$ *and state* τ *is identical to* σ *except that at each position* $\boldsymbol{o}$ *of* f*: if* $\sigma_i^e[\![\boldsymbol{s}]\!] = \boldsymbol{o}$ *for some object* $e \in \sigma(A)$*, then* $\tau(f)(\sigma_i^e[\![\boldsymbol{s}]\!]) = \sigma_i^e[\![\theta]\!]$.$\}$
2. $\mu(\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi) = \{(\varphi) : 0 \leq r \in \mathbb{R}$ *and* $\varphi : [0, r] \to \mathcal{S}$ *is a function satisfying the following conditions. At each time* $t \in [0, r]$*, state* $\varphi(t)$ *is identical to* $\varphi(0)$*, except that at each position* $\boldsymbol{o}$ *of* f *: if* $\sigma_i^e[\![\boldsymbol{s}]\!] = \boldsymbol{o}$ *for some object* $e \in \sigma(A)$*, then, at each time* $\zeta \in [0, r]$*:*
   – *The differential equations hold and derivatives exist (trivial for* $r = 0$*):*
$$\frac{\mathsf{d}(\varphi(t)_i^e[\![f(\boldsymbol{s})]\!])}{\mathsf{d}t}(\zeta) = (\varphi(\zeta)_i^e[\![\theta]\!])$$
   – *The evolution domains is respected:* $\varphi(\zeta)_i^e[\![\chi]\!] = true.\}$
3. $\mu(?\chi) = \{(\hat{\sigma}) : \sigma[\![\chi]\!] = true\} \cup \{(\hat{\sigma}, \hat{\Lambda}) : \sigma[\![\chi]\!] = false\}$
4. $\mu(\alpha \cup \beta) = \mu(\alpha) \cup \mu(\beta)$
5. $\mu(\alpha; \beta) = \{\nu \circ \varsigma : \nu \in \mu(\alpha), \varsigma \in \mu(\beta)$ *when* $\nu \circ \varsigma$ *is defined*$\}$*; the composition of* $\nu = (\nu_0, \nu_1, \nu_2, \ldots)$ *and* $\varsigma = (\varsigma_0, \varsigma_1, \varsigma_2, \ldots)$ *is*
$$\nu \circ \varsigma = \begin{cases} (\nu_0, \ldots, \nu_n, \varsigma_0, \varsigma_1, \ldots) & \text{if } \nu \text{ terminates at } \nu_n \text{ and } \mathrm{last}\,\nu = \mathrm{first}\,\varsigma \\ \nu & \text{if } \nu \text{ does not terminate} \\ \text{not defined} & \text{otherwise} \end{cases}$$
6. $\mu(\alpha^*) = \bigcup_{n \in \mathbb{N}} \mu(\alpha^n)$*, where* $\alpha^{n+1} := (\alpha^n; \alpha)$ *for* $n \geq 1$*, as well as* $\alpha^1 := \alpha$ *and* $\alpha^0 := (?true)$*.*

4.2 Valuation of State and Trace Formulas

Definition 3 (Valuation of Formulas). *The valuation of state and trace formulas is defined respectively. For state formulas, the* valuation $\sigma[\![\cdot]\!]$ *with respect to state* σ *is defined as follows:*

1. $\sigma[\![\theta_1 = \theta_2]\!] = true$ *iff* $\sigma[\![\theta_1]\!] = \sigma[\![\theta_2]\!]$*; accordingly for* $\geq$.
2. $\sigma[\![\phi \wedge \psi]\!] = true$ *iff* $\sigma[\![\phi]\!] = true$ *and* $\sigma[\![\psi]\!] = true$*; accordingly for* $\neg$.
3. $\sigma[\![\forall i : A\ \phi]\!] = true$ *iff* $\sigma_i^e[\![\phi]\!] = true$ *for all objects* $e \in \sigma(A)$.
4. $\sigma[\![\exists i : A\ \phi]\!] = true$ *iff* $\sigma_i^e[\![\phi]\!] = true$ *for some object* $e \in \sigma(A)$.
5. $\sigma[\![[\alpha]\pi]\!] = true$ *iff for each trace* $\nu \in \mu(\alpha)$ *that starts in* first $\nu = \sigma$*, if* $\nu[\![\pi]\!]$ *is defined, then* $\nu[\![\pi]\!] = true$.
6. $\sigma[\![\langle\alpha\rangle\pi]\!] = true$ *iff there is a trace* $\nu \in \mu(\alpha)$ *starting in* first $\nu = \sigma$ *such that* $\nu[\![\pi]\!]$ *is defined and* $\nu[\![\pi]\!] = true$.

For trace formulas, the valuation $\nu[\![\cdot]\!]$ *with respect to trace* ν *is defined as follows:*

1. *If* ϕ *is a state formula, then* $\nu[\![\phi]\!] =$ last $\nu[\![\phi]\!]$ *if* ν *terminates, whereas* $\nu[\![\phi]\!]$ *is not defined if* ν *does not terminate.*
2. $\nu[\![\Box\phi]\!] = true$ *iff* $\nu_k(\zeta)[\![\phi]\!] = true$ *for all positions* (k, ζ) *of* ν *with* $\nu_k(\zeta) \neq \Lambda$.
3. $\nu[\![\Diamond\phi]\!] = true$ *iff* $\nu_k(\zeta)[\![\phi]\!] = true$ *for some position* (k, ζ) *of* ν *with* $\nu_k(\zeta) \neq \Lambda$.

As usual, a (state) formula is *valid* if it is true in all states. Further for (state) formula ϕ and state σ we write $\sigma \models \phi$ iff $\sigma[\![\phi]\!] = true$. We write $\sigma \not\models \phi$ iff $\sigma[\![\phi]\!] = false$. Likewise, for trace formula π and trace ν we write $\nu \models \pi$ iff $\nu[\![\pi]\!] = true$ and $\nu \not\models \pi$ iff $\nu[\![\pi]\!] = false$. In particular, we only write $\nu \models \pi$ or $\nu \not\models \pi$ if $\nu[\![\pi]\!]$ is defined, which it is not the case if π is a state formula and ν does not terminate.

4.3 Conservative Temporal Extension

The following result shows that the extension of QdTL by temporal operators does not change the meaning of non-temporal Qd$\mathcal{L}$ formulas. The trace semantics given in Definition 3 is equivalent to the final state reachability relation semantics [21,22] for the sublogic Qd$\mathcal{L}$ of QdTL.

Proposition 1. *The logic* QdTL *is a* conservative extension *of non-temporal* Qd$\mathcal{L}$*, i.e., the set of valid* Qd$\mathcal{L}$ *formulas is the same with respect to transition reachability semantics of* Qd$\mathcal{L}$ *[21,22] as with respect to the trace semantics of* QdTL *(Definition 3).*

5 Safety Properties in Distributed Air Traffic Control

In air traffic control, collision avoidance maneuvers [8,26] are used to resolve conflicting flight paths that arise during free flight. We consider the roundabout collision avoidance maneuver for air traffic control [26]. In the literature, formal verification of the hybrid dynamics of air traffic control focused on a fixed number of aircraft, usually two. In reality, many more aircraft are in the same flight corridor, even if not all of them participate in the same maneuver. However, they may be involved in multiple distributed maneuvers at the same time. Perfect global trajectory planning quickly becomes infeasible then. The verification itself also becomes much more complicated for three aircraft already. Explicit replication of the system dynamics n times is computationally infeasible for larger n. Yet, collision avoidance maneuvers need to work for an (essentially) unbounded number of aircraft. Because global trajectory planning is infeasible, the appearance of other aircraft into a local collision avoidance maneuver always has to be expected and managed safely. See Fig. 1 for a general illustration of roundabout-style collision avoidance maneuvers and the phenomenon of dynamic appearance of some new aircraft z into the horizon of relevance.

The resulting flight control system has several characteristics of hybrid dynamics. But it is not a hybrid system and does not even have a fixed finite number of variables in a fixed finite-dimensional state space. The system forms a distributed hybrid system, in which all aircraft fly at the same time and new aircraft may appear from remote areas into the local flight scenario. Let A be the sort of all aircraft. Each aircraft i has a position $x(i) = (x_1(i), x_2(i))$ and a velocity vector $d(i) = (d_1(i), d_2(i))$. We model the continuous dynamics of an aircraft i that follows a flight curve with an angular velocity $\omega(i)$ by the (function) differential equation:

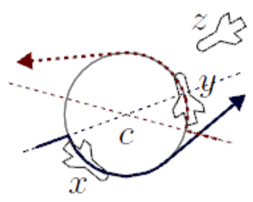


Fig. 1. Roundabout collision avoidance maneuver with new appearance

$$x_1(i)' = d_1(i), x_2(i)' = d_2(i), d_1(i)' = -\omega(i)d_2(i), d_2(i)' = \omega(i)d_1(i) \quad (\mathcal{F}_{\omega(i)}(i))$$

This differential equation, which we denote by $\mathcal{F}_{\omega(i)}(i)$, is the standard equation for curved flight from the literature [26], but lifted to function symbols that are parameterized by aircraft i. Now the quantified differential equation $\forall i : A\ \mathcal{F}_{\omega(i)}(i)$ characterizes that *all* aircraft i fly along their respective (function) differential equation $\mathcal{F}_{\omega(i)}(i)$ according to their respective angular velocities $\omega(i)$ at the same time. This quantified differential equation captures what no finite-dimensional differential equation system could ever do. It characterizes the simultaneous movement of an unbounded, arbitrary, and even growing or shrinking set of aircraft.

Two aircraft i and j have violated the safe separation property if they falsify the following formula

$$\mathcal{P}(i,j) \equiv i = j \lor (x_1(i) - x_1(j))^2 + (x_2(i) - x_2(j))^2 \geq p^2$$

which says that aircraft i and j are either identical or separated by at least the protected zone p (usually 5mi). For the aircraft control system to be safe, all aircraft have to be safely separated, i.e., need to satisfy $\forall i, j : A\ \mathcal{P}(i,j)$. It is crucial that this formula holds at *every* point in time during the system evolution, not only at its beginning or at its end. Hence, we need to consider temporal safety properties. For instance, QdTL can analyze the following temporal safety properties of a part of the distributed roundabout collision avoidance maneuver for air traffic control:

$$\forall i, j : A\ \mathcal{P}(i,j) \land \forall i, j : A\ \mathcal{T}(i,j) \to [\forall i : A\ \mathcal{F}_{\omega(i)}(i)]\Box\forall i, j : A\ \mathcal{P}(i,j) \qquad (1)$$

$$\forall i, j : A\ \mathcal{P}(i,j) \land \forall i, j : A\ \mathcal{T}(i,j) \rightarrow$$
$$[\forall i : A\ t := 0; \forall i : A\ \mathcal{F}_{\omega(i)}(i), t' = 1\ \&\ \forall i : A\ t \leq T; ?(\forall i : A\ t = T)]\Box \forall i, j : A\ \mathcal{P}(i,j) \tag{2}$$

where $\mathcal{T}(i,j) \equiv d_1(i) - d_1(j) = -\omega(x_2(i) - x_2(j)) \land d_2(i) - d_2(j) = \omega(x_1(i) - x_1(j))$, t is a clock variable, and T is some bounded time.

The temporal safety invariant in (1) expresses that the circle phase of roundabout maneuver *always* stays collision-free indefinitely for an arbitrary number of aircraft. That is the most crucial part because we have to know the aircraft *always* remain safe during the actual roundabout collision avoidance circle. The condition $\forall i, j : A\ \mathcal{T}(i,j)$ characterizes compatible tangential maneuvering choices. Without a condition like $\mathcal{T}(i,j)$, roundabouts can be unsafe [20]. For a systematic derivation of how to construct $\mathcal{T}(i,j)$, we refer to the work [20]. As a variation of (1), the temporal safety property in (2) states that, for an arbitrary number of aircraft, the circle procedure of roundabout maneuver cannot produce collisions at any point in its bounded duration T. This variation restricts the continuous evolution to take exactly T time units (the evolution domain region restricts the evolution to $t \leq T$ and the subsequent test to $?(\forall i : A\ t = T)$) and no intermediate state is visible as a final state anymore. Thus, the temporal modality $\Box$ in (2) is truly necessary. We will use the techniques developed in this paper to verify these temporal safety properties in the distributed roundabout flight collision avoidance maneuver.

6 Proof Calculus for Temporal Properties

In this section, we introduce a sequent calculus for verifying temporal specifications of distributed hybrid systems in QdTL. With the basic idea being to perform a symbolic decomposition, the calculus transforms quantified hybrid programs successively into simpler logical formulas describing their effects. Statements about the temporal behavior of a quantified hybrid program are successively reduced to corresponding non-temporal statements about the intermediate states.

6.1 Proof Rules

In Fig. 2, we present a proof calculus for QdTL that inherits the proof rules of Qd$\mathcal{L}$ from [21,22] and adds new proof rules for temporal modalities. We use the sequent notation informally for a systematic proof structure. A *sequent* is of the form $\Gamma \rightarrow \Delta$, where the *antecedent* Γ and *succedent* Δ are finite sets of formulas. The semantics of $\Gamma \rightarrow \Delta$ is that of the formula $\bigwedge_{\phi \in \Gamma} \phi \rightarrow \bigvee_{\psi \in \Delta} \psi$ and will be treated as an abbreviation. As usual in sequent calculus, the proof rules are applied backwards from the *conclusion* (goal below horizontal bar) to the *premises* (subgoals above bar).

Inherited Non-temporal Rules. The QdTL calculus inherits the (non-temporal) Qd$\mathcal{L}$ proof rules. For propositional logic, standard rules *ax–cut* are

$$(ax)\ \frac{}{\phi \rightarrow \phi} \quad (\neg\mathrm{r})\ \frac{\phi \rightarrow}{\rightarrow \neg\phi} \quad (\neg\mathrm{l})\ \frac{\rightarrow \phi}{\neg\phi \rightarrow} \quad (\wedge\mathrm{r})\ \frac{\rightarrow \phi \quad \rightarrow \psi}{\rightarrow \phi \wedge \psi} \quad (\wedge\mathrm{l})\ \frac{\phi, \psi \rightarrow}{\phi \wedge \psi \rightarrow} \quad (cut)\ \frac{\rightarrow \phi \quad \phi \rightarrow}{\rightarrow}$$

$$([;])\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \quad (\langle;\rangle)\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \quad ([\cup])\ \frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi} \quad (\langle\cup\rangle)\ \frac{\langle\alpha\rangle\phi \vee \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi} \quad ([?])\ \frac{\chi \rightarrow \phi}{[?\chi]\phi} \quad (\langle?\rangle)\ \frac{\chi \wedge \phi}{\langle?\chi\rangle\phi}$$

$$(['])\ \frac{\forall t \geq 0((\forall 0 \leq \tilde{t} \leq t[\forall i : A\ \mathcal{S}(\tilde{t})]\chi) \rightarrow [\forall i : A\ \mathcal{S}(t)]\phi)}{[\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\phi}{}^{1} \quad (\langle'\rangle)\ \frac{\exists t \geq 0((\forall 0 \leq \tilde{t} \leq t\langle\forall i : A\ \mathcal{S}(\tilde{t})\rangle\chi) \wedge \langle\forall i : A\ \mathcal{S}(t)\rangle\phi)}{\langle\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi\rangle\phi}{}^{1}$$

$$([:=])\ \frac{\mathsf{if}\ \exists i : A\ \boldsymbol{s} = [\mathcal{A}]\boldsymbol{u}\ \mathsf{then}\ \forall i : A\ (\boldsymbol{s} = [\mathcal{A}]\boldsymbol{u} \rightarrow \phi(\theta))\ \mathsf{else}\ \phi(f([\mathcal{A}]\boldsymbol{u}))\ \mathsf{fi}}{\phi([\forall i : A\ f(\boldsymbol{s}) := \theta]f(\boldsymbol{u}))}{}^{2}$$

$$(\langle:=\rangle)\ \frac{\mathsf{if}\ \exists i : A\ \boldsymbol{s} = \langle\mathcal{A}\rangle\boldsymbol{u}\ \mathsf{then}\ \exists i : A\ (\boldsymbol{s} = \langle\mathcal{A}\rangle\boldsymbol{u} \wedge \phi(\theta))\ \mathsf{else}\ \phi(f(\langle\mathcal{A}\rangle\boldsymbol{u}))\ \mathsf{fi}}{\phi(\langle\forall i : A\ f(\boldsymbol{s}) := \theta\rangle f(\boldsymbol{u}))}{}^{2}$$

$$(skip)\ \frac{\varUpsilon([\forall i : A\ f(\boldsymbol{s}) := \theta]\boldsymbol{u})}{[\forall i : A\ f(\boldsymbol{s}) := \theta]\varUpsilon(\boldsymbol{u})}{}^{3} \quad ([:*])\ \frac{\forall j : A\ \phi(\theta)}{[\forall j : A\ n := \theta]\phi(n)} \quad (\langle:*\rangle)\ \frac{\exists j : A\ \phi(\theta)}{\langle\forall j : A\ n := \theta\rangle\phi(n)} \quad (ex)\ \frac{true}{\exists n : A\ \mathsf{E}(n) = 0}$$

$$(\forall\mathrm{r})\ \frac{\varGamma \rightarrow \phi(f(X_1,\ldots,X_n)), \varDelta}{\varGamma \rightarrow \forall x\phi(x), \varDelta}{}^{4} \quad (\exists\mathrm{r})\ \frac{\varGamma \rightarrow \phi(\theta), \exists x\phi(x), \varDelta}{\varGamma \rightarrow \exists x\phi(x), \varDelta}{}^{5} \quad (\forall\mathrm{l})\ \frac{\varGamma, \phi(\theta), \forall x\phi(x) \rightarrow \varDelta}{\varGamma, \forall x\phi(x) \rightarrow \varDelta}{}^{5} \quad (\exists\mathrm{l})\ \frac{\varGamma, \phi(f(X_1,\ldots,X_n)) \rightarrow \varDelta}{\varGamma, \exists x\phi(x) \rightarrow \varDelta}{}^{4}$$

$$(\mathrm{i}\forall)\ \frac{\mathrm{QE}(\forall X, Y\,(\mathsf{if}\ \boldsymbol{s} = \boldsymbol{t}\ \mathsf{then}\ \varPhi(X) \rightarrow \varPsi(X)\ \mathsf{else}\ \varPhi(X) \rightarrow \varPsi(Y)\ \mathsf{fi}))}{\varPhi(f(\boldsymbol{s})) \rightarrow \varPsi(f(\boldsymbol{t}))}{}^{6} \quad (\mathrm{i}\exists)\ \frac{\mathrm{QE}(\exists X \bigwedge_i(\varPhi_i \rightarrow \varPsi_i))}{\varPhi_1 \rightarrow \varPsi_1 \ \ldots\ \varPhi_n \rightarrow \varPsi_n}{}^{7}$$

$$([]gen)\ \frac{\phi \rightarrow \psi}{\varGamma, [\alpha]\phi \rightarrow [\alpha]\psi, \varDelta} \quad (\langle\rangle gen)\ \frac{\phi \rightarrow \psi}{\varGamma, \langle\alpha\rangle\phi \rightarrow \langle\alpha\rangle\psi, \varDelta} \quad (ind)\ \frac{\phi \rightarrow [\alpha]\phi}{\varGamma, \phi \rightarrow [\alpha^*]\phi, \varDelta} \quad (con)\ \frac{v > 0 \wedge \varphi(v) \rightarrow \langle\alpha\rangle\varphi(v-1)}{\varGamma, \exists v\,\varphi(v) \rightarrow \langle\alpha^*\rangle\exists v \leq 0\,\varphi(v), \varDelta}{}^{8}$$

$$(DI)\ \frac{\chi \rightarrow [\forall i : A\ f(\boldsymbol{s})' := \theta]D(\phi)}{\phi \rightarrow [\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\phi}{}^{9} \quad (DC)\ \frac{\phi \rightarrow [\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\psi \quad \phi \rightarrow [\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi \wedge \psi]\phi}{\phi \rightarrow [\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\phi}$$

$$([\cup]\Box)\ \frac{[\alpha]\pi \wedge [\beta]\pi}{[\alpha \cup \beta]\pi}{}^{10} \quad (\langle\cup\rangle\Diamond)\ \frac{\langle\alpha\rangle\pi \vee \langle\beta\rangle\pi}{\langle\alpha \cup \beta\rangle\pi}{}^{10} \quad ([;]\Box)\ \frac{[\alpha]\Box\phi \wedge [\alpha][\beta]\Box\phi}{[\alpha;\beta]\Box\phi} \quad (\langle;\rangle\Diamond)\ \frac{\langle\alpha\rangle\Diamond\phi \vee \langle\alpha\rangle\langle\beta\rangle\Diamond\phi}{\langle\alpha;\beta\rangle\Diamond\phi}$$

$$([?]\Box)\ \frac{\phi}{[?\chi]\Box\phi} \quad (\langle?\rangle\Diamond)\ \frac{\phi}{\langle?\chi\rangle\Diamond\phi}$$

$$([:=]\Box)\ \frac{\phi \wedge [\forall i : A\ f(\boldsymbol{s}) := \theta]\phi}{[\forall i : A\ f(\boldsymbol{s}) := \theta]\Box\phi} \quad (\langle:=\rangle\Diamond)\ \frac{\phi \vee \langle\forall i : A\ f(\boldsymbol{s}) := \theta\rangle\phi}{\langle\forall i : A\ f(\boldsymbol{s}) := \theta\rangle\Diamond\phi}$$

$$([']\Box)\ \frac{[\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\phi}{[\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi]\Box\phi} \quad (\langle'\rangle\Diamond)\ \frac{\langle\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi\rangle\phi}{\langle\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi\rangle\Diamond\phi}$$

$$([^{*n}]\Box)\ \frac{[\alpha;\alpha^*]\Box\phi}{[\alpha^*]\Box\phi} \quad (\langle^{*n}\rangle\Diamond)\ \frac{\langle\alpha;\alpha^*\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi} \quad ([^*]\Box)\ \frac{[\alpha^*][\alpha]\Box\phi}{[\alpha^*]\Box\phi} \quad (\langle^*\rangle\Diamond)\ \frac{\langle\alpha^*\rangle\langle\alpha\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi}$$

[1] $t, \tilde{t}$ are new variables, $\forall i : A\ \mathcal{S}(t)$ is the quantified assignment $\forall i : A\ f(\boldsymbol{s}) := y_{\boldsymbol{s}}(t)$ with solutions $y_{\boldsymbol{s}}(t)$ of the (injective) differential equations and $f(\boldsymbol{s})$ as initial values. See [21, 22] for the definition of a *injective* quantified assignment or quantified differential equation.

[2] Occurrence $f(\boldsymbol{u})$ in $\phi(f(\boldsymbol{u}))$ is not in scope of a modality (admissible substitution) and we abbreviate assignment $\forall i : A\ f(\boldsymbol{s}) := \theta$ by $\mathcal{A}$, which is assumed to be injective.

[3] $f \neq \varUpsilon$ and the quantified assignment $\forall i : A\ f(\boldsymbol{s}) := \theta$ is injective. The same rule applies for $\langle\forall i : A\ f(\boldsymbol{s}) := \theta\rangle$ instead of $[\forall i : A\ f(\boldsymbol{s}) := \theta]$.

[4] f is a new (Skolem) function and $X_1, \ldots, X_n$ are all free logical variables of $\forall x\,\phi(x)$.

[5] θ is an abbreviate term, often a new logical variable.

[6] X, Y are new variables of sort $\mathbb{R}$. QE needs to be applicable in the premises.

[7] Among all open branches, the free (existential) logical variable X of sort $\mathbb{R}$ only occurs in the branch $\varPhi_i \rightarrow \varPsi_i$. QE needs to be defined for the formula in the premises, especially, no Skolem dependencies on X occur.

[8] logical variable v does not occur in α.

[9] The operator D, as defined in [21, 22], is used to computer syntactic total derivations of formulas algebraically.

[10] π is a trace formula, whereas ϕ and ψ are (state) formulas. Unlike ϕ and ψ, the trace formula π may thus begin with a temporal modality $\Box$ or $\Diamond$.

Fig. 2. Rule schemata of the proof calculus for quantified differential temporal dynamic logic

listed in Fig. 2. Rules $[;]$–$\langle?\rangle$ work similar to those in [12]. Rules $[']$, $\langle'\rangle$ handle continuous evolutions for quantified differential equations with first-order definable solutions. Rules $[:=]$–$\langle:*\rangle$ handle discrete changes for quantified assignments. Axiom *ex* expresses that, for sort $A \neq \mathbb{R}$, there always is a new object n that has not been created yet ($\mathsf{E}(n) = 0$), because domains are infinite. The quantifier rules $\forall$r–i$\exists$ combine quantifier handling of many-sorted logic based on instantiation with theory reasoning by *quantifier elimination* (QE) for the theory of reals. The global rules $[]gen$, $\langle\rangle gen$ are Gödel generalization rules and *ind* is an induction schema for loops with *inductive invariant* ϕ [12]. Similarly, *con* generalizes Harel's convergence rule [12] to the hybrid case with decreasing variant φ [19]. *DI* and *DC* are rules for quantified differential equations with *quantified differential invariants* [21,22]. Notice that $[\cup]$, $\langle\cup\rangle$ can be generalized to apply to formulas of the form $[\alpha \cup \beta]\pi$ where π is an arbitrary trace formula, and not just a state formula as in Qd$\mathcal{L}$. Thus, π may begin with $\Box$ and $\Diamond$, which is why the rules are repeated in this generalized form as $[\cup]\Box$ and $\langle\cup\rangle\Diamond$ in Fig. 2.

Temporal Rules. The new temporal rules in Fig. 2 for the QdTL calculus successively transform temporal specifications of quantified hybrid programs into non-temporal Qd$\mathcal{L}$ formulas. The idea underlying this transformation is to decompose quantified hybrid programs and recursively augment intermediate state transitions with appropriate specifications.

Rule $[;]\Box$ decomposes invariants of $\alpha;\beta$ (i.e., $[\alpha;\beta]\Box\phi$ holds) into an invariant of α (i.e., $[\alpha]\Box\phi$) and an invariant of β that holds when β is started in *any* final state of α (i.e., $[\alpha]([\beta]\Box\phi)$). Its difference with the Qd$\mathcal{L}$ rule $[;]$ thus is that the QdTL rule $[;]\Box$ also checks safety invariant ϕ at the symbolic states in between the execution of α and β, and recursively so because of the temporal modality $\Box$. Rule $[:=]\Box$ expresses that invariants of quantified assignments need to hold before and after the discrete change (similarly for $[?]\Box$, except that tests do not lead to a state change, so ϕ holding before the test is all there is to it). Rule $[']\Box$ can directly reduce invariants of continuous evolutions to non-temporal formulas as restrictions of solutions of quantified differential equations are themselves solutions of different duration and thus already included in the continuous evolutions of $\forall i : A\ f(\boldsymbol{s})' = \theta$. The (optional) iteration rule $[^{*n}]\Box$ can partially unwind loops. It relies on rule $[;]\Box$. The dual rules $\langle\cup\rangle\Diamond$,$\langle;\rangle\Diamond$,$\langle:=\rangle\Diamond$,$\langle?\rangle\Diamond$,$\langle'\rangle\Diamond$,$\langle^{*n}\rangle\Diamond$ work similarly. Rules for handling $[\alpha]\Diamond\phi$ and $\langle\alpha\rangle\Box\phi$ are discussed in Section 8.

The inductive definition rules $[^*]\Box$ and $\langle^*\rangle\Diamond$ completely reduce temporal properties of loops to QdTL properties of standard non-temporal Qd$\mathcal{L}$ modalities such that standard induction (*ind*) or convergence (*con*) rules, as listed in Fig. 2, can be used for the outer non-temporal modality of the loop. Hence, after applying the inductive loop definition rules $[^*]\Box$ and $\langle^*\rangle\Diamond$, the standard Qd$\mathcal{L}$ loop invariant and variant rules can be used for verifying temporal properties of loops without change, except that the postcondition contains temporal modalities.

6.2 Soundness and Completeness

The following result shows that verification with the QdTL calculus always produces correct results about the safety of distributed hybrid systems, i.e., the QdTL calculus is sound.

Theorem 1 (Soundness of QdTL). *The* QdTL *calculus is sound, i.e., every* QdTL *(state) formula that can be proven is valid.*

The verification for temporal safety ($[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$), temporal liveness ($[\alpha]\Diamond\phi$ or $\langle\alpha\rangle\Box\phi$), and non-temporal ($[\alpha]\phi$ or $\langle\alpha\rangle\phi$) fragments of distributed hybrid systems has *three independent sources* of undecidability. Thus, no verification technique can be effective. Hence, QdTL cannot be effectively axiomatizable. Both its discrete and its continuous fragments alone are subject to Gödel's incompleteness theorem [19]. The fragment with only structural and dimension-changing dynamics is not effective either, because it can encode two-counter machines.

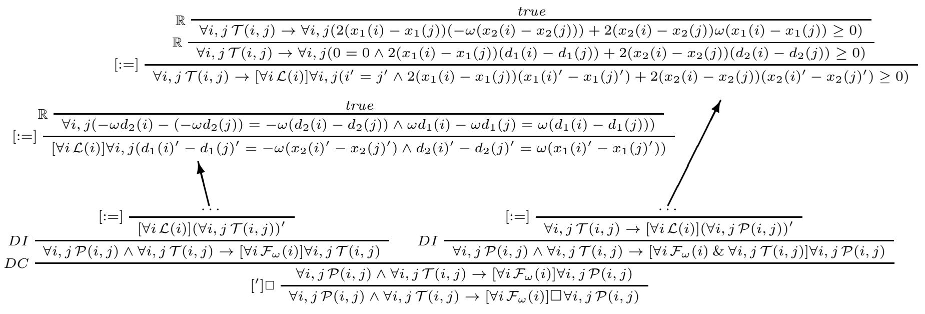
Qd$\mathcal{L}$ has been proved to be complete relative to quantified differential equations [21,22]. Due to the modular construction of the QdTL calculus, we can lift the relative completeness result from Qd$\mathcal{L}$ to QdTL. We essentially show that QdTL is complete relative to Qd$\mathcal{L}$, which directly implies that QdTL calculus is even complete relative to an oracle for the fragment of QdTL that has only quantified differential equations in modalities. Again, we restrict our attention to homogeneous combinations of path and trace quantifiers like $[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$.

Theorem 2 (Relative Completeness of QdTL). *The calculus in Fig. 2 is a complete axiomatization of* QdTL *relative to quantified differential equations.*

This result shows that both temporal and non-temporal properties of distributed hybrid systems can be proven to exactly the same extent to which properties of quantified differential equations can be proven. It also gives a formal justification that the QdTL calculus reduces temporal properties to non-temporal Qd$\mathcal{L}$ properties.

7 Verification of Distributed Air Traffic Control Safety Properties

Continuing the distributed air traffic control study from Section 5, the QdTL proofs of the temporal safety invariant in (1) and the temporal safety property in (2) are presented in Fig. 3 and Fig. 4, respectively (for the purpose of simplifying the presentation, we ignore typing information A for aircraft in the proof, because it is clear from the context). Note that temporal and non-temporal properties of the maneuver cannot be proven using any hybrid systems verification technique, because the dimension is parametric and unbounded and may even change dynamically during the remainder of the maneuver. The single proof in Fig. 3 or Fig. 4 corresponds to infinitely many proofs for systems with n aircraft for all n.



$$[:=]\ \dfrac{\mathbb{R}\ \dfrac{\mathbb{R}\ \dfrac{true}{\forall i,j\,\mathcal{T}(i,j) \to \forall i,j(2(x_1(i)-x_1(j))(-\omega(x_2(i)-x_2(j))) + 2(x_2(i)-x_2(j))\omega(x_1(i)-x_1(j)) \geq 0)}}{\forall i,j\,\mathcal{T}(i,j) \to \forall i,j(0=0 \land 2(x_1(i)-x_1(j))(d_1(i)-d_1(j)) + 2(x_2(i)-x_2(j))(d_2(i)-d_2(j)) \geq 0)}}{\forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{L}(i)]\forall i,j(i'=j' \land 2(x_1(i)-x_1(j))(x_1(i)'-x_1(j)') + 2(x_2(i)-x_2(j))(x_2(i)'-x_2(j)') \geq 0)}$$

$$[:=]\ \dfrac{\mathbb{R}\ \dfrac{true}{\forall i,j(-\omega d_2(i) - (-\omega d_2(j)) = -\omega(d_2(i)-d_2(j)) \land \omega d_1(i) - \omega d_1(j) = \omega(d_1(i)-d_1(j)))}}{[\forall i\,\mathcal{L}(i)]\forall i,j(d_1(i)'-d_1(j)' = -\omega(x_2(i)'-x_2(j)') \land d_2(i)'-d_2(j)' = \omega(x_1(i)'-x_1(j)'))}$$

$$[']\Box\ \dfrac{DC\ \dfrac{DI\ \dfrac{[:=]\ \dfrac{\dots}{[\forall i\,\mathcal{L}(i)](\forall i,j\,\mathcal{T}(i,j))'}}{\forall i,j\,\mathcal{P}(i,j) \land \forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{F}_\omega(i)]\forall i,j\,\mathcal{T}(i,j)} \qquad DI\ \dfrac{[:=]\ \dfrac{\dots}{\forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{L}(i)](\forall i,j\,\mathcal{P}(i,j))'}}{\forall i,j\,\mathcal{P}(i,j) \land \forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{F}_\omega(i)\ \&\ \forall i,j\,\mathcal{T}(i,j)]\forall i,j\,\mathcal{P}(i,j)}}{\forall i,j\,\mathcal{P}(i,j) \land \forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{F}_\omega(i)]\forall i,j\,\mathcal{P}(i,j)}}{\forall i,j\,\mathcal{P}(i,j) \land \forall i,j\,\mathcal{T}(i,j) \to [\forall i\,\mathcal{F}_\omega(i)]\Box\forall i,j\,\mathcal{P}(i,j)}$$

Abbreviation: $\mathcal{L}(i) \equiv x_1(i)' := d_1(i), x_2(i)' := d_2(i), d_1(i)' := -\omega d_2(i), d_2(i)' := \omega d_1(i)$

Fig. 3. Proof for temporal collision freedom of roundabout collision avoidance maneuver circle

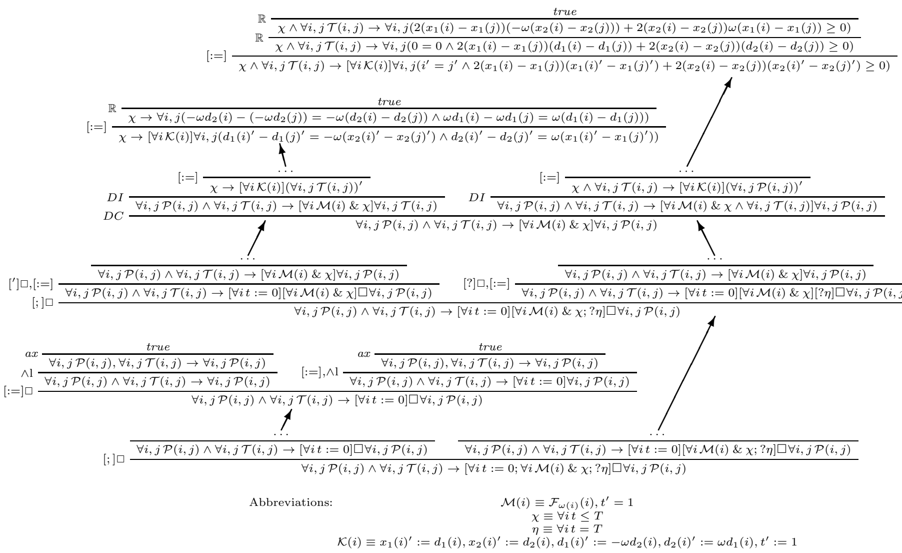


Fig. 4. Proof for temporal collision freedom of roundabout collision avoidance maneuver circle in bounded time

Our proofs show that the distributed roundabout maneuver always safely avoids collisions for arbitrarily many aircraft (even with dynamic appearance of new aircraft). The above maneuver still requires all aircraft in the horizon of relevance to participate in the collision avoidance maneuver. In fact, we can show that this is unnecessary for aircraft that are far enough away and that may be engaged in other roundabouts. Yet, this is beyond the scope of this paper.

8 Liveness by Quantifier Alternation

Liveness specifications of the form $[\alpha]\Diamond\phi$ or $\langle\alpha\rangle\Box\phi$ are sophisticated (Σ_1^1-hard because they can express infinite occurrence in Turing machines). Beckert and Schlager [2], for instance, note that they failed to find sound rules for a discrete case that corresponds to $[\alpha;\beta]\Diamond\phi$.

For *finitary liveness semantics*, we can still find proof rules. In this section, we modify the meaning of $[\alpha]\Diamond\phi$ to refer to all *terminating* traces of α. Then, the straightforward generalization $[;]\diamond$ in Fig. 5 is sound, even in the hybrid case. But $[;]\diamond$ still leads to an incomplete axiomatization as it does not cover the case where, in some traces, ϕ becomes true at some point during α, and in other traces, ϕ only becomes true during β. To overcome this limitation, we use a program transformation approach. We instrument the quantified hybrid program to monitor the occurrence of ϕ during all changes: In $[\alpha]\diamond$, $\breve{\alpha}$ results from replacing all occurrences of $\forall i : A\ f(\boldsymbol{s}) := \theta$ with $\forall i : A\ f(\boldsymbol{s}) := \theta; ?(\phi \to t = 1)$ and $\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi$ with $\forall i : A\ f(\boldsymbol{s})' = \theta \,\&\, \chi \land (\phi \to t = 1)$. The latter is a continuous evolution restricted to the region that satisfies χ and $\phi \to t = 1$.

The effect is that t detects whether ϕ has occurred during any change in α. In particular, t is guaranteed to be 1 after all runs if ϕ occurs at least once along all traces of α.

$$([;]\Diamond)\ \frac{\to [\alpha]\Diamond\phi, [\alpha][\beta]\Diamond\phi}{\to [\alpha;\beta]\Diamond\phi} \qquad ([\alpha]\Diamond)\ \frac{\phi \lor \forall t:\mathbb{R}\ [\check{\alpha}]t = 1}{[\alpha]\Diamond\phi}$$

Fig. 5. Transformation rules for alternating temporal path and trace quantifiers

9 Conclusions and Future Work

For reasoning about distributed hybrid systems, we have introduced a temporal dynamic logic, QdTL, with modal path quantifiers over traces and temporal quantifiers along the traces. It combines the capabilities of quantified differential dynamic logic to reason about possible distributed hybrid system behavior with the power of temporal logic in reasoning about the behavior along traces. Furthermore, we have presented a proof calculus for verifying temporal safety specifications of quantified hybrid programs in QdTL.

Our sequent calculus for QdTL is a completely modular combination of temporal and non-temporal reasoning. Temporal formulas are handled using rules that augment intermediate state transitions with corresponding sub-specifications. Purely non-temporal rules handle the effects of discrete transitions, continuous evolutions, and structural/dimension changes. The modular nature of the QdTL calculus further enables us to lift the relative completeness result from Qd$\mathcal{L}$ to QdTL. This theoretical result shows that the QdTL calculus is a sound and complete axiomatization of the temporal behavior of distributed hybrid systems relative to differential equations. As an example, we demonstrate that our logic is suitable for reasoning about temporal safety properties in a distributed air traffic control system.

We are currently extending our verification tool for distributed hybrid systems, which is an automated theorem prover called KeYmaeraD [25], to cover the full QdTL calculus. Future work includes extending QdTL with CTL*-like [10] formulas of the form $[\alpha](\psi \land \Box\phi)$ to avoid splitting of the proof into two very similar sub-proofs for temporal parts $[\alpha]\Box\phi$ and non-temporal parts $[\alpha]\psi$ arising in $[;]\Box$. Our combination of temporal logic with dynamic logic is more suitable for this purpose than the approach in [2], since QdTL has uniform modalities and uniform semantics for temporal and non-temporal specifications. This extension will also simplify the treatment of alternating liveness quantifiers conceptually.

References

1. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking for real-time systems. In: LICS, pp. 414–425. IEEE Computer Society (1990)
2. Beckert, B., Schlager, S.: A Sequent Calculus for First-Order Dynamic Logic with Trace Modalities. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 626–641. Springer, Heidelberg (2001)

3. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
4. Damm, W., Hungar, H., Olderog, E.-R.: On the Verification of Cooperating Traffic Agents. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2003. LNCS, vol. 3188, pp. 77–110. Springer, Heidelberg (2004)
5. Davoren, J.M., Coulthard, V., Markey, N., Moor, T.: Non-deterministic Temporal Logics for General Flow Systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 280–295. Springer, Heidelberg (2004)
6. Davoren, J.M., Nerode, A.: Logics for hybrid systems. Proceedings of the IEEE 88(7), 985–1010 (2000)
7. Deshpande, A., Göllü, A., Varaiya, P.: SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In: Hybrid Systems, pp. 113–133 (1996)
8. Dowek, G., Muñoz, C., Carreño, V.A.: Provably safe coordinated strategy for distributed conflict resolution. In: AIAA Proceedings, AIAA-2005-6047, pp. 278–292 (2005)
9. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Sci. Comput. Program. 2(3), 241–266 (1982)
10. Emerson, E.A., Halpern, J.Y.: “Sometimes” and “Not Never” revisited: on branching versus linear time temporal logic. J. ACM 33(1), 151–178 (1986)
11. Faber, J., Meyer, R.: Model checking data-dependent real-time properties of the European Train Control System. In: FMCAD, pp. 76–77. IEEE Computer Society Press (November 2006)
12. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. MIT Press, Cambridge (2000)
13. Henzinger, T.A.: The theory of hybrid automata. In: LICS, pp. 278–292 (1996)
14. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. In: LICS, pp. 394–406. IEEE Computer Society (1992)
15. Hsu, A., Eskafi, F., Sachs, S., Varaiya, P.: Design of platoon maneuver protocols for IVHS. Technical Report PATH Research Report UCB-ITS-PRR-91-6, UC Berkeley (1991)
16. Kratz, F., Sokolsky, O., Pappas, G.J., Lee, I.: R-Charon, a Modeling Language for Reconfigurable Hybrid Systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 392–406. Springer, Heidelberg (2006)
17. Leivant, D.: Partial Correctness Assertions Provable in Dynamic Logics. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 304–317. Springer, Heidelberg (2004)
18. Mysore, V., Piazza, C., Mishra, B.: Algorithmic Algebraic Model Checking II: Decidability of Semi-algebraic Model Checking and Its Applications to Systems Biology. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 217–233. Springer, Heidelberg (2005)
19. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. 41(2), 143–189 (2008)
20. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. 20(1), 309–352 (2010)
21. Platzer, A.: Quantified Differential Dynamic Logic for Distributed Hybrid Systems. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 469–483. Springer, Heidelberg (2010)
22. Platzer, A.: Quantified differential dynamic logic for distributed hybrid systems. Technical Report CMU-CS-10-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (May 2010)

23. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE (1977)
24. Pratt, V.R.: Process logic. In: POPL, pp. 93–100 (1979)
25. Renshaw, D.W., Loos, S.M., Platzer, A.: Distributed Theorem Proving for Distributed Hybrid Systems. In: Qin, S., Qiu, Z. (eds.) ICFEM 2011. LNCS, vol. 6991, pp. 356–371. Springer, Heidelberg (2011)
26. Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management: a study in multi-agent hybrid systems. IEEE T. Automat. Contr. 43(4), 509–521 (1998)
27. van Beek, D.A., Man, K.L., Reniers, M.A., Rooda, J.E., Schiffelers, R.R.H.: Syntax and consistent equation semantics of hybrid chi. J. Log. Algebr. Program. 68(1-2), 129–210 (2006)

Computational Hardness of Validity in Probability Logic

Rutger Kuyper[⋆]

Radboud University Nijmegen
Department of Mathematics,
P.O. Box 9010
6500 GL Nijmegen,
The Netherlands
r.kuyper@math.ru.nl

Abstract. We consider the complexity of validity in ε-logic, a probability logic introduced by Terwijn. We prove that the set of valid formulas is Π^1_1-hard, improving a previous undecidability result by Terwijn.

1 Introduction

Over the years, there have been many attempts at combining logic and probability through so-called *probability logics*. We study the computational aspects of a probability logic introduced by Terwijn in [4]. This logic has two main characteristics whose combination sets it apart from earlier attempts: first, the logic is closely related to probabilistic induction and Valiant's pac-model; and second, it is a probabilistic interpretation of first-order logic instead of a probability logic with an entirely new syntax. Terwijn's probability logic depends on a fixed error parameter ε and is hence called *ε-logic*.

Valiant [6] also introduced a probability logic related to his pac-model. Nonetheless, the logic most closely related to ε-logic is the logic $\mathcal{L}_{\omega P}$ introduced by Keisler, surveyed in Keisler [2]. This logic contains quantifiers of the form $(Px \geq r)$ which should be read as "holds for at least measure r many x". However, Keisler's logic does not contain the classical universal and existential quantifiers and does not attempt to model probabilistic induction in any way. Nevertheless, it turns out we can adapt some of the ideas used to prove results about $\mathcal{L}_{\omega P}$ to obtain similar results for ε-logic. For a discussion of more probability logics related to ours, we refer to the introduction of Kuyper and Terwijn [3].

Unfortunately, it turns out that ε-logic is computationally quite hard. Previously, Terwijn [5] has shown that the set of ε-tautologies is undecidable. This should not be too surprising, because this is of course also the case for classical first-order logic. For $\mathcal{L}_{\omega P}$, Hoover [1] has shown that validity is Π^1_1-complete

[⋆] Research supported by the Dutch mathematics research cluster DIAMANT and by the John Templeton Foundation Grant ID#15619: 'Mind, Mechanism and Mathematics: Turing Centenary Research Project'.

(i.e. of the same complexity as first-order arithmetic with second-order universal quantifiers), which is computationally much harder than first-order logic. In this paper, we will combine some of his ideas with our own to show that ε-logic is Π^1_1-hard. This shows that ε-logic is computationally much harder than first-order logic and that we cannot hope to find an effective calculus for it.

In the next section we will briefly recall the definition of ε-logic and some elementary facts. After that, we will prove in section 3 that there exist many-one reductions between ε_0-logic and ε_1-logic for $\varepsilon_0 \neq \varepsilon_1$. Finally, in section 4 we will prove that ε-logic is Π^1_1-hard by utilising these reductions.

2 ε-Logic

As mentioned above, ε-logic was introduced in Terwijn [4]. Afterwards, the definition has gone through a few minor modifications. We use the definition from Kuyper and Terwijn [3]. For a discussion of this definition and more information about ε-logic, we refer to the same paper.

Definition 2.1. Let $\mathcal{L}$ be a first-order language, possibly containing equality, of a countable signature. Let $\varphi = \varphi(x_1, \dots, x_n)$ be a first-order formula in the language $\mathcal{L}$, and let $\varepsilon \in [0,1]$. Furthermore, let $\mathcal{M}$ be a classical first-order model for $\mathcal{M}$ and let $\mathcal{D}$ be a probability measure on $\mathcal{M}$. Then we inductively define the notion of *ε-truth*, denoted by $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi$, as follows (where we leave the parameters implicit).

(i) For every atomic formula φ:

$$(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \text{ if } \mathcal{M} \models \varphi.$$

(ii) We treat the logical connectives $\wedge$ and $\vee$ classically, e.g.

$$(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \wedge \psi \text{ if } (\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \text{ and } (\mathcal{M}, \mathcal{D}) \models_\varepsilon \psi.$$

(iii) The existential quantifier is treated classically as well:

$$(\mathcal{M}, \mathcal{D}) \models_\varepsilon \exists x \varphi(x)$$

if there exists an $a \in \mathcal{M}$ such that $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(a)$.

(iv) The case of negation is split into sub-cases as follows:
  (a) For φ atomic, $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\varphi$ if $(\mathcal{M}, \mathcal{D}) \not\models_\varepsilon \varphi$.
  (b) $\neg$ distributes in the classical way over $\wedge$ and $\vee$, e.g.

$$(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg(\varphi \wedge \psi) \text{ if } (\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\varphi \vee \neg\psi.$$

  (c) $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\neg\varphi$ if $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi$.
  (d) $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg(\varphi \to \psi)$ if $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \wedge \neg\psi$.
  (e) $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\exists x \varphi(x)$ if $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \forall x \neg\varphi(x)$.
  (f) $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\forall x \varphi(x)$ if $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \exists x \neg\varphi(x)$.

(v) $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \to \psi$ if $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\varphi \vee \psi$.

(vi) Finally, we define $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \forall x \varphi(x)$ if

$$\Pr_{\mathcal{D}}\big[a \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(a)\big] \geq 1 - \varepsilon.$$

Thus, the crucial change is that the universal quantifier is not treated classically: instead of saying that we have $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(a)$ for *all* elements $a \in \mathcal{M}$, we merely say that it holds for "many" of the elements, where "many" depends on the error parameter ε.

The main reason for this change is that we want our logic to be *learnable*, in the sense defined in Terwijn [4] (whose definition of learning is closely related to Valiant's pac-model). We do not want to add the classical universal quantifier to our logic, since it is impossible to decide if a universal quantifier holds from just a finite amount of information. Therefore we take special care in defining our negation: we do not want $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \neg\exists x \varphi(x)$ to mean $(\mathcal{M}, \mathcal{D}) \not\models_\varepsilon \exists x \varphi(x)$, because the latter is equivalent to saying that $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(x)$ holds *classically* for all $x \in \mathcal{M}$, which is exactly what we wanted to avoid. We define our negation in such a way that it still behaves in a classical way on the propositional level, while it interchanges the existential and universal quantifiers.

One other important point is that both $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \forall x \varphi(x)$ and $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \exists x \neg\varphi(x)$ may hold simultaneously, i.e. both a formula and its negation might hold at the same time. Thus, ε-logic is *paraconsistent*. For our current work this has one important implication: it is no longer the case that φ is satisfiable if and only if its negation $\neg\varphi$ is not a tautology, as demonstrated in Example 2.4 below.

To make sure that all necessary sets are measurable, we need to restrict ourselves to the right set of models.

Definition 2.2. Let $\mathcal{L}$ be a first-order language of a countable signature, possibly containing equality, and let $\varepsilon \in [0, 1]$. Then an *ε-model* $(\mathcal{M}, \mathcal{D})$ for the language $\mathcal{L}$ consists of a classical first-order $\mathcal{L}$-model $\mathcal{M}$ together with a probability distribution $\mathcal{D}$ over $\mathcal{M}$ such that:

(i) For all formulas $\varphi = \varphi(x_1, \dots, x_n)$ and all $a_1, \dots, a_{n-1} \in \mathcal{M}$, the set

$$\{a_n \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(a_1, \dots, a_n)\}$$

is $\mathcal{D}$-measurable (i.e. all definable sets of dimension 1 are measurable).

(ii) All relations of arity n are $\mathcal{D}^n$-measurable (including equality, if it is in $\mathcal{L}$) and all functions of arity n are measurable as functions from $(\mathcal{M}^n, \mathcal{D}^n)$ to $(\mathcal{M}, \mathcal{D})$ (where $\mathcal{D}^n$ denotes the n-fold product measure). In particular, constants are $\mathcal{D}$-measurable.

A *probability model* is a pair $(\mathcal{M}, \mathcal{D})$ that is an ε-model for every $\varepsilon \in [0, 1]$.

Definition 2.3. A formula $\varphi(x_1, \dots, x_n)$ is an *ε-tautology* or is *ε-valid* (notation: $\models_\varepsilon \varphi$) if for all probability models $(\mathcal{M}, \mathcal{D})$ and all $a_1, \dots, a_n \in \mathcal{M}$ it holds that $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi(a_1, \dots, a_n)$. Similarly, we say that φ is *ε-satisfiable* if there exists a probability model $(\mathcal{M}, \mathcal{D})$ and there exist $a_1, \dots, a_n \in \mathcal{M}$ such that $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi$.

We remark that, for satisfiability, it would be equivalent to use the more satisfying definition of letting φ be ε-satisfiable if there exists just an ε-model $(\mathcal{M}, \mathcal{D})$ such that $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi$, as follows directly from Kuyper and Terwijn [3, Proposition 5.1 and Theorem 5.9]. Unfortunately, we do not know of a similar result for validity. Our proof below needs our models to be probability models, hence our choice for this definition of an ε-tautology.

Example 2.4. Let Q be a unary predicate. Then $\varphi = \forall x Q(x) \vee \forall x \neg Q(x)$ is a $\frac{1}{2}$-tautology. Namely, in every probability model, either the set on which Q holds or its complement has measure at least $\frac{1}{2}$. However, φ is not an ε-tautology for $\varepsilon < \frac{1}{2}$. Furthermore, both φ and $\neg\varphi$ are classically satisfiable and hence ε-satisfiable for every ε; in particular we see that φ can be an ε-tautology while simultaneously $\neg\varphi$ is ε-satisfiable.

The next result will be used below.

Proposition 2.5. (Terwijn [4]) *Every formula φ is semantically equivalent to a formula φ' in prenex normal form; i.e. $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi \Leftrightarrow (\mathcal{M}, \mathcal{D}) \models_\varepsilon \varphi'$ for all $\varepsilon \in [0, 1]$ and all ε-models $(\mathcal{M}, \mathcal{D})$.*

Note that 1-logic is fairly trivial: every formula in prenex normal form containing a universal quantifier is trivially true, so the only interesting fragment is the existential fragment, which is just the classical fragment. It turns out that 0-validity also does not contain much interesting information.

Proposition 2.6. (Terwijn [5, Proposition 3.2]) *The 0-valid formulas coincide with the classically valid formulas.*

That the 0-tautologies and classical tautologies coincide does not mean that 0-logic is the same as classical logic, because in a fixed 0-model $(\mathcal{M}, \mathcal{D})$ it might be the case that some statement $\varphi(x)$ holds for almost all elements of the model, but not for all; hence $(\mathcal{M}, \mathcal{D}) \models_\varepsilon \forall x \varphi(x)$ but not $\mathcal{M} \models \forall x \varphi(x)$.

Thus, 0-logic and 1-logic have been taken care of from a computational point of view (the first is computably enumerable, while the second is decidable). This paper will deal with rational $\varepsilon \in (0, 1)$.

3 Many-One Reductions between Different ε

In this section we will show that for rational $\varepsilon_0, \varepsilon_1 \in (0, 1)$, the ε_0-tautologies many-one reduce to the ε_1-tautologies. Not only does this show that we only need to consider one fixed ε for our hardness results (we will take $\varepsilon = \frac{1}{2}$ below), but in our proof of the Π^1_1-hardness of ε-validity these reductions will also turn out to be useful in a different way.

We will begin with reducing to bigger ε_1. To do this, we refine the argument by Terwijn [5], where it is shown that the 0-tautologies many-one reduce to the ε-tautologies for $\varepsilon \in [0, 1)$. Our argument is similar to the one given in Kuyper and Terwijn [3], where we discuss reductions for satisfiability instead of for validity.

Theorem 3.1. *Let $\mathcal{L}$ be a countable first-order language not containing equality. Then, for all rationals $0 \leq \varepsilon_0 \leq \varepsilon_1 < 1$, the ε_0-tautologies many-one reduce to the ε_1-tautologies.*

Proof. We can choose integers n and $0 < m \leq n$ so that $\frac{m}{n} = \frac{1-\varepsilon_1}{1-\varepsilon_0}$. Let $\varphi(y_1, \dots, y_k)$ be a formula in prenex normal form (see Proposition 2.5). For simplicity we write $\vec{y} = y_1, \dots, y_k$. Also, for a function π we let $\pi(\vec{y})$ denote the vector $\pi(y_1), \dots, \pi(y_k)$. We use formula-induction to define a computable function f such that for every formula φ,

$$\varphi \text{ is an } \varepsilon_0\text{-tautology if and only if } f(\varphi) \text{ is an } \varepsilon_1\text{-tautology.} \tag{1}$$

For propositional formulas and existential quantifiers, there is nothing to be done and we use the identity map. Next, we consider the universal quantifiers. Let $\varphi = \forall x \psi(\vec{y}, x)$. The idea is to introduce new unary predicates, that can be used to vary the strength of the universal quantifier. We will make these predicates split the model into disjoint parts. If we split it into just the right number of parts (in this case n), then we can choose m of these parts to get just the right strength.

So, we introduce new unary predicates $X_1, \dots, X_n$. We define the sentence n-split by:

$$\forall x \left((X_1(x) \vee \ldots \vee X_n(x)) \wedge \bigwedge_{1 \leq i < j \leq n} \neg (X_i(x) \wedge X_j(x)) \right).$$

Then one can verify that in any model, $\neg n$-split does *not* hold if and only if the sets X_i disjointly cover the entire model.

Now define $f(\varphi)$ to be the formula

$$\neg n\text{-split} \vee \bigvee_{i_1, \dots, i_m} \forall x \big((X_{i_1}(x) \vee \cdots \vee X_{i_m}(x)) \wedge f(\psi)(\vec{y}, x) \big)$$

where the disjunction is over all subsets of size m from $\{1, \dots, n\}$. (It will be clear from the construction that $f(\psi)$ has the same arity as ψ.) Thus, $f(\varphi)$ expresses that for some choice of m of the n parts, $f(\psi)(x)$ holds often enough when restricted to the resulting part of the model.

We will now prove claim (1) above. For the implication from right to left, we will prove the following strengthening:

For every formula $\varphi(\vec{y})$ and every probability model $(\mathcal{M}, \mathcal{D})$ there exists a probability model $(\mathcal{N}, \mathcal{E})$ together with a measure-preserving surjective measurable function $\pi : \mathcal{N} \to \mathcal{M}$ (i.e. for all $\mathcal{D}$-measurable A we have that $\mathcal{E}(\pi^{-1}(A)) = \mathcal{D}(A)$) such that for all $\vec{y} \in \mathcal{N}$ we have that

$$(\mathcal{N}, \mathcal{E}) \not\models_{\varepsilon_1} f(\varphi)(\vec{y}) \textit{ if and only if } (\mathcal{M}, \mathcal{D}) \not\models_{\varepsilon_0} \varphi(\pi(\vec{y})).$$

In particular, if $f(\varphi)$ is an ε_1-tautology, then φ is an ε_0-tautology. We prove this by formula-induction over the formulas in prenex normal form. For propositional formulas, there is nothing to be done (we can simply take the models to be equal and π the identity). For the existential quantifier, let $\varphi = \forall x \psi(x)$ and apply the induction hypothesis to ψ to find a model $(\mathcal{N}, \mathcal{E})$ and a mapping π. Then we can take the same model and mapping for φ, as easily follows from the fact that π is surjective.

Next, we consider the universal quantifier. Suppose $\varphi = \forall x \psi(\vec{y}, x)$ and let $(\mathcal{M}, \mathcal{D})$ be a probability model. Use the induction hypothesis to find a model $(\mathcal{N}, \mathcal{E})$ and a measure-preserving surjective measurable function $\pi : \mathcal{N} \to \mathcal{M}$ such that for all $\vec{y}, x \in \mathcal{N}$ we have that

$$(\mathcal{N}, \mathcal{E}) \not\models_{\varepsilon_1} f(\psi)(\vec{y}, x) \text{ if and only if } (\mathcal{M}, \mathcal{D}) \not\models_{\varepsilon_0} \varphi(\pi(\vec{y}), \pi(x)).$$

Now form the probability model $(\mathcal{N}', \mathcal{E}')$ which consists of n disjoint copies $\mathcal{N}_1, \dots, \mathcal{N}_n$ of $(\mathcal{N}, \mathcal{E})$, each with weight $\frac{1}{n}$. That is, $\mathcal{E}'$ is the sum of n copies of $\frac{1}{n}\mathcal{E}$. Let $\pi' : \mathcal{N}' \to \mathcal{M}$ be the composition of the projection map $\sigma : \mathcal{N}' \to \mathcal{N}$ with π. Relations in $\mathcal{N}'$ are defined just as on $\mathcal{N}$, that is, for a t-ary relation R we define $R^{\mathcal{N}'}(x_1, \dots, x_t)$ by $R^{\mathcal{N}}(\sigma(x_1), \dots, \sigma(x_t))$. Observe that this is the same as defining $R^{\mathcal{N}'}(x_1, \dots, x_t)$ by $R^{\mathcal{M}}(\pi'(x_1), \dots, \pi'(x_t))$. We interpret constants $c^{\mathcal{N}'}$ by embedding $c^{\mathcal{N}}$ into the first copy $\mathcal{N}_1$. For functions f of arity t, first note that we can see $f^{\mathcal{N}}$ as a function from $\mathcal{N}^t \to \mathcal{N}'$ by embedding its codomain $\mathcal{N}$ into the first copy $\mathcal{N}_1$. We now interpret $f^{\mathcal{N}'}$ as the composition of this $f^{\mathcal{N}}$ with π'. Finally, we let each X_i be true exactly on the copy $\mathcal{N}_i$.

Then π' is clearly surjective. To show that it is measure-preserving, it is enough to show that σ is measure-preserving. If A is $\mathcal{E}$-measurable, then $\sigma^{-1}(A)$ consists of n disjoint copies of A, each having measure $\frac{1}{n}\mathcal{E}(A)$, so $\pi^{-1}(A)$ has $\mathcal{E}'$-measure exactly $\mathcal{E}(A)$.

Now, since $(\mathcal{N}', \mathcal{E}')$ does not satisfy $\neg n$-split (because the X_i disjointly cover $\mathcal{N}'$), we see that

$$(\mathcal{N}', \mathcal{E}') \not\models_{\varepsilon_1} f(\varphi)(\vec{y}) \tag{2}$$

is equivalent to the statement that for all $1 \le i_1 < \cdots < i_m \le n$ we have

$$\Pr_{\mathcal{E}'}\big[x \in \mathcal{N}' \mid (\mathcal{N}', \mathcal{E}') \not\models_{\varepsilon_1} (X_{i_1}(x) \vee \cdots \vee X_{i_m}(x)) \wedge f(\psi)(\vec{y}, x)\big] > \varepsilon_1. \tag{3}$$

By Lemma 3.2 below we have that $(\mathcal{N}', \mathcal{E}') \models_{\varepsilon_1} f(\psi)(\vec{y}, x)$ holds if and only if $(\mathcal{N}, \mathcal{E}) \models_{\varepsilon_1} f(\psi)(\sigma(\vec{y}), \sigma(x))$ holds. In particular, we see for every $1 \le i \le n$ that

$$\begin{aligned}&\Pr_{\mathcal{E}'}\big[x \in \mathcal{N}' \mid (\mathcal{N}', \mathcal{E}') \models_{\varepsilon_1} X_i(x) \text{ and } (\mathcal{N}', \mathcal{E}') \not\models_{\varepsilon_1} f(\psi)(\vec{y}, x)\big] \\ &= \frac{1}{n} \Pr_{\mathcal{E}}\big[x \in \mathcal{N} \mid (\mathcal{N}, \mathcal{E}) \not\models_{\varepsilon_1} f(\psi)(\sigma(\vec{y}), x)\big].\end{aligned}$$

It follows that (3) is equivalent to

$$\frac{n-m}{n} + \frac{m}{n} \Pr_{\mathcal{E}}\big[x \in \mathcal{N} \mid (\mathcal{N}, \mathcal{E}) \not\models_{\varepsilon_1} f(\psi)(\sigma(\vec{y}), x)\big] > \varepsilon_1.$$

The induction hypothesis tells us that this is equivalent to

$$\frac{n-m}{n} + \frac{m}{n} \Pr_{\mathcal{E}}\big[x \in \mathcal{N} \mid (\mathcal{M}, \mathcal{D}) \not\models_{\varepsilon_0} \psi(\pi(\sigma(\vec{y})), \pi(x))\big] > \varepsilon_1$$

and since π is surjective and measure-preserving, this is the same as

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_0} \psi(\pi(\sigma(\vec{y})),x)\big] > \frac{n}{m}\left(\varepsilon_1 - \frac{n-m}{n}\right)$$
$$= \frac{n}{m}(\varepsilon_1 - 1) + 1 = \varepsilon_0.$$

This proves that we have $(\mathcal{N}',\mathcal{E}') \not\models_{\varepsilon_1} f(\varphi)(\vec{y})$ if and only if $(\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_0} \varphi(\pi'(\vec{y}))$.

To prove the left to right direction of (1) we will use induction to prove the following stronger statement:

If $(\mathcal{M},\mathcal{D})$ is a probability model and $\vec{y} \in \mathcal{M}$ are such that $(\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_1} f(\varphi)(\vec{y})$, then we also have $(\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_0} \varphi(\vec{y})$.

In particular, if φ is an ε_0-tautology, then $f(\varphi)$ is an ε_1-tautology. The only interesting case is the universal case, so let $\varphi = \forall x \psi(\vec{y},x)$. Let $\vec{y} \in \mathcal{M}$ be such that $(\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_1} f(\varphi)(\vec{y})$. Assume, towards a contradiction, that $(\mathcal{M},\mathcal{D}) \models_{\varepsilon_0} \varphi(\vec{y})$. Then

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \models_{\varepsilon_0} \psi(\vec{y},x)\big] \geq 1 - \varepsilon_0$$

and by the induction hypothesis we have

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \models_{\varepsilon_1} f(\psi)(\vec{y},x)\big] \geq 1 - \varepsilon_0. \tag{4}$$

Because $(\mathcal{M},\mathcal{D}) \not\models_{\varepsilon_1} \neg n$-split, the X_i disjointly cover $\mathcal{M}$, as discussed above. Now, by taking those m of the X_i (say $X_{i_1},\dots,X_{i_m}$) which have the largest intersection with this set we find that

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \models_{\varepsilon_1} (X_{i_1} \vee \dots \vee X_{i_m}) \wedge f(\psi)(\vec{y},x)\big] \geq \frac{m}{n}(1-\varepsilon_0)$$
$$= 1 - \varepsilon_1$$

which contradicts our choice of $(\mathcal{M},\mathcal{D})$. □

Lemma 3.2. (Kuyper and Terwijn [3, Lemma 7.3]) *Let $(\mathcal{N}',\mathcal{E}')$ and $(\mathcal{N},\mathcal{E})$ be as in the proof of Theorem 3.1 above. Then for every formula $\zeta(x_1,\dots,x_t)$ in the language of $\mathcal{M}$, for every $\varepsilon \in [0,1]$ and all $x_1,\dots,x_t \in \mathcal{N}'$: $(\mathcal{N}',\mathcal{E}') \models_\varepsilon \zeta(x_1,\dots,x_t)$ if and only if $(\mathcal{N},\mathcal{E}) \models_\varepsilon \zeta(\sigma(x_1),\dots,\sigma(x_t))$.*

Proof. By induction on the structure of the formulas in prenex normal form. The base case holds by definition of the relations in $\mathcal{N}'$. The only interesting induction step is the one for the universal quantifier. So, let $\zeta = \forall x_0 \zeta'(x_0,\dots,x_t)$ and let $x_1,\dots,x_t \in \mathcal{N}'$. Using the induction hypothesis, we find that the set $A = \{x_0 \in \mathcal{N}' \mid (\mathcal{N}',\mathcal{E}') \models_\varepsilon \zeta'(x_0,\dots,x_t)\}$ is equal to the set $\{x_0 \in \mathcal{N}' \mid (\mathcal{N},\mathcal{E}) \models_\varepsilon \zeta'(\sigma(x_0),\dots,\sigma(x_t))\}$, which consists of n disjoint copies of the set $B = \{x_0 \in \mathcal{N} \mid (\mathcal{N},\mathcal{E}) \models_\varepsilon \zeta'(x_0,\sigma(x_1),\dots,\sigma(x_t))\}$; denote the copy of B living inside $\mathcal{N}_i$ by B_i. Then

$$\mathcal{D}(A) = \sum_{i=1}^{n} \mathcal{E}'(B_i) = \sum_{i=1}^{n} \tfrac{1}{n}\mathcal{E}(B) = \mathcal{E}(B)$$

from which we directly see that $(\mathcal{N}', \mathcal{E}') \models_\varepsilon \zeta(x_1, \dots, x_t)$ if and only if $(\mathcal{N}, \mathcal{E}) \models_\varepsilon \zeta(\sigma(x_1), \dots, \sigma(x_t))$. □

Theorem 3.3. *Let $\mathcal{L}$ be a countable first-order language not containing equality. Then, for all rationals $0 < \varepsilon_1 \leq \varepsilon_0 \leq 1$, the ε_0-tautologies many-one reduce to the ε_1-tautologies.*

Proof. We can choose integers n and $m < n$ such that $\frac{m}{n} = \frac{\varepsilon_0 - \varepsilon_1}{\varepsilon_0}$. We construct a many-one reduction f such that for all formulas φ,

$$\varphi \text{ is an } \varepsilon_0\text{-tautology if and only if } f(\varphi) \text{ is an } \varepsilon_1\text{-tautology.}$$

Again, we only consider the nontrivial case where φ is a universal formula $\forall x \psi(\vec{y}, x)$. We define $f(\varphi)$ to be the formula

$$\neg\text{-}n\text{-split} \wedge \bigvee_{i_1, \dots, i_m} \forall x \big(X_{i_1}(x) \vee \dots \vee X_{i_m}(x) \vee f(\psi)(\vec{y}, x) \big)$$

where the disjunction is over all subsets of size m from $\{1, \dots, n\}$.

The proof is almost the same as for Theorem 3.1. In the proof for the implication from right to left, follow the proof up to (2), i.e.

$$(\mathcal{N}', \mathcal{E}') \not\models_{\varepsilon_1} f(\varphi)(\vec{y}).$$

This is equivalent to the statement that for all $1 \leq i_1 < \dots < i_m \leq n$ we have

$$\Pr_{\mathcal{E}'}\big[x \in \mathcal{N}' \mid (\mathcal{N}', \mathcal{E}') \not\models_{\varepsilon_1} X_{i_1}(x) \vee \dots \vee X_{i_m}(x) \vee f(\psi)(\vec{y}, x)\big] > \varepsilon_1.$$

Similar as before, using Lemma 3.2, we find that this is equivalent to

$$\frac{n-m}{n} \Pr_{\mathcal{E}}\big[x \in \mathcal{N} \mid (\mathcal{N}, \mathcal{E}) \not\models_{\varepsilon_1} f(\psi)(\sigma(\vec{y}), x)\big] > \varepsilon_1.$$

Again, using the induction hypothesis and the fact that π is measure-preserving we find that this is equivalent to

$$\begin{aligned}&\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \not\models_{\varepsilon_0} \psi(\pi(\sigma(\vec{y})), x)\big] \\ &> \frac{n}{n-m}\varepsilon_1 = \frac{\varepsilon_0}{\varepsilon_1}\varepsilon_1 = \varepsilon_0.\end{aligned}$$

This proves that $(\mathcal{N}', \mathcal{E}') \models_{\varepsilon_1} f(\varphi)(\vec{y})$ if and only if $(\mathcal{M}, \mathcal{D}) \models_{\varepsilon_0} \varphi(\vec{y})$.

For the converse implication, we also need to slightly alter the proof of Theorem 3.1. Assuming that $(\mathcal{M}, \mathcal{D}) \not\models_{\varepsilon_1} f(\varphi)(\vec{y})$, follow the proof up to (4), where we obtain

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \models_{\varepsilon_1} f(\psi)(\vec{y}, x)\big] \geq 1 - \varepsilon_0. \tag{5}$$

Define

$$\eta = \Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \models_{\varepsilon_1} f(\psi)(\vec{y}, x)\big]$$

and take those m of the X_i (say $X_{i_1}, \dots, X_{i_m}$) which have the smallest intersection with this set. Note that by (5) we have $\eta \geq 1 - \varepsilon_0$. Then we find that

$$\begin{aligned}
&\Pr_{\mathcal{D}}\big[x \in \mathcal{M} \mid (\mathcal{M}, \mathcal{D}) \models_{\varepsilon_1} X_{i_1} \vee \dots \vee X_{i_m} \vee f(\psi)(\vec{y}, x)\big] \\
&\geq \frac{m}{n} + \Big(1 - \frac{m}{n}\Big)\eta = \frac{\varepsilon_0 - \varepsilon_1}{\varepsilon_0} + \frac{\varepsilon_1}{\varepsilon_0}\eta \\
&\geq \frac{\varepsilon_0 - \varepsilon_1}{\varepsilon_0} + \frac{\varepsilon_1}{\varepsilon_0}(1 - \varepsilon_0) = 1 - \varepsilon_1.
\end{aligned}$$

which contradicts our choice of $(\mathcal{M}, \mathcal{D})$. □

Observe that, because of the inductive nature of the reductions above, we can perform these reductions per quantifier. In particular, we can talk about what it means for a formula with variable ε (that is, a separate ε for each quantifier) to be a tautology. This way, we get something like Keisler's probability logic mentioned in the introduction; however, remember that we still have our non-classical negation (unlike Keisler). This idea will be crucial in our hardness proof.

4 Validity Is Π^1_1-hard

To show that the set of ε-tautologies is indeed Π^1_1-hard, we adapt a proof by Hoover [1] which shows that $\mathcal{L}_{\omega P}$ is Π^1_1-complete. We will show that, to a certain extent, we can define the natural numbers within probability logic.

Definition 4.1. Let φ be a formula in prenex normal form and N a unary predicate. Then φ^N, or φ **relativised to** N, is defined as the formula where each $\forall x \psi(x)$ is replaced by $\forall x (N(x) \to \psi(x))$ and each $\exists x \psi(x)$ is replaced by $\exists x (N(x) \wedge \psi(x))$.

Theorem 4.2. *Let $\mathcal{L}$ be the language consisting of a constant symbol 0, a unary relation $N(x)$, binary relations $x = y$,*[1] *$S(x) = y$ and $R(x, y)$, and ternary relations $x + y = z$ and $x \cdot y = z$. Furthermore, let f be the reduction from 0-tautologies to $\frac{1}{2}$-tautologies from Proposition 3.1. Then there exists finite theories T, T' in the language $\mathcal{L}$ such that, for every first-order sentence φ containing a new predicate symbol Q, the following are equivalent:*

(i) $\models_{\frac{1}{2}} f(\neg(\bigwedge T)) \vee \neg\left(\bigwedge T'\right) \vee f\left(\neg\varphi^N\right)$;
(ii) $\mathbb{N} \models \forall Q \neg\varphi(Q)$.[2]

Proof. We will prove the contrapositives of the implications (i) → (ii) and (ii) → (i). During the entire proof, one should mainly think about what it means for a formula ψ that its negation $\neg\psi$ does not hold. Note that we have that $(\mathcal{M}, \mathcal{D}) \not\models_0$

[1] Here we do not mean true equality, but rather a binary relation that we will use to represent equality.

[2] We denote by $\forall Q \neg\varphi(Q)$ the second-order formula $\forall X \neg\varphi(X/Q)$, where $\varphi(X/Q)$ is the formula where the predicate symbol Q is replaced by a second-order variable X.

$\neg\psi$ if and only if all universal quantifiers hold classically and all existential quantifiers hold on a set of strictly positive measure. Likewise, $(\mathcal{M}, \mathcal{D}) \not\models_{\frac{1}{2}} \neg\psi$ holds if and only if all universal quantifiers hold classically and all existential quantifiers hold on a set of measure strictly greater than $\frac{1}{2}$.

Inspired by this, we form the theories T and T'. T consists of Robinson's Q relativised to N, axioms specifying that the arithmetical relations only hold on N, and some special axioms for N and R. That is, we put the following axioms in T (keeping in mind that we are mostly interested in what happens when the negation of these formulas does not hold, i.e. one should read the $\forall$ as a classical universal quantifier and the $\exists$ as saying that the statement holds on a set of strictly positive measure):

All equality axioms. For example:

$$\forall x(x = x)$$
$$\forall x \forall y((N(x) \wedge x = y) \rightarrow N(y))$$

We should guarantee that 0 is in N:

$$N(0)$$

We now give the axioms for the successor function:

$$\forall x \forall y(S(x) = y \rightarrow (N(x) \wedge N(y)))$$
$$(\forall x \exists y S(x) = y)^N$$
$$(\forall x \forall y \forall u \forall v((S(x) = y \wedge S(u) = v \wedge x = u) \rightarrow y = v))^N$$
$$(\forall x \neg S(x) = 0)^N$$
$$(\forall x(x = 0 \vee \exists y S(y) = x))^N.$$ [3]

In the axioms below, we will leisurely denote by $\psi(S(x))$ the formula $\forall y(S(x) = y \rightarrow \psi(y))$ and similarly for $x + y$ and $x \cdot y$. We proceed with the inductive definitions of $+$ and $\cdot$:

$$(\forall x \forall y \forall z(x + y = z \rightarrow (N(x) \wedge N(y) \wedge N(z))))$$
$$(\forall x(x + 0 = x))^N$$
$$(\forall x \forall y(x + S(y) = S(x + y)))^N$$
$$(\forall x \forall y \forall z(x \cdot y = z \rightarrow (N(x) \wedge N(y) \wedge N(z))))^N$$
$$(\forall x(x \cdot 0 = 0))^N$$
$$(\forall x \forall y(x \cdot S(y) = (x \cdot y) + x))^N.$$

[3] We do not really need this last axiom, but we have added it anyway so that all axioms of Robinson's Q are in T.

Finally, we introduce a predicate R. This predicate is meant to function as a sort of 'padding'. The goal of this predicate is to force the measure of a point $S^n(0)$ to be larger than the measure of $\{x \mid N(x) \wedge x > S^n(0)\}$ (the precise use will be made clear in the proof below).

$$(\forall x \forall y \neg R(x,y))^N$$

The last two axioms will be in T' instead of in T, because these need to be evaluated for $\varepsilon = \frac{1}{2}$ while the rest will be evaluated for $\varepsilon = 0$. So, because we will be looking at when the negation does not hold, the existential quantifier should be read as "strictly more than measure $\frac{1}{2}$ many".

$$\forall x(N(x) \to \exists y(R(x,y) \vee x = y))$$
$$\forall x(N(x) \to \exists y \neg(R(x,y) \vee x < y))$$

Here, $x < y$ is short for $f(\exists z(x + S(z) = y))$, i.e. the usual definition of $x < y$ evaluated for $\varepsilon = 0$.

Note that for universal formulas it does not matter if they are in T or T' because in both cases the negation of the formula does not hold if and only if the formula holds classically.

We will now show that these axioms indeed do what we promised. First, we show that (i) implies (ii). So, assume $\mathbb{N} \not\models \forall Q(\neg\varphi(Q))$. Fix a predicate $Q^{\mathbb{N}}$ such that $\mathbb{N} \not\models \neg\varphi(Q)$. Now take the model $\mathcal{M} = \omega \times \{0,1\}$ to be the disjoint union of two copies of ω, where we define $S, +, \cdot, \leq, 0$ on the first copy $\omega \times \{0\}$ of ω as usual, and let these be undefined elsewhere. Let

$$N := \omega \times \{0\} \text{ and } R := \left\{((a,0),(b,1)) \mid \mu k \left[2^{k+1} > 3^{a+1}\right] \neq b\right\}.$$

We let $Q^{\mathcal{M}}(a,0)$ hold if $Q^{\mathbb{N}}(a)$ and we never let it hold on the second copy of ω. Finally, define $\mathcal{D}$ by

$$\mathcal{D}(a,0) = \mathcal{D}(a,1) := \frac{1}{3^{a+1}}.$$

Then it is directly verified that

$$(\mathcal{M},\mathcal{D}) \not\models_0 \neg\Big(\bigwedge T\Big) \vee \neg\varphi^N,$$

i.e. all formulas in $T \cup \{\varphi^N\}$ hold in $(\mathcal{M},\mathcal{D})$ if universal quantifiers are interpreted classically and existential quantifiers as expressing that there exists a set of positive measure. Note that because all points have positive measure this is equivalent to the classical existential quantifier, so all we are really saying is that T and φ^N hold classically in $\mathcal{M}$.

Furthermore, if we let $a \in \omega$ and denote b for $\mu k[2^{k+1} > 3^{a+1}]$ then we have that

$$\begin{aligned}&\Pr_{\mathcal{D}}\left[y \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \models_{\frac{1}{2}} R((a,0),y) \vee (a,0) = y\right] \\ &= \frac{1}{2} - \frac{1}{2^{b+1}} + \frac{1}{3^{a+1}} \\ &> \frac{1}{2}\end{aligned}$$

while we also have that

$$\begin{aligned}
&\Pr_{\mathcal{D}}\big[y \in \mathcal{M} \mid (\mathcal{M},\mathcal{D}) \models_{\frac{1}{2}} \neg(R((a,0),y) \vee (a,0) < y)\big] \\
&= 1 - \left(\frac{1}{2} - \frac{1}{2^{b+1}} + \sum_{i=a+2}^{\infty} 3^{-i}\right) \\
&= 1 - \frac{1}{2}\left(1 - \frac{1}{2^b} + \frac{1}{3^{a+1}}\right) \\
&> \frac{1}{2}
\end{aligned}$$

where the last inequality follows from the fact that b is the smallest $k \in \omega$ such that $2^{k+1} > 3^{a+1}$, so that $2^b \leq 3^{a+1}$. Thus, we see that $(\mathcal{M},\mathcal{D}) \not\models_{\frac{1}{2}} \neg(\bigwedge T')$. But then we see from (the proof of) Theorem 3.1, together with the remark below Theorem 3.3 that there is a probability model $(\mathcal{N},\mathcal{E})$ such that

$$(\mathcal{N},\mathcal{E}) \not\models_{\frac{1}{2}} f\left(\neg\Big(\bigwedge T\Big)\right) \vee \neg\Big(\bigwedge T'\Big) \vee f\left(\neg\varphi^N\right),$$

i.e. (i) does not hold.

Conversely, assume that statement (i) does not hold. Without loss of generality, we may assume the equality relation on $\mathcal{M}$ to be true equality; otherwise, because (i) does not hold and all equality axioms are in T we could look at $\mathcal{M}/=$ instead.

Again, from (the proof of) Theorem 3.1 we see that

$$(\mathcal{M},\mathcal{D}) \not\models_0 \neg\Big(\bigwedge T\Big) \vee \neg\varphi^N \text{ and } (\mathcal{M},\mathcal{D}) \not\models_{\frac{1}{2}} \neg\Big(\bigwedge T'\Big).$$

We will now use the three axioms involving R. Let $m \in \mathcal{M}$ with $\mathcal{M} \models N(m)$. Then $\{a \in \mathcal{M} \mid \mathcal{M} \models a = m\} \subseteq N^{\mathcal{M}}$ by the equality axioms, and similarly $\{a \in \mathcal{M} \mid m < a\} \subseteq N^{\mathcal{M}}$. So the axiom $(\forall x \forall y \neg R(x,y))^N$ tells us that these two sets are disjoint from $\{a \in \mathcal{M} \mid \mathcal{M} \models R(m,a)\}$. Therefore, from the two axioms in T' it now follows that

$$\begin{aligned}
\Pr_{\mathcal{D}}\big[a \in \mathcal{M} \mid \mathcal{M} \models m = a\big] &> \frac{1}{2} - \Pr_{\mathcal{D}}\big[a \in \mathcal{M} \mid \mathcal{M} \models R(m,a)\big] \\
&> \Pr_{\mathcal{D}}\big[a \in \mathcal{M} \mid \mathcal{M} \models m < a\big].
\end{aligned}$$

Thus,

$$\Pr_{\mathcal{D}}\big[\mathcal{M}\big] > \frac{1}{2} \Pr_{\mathcal{D}}\big[a \in \mathcal{M} \mid m \leq a\big] \tag{6}$$

We now claim that, if we denote $S(x)$ for the unique y such that $S(x) = y$ (as guaranteed to exist and be unique by T):

$$\Pr_{\mathcal{D}}\left[\left\{0,\ldots,S^k(0)\right\}\right] > \left(1 - \frac{1}{2^{k+1}}\right) \Pr_{\mathcal{D}}\big[N\big].$$

For $k = 0$ this is clear: from the axioms in T it follows that for all elements $a \in N$ different from 0 we have $a > 0$, and therefore $\Pr_{\mathcal{D}}\left[\{0\}\right] > \frac{1}{2}\Pr_{\mathcal{D}}\left[N\right]$ by (6). Similarly, assume this holds for $k \in \omega$. Then we have by (6):

$$\begin{aligned}&\Pr_{\mathcal{D}}\left[\{0,\dots,S^{k+1}(0)\}\right]\\&> \Pr_{\mathcal{D}}\left[\{0,\dots,S^{k}(0)\}\right] + \frac{1}{2}\left(\Pr_{\mathcal{D}}\left[N\right] - \Pr_{\mathcal{D}}\left[\{0,\dots,S^{k}(0)\}\right]\right)\end{aligned}$$

so from the induction hypothesis we obtain

$$\begin{aligned}&> \left(1 - \frac{1}{2^{k+1}}\right)\Pr_{\mathcal{D}}\left[N\right] + \frac{1}{2^{k+2}}\Pr_{\mathcal{D}}\left[N\right]\\&= \left(1 - \frac{1}{2^{k+2}}\right)\Pr_{\mathcal{D}}\left[N\right].\end{aligned}$$

Because this converges to $\Pr_{\mathcal{D}}\left[N\right]$ if k goes to infinity, we see that all weight of N rests on $X := \{S^n(0) \mid n \in \omega\} \subseteq N$. Now, if some universal quantifier holds when relativised to N, it certainly holds when restricted to X. Furthermore, if some existential quantifier holds with positive measure in N, then it also has to hold with positive measure in X because $X \subseteq N$ has the same measure as N. Therefore, we see that $(\mathcal{M}, \mathcal{D}) \not\models_0 \neg\varphi^N$ implies that also $(\mathcal{M}{\restriction}X, \mathcal{D}{\restriction}X) \not\models_0 \neg\varphi^X$ (see the discussion at the beginning of the proof about what it means for the negation of a formula to not hold).

However, we can directly verify that $\mathcal{M}{\restriction}X$ is isomorphic to the standard natural numbers $\mathbb{N} = (\omega, S, +, \cdot, 0)$. So, by transferring the predicate Q from $\mathcal{M}$ to $\mathbb{N}$ (i.e. letting $Q^{\mathbb{N}}(k)$ hold if $Q^{\mathcal{M}}(S^k(0))$ holds) we find that indeed $\mathbb{N} \not\models \forall Q \neg\varphi(Q)$. □

Putting this together, we reach our conclusion.

Theorem 4.3. *For rational $\varepsilon \in (0, 1)$, the set of ε-tautologies is Π^1_1-hard.*

Proof. From Theorem 3.1, Theorem 3.3 and Theorem 4.2. □

In fact, we have shown that even for languages not containing function symbols or equality, ε-validity is already Π^1_1-hard. Our proof above uses one constant: 0. However, we could also replace 0 by a unary relation representing $0 = x$ and modify the proof to show that the relational fragment of ε-validity is Π^1_1-hard.

We do not yet know of an upper bound for the complexity of ε-validity. While we have developed methods for proving upper bounds for ε-satisfiability, which will be discussed in a future paper, these methods do not seem to work for proving any results about ε-validity. Thus, the exact complexity of ε-validity is still an open problem.

References

1. Hoover, D.N.: Probability logic. Annals of Mathematical Logic 14, 287–313 (1978)
2. Keisler, H.J.: Probability quantifiers. In: Barwise, J., Feferman, S. (eds.) Model-Theoretic Logics. Perspectives in Mathematical Logic, vol. 8. Springer (1985)
3. Kuyper, R., Terwijn, S.A.: Model theory of measure spaces and probability logic (submitted, 2012)
4. Terwijn, S.A.: Probabilistic logic and induction. Journal of Logic and Computation 15(4), 507–515 (2005)
5. Terwijn, S.A.: Decidability and Undecidability in Probability Logic. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 441–450. Springer, Heidelberg (2008)
6. Valiant, L.G.: Robust logics. Artificial Intelligence 117, 231–253 (2000)

Update as Evidence: Belief Expansion

Roman Kuznets[⋆] and Thomas Studer

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

Abstract. We introduce a justification logic with a novel constructor for evidence terms, according to which the new information itself serves as evidence for believing it. We provide a sound and complete axiomatization for belief expansion and minimal change and explain how the minimality can be graded according to the strength of reasoning. We also provide an evidential analog of the Ramsey axiom.

1 Introduction

Like modal logics, *justification logics* are epistemic logics that provide means to formalize properties of knowledge and belief. Modal logics use formulas $\Box A$ to state that A is known (or believed), where the modality $\Box$ can be seen as an *implicit* knowledge operator since it does not provide any reason why A is known. Justification logics operate with *explicit* evidence for an agent's knowledge using formulas of the form $t : A$ to state that A is known for reason t. The evidence term t may represent a formal mathematical proof of A or an informal reason for believing A such as a public announcement or direct observation of A.

Artemov developed the first justification logic, the Logic of Proofs, to give a classical provability semantics for intuitionistic logic [2–4]. In the area of formal epistemology, justification logics provide a novel approach to certain epistemic puzzles and problems of multiagent systems [5–7, 9, 13].

The study of dynamic justification logics took off with Renne's PhD thesis [21] and his work on eliminating unreliable evidence [22]. He also investigated the expressive power of certain justification logics with announcements [23]. In a series of papers [12, 14, 15] we examined two alternative justification counterparts of Gerbrandy–Groeneveld's public announcement logic [18]. Last but not least, Baltag et al. [10] introduced a justification logic for belief change, soft evidence, and defeasible knowledge.

In the present paper we introduce the justification logic $\mathsf{JUP}_{\mathsf{CS}}$ that provides a sound and complete axiomatization for belief expansion and minimal change. Our logic includes a new evidence term construct $\mathsf{up}(A)$ that represents the update with A. Hence, after an update with A, the term $\mathsf{up}(A)$ becomes a reason to believe A. Formally, this is modeled by the axiom $[A]\Big(\mathsf{up}(A) : A\Big)$.

[⋆] Supported by Swiss National Science Foundation grant PZ00P2–131706.

In addition, the presence of explicit evidence makes it possible to axiomatize the principle of minimal change within the object language. For instance, in Lemma [20] we prove that for each term t that does not contain $\mathsf{up}(A)$ as a subterm,

$$\mathsf{JUP_{CS}} \vdash \quad t : B \leftrightarrow [A](t : B) \ .$$

The direction from left to right is the persistence principle saying that we deal only with belief expansion. The direction from right to left states that if after an update with A an agent believes B for a reason that is independent from the update, then before the update the agent already believed B for the same reason. Note that a principle of this kind cannot be formulated in a purely modal language. When $[A]\Box B$ holds, it is not clear whether $\Box B$ should be the case since it is not known whether the belief in B depends on the update or is due to another, unrelated reason.

2 The Logic $\mathsf{JUP_{CS}}$

We start with countably many constants c_i, countably many variables x_i, and countably many atomic propositions P_i. The (*evidence*) *terms* and formulas of the language of JUP are defined as follows:

– *Evidence terms.*
  - Every constant c_i and every variable x_i is an atomic term. If A is a formula, then $\mathsf{up}(A)$ is an atomic term. Every atomic term is a term.
  - If t and s are terms and A is a formula, then $(t \cdot_A s)$ is a term.

– *Formulas.*
  - Every atomic proposition P_i is a formula.
  - If A and B are formulas, Γ is a finite set of formulas, and t is a term, then $\neg A$, $(A \to B)$, $t : A$, and $[\Gamma]A$ are formulas.

We write $[A_1, \ldots, A_n]B$ instead of $[\{A_1, \ldots, A_n\}]B$, usually assuming all A_i's to be pairwise distinct.

ATm, Prop, Tm, and Fml denote the set of atomic terms, the set of atomic propositions, the set of evidence terms, and the set of formulas respectively. A formula $t : A$ means that A *is believed for reason* t and $[\Gamma]A$ stands for A *holds after an update with all formulas in* Γ. As usual, we define $(A \wedge B) := \neg(A \to \neg B)$ and $(A \leftrightarrow B) := ((A \to B) \wedge (B \to A))$. We employ the standard conventions on the omission of brackets and postulate that both the colon operator in $t : A$ and the update operator in $[\Gamma]A$ bind stronger than any Boolean connective.

The set of axioms of JUP can be found in Fig. [1]. Note that $\Gamma \cup \Delta$ in the axiom (It) is a finite set of formulas whenever Γ and Δ are. The axioms (Taut) and (App) become the usual axioms of the justification logic J (see [6]) if (App) is formulated as an implication instead of the equivalence above. The present version with the equivalence yields a justification logics with *minimal evidence*. Later, when we define the semantics, this will correspond to the fact that the evidence relation is the *least* fixed point. This also explains why we need to annotate the application operator $\cdot$ by a formula: otherwise we would not be able to formulate the direction from right to left. Renne [22] was the first to use this kind of annotation. Axioms (Red.1)–(Red.3) are called *reduction axioms*. They make it possible to reduce the situation after an update occurred to the situation before the update. For instance, (Red.1) states that atomic facts are not affected by updates. Axiom (Pers) postulates that beliefs are *persistent*, i.e., that no contraction takes place because of an update and, consequently, the belief set can only be expanded. The update axiom (Up) claims that updates are *introspectively successful*: after an update with a formula A, the agent believes A and the term $\mathsf{up}(A)$ represents a reason for that belief, which is the update itself. The axiom (Init) postulates the special status of terms $\mathsf{up}(A)$, which initially, i.e., before any updates, cannot serve as a basis for belief in anything. Axioms (MC.1) and (MC.2) formalize the principle of *minimal change*: an update should only lead to the smallest necessary change in the belief set. That means only those beliefs should be added that "logically" follow from the update and from what is already believed before. An interesting feature of our system is that the strength of the logic used for the deductive closure can be regulated. Finally, the axiom (It) explains how to deal with *iterated* updates.

1. All propositional tautologies (Taut)
2. $t : (A \to B) \wedge s : A \ \leftrightarrow\ t \cdot_A s : B$ (App)
3. $[\Gamma]P \ \leftrightarrow\ P$ (Red.1)
4. $[\Gamma]\neg B \ \leftrightarrow\ \neg[\Gamma]B$ (Red.2)
5. $[\Gamma](B \to C) \ \leftrightarrow\ ([\Gamma]B \to [\Gamma]C)$ (Red.3)
6. $t : B \ \to\ [\Gamma]t : B$ (Pers)
7. $\neg\mathsf{up}(A) : B$ (Init)
8. $[\Gamma]\mathsf{up}(A) : A$ if $A \in \Gamma$ (Up)
9. $[\Gamma]t : A \ \to\ t : A$
   if $t \in \mathsf{ATm}$ and either $t \neq \mathsf{up}(A)$ or $A \notin \Gamma$ (MC.1)
10. $[\Gamma]t \cdot_A s : B \ \leftrightarrow\ [\Gamma]t : (A \to B) \wedge [\Gamma]s : A$ (MC.2)
11. $[\Gamma][\Delta]A \ \leftrightarrow\ [\Gamma \cup \Delta]A$ (It)

Fig. 1. Axioms of JUP

A *constant specification* CS (for JUP) is any subset

$$\mathsf{CS} \subseteq \{(c,\quad c_1 : c_2 : \ldots : c_n : A) \mid \\ n \geq 0,\ c, c_1, c_2, \ldots, c_n \text{ are constants, and } A \text{ is an axiom of } \mathsf{JUP}\}.$$

For a constant specification CS the deductive system $\mathsf{JUP}_{\mathsf{CS}}$ is the Hilbert system given by the axioms of JUP and by the rules modus ponens and axiom necessitation:

$$\frac{A \quad A \to B}{B}\,(\mathsf{MP})\ , \qquad \frac{(c, B) \in \mathsf{CS}}{c : B}\,(\mathsf{AN})\ .$$

We write $\mathsf{JUP}_{\mathsf{CS}} \vdash A$ if the formula A is derivable in $\mathsf{JUP}_{\mathsf{CS}}$.

We are now going to introduce a semantics for $\mathsf{JUP}_{\mathsf{CS}}$ that uses basic modular models. Artemov [8] introduced them for the basic justification logic J in order to provide an ontologically transparent semantics for justifications. Kuznets and Studer [19] later extended this construction to all justification counterparts of the logics from the modal cube between K and $\mathsf{S5}$. The very first semantics of this kind, however, was presented by Mkrtychev [20].

Definition 1 (Evidence closure). *Let* $\mathcal{B} \subseteq \mathsf{ATm} \times \mathsf{Fml}$*. For an arbitrary set* $X \subseteq \mathsf{Tm} \times \mathsf{Fml}$ *we define* $\mathsf{cl}_{\mathcal{B}}(X)$ *by:*

1. *if* $(t, A) \in \mathcal{B}$*, then* $(t, A) \in \mathsf{cl}_{\mathcal{B}}(X)$*;*
2. *if* $(s, A) \in X$ *and* $(t, A \to B) \in X$*, then* $(t \cdot_A s, B) \in \mathsf{cl}_{\mathcal{B}}(X)$*.*

Note that $\mathsf{cl}_{\mathcal{B}}$ is a monotone operator on $\mathsf{Tm} \times \mathsf{Fml}$, that is

$$X \subseteq Y \quad \text{implies} \quad \mathsf{cl}_{\mathcal{B}}(X) \subseteq \mathsf{cl}_{\mathcal{B}}(Y)$$

for all $X, Y \subseteq \mathsf{Tm} \times \mathsf{Fml}$. Hence, $\mathsf{cl}_{\mathcal{B}}$ has a least fixed point, which is shown as usual, see, e.g., [11].

Lemma 2 (Least fixed point). *There is a unique* $R \subseteq \mathsf{Tm} \times \mathsf{Fml}$ *such that*

1. $\mathsf{cl}_{\mathcal{B}}(R) = R$*,*
2. *for any* $S \subseteq \mathsf{Tm} \times \mathsf{Fml}$*, if* $\mathsf{cl}_{\mathcal{B}}(S) \subseteq S$*, then* $R \subseteq S$*.*

Proof. Let $C := \{S \subseteq \mathsf{Tm} \times \mathsf{Fml} \mid \mathsf{cl}_{\mathcal{B}}(S) \subseteq S\}$. Since $\mathsf{Tm} \times \mathsf{Fml} \in C$, we know that C is non-empty. Let $R := \bigcap C$. The second claim now holds by definition. And the uniqueness of R is an easy corollary of the second claim.

It remains to establish $\mathsf{cl}_{\mathcal{B}}(R) = R$. Let $S \in C$. Since $R \subseteq S$ and $\mathsf{cl}_{\mathcal{B}}$ is monotone, we find $\mathsf{cl}_{\mathcal{B}}(R) \subseteq \mathsf{cl}_{\mathcal{B}}(S)$. We also have $\mathsf{cl}_{\mathcal{B}}(S) \subseteq S$, so $\mathsf{cl}_{\mathcal{B}}(R) \subseteq S$. Since S is an arbitrary element of C and $R = \bigcap C$, this implies $\mathsf{cl}_{\mathcal{B}}(R) \subseteq R$.

To show $R \subseteq \mathsf{cl}_{\mathcal{B}}(R)$, we first observe that since $\mathsf{cl}_{\mathcal{B}}(R) \subseteq R$, we have $\mathsf{cl}_{\mathcal{B}}(\mathsf{cl}_{\mathcal{B}}(R)) \subseteq \mathsf{cl}_{\mathcal{B}}(R)$ by monotonicity. Thus $\mathsf{cl}_{\mathcal{B}}(R) \in C$, yielding $R \subseteq \mathsf{cl}_{\mathcal{B}}(R)$ because $R = \bigcap C$. □

Definition 3 (Evidence relation). *Let* $\mathcal{B} \subseteq \mathsf{ATm} \times \mathsf{Fml}$*. We define the* minimal evidence relation $\mathcal{E}(\mathcal{B})$ *as the least fixed point of* $\mathsf{cl}_{\mathcal{B}}$*.*

It follows directly from the definition of $\mathsf{cl}_{\mathcal{B}}$ that

Lemma 4 (Properties of fixed points of $\mathsf{cl}_{\mathcal{B}}$). *For any* $\mathcal{B} \subseteq \mathsf{ATm} \times \mathsf{Fml}$ *and any fixed point* F *of* $\mathsf{cl}_{\mathcal{B}}$*, e.g., for* $F = \mathcal{E}(\mathcal{B})$*:*

1. $(t, A) \in F$ *iff* $(t, A) \in \mathcal{B}$ *for any* $t \in \mathsf{ATm}$*.*
2. $(t \cdot_A s, B) \in F$ *iff* $(t, A \to B) \in F$ *and* $(s, A) \in F$*.*

Further, we get the following lemma.

Lemma 5 (Monotonicity of $\mathcal{E}$). $\mathcal{E}(\mathcal{B}) \subseteq \mathcal{E}(\mathcal{B} \cup \mathcal{C})$ *for* $\mathcal{B}, \mathcal{C} \subseteq \mathsf{ATm} \times \mathsf{Fml}$*.*

Proof. By induction on the construction of t we show $(t, A) \in \mathcal{E}(\mathcal{B})$ implies $(t, A) \in \mathcal{E}(\mathcal{B} \cup \mathcal{C})$ for all formulas A. Assume $(t, A) \in \mathcal{E}(\mathcal{B})$. We have one of the following cases.

1. $t \in \mathsf{ATm}$. Then $(t, A) \in \mathcal{B}$ by Lemma 4.1. Since $(t, A) \in \mathcal{B} \cup \mathcal{C}$, it follows from the same lemma that $(t, A) \in \mathcal{E}(\mathcal{B} \cup \mathcal{C})$.
2. $t = r \cdot_B s$. Then $\{(s, B), (r, B \to A)\} \subseteq \mathcal{E}(\mathcal{B})$ by Lemma 4.2. By IH we find $\{(s, B), (r, B \to A)\} \subseteq \mathcal{E}(\mathcal{B} \cup \mathcal{C})$. We get $(t, A) = (r \cdot_B s, A) \in \mathcal{E}(\mathcal{B} \cup \mathcal{C})$ by Lemma 4.2. □

Definition 6 (Model, initial model, updated model). *A* model *is a pair* $\mathcal{M} = (\mathsf{v}, \mathcal{B})$ *where* $\mathsf{v} \subseteq \mathsf{Prop}$ *and* $\mathcal{B} \subseteq \mathsf{ATm} \times \mathsf{Fml}$. *For a constant specification* CS, *the model* $\mathcal{M}$ *is called a* CS-model *if* $\mathsf{CS} \subseteq \mathcal{B}$. *The model* $\mathcal{M}$ *is called* initial *if* $(\mathsf{up}(A), B) \notin \mathcal{B}$ *for any formulas* A *and* B.

For a finite set Γ *of formulas, the* updated model $\mathcal{M}^\Gamma := (\mathsf{v}, \mathcal{B}^\Gamma)$ *is defined by* $\mathcal{B}^\Gamma := \mathcal{B} \cup \mathcal{U}_\Gamma$ *with* $\mathcal{U}_\Gamma := \{(\mathsf{up}(A), A) \mid A \in \Gamma\}$. *For a singleton set* $\Gamma = \{A\}$ *we write* $\mathcal{M}^A$ *and* $\mathcal{B}^A$ *instead of* $\mathcal{M}^{\{A\}}$ *and* $\mathcal{B}^{\{A\}}$ *respectively.*

Remark 7. Note that our definition of a model update is independent of which formulas are true, unlike Plaza-style, Gerbrandy–Groeneveld-style, or action-model-style updates, where the definitions of model update and truth in the model have to be given by simultaneous induction. This ontological separation of reasons for belief from truth is inherent in Artemov's semantics of modular models [8], which we adopt and adapt in this paper.

Lemma 8 (Properties of updated models)

1. $\mathcal{M}^\varnothing = \mathcal{M}$,
2. $(\mathcal{M}^\Gamma)^\Delta = \mathcal{M}^{\Gamma \cup \Delta}$,
3. *For any constant specification* CS, *any* CS*-model* $\mathcal{M}$, *and any finite set* Γ *of formulas, the model* $\mathcal{M}^\Gamma$ *is a* CS*-model.*

Proof. Immediately follows from $\mathcal{U}_\varnothing = \varnothing$, $\mathcal{U}_\Gamma \cup \mathcal{U}_\Delta = \mathcal{U}_{\Gamma \cup \Delta}$, and $\mathcal{B} \subseteq \mathcal{B}^\Gamma$ respectively. □

Definition 9 (Truth). *Let* $\mathcal{M} = (\mathsf{v}, \mathcal{B})$ *be a model and* D *be a formula. We define the relation* $\mathcal{M} \Vdash D$ *by*

1. $\mathcal{M} \Vdash P$ *iff* $P \in \mathsf{v}$
2. $\mathcal{M} \Vdash \neg A$ *iff* $\mathcal{M} \nVdash A$
3. $\mathcal{M} \Vdash A \to B$ *iff* $\mathcal{M} \nVdash A$ *or* $\mathcal{M} \Vdash B$
4. $\mathcal{M} \Vdash t : A$ *iff* $(t, A) \in \mathcal{E}(\mathcal{B})$
5. $\mathcal{M} \Vdash [\Gamma]A$ *iff* $\mathcal{M}^\Gamma \Vdash A$.

A formula D *is* valid with respect to a constant specification CS *if* $\mathcal{M} \Vdash D$ *for all initial* CS*-models* $\mathcal{M}$.

3 Soundness

For this section and the next one, we assume CS to be a fixed but arbitrary constant specification. In later sections, the use of soundness and completeness with respect to models with no CS specified should be understood as soundness and completeness with respect to initial $\varnothing$-models because any model is an $\varnothing$-model.

Theorem 10 (Soundness). *For all formulas D,*

$$\mathsf{JUP}_{\mathsf{CS}} \vdash D \quad \textit{implies} \quad D \textit{ is valid with respect to } \mathsf{CS}.$$

Proof. As usual the proof is by induction on the length of the derivation of D. Let $\mathcal{M} = (\mathsf{v}, \mathcal{B})$ be an initial CS-model.

1. (Taut). All instances of propositional tautologies hold under $\mathcal{M}$.
2. (App). $\mathcal{M} \Vdash t{:}(A \to B) \wedge s{:}A$ iff $\{(t, A \to B), (s, A)\} \subseteq \mathcal{E}(\mathcal{B})$. By Lemma 4.2, this is equivalent to $(t \cdot_A s, B) \in \mathcal{E}(\mathcal{B})$, in other words to $\mathcal{M} \Vdash t \cdot_A s : B$.
3. $(\mathsf{Red.1})$. $\mathcal{M} \Vdash [\Gamma]P$ iff $\mathcal{M}^\Gamma \Vdash P$ iff $P \in \mathsf{v}$ iff $\mathcal{M} \Vdash P$.
4. $(\mathsf{Red.2})$. $\mathcal{M} \Vdash [\Gamma]\neg B$ iff $\mathcal{M}^\Gamma \Vdash \neg B$ iff $\mathcal{M}^\Gamma \nVdash B$ iff $\mathcal{M} \nVdash [\Gamma]B$ iff $\mathcal{M} \Vdash \neg[\Gamma]B$.
5. $(\mathsf{Red.3})$. Similar to the previous case.
6. (Pers). Follows immediately from Lemma 5.
7. (Init). $(\mathsf{up}(A), B) \notin \mathcal{B}$ since $\mathcal{M}$ is initial. $(\mathsf{up}(A), B) \notin \mathcal{E}(\mathcal{B})$ by Lemma 4.1. Thus, $\mathcal{M} \Vdash \neg\mathsf{up}(A) : B$.
8. (Up). If $A \in \Gamma$, then $(\mathsf{up}(A), A) \in \mathcal{U}_\Gamma \subseteq \mathcal{B}^\Gamma$, and $(\mathsf{up}(A), A) \in \mathcal{E}(\mathcal{B}^\Gamma)$ by Lemma 4.1. It follows that $\mathcal{M}^\Gamma \Vdash \mathsf{up}(A) : A$ and $\mathcal{M} \Vdash [\Gamma]\mathsf{up}(A) : A$.
9. $(\mathsf{MC.1})$. Assume $\mathcal{M} \Vdash [\Gamma]t : A$ for $t \in \mathsf{ATm}$ such that either $t \neq \mathsf{up}(A)$ or $A \notin \Gamma$. Then $\mathcal{M}^\Gamma \Vdash t : A$ and $(t, A) \in \mathcal{E}(\mathcal{B}^\Gamma)$. Since $t \in \mathsf{ATm}$, we get $(t, A) \in \mathcal{B}^\Gamma = \mathcal{B} \cup \mathcal{U}_\Gamma$ by Lemma 4.1. Clearly, $(t, A) \notin \mathcal{U}_\Gamma$. Hence, $(t, A) \in \mathcal{B}$, and $(t, A) \in \mathcal{E}(\mathcal{B})$ by Lemma 4.1. Therefore, we conclude that $\mathcal{M} \Vdash t : A$.
10. $(\mathsf{MC.2})$. Similar to Case 2 but for $\mathcal{M}^\Gamma$.
11. (It). $\mathcal{M} \Vdash [\Gamma][\Delta]A$ iff $\mathcal{M}^\Gamma \Vdash [\Delta]A$ iff $(\mathcal{M}^\Gamma)^\Delta \Vdash A$. Then $(\mathcal{M}^\Gamma)^\Delta = \mathcal{M}^{\Gamma \cup \Delta}$ by Lemma 8. The equivalence continues as $\mathcal{M}^{\Gamma \cup \Delta} \Vdash A$ iff $\mathcal{M} \Vdash [\Gamma \cup \Delta]A$.
12. (MP). It is trivial to see that modus ponens preserves truth in a model.
13. (AN). For any $(c, B) \in \mathsf{CS}$, we have $(c, B) \in \mathcal{B}$ by definition of a CS-model. Further, $(c, B) \in \mathcal{E}(\mathcal{B})$ by Lemma 4.1, and $\mathcal{M} \Vdash c : B$. □

4 Completeness

Definition 11 (Consistency). *A set Φ of formulas, finite or infinite, is called* consistent *if* $\mathsf{JUP}_{\mathsf{CS}} \nvdash \neg(A_1 \wedge \cdots \wedge A_n)$ *for any finite subset* $\{A_1, \ldots, A_n\} \subseteq \Phi$.

A set Φ is called maximal consistent *if it is consistent whereas no proper superset of Φ is.*

Definition 12 (Induced model). *Let Φ be a maximal consistent set of formulas. The model $\mathcal{M}_\Phi = (\mathsf{v}_\Phi, \mathcal{B}_\Phi)$ that is* induced by Φ *is given by*

1. $P \in \mathsf{v}_\Phi$ *iff* $P \in \Phi \cap \mathsf{Prop}$.
2. $(t, A) \in \mathcal{B}_\Phi$ *iff* $t \in \mathsf{ATm}$ *and* $t : A \in \Phi$.

$\mathcal{M}_\Phi$ is an initial CS-model. Indeed, by the maximal consistency of Φ, we have

– $(c, B) \in \mathcal{B}_\Phi$ since $c : B \in \Phi$ since $\mathsf{JUP}_{\mathsf{CS}} \vdash c : B$ for every $(c, B) \in \mathsf{CS}$;
– $(\mathsf{up}(A), B) \notin \mathcal{B}_\Phi$ since $\mathsf{up}(A) : B \notin \Phi$ since $\neg\mathsf{up}(A) : B \in \Phi$ because we have $\mathsf{JUP}_{\mathsf{CS}} \vdash \neg\mathsf{up}(A) : B$ for arbitrary A and B.

Lemma 13 (Canonical evidence). *Let Φ be a maximal consistent set. Then*

$$t : A \in \Phi \iff (t, A) \in \mathcal{E}(\mathcal{B}_\Phi) .$$

Proof. By induction on the construction of t.

1. $t \in \mathsf{ATm}$. We have $t : A \in \Phi$ iff (by definition) $(t, A) \in \mathcal{B}_\Phi$ iff (by Lemma 4.1) $(t, A) \in \mathcal{E}(\mathcal{B}_\Phi)$.
2. $t = r \cdot_B s$. We have $r \cdot_B s : A \in \Phi$ iff (by (App) and the maximal consistency of Φ) $\{s : B, r : (B \to A)\} \subseteq \Phi$ iff (by IH) $\{(s, B), (r, B \to A)\} \subseteq \mathcal{E}(\mathcal{B}_\Phi)$ iff (by Lemma 4.2) $(r \cdot_B s, A) \in \mathcal{E}(\mathcal{B}_\Phi)$. □

Definition 14 (Rank). *We inductively define the* rank *of a term by*

1. $\mathsf{rk}(t) := 1$ *if* $t \in \mathsf{ATm}$*;*
2. $\mathsf{rk}(s \cdot_A t) := \max(\mathsf{rk}(s), \mathsf{rk}(t)) + 1$*;*

and the rank of a formula by

1. $\mathsf{rk}(P) := 1$ *if* $P \in \mathsf{Prop}$*;*
2. $\mathsf{rk}(\neg A) := \mathsf{rk}(A) + 1$*;*
3. $\mathsf{rk}(A \to B) := \max(\mathsf{rk}(A), \mathsf{rk}(B)) + 1$*;*
4. $\mathsf{rk}(t : A) := \mathsf{rk}(t)$*;*
5. $\mathsf{rk}([\Gamma]B) := 2 \cdot \mathsf{rk}(B)$.

We immediately get the following properties of rk.

Lemma 15 (Reduction)

1. $\mathsf{rk}([\Gamma]A) > \mathsf{rk}(A)$.
2. $\mathsf{rk}([\Gamma]\neg B) > \mathsf{rk}(\neg[\Gamma]B)$.
3. $\mathsf{rk}([\Gamma](A \to B)) > \mathsf{rk}([\Gamma]A \to [\Gamma]B)$.
4. $\mathsf{rk}([\Gamma]r \cdot_C s : B) > \mathsf{rk}([\Gamma]r : (C \to B))$ *and* $\mathsf{rk}([\Gamma]r \cdot_C s : B) > \mathsf{rk}([\Gamma]s : C)$.
5. $\mathsf{rk}([\Gamma][\Delta]A) > \mathsf{rk}([\Gamma \cup \Delta]A)$.

Lemma 16 (Truth lemma). *Let Φ be a maximal consistent set of formulas. Then*

$$A \in \Phi \iff \mathcal{M}_\Phi \Vdash A .$$

Proof. By induction on $\mathsf{rk}(A)$.

1. $A \in \mathsf{Prop}$. We have $A \in \Phi$ iff (by definition) $A \in \mathsf{v}_\Phi$ iff (by definition) $\mathcal{M}_\Phi \Vdash A$.
2. $A = \neg B$. We have $\neg B \in \Phi$ iff (by the maximal consistency of Φ) $B \notin \Phi$ iff (by IH) $\mathcal{M}_\Phi \nVdash B$ iff $\mathcal{M}_\Phi \Vdash \neg B$.
3. $A = B \to C$. We have $B \to C \in \Phi$ iff (by the maximal consistency of Φ) $B \notin \Phi$ or $C \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \nVdash B$ or $\mathcal{M}_\Phi \Vdash C$ iff $\mathcal{M}_\Phi \Vdash B \to C$.
4. $A = t{:}B$. We have $t{:}B \in \Phi$ iff (by Lemma 13) $(t, B) \in \mathcal{E}(\mathcal{B}_\Phi)$ iff (by definition) $\mathcal{M}_\Phi \Vdash t : B$.
5. $A = [\Gamma]P$. We have $[\Gamma]P \in \Phi$ iff (by ($\mathsf{Red.1}$) and the maximal consistency of Φ) $P \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \Vdash P$ iff (by ($\mathsf{Red.1}$) and soundness) $\mathcal{M}_\Phi \Vdash [\Gamma]P$.
6. $A = [\Gamma]\neg B$. Then $[\Gamma]\neg B \in \Phi$ iff (by ($\mathsf{Red.2}$) and the maximal consistency of Φ) $\neg[\Gamma]B \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \Vdash \neg[\Gamma]B$ iff (by ($\mathsf{Red.2}$) and soundness) $\mathcal{M}_\Phi \Vdash [\Gamma]\neg B$.
7. $A = [\Gamma](B \to C)$. We have $[\Gamma](B \to C) \in \Phi$ iff (by ($\mathsf{Red.3}$) and the maximal consistency of Φ) $[\Gamma]B \to [\Gamma]C \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \Vdash [\Gamma]B \to [\Gamma]C$ iff (by ($\mathsf{Red.3}$) and soundness) $\mathcal{M}_\Phi \Vdash [\Gamma](B \to C)$.
8. $A = [\Gamma]t : B$. We distinguish the following cases for t.
   (a) $t \in \mathsf{ATm}$. There are two possibilities:
      – $t = \mathsf{up}(B)$ and $B \in \Gamma$. In this case, $A = [\Gamma]\mathsf{up}(B) : B$ is an axiom. Therefore, we have $A \in \Phi$ by the maximal consistency of Φ and $\mathcal{M}_\Phi \Vdash A$ by soundness;
      – either $t \neq \mathsf{up}(B)$ or $B \notin \Gamma$. We have that $[\Gamma]t : B \in \Phi$ iff (by (Pers), ($\mathsf{MC.1}$), and the maximal consistency of Φ) $t : B \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \Vdash t : B$ iff (by (Pers), ($\mathsf{MC.1}$), and soundness) $\mathcal{M}_\Phi \Vdash [\Gamma]t : B$.
   (b) $t = r \cdot_C s$. We have $[\Gamma]r \cdot_C s : B \in \Phi$ iff (by ($\mathsf{MC.2}$) and the maximal consistency of Φ) $\{[\Gamma]r : (C \to B), [\Gamma]s : C\} \subseteq \Phi$ iff (by IH) we have $\mathcal{M}_\Phi \Vdash [\Gamma]r : (C \to B) \wedge [\Gamma]s : C$ iff (by ($\mathsf{MC.2}$) and soundness) we have $\mathcal{M}_\Phi \Vdash [\Gamma]r \cdot_C s : B$.
9. $A = [\Gamma][\Delta]B$. Then $[\Gamma][\Delta]B \in \Phi$ iff (by (It) and the maximal consistency of Φ) $[\Gamma \cup \Delta]B \in \Phi$ iff (by IH) $\mathcal{M}_\Phi \Vdash [\Gamma \cup \Delta]B$ iff (by (It) and soundness) $\mathcal{M}_\Phi \Vdash [\Gamma][\Delta]B$. □

Theorem 17 (Completeness). *For all formulas D,*

$$D \textit{ is valid with respect to } \mathsf{CS} \quad \textit{implies} \quad \mathsf{JUP_{CS}} \vdash D \ .$$

Proof. Assume $\mathsf{JUP_{CS}} \nvdash D$. Then $\{\neg D\}$ is consistent and, hence, contained in a maximal consistent set Φ. By the previous lemma we find $\mathcal{M}_\Phi \Vdash \neg D$. Thus, we conclude $\mathcal{M}_\Phi \nVdash D$, which means that D is not valid with respect to CS since $\mathcal{M}_\Phi$ is an initial CS-model.

We now show that the update with the empty set $\Gamma = \varnothing$, i.e., the update with no additional information, has no effect.

Lemma 18 (Uninformative update). $\mathsf{JUP_{CS}} \vdash [\varnothing]A \leftrightarrow A$.

Proof. Follows from $\mathcal{M}^\varnothing = \mathcal{M}$ (Lemma 8) and completeness. □

We show the following principle of minimal change: an update with A has no effect on beliefs that are justified without reference to that update.

Definition 19 (Subterms). Subterms *of a term t are defined by induction as follows.* $Sub(t) := \{t\}$ *if* $t \in \mathsf{ATm}$.

$$Sub(t \cdot_A s) := Sub(t) \cup Sub(s) \cup \{t \cdot_A s\} \ .$$

Lemma 20 (Minimal change). *Let Γ be a finite set of formulas and t be a term that does not contain $\mathsf{up}(A)$ as a subterm for any $A \in \Gamma$. Then*

$$\mathsf{JUP_{CS}} \vdash [\Gamma]t : B \leftrightarrow t : B \ .$$

Proof. The direction from right to left follows from (Pers). To show the other direction, let $\mathcal{M} = (\mathsf{v}, \mathcal{B})$ be an initial CS-model with $\mathcal{M} \Vdash [\Gamma]t : B$. We prove $\mathcal{M} \Vdash t : B$ by induction on the construction of t.

1. $t \in \mathsf{ATm}$. Then, $(t, B) \in \mathcal{B}^\Gamma = \mathcal{B} \cup \{(\mathsf{up}(A), A) \mid A \in \Gamma\}$ by Lemma 4.1. Since $t \neq \mathsf{up}(A)$ for any $A \in \Gamma$, we find $(t, B) \in \mathcal{B}$. Thus, $(t, B) \in \mathcal{E}(\mathcal{B})$ by Lemma 4.1, and $\mathcal{M} \Vdash t : B$ follows.
2. $t = r \cdot_C s$. Then $\mathcal{M} \Vdash [\Gamma]s : C \wedge [\Gamma]r : (C \rightarrow B)$ by ($\mathsf{MC.2}$) and soundness. By IH we find $\mathcal{M} \Vdash s : C \wedge r : (C \rightarrow B)$. Thus, we conclude by ($\mathsf{App}$) and soundness that $\mathcal{M} \Vdash t : B$.

The claim follows by completeness. □

5 AGM Postulates

In the now classic paper [1], Alchourrón, Gärdenfors, and Makinson introduced their famous postulates for belief contraction and revision where the underlying principle is that of minimal change. Later Gärdenfors [17] added postulates for belief expansion. We are going to show that the update operator of $\mathsf{JUP_{CS}}$ satisfies these postulates for expansion, see Lemma 27.

Before we can state and prove Gärdenfors's postulates, we need to introduce the notion of *belief set* and of *belief set induced by a model.*

Definition 21 (Belief set). *A* belief set *is a set $X \subseteq \mathsf{Fml}$ of formulas that satisfies*

$$\textit{if } A \in X \textit{ and } A \rightarrow B \in X, \textit{ then } B \in X.$$

Definition 22 (Induced beliefs). *Let $\mathcal{M}$ be a model. We define the* beliefs $\Box_\mathcal{M}$ induced by $\mathcal{M}$ *as*

$$\Box_\mathcal{M} := \{A \in \mathsf{Fml} \mid \mathcal{M} \Vdash t : A \textit{ for some } t \in \mathsf{Tm}\} \ .$$

Lemma 23 (Induced beliefs). *Let $\mathcal{M}$ be a model. Then $\Box_\mathcal{M}$ is a belief set.*

Proof. We have to show the condition of Definition 21. Assume that $A \in \Box_\mathcal{M}$ and $A \to B \in \Box_\mathcal{M}$. Then there are terms s and t such that $\mathcal{M} \Vdash t : A$ and $\mathcal{M} \Vdash s : (A \to B)$. Therefore, $\mathcal{M} \Vdash s \cdot_A t : B$ and hence $B \in \Box_\mathcal{M}$. □

Definition 24 (Expansion). *Let $\mathcal{M}$ be a model and A be a formula. We define*

$$\Box_\mathcal{M} \oplus A := \Box_{\mathcal{M}^A} \quad .$$

Definition 25 (Appropriate constant specification). *A constant specification* CS *is called*

– propositionally appropriate *if for every A that is a propositional tautology there exists a constant c such that $(c, A) \in \mathsf{CS}$;*
– axiomatically appropriate *if for every A that is an axiom of* JUP *there exists a constant c such that $(c, A) \in \mathsf{CS}$;*
– JUP_CS-appropriate *if it is axiomatically appropriate and also for every pair $(c, B) \in \mathsf{CS}$ there exists a constant c' such that $(c', c : B) \in \mathsf{CS}$.*

Lemma 26 (CS appropriateness as a measure of reasoning strength). *Let $\mathcal{M}$ be an initial CS-model. If CS is propositionally appropriate (axiomatically appropriate, JUP_CS-appropriate), belief sets $\Box_\mathcal{M}$ and $\Box_\mathcal{M} \oplus A$ are closed with respect to reasoning in classical propositional logic (in $\mathsf{JUP}_\varnothing$, in JUP_CS).*

Proof. What we need to prove is that whenever $C \in \Box_\mathcal{M}$ ($C \in \Box_\mathcal{M} \oplus A$) and $C \vdash_{Th} D$, it follows that $D \in \Box_\mathcal{M}$ ($D \in \Box_\mathcal{M} \oplus A$), where Th stands for classical propositional logic in the language of JUP in the case of a propositionally appropriate CS, for $\mathsf{JUP}_\varnothing$ in the case of an axiomatically appropriate CS, and for JUP_CS in the case of a JUP_CS-appropriate CS. This can be easily demonstrated by induction on the derivation in the respective logic.

Lemma 27 (Postulates for expansion). *Let $\mathcal{M} = (\mathsf{v}, \mathcal{B})$ be a model and A be a formula. Then $X = \Box_\mathcal{M} \oplus A$ satisfies the following properties:*

1. *X is a belief set.*
2. *$A \in X$.*
3. *$\Box_\mathcal{M} \subseteq X$.*

Moreover, $\Box_\mathcal{M} \oplus A$ is the smallest set satisfying Properties 1–3:

4. *for any set $X \subseteq \mathsf{Fml}$ satisfying Properties 1–3 we have $\Box_\mathcal{M} \oplus A \subseteq X$.*

Proof. 1. Since Lemma 23 holds for arbitrary models, we immediately obtain that $\Box_\mathcal{M} \oplus A = \Box_{\mathcal{M}^A}$ is a belief set.
2. By (Up) and soundness we have $\mathcal{M} \Vdash [A]\mathsf{up}(A) : A$, in other words, we have $\mathcal{M}^A \Vdash \mathsf{up}(A) : A$. Thus we get $A \in \Box_{\mathcal{M}^A}$, i.e., $A \in \Box_\mathcal{M} \oplus A$.
3. Assume $B \in \Box_\mathcal{M}$. There exists a term t such that $\mathcal{M} \Vdash t : B$. By (Pers) and soundness, we have $\mathcal{M} \Vdash [A]t : B$, in other words, $\mathcal{M}^A \Vdash t : B$. Thus, we get $B \in \Box_{\mathcal{M}^A}$, i.e., $B \in \Box_\mathcal{M} \oplus A$.

4. Let X satisfy Properties 1–3. We have to show $\Box_{\mathcal{M}} \oplus A \subseteq X$. By the definition of $\Box_{\mathcal{M}} \oplus A$, this amounts to showing
$$\mathcal{M}^A \Vdash t : B \quad \text{implies} \quad B \in X \ . \tag{1}$$
Let $\mathcal{M}^A = (\mathsf{v}, \mathcal{B}^A)$. To establish (1) it is enough to show
$$(t, B) \in \mathcal{E}(\mathcal{B}^A) \quad \text{implies} \quad B \in X \ . \tag{2}$$
We prove (2) by induction on the construction of t. Assume $(t, B) \in \mathcal{E}(\mathcal{B}^A)$. We have one of the following cases:
   (a) $t \in \mathsf{ATm}$. By Lemma 4.1, $(t, B) \in \mathcal{B}^A = \mathcal{B} \cup \{(\mathsf{up}(A), A)\}$. If $(t, B) \in \mathcal{B}$, then $(t, B) \in \mathcal{E}(\mathcal{B})$ by Lemma 4.1, thus, $\mathcal{M} \Vdash t : B$, i.e., $B \in \Box_{\mathcal{M}}$. By Property 3 for X, we find $B \in X$.
   If $(t, B) = (\mathsf{up}(A), A)$, then $B = A$, and $B \in X$ follows by Property 2 for X.
   (b) $t = r \cdot_C s$. Then $\{(s, C), (r, C \to B)\} \subseteq \mathcal{E}(\mathcal{B}^A)$ by Lemma 4.2. By IH we find $\{C, C \to B\} \subseteq X$. By Property 1 for X we know that $C \in X$ and $C \to B \in X$ imply $B \in X$. Hence, we conclude that $B \in X$. □

Remark 28. Gärdenfors [17] presented two more postulates that in our context read as

1. if $A \in \Box_{\mathcal{M}}$, then $\Box_{\mathcal{M}} = \Box_{\mathcal{M}} \oplus A$
2. if $\Box_{\mathcal{M}} \subseteq \Box_{\mathcal{M}'}$, then $\Box_{\mathcal{M}} \oplus A \subseteq \Box_{\mathcal{M}'} \oplus A$.

It is standard [16] to show that these two additional postulates follow from the properties established in Lemma 27.

6 Ramsey Axiom

The Ramsey axiom makes it possible to express the beliefs after an update in terms of the beliefs before the update. In dynamic doxastic logic, for example, Segerberg [24] formulates the Ramsey axiom as
$$[A]\Box B \leftrightarrow \Box(A \to B) \ . \tag{3}$$
Thus, it states that an agent believes B after an update with A if and only if the agent believes that A implies B before the update.

We can establish an explicit analog of the Ramsey axiom in $\mathsf{JUP_{CS}}$ for propositionally appropriate constant specifications.

We show the two implications of the Ramsey axiom separately. First, the direction from right to left.

Lemma 29 (Ramsey I). $\mathsf{JUP_{CS}} \vdash t : (A \to B) \to [A]t \cdot_A \mathsf{up}(A) : B$.

Proof. Let $\mathcal{M}$ be an initial CS-model. Assume $\mathcal{M} \Vdash t : (A \to B)$ for some term t. By the axiom (Pers) and soundness, $\mathcal{M} \Vdash [A]t : (A \to B)$. Moreover, using the axiom (Up) we find $\mathcal{M} \Vdash [A]\mathsf{up}(A) : A$ and by ($\mathsf{MC.2}$) we obtain $\mathcal{M} \Vdash [A]t \cdot_A \mathsf{up}(A) : B$. The claim follows by completeness. □

The direction from left to right of (3) need not hold in general. Here is a simple counter-example. By (Up) we have $\mathsf{JUP}_{\mathsf{CS}} \vdash [A]\mathsf{up}(A){:}A$. However, if the constant specification CS is not propositionally appropriate, e.g., for $\mathsf{CS} = \varnothing$, any model $\mathcal{M} = (\mathsf{v}, \varnothing)$ is an initial $\varnothing$-model. It is easy to see that $\mathcal{E}(\varnothing) = \varnothing$ and $\mathcal{M} \nVdash t : B$ for any term t and any formula B. Now $\mathsf{JUP}_{\varnothing} \nvdash t : (A \to A)$ by completeness.

For a propositionally appropriate constant specification, we do have an explicit version of the direction from left to right.

Lemma 30 (Ramsey II). *Let* CS *be a propositionally appropriate constant specification. For each term t there exists a term s such that*

$$\mathsf{JUP}_{\mathsf{CS}} \vdash [A]t : B \to s : (A \to B) \ . \tag{4}$$

Proof. By induction on the construction of t we show that there exists a term s such that $\mathcal{M} \Vdash s : (A \to B)$ for any initial CS-model $\mathcal{M}$ whenever $\mathcal{M} \Vdash [A]t : B$. Then (4) follows by completeness. We distinguish the following cases for t:

1. $t \in \mathsf{ATm}$. There are two possibilities:
   - $t \neq \mathsf{up}(B)$ or $A \neq B$. Since CS is propositionally appropriate, there exists a constant c such that $\mathsf{JUP}_{\mathsf{CS}} \vdash c{:}(B \to (A \to B))$ and we set $s := c \cdot_B t$. If $\mathcal{M} \Vdash [A]t{:}B$, then $\mathcal{M} \Vdash t{:}B$ by the axiom of minimal change $(\mathsf{MC.1})$. Hence, we conclude $\mathcal{M} \Vdash c \cdot_B t : (A \to B)$.
   - $t = \mathsf{up}(B)$ and $A = B$. Since CS is propositionally appropriate, there is a constant c such that $\mathsf{JUP}_{\mathsf{CS}} \vdash c : (A \to B)$ and we set $s := c$. Then, $\mathcal{M} \Vdash c : (A \to B)$
2. $t = r \cdot_C s$. By IH there are terms r' and s' such that $\mathcal{M} \Vdash r'{:}\big(A \to (C \to B)\big)$ whenever $\mathcal{M} \Vdash [A]r{:}(C \to B)$ and $\mathcal{M} \Vdash s'{:}(A \to C)$ whenever $\mathcal{M} \Vdash [A]s{:}C$ for any initial CS-model $\mathcal{M}$. Assume $\mathcal{M} \Vdash [A]r \cdot_C s : B$. It follows by $(\mathsf{MC.2})$ that $\mathcal{M} \Vdash [A]r : (C \to B)$ and $\mathcal{M} \Vdash [A]s : C$. Since CS is propositionally appropriate, there exists a constant c such that

$$\mathsf{JUP}_{\mathsf{CS}} \vdash c : \Big(\big(A \to (C \to B)\big) \to \big((A \to C) \to (A \to B)\big)\Big) \ .$$

   Then for $s := (c \cdot_{A \to (C \to B)} r') \cdot_{A \to C} s'$ we have $\mathcal{M} \Vdash s : (A \to B)$. □

7 Conclusion

We have introduced $\mathsf{JUP}_{\mathsf{CS}}$, a justification logic for belief expansion. The explicit evidence terms in $\mathsf{JUP}_{\mathsf{CS}}$ keep track of the effect an update has on an agent's beliefs, which makes it possible to axiomatize in the object language the principle of minimal change and establish soundness and completeness.

There are two directions for further research. One is to study belief contraction and revision in the context of justification logic. This is closely related to [22] where evidence elimination is studied.

A second line of research is to consider introspective agents. It is straightforward to add positive introspection to $\mathsf{JUP}_{\mathsf{CS}}$ since semantically this corresponds to a positive operator and, therefore, the least fixed point construction of the evidence relation still works. However, the properties with respect to belief sets and the Ramsey axiom will be different and, not surprisingly, the Moore's paradox will reappear. Adding negative introspection is also possible. The non-monotone inductive definitions used in the model constructions [25] for negative introspection provide a short preview of its belief dynamics.

References

1. 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)
2. Artemov, S.N.: Operational modal logic. Technical Report MSI 95–29, Cornell University (December 1995)
3. Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
4. Artemov, S.N.: Kolmogorov and Gödel's approach to intuitionistic logic: current developments. Russian Mathematical Surveys 59(2), 203–229 (2004) (originally published in Russian)
5. Artemov, S.: Justified common knowledge. Theoretical Computer Science 357(1-3), 4–22 (2006)
6. Artemov, S.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
7. Artemov, S.: Tracking Evidence. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Fields of Logic and Computation. LNCS, vol. 6300, pp. 61–74. Springer, Heidelberg (2010)
8. Artemov, S.N.: The Ontology of Justifications in the Logical Setting. Studia Logica 100(1-2), 17–30 (2012)
9. Artemov, S., Kuznets, R.: Logical omniscience as a computational complexity problem. In: Heifetz, A. (ed.) TARK 2009, Stanford University, California, July 6–8, pp. 14–23. ACM (2009)
10. Baltag, A., Renne, B., Smets, S.: The Logic of Justified Belief Change, Soft Evidence and Defeasible Knowledge. In: Ong, L., de Queiroz, R. (eds.) WoLLIC 2012. LNCS, vol. 7456, pp. 168–190. Springer, Heidelberg (2012)
11. Barwise, J.: Admissible Sets and Structures. Springer (1975)
12. Bucheli, S., Kuznets, R., Renne, B., Sack, J., Studer, T.: Justified Belief Change. In: Arrazola, X., Ponte, M. (eds.) LogKCA-10, Proceedings of the Second ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, pp. 135–155. University of the Basque Country Press (2010)
13. Bucheli, S., Kuznets, R., Studer, T.: Justifications for common knowledge. Journal of Applied Non-Classical Logics 21(1), 35–60 (2011)
14. Bucheli, S., Kuznets, R., Studer, T.: Partial Realization in Dynamic Justification Logic. In: Beklemishev, L.D., de Queiroz, R. (eds.) WoLLIC 2011. LNCS (LNAI), vol. 6642, pp. 35–51. Springer, Heidelberg (2011)
15. Bucheli, S., Kuznets, R., Studer, T.: Realizing public announcements by justifications. Journal of Computer and System Sciences (to appear)
16. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer (2007)
17. Gärdenfors, P.: Knowledge in Flux. The MIT Press (1988)

18. Gerbrandy, J., Groeneveld, W.: Reasoning about Information Change. Journal of Logic, Language and Information 6(2), 147–169 (1997)
19. Kuznets, R., Studer, T.: Justifications, Ontology, and Conservativity. In: Bolander, T., Braüner, T., Ghilardi, S., Moss, L. (eds.) Advances in Modal Logic, vol. 9, pp. 437–458. College Publications (2012)
20. Mkrtychev, A.: Models for the Logic of Proofs. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)
21. Renne, B.: Dynamic Epistemic Logic with Justification. PhD thesis, CUNY Graduate Center (May 2008)
22. Renne, B.: Evidence elimination in multi-agent justification logic. In: Heifetz, A. (ed.) TARK 2009, Stanford University, California, July 6–8, pp. 227–236. ACM (2009)
23. Renne, B.: Public communication in justification logic. Journal of Logic and Computation 21(6), 1005–1034 (2011)
24. Segerberg, K.: Belief Revision From the Point of View of Doxastic Logic. Logic Journal of the IGPL 3(4), 535–553 (1995)
25. Studer, T.: Decidability for some justification logics with negative introspection. Journal of Symbolic Logic (to appear)

Separating the Fan Theorem and Its Weakenings

Robert S. Lubarsky[1] and Hannes Diener[2]

[1] Florida Atlantic University, Boca Raton, FL 33431, USA
Robert.Lubarsky@alum.mit.edu
[2] University of Siegen, 57068 Siegen, Germany
diener@math.uni-siegen.de

Abstract. Varieties of the Fan Theorem have recently been developed in reverse constructive mathematics, corresponding to different continuity principles. They form a natural implicational hierarchy. Some of the implications have been shown to be strict, others strict in a weak context, and yet others not at all, using disparate techniques. Here we present a family of related Kripke models which suffices to separate all of the as yet identified fan theorems.

Keywords: fan theorems, Kripke models, forcing, non-standard models, Heyting-valued models.

AMS 2010 MSC: 03C90, 03F50, 03F60, 03H05.

1 Introduction

To be able to talk about fans, Cantor space, and similar objects properly, we will start by introducing some notation. The space of all infinite binary sequences, endowed with the usual metric, will be denoted by $2^{\mathbb{N}}$; the space of all finite binary sequences will be denoted by 2^*. The concatenation of $u, v \in 2^*$ will be denoted by $u * v$. The length of a finite sequence $u \in 2^*$ will be denoted by $|u|$. For $\alpha \in 2^{\mathbb{N}}$ and $n \in \mathbb{N}$, the first n elements of α form a finite sequence denoted by $\overline{\alpha}n$. A subset $B \subset 2^*$ is called a **bar** if

$$\forall \alpha \in 2^{\mathbb{N}} \exists n \in \mathbb{N}(\overline{\alpha}n \in B),$$

and a bar is called **uniform** if

$$\exists n \in \mathbb{N} \forall \alpha \in 2^{\mathbb{N}} \exists m \leqslant n(\overline{\alpha}m \in B).$$

Notice that if a bar B is closed under extensions, that is if

$$\forall u \in 2^*(u \in B \implies \forall w \in 2^* u * w \in B),$$

then it is uniform if and only if

$$\exists n \in \mathbb{N} \forall \alpha \in 2^{\mathbb{N}}(\overline{\alpha}n \in B).$$

Not all of the bars we consider will be so closed.

There are currently four versions of Brouwer's fan theorem in common use. All of them enable one to conclude that a given bar is uniform. The differences among them lie in the required complexity of the bar, which ranges from the very strongest requirement to no restriction on the bar at all. A bar $C \subset 2^*$ is **decidable** if it is decidable as a set:

$$\forall u \in 2^* \; u \in C \vee u \notin C.$$

A bar $C \subset 2^*$ is called a ***c*-bar** if there exists a decidable set $C' \subset 2^*$ such that

$$u \in C \iff \forall w \in 2^* \; (u * w \in C') \, .$$

A bar $B \subset 2^*$ is called a $\mathbf{\Pi^0_1}$**-bar** if there exist a decidable set $S \subset 2^* \times \mathbb{N}$ such that

$$u \in B \iff \forall n (u, n) \in S \; .$$

We can now state four commonly used versions of the fan theorem.

FAN_{Δ}:	Every decidable bar is uniform.
FAN_{c}:	Every c-bar is uniform.
$\mathrm{FAN}_{\Pi^0_1}$:	Every Π^0_1-bar is uniform.
$\mathrm{FAN}_{\mathrm{full}}$:	Every bar is uniform.

Notice that every decidable bar can be taken to be closed under extensions; that is, the closure of a decidable bar under extension is still decidable. If there is no restriction on the definability of a bar, then every bar can be taken to be so closed, by working with the closure of any given bar. Every c-bar is already closed under extension. In contrast, Π^0_1-bars seemingly cannot be replaced by their closures while remaining Π^0_1. These principles were developed within reverse constructive mathematics, because they are equivalent with certain continuity principles [2, 5, 7].

The following implications hold trivially [2, 4]:

$$\mathrm{FAN}_{\mathrm{full}} \implies \mathrm{FAN}_{\Pi^0_1} \implies \mathrm{FAN}_{c} \implies \mathrm{FAN}_{\Delta}.$$

One naturally wonders whether any of the implications can be reversed, including whether FAN_{Δ} is outright provable in constructive set theory. Some such non-implications have already been determined. It is well-known (see [1] for instance) that FAN_{Δ} is not provable, via recursive realizability. Berger [3] shows that FAN_{Δ} does not imply FAN_{c} over a very weak base system. Fourman and Hyland [6] present a Heyting-valued, almost topological, model in which $\mathrm{FAN}_{\mathrm{full}}$ fails; we show below that $\mathrm{FAN}_{\Pi^0_1}$ holds in their model, separating the left-most pair of principles in the diagram above. We are not aware of any prior proofs separating FAN_{c} and $\mathrm{FAN}_{\Pi^0_1}$.

The goal of this paper is to separate all of these principles via a uniform technique, based on Kripke models. To build a tree we could control, along with its paths, over full set theory, we turned to forcing. (Because of space limitations, we assume familiarity with forcing, including standard notation, and refer the reader to any standard text for this background.) In order to have the trees be decidable, yet not completely pinned down, as required by the theories in question, we were forced to use non-standard integers, to provide non-standard levels on the trees.

2 The Fan Theorem in Heyting-Valued Models

Fourman and Hyland show that:

Proposition 1. *(Fourman-Hyland [6]) In any topological model* FAN_{full} *holds.*

Their proof suggests that if we are looking for models in which some form of the fan theorem fails we need to "delete points". This was done in [6], section 4, where they consider $K(T)$, the coperfect open sets of a topological space T. This can be viewed as the equivalence classes of the open sets of T, under which an open set is identified with its smallest coperfect superset. In this setting, removing a point from an open set does not change the set.

Definition 1. *A Heyting algebara is connected if* $A \vee B = \top$ *and* $A \wedge B = \bot$ *implies that either* $A = \top$ *or* $A = \bot$.

Let Ω be $K([0,1] \times [0,1])$. It is easy to see that Ω is connected.

Proposition 2. *If* H *is a connected Heyting algebra, then* $H \Vdash FAN_{\Pi^0_1}$.

Proof. Suppose $H \Vdash$ "B is a Π^0_1-bar, given say by $S : u \in B$ iff $\forall n \in \mathbb{N}\ (u,n) \in S$." Since $H \Vdash$ "S is decidable," for any $u \in 2^*$ and $n \in \mathbb{N}$,

$$H \Vdash \text{“}(u,n) \in S \vee (u,n) \notin S.\text{”}$$

By the connectedness of H either $H \Vdash$ "$(u,n) \in S$" or $H \Vdash$ "$(u,n) \notin S$." So define a set $\tilde{B} \subset 2^*$ in the metatheory by

$$u \in \tilde{B} \iff \forall n \in \mathbb{N}\ H \Vdash \text{“}(u,n) \in S.\text{”}$$

$\tilde{B}$ is itself a bar: for let $\alpha \in 2^{\mathbb{N}}$ be arbitrary. If $\overline{\alpha}n \notin \tilde{B}$ for all $n \in \mathbb{N}$ then for all n there exist i_n such that $[\![(\overline{\alpha}n, i_n) \in B]\!] = \bot$. Thus

$$[\![\forall m \in \mathbb{N}\ (\overline{\alpha}n, m) \in B]\!] = \bot$$

for any $n \in \mathbb{N}$, and therefore

$$[\![\exists n \in \mathbb{N} \forall m \in \mathbb{N}\ (\overline{\alpha}n, m) \in B]\!] = \bot\ ;$$

a contradiction to B being a bar internally. Hence $\tilde{B}$ is bar externally, and therefore, working with a classical metatheory (or simply the Fan Theorem), it is uniform. So there exists N such that for all $u \in 2^N$ some initial segment of u is in $\tilde{B}$. Then it is easy to see, that this same N witnesses the uniformity of B internally.

Corollary 3. $FAN_{\Pi^0_1}$ *does not imply* FAN_{full} *(over IZF).*

Proof. In [6] it shown that $\Omega \not\Vdash \text{FAN}_{\text{full}}$.

3 FAN_Δ Is Not Provable

We will build a Kripke model, working within ZFC. To construct a bar, it will be crucial to control what paths exist. This is most easily done with a generic.

Definition 2. *Let the forcing partial order P be the set of appropriate labelings of finitely many nodes from 2^*. A labeling of nodes assigns to each one either IN, OUT, or ∞, with the following restrictions. Any node labeled IN has no descendant, the idea being that once a node gets into the eventual bar so are all of its descendants automatically, so nothing more need be said. Any descendant of a node labeled OUT must labeled IN or OUT. Finally, for any node labeled ∞, if both children are labeled, then at least one of them must be labeled ∞.*

Let G be a generic through the condition that labels $\langle\rangle$ with ∞. By straightforward density arguments, any node labeled OUT by G has a uniform bar above (resp. below) it all labeled IN, and every node labeled ∞ has a path through it always labeled ∞, in fact a perfect set of such.

Let $B = \{\alpha \in 2^* | \text{ for some } n\ G(\alpha \restriction n) = \text{IN}\}$. $B \in M[G]$ is the interpretation σ_B^G of the term $\sigma_B = \{\langle p, \hat{\alpha}\rangle \mid \text{for some } n\ p(\alpha \restriction n) = \text{IN}\}$. (As usual, the function $\hat{}$ is the canonical injection of the ambient universe into the terms: $\hat{x} = \{\langle \emptyset, \hat{y}\rangle \mid y \in x\}$.) Because of these latter ∞-paths, B is not a bar. However, we might reasonably think that if we no longer had access to the distinction between the OUT and the ∞ nodes, we might no longer be able to build a path avoiding B. This intuition is confirmed by the next lemma.

Definition 3. *The shadow forcing Q is the set of functions from finite subtrees of 2^* to $\{$IN, OUT$\}$ such that any node labeled IN has no descendant. Equivalently, Q is the sub-p.o. of P beneath the condition labeling $\langle\rangle$ with OUT (together with the condition which labels $\langle\rangle$ IN, which has no extension). The canonical projection $proj_Q$ of P onto Q replaces all occurrences of ∞ with OUT. The canonical projection of the terms of P's forcing language to those of Q's, ambiguously also called $proj_Q$, acts by applying $proj_Q$ to the conditions that appear in the terms, hereditarily. (Notice that Q term are also P terms.)*

Notice that a P-filter projects to a Q-filter. If G is a generic P-filter, then $\mathrm{proj}_Q(G)$ will not be Q-generic, because in Q the terminal conditions are dense. Still, $\mathrm{proj}_Q(G)$ induces an interpretation $\sigma^{proj_Q(G)}$ of the terms σ of Q. These interpretations are in $M[G]$, as they are easily definable from σ and G; alternatively, $\sigma^{proj_Q(G)} = (proj_Q^{-1}{}''\sigma)^G$.

For any P-filter G, $\mathrm{proj}_Q(\sigma_B)^{proj_Q(G)} = B$: the induced interpretation of the projection of B is just B itself. Effectively, B as a P-term is already a Q-term.

Proposition 4. *If σ is a Q-term and $p \Vdash_P$ "$proj_Q^{-1}{}''\sigma$ is an infinite branch through 2^*," then $p \Vdash_P$ "$proj_Q^{-1}{}''\sigma$ goes through σ_B."*

Proof. By standard forcing technology, it suffices to extend p to some condition forcing "$proj_Q^{-1}{}''\sigma$ goes through σ_B," as then it will be dense beneath p to force as much, and so will happen generically.

First extend p so that every sequence in 2^* of length 2^{n-1} for some n either is labeled OUT or ∞ or has a proper initial segment labeled IN. Then extend again by adjoining both children to all nodes of length 2^{n-1}, and labeling them ∞ whenever possible (otherwise IN or OUT). For a technical reason soon to become clear, we must extend yet again. This time have the domain include all length k descendants of the length n nodes not labeled IN, and label them so that every length n node labeled ∞ has a unique descendant of length k labeled ∞, and, most importantly, for each pair of nodes α and β of length k labeled ∞, there is some i with $\alpha(i) = 1$ and $\beta(i) = 0$. One way of doing this is to let s be the number of nodes of length n labeled ∞, to let k be $n + s$, and to build the ∞-labeled descendant of the j^{th} such node by adjoining to it $j - 1$ 0's, a 1, and then $s - j$ 0's, all other descendants of length k being labeled OUT.

Extend one last time to $q \Vdash proj_Q^{-1}{}''\sigma(\hat{k}) = \hat{\alpha}$ for some fixed α, where as usual $\hat{x}$ is the standard term for the internalization of the set x. Moreover, q should force the equality in the strong sense that for each $j < k$ there is a term τ and a condition $r \geq q$ with $\langle r, \tau\rangle \in proj_Q^{-1}{}''\sigma$ and $q \Vdash \tau = \langle \hat{j}, \hat{\alpha}(\hat{j})\rangle$; even further, if $\alpha(j) = 1$ then q forces a particular element to be in τ's second component.

If q labels some initial segment of α IN then we're done.

If q labels α OUT then it is dense beneath q that all descendants of α of some fixed length are labeled IN, and again we're done.

If q labels α ∞ then let q_{alt} be identical to q except that all descendants of $\alpha \upharpoonright n$ labeled ∞ by q are labeled OUT by q_{alt}. Observe first that q_{alt} extends p. Then note that, because $proj_Q(q_{alt}) = proj_Q(q)$, the strong forcing facts posited of q hold for q_{alt} as well: for the same τ and j as above, $q_{alt} \Vdash \tau \in proj_Q^{-1}{}''\sigma$ and $q_{alt} \Vdash$ "τ is an ordered pair with first component $\hat{j}$," and if q forced τ's second component to be non-empty, q_{alt} also forces it to be non-empty, containing the same term as for q. The difference between q and q_{alt}, from σ's point of view, is that q_{alt} has more extensions than q: there are conditions extending q_{alt} which bar the tree beneath α, which is not so for q. That means that it is possible for extensions of q_{alt} to force sets into Q-terms that no extension of q could. In the case of $proj_Q^{-1}{}''\sigma(\hat{k})$, though, such opportunities are limited. That term is already forced by p to be a function with domain k; for each $j < k$ there is already a fixed term forced to stand for $\langle j, (proj_Q^{-1}{}''\sigma(\hat{k}))(j)\rangle$; if that function value at j was forced by q to be 1 then it must retain a member and so is also forced by q_{alt} to be 1. The only change possible is that something formerly forced to be empty (i.e. be 0) could now be forced by some extension to have an element (i.e. be 1). Recall, though, the construction of q on level k: if $proj_Q^{-1}{}''\sigma(\hat{k})$ is ever forced by some $r \leq q_{alt}$ to be some $\beta \neq \alpha$ by flipping some 0's to 1's, by α's distinguished 1 r cannot label β ∞. So r can be extended so that all extensions of β of a certain length are labeled IN, forcing $proj_Q^{-1}{}''\sigma$ to hit σ_B. Of course, any extension of q_{alt} forcing $proj_Q^{-1}{}''\sigma(\hat{k})$ to be α works the same way as such an r does, since q_{alt} already labels α OUT. In either case we have an extension of p forcing $proj_Q^{-1}{}''\sigma$ go through σ_B.

Even though we have just seen that B is a bar relative to the Q-paths, we will perhaps surprisingly have occasion to consider weaker situations, where B is larger and hence even easier to hit. The case of interest is if we were to change some ∞'s in G to OUTs, thereby allowing uniform bars above those nodes. Notice that if α's sibling is not labeled ∞, then α's label could not consistently be changed from ∞, as then α's parent, labeled ∞, would then have both children not labeled ∞. Such considerations do not apply when $\alpha = \langle\rangle$.

Definition 4. *H is a legal weakening of G if H can be constructed by choosing finitely many nodes labeled ∞ by G, changing those labels (to either IN or OUT), also changing the labeling of finitely many descendants of those nodes from ∞ or OUT to OUT or IN in such a way that each node labeled OUT has a uniform bar above it labeled IN, and then eliminating all descendants of nodes labeled IN. Furthermore, this must be done in such a manner that H is a filter through P (avoiding, for instance, the problem posed just before this definition).*

Notice that the difference between H and G can be summarized in one condition p, which contains the new bars, all labeled IN, and all of their ancestors. Hence we use the notation G_p to stand for this H: to build G_p, make the minimal change to each condition in G in order to be consistent with p.

Lemma 5. *If G_p is a legal weakening of G then G_p is generic through p.*

Remark 6. *Notice that if p labels the empty sequence IN or OUT then $p = G_p$ is a terminal condition in P, trivially satisfying the lemma.*

Proof. Let D be dense beneath p. Notice that $G \restriction dom(p)$ is a condition in P contained in G. It is not hard to define the notion of projection beneath p, $proj_p$, by making the minimal changes in a condition necessary to be compatible with p. We claim that $proj_p^{-1}{}''D$ is dense beneath $G \restriction dom(p)$. To see this, let $q \leq G \restriction dom(p)$. Extend $proj_p(q)$ to $r \in D$. The only way r can extend $proj_p(q)$ is by labeling extensions α of nodes which are unchanged by $proj_p$: if $\alpha \in dom(r) \backslash dom(proj_p(q))$ then, for $\alpha \restriction n \in dom(q)$, $q(\alpha \restriction n) = proj_p(q)(\alpha \restriction n)$. Extend q to q_r by labeling those same extensions the same way: for $\alpha \in dom(r) \backslash dom(proj_p(q))$ $q_r(\alpha) = r(\alpha)$. We have that $proj_p(q_r) = r$, hence $q_r \in proj_p^{-1}{}''D$. So $proj_p^{-1}{}''D$ is dense beneath $G \restriction dom(p)$, hence contains a member of G, say q. Then $proj_p(q)$ is in both D and G_p.

We can now start to describe the ultimate Kripke model. Recall that G is generic for P over M and labels the empty sequence with ∞. The bottom node $\perp$ of the Kripke model consists of the Q-terms, with membership (not equality!) as interpreted by $proj_Q(G)$. Let N be an ultrapower of $M[G]$ using any non-principal ultrafilter on ω, with elementary embedding $f : M[G] \to N$. This necessarily produces non-standard integers. Let $\mathcal{H}$ be the set of legal weakenings of $f(G)$, as defined in N, which induce the same B on the standard levels of 2^*, which restriction is definable only in $M[G]$. That is, any standard node labeled ∞ by G can only be changed to OUT by the legal weakening. $\mathcal{H}$ will index the successors of $\perp$. At the node indexed by $f(G)_p$, the universe will be the Q-terms of N as interpreted by $proj_Q(f(G)_p)$. Regarding the embeddings from $\perp$, for a Q-term $\sigma \in M$, $f(\sigma)$ is an $f(Q)$-term in N, so send σ to $f(\sigma)$. If $f(G)_p$ is a terminal condition in P, then the node indexed by $f(G)_p$ is terminal in the Kripke ordering.

Else iterate. That is, suppose $f(G)_p$ is non-terminal. The structure at its node can be built in N. As an ultrapower of $M[G]$, N internally looks like $f(M)[f(G)]$; internally, $f(G)$ is $f(P)$-generic over the ground model $f(M)$. The structure at node $f(G)_p$ could be built in $f(M)[f(G)_p]$, where, by the previous lemma, $f(G)_p$ is generic through $f(P)$, and also non-terminal. Hence the construction just described, using an ultrapower and legal weakenings to get additional nodes, can be performed in $f(M)[f(G)_p]$ just as above. Continue through ω-many levels. We will ambiguously use f to stand for any of the elementary embeddings, including compositions of such (making f a sort-of polymorhpic transition function). Notice that the construction relativizes: the Kripke structure from node $f(G)_p$ onwards is definable in $f(M)[f(G)_p]$ just as the entire structure is definable in $M[G]$.

This defines a Kripke structure interpreting membership. Equality at any node can now be defined as extensional equality beyond that node in this structure, inductively on the ranks of the terms, even though the model is not well-founded, thanks to the elementarity present. That is, working at $\perp$, suppose σ and τ are terms of rank at most α, and we have defined equality at $\perp$ for all terms of rank less than α. Moreover, suppose (strengthening the inductive assumption here) that this definability was forced in M by the empty condition $\emptyset$. At node $f(G)_p$ the structure is definable over $f(M)[f(G)_p]$, and, by elementarity, in $f(M)$, $\emptyset \Vdash$ "Equality in the Kripke model is unambiguously definable for all terms of rank less that $f(\alpha)$." So at that node we can see whether there is a witness to $f(\sigma)$ and $f(\tau)$ being unequal. If there is such a witness at any node $f(G)_p$, then σ and τ are unequal at $\perp$, else they are equal at $\perp$. This extends the definability of equality to all terms of rank α. Hence inductively equality is definable for all terms.

Proposition 7. $\perp \not\Vdash$ *FAN_Δ*.

Proof. It is immediate that B is a bar: any node is internally of the form $f(M)[f(G)_p]$; by the lemma, $f(G)_p$ is always $f(P)$-generic; by the proposition, no path given by a Q-term can avoid the B as given by an $f(P)$-generic. Moreover, B is decidable, as $f(G)_p$ agrees with G on the standard part of 2^*, the only part that exists at $\perp$, and that argument relativizes to all nodes. However, B is not uniform at any non-terminal node, since $f(G)_p$, when non-terminal, has labels of ∞ at every level.

What remains to show is that our model satisfies IZF. In order to do this, we will need to get a handle on internal truth in the model. This is actually unnecessary for most of the IZF axioms, but for Separation in particular we will have to deal with truth in the model. When forcing, this is done via the forcing and truth lemmas: $M[G] \models \phi$ iff for some $p \in G$ $p \Vdash \phi$, where $\Vdash$ is definable in M. Since our Kripke model is built in $M[G]$, statements about it are statements within $M[G]$, and so are forced by conditions in G. The problem is that the Kripke model internally does not have access to G, but only to B. In detail, Separation for $M[G]$ is proven as follows: given ϕ and σ, it suffices to consider $\{\langle q, \tau\rangle \mid$ for some $\langle p, \tau\rangle \in \sigma$ $q \le p$ and $q \Vdash \phi(\tau)\}$. The problem we face is that that set seems not to be in the Kripke model, even if σ is. What we need to show is that if σ and ϕ's parameters are Q-terms then that separating set is given by a Q-term.

Recall that proj_Q operates by replacing all occurrences of ∞ by OUT.

Definition 5. *$p \sim p'$ if $proj_Q(p) = proj_Q(p')$.*

Definition 6. *$p \Vdash^* \phi$, for ϕ in the language of the Kripke model, i.e. when ϕ's parameters are Q-terms, inductively on ϕ :*

- *$p \Vdash^* \sigma \in \tau$ if for some $\langle q, \rho\rangle \in \tau$ $q \geq_Q proj_Q(p)$ and $p \Vdash^* \sigma = \rho$.*
- *$p \Vdash^* \sigma = \tau$ if for all $p'' \leq_P p' \sim p$ and $\langle q, \rho\rangle \in \sigma$ if $proj_Q(p'') \leq_Q q$ then there is a $p''' \leq_P p''$ such that $p''' \Vdash^* \rho \in \tau$, and symmetrically.*
- *$p \Vdash^* \phi \wedge \theta$ if $p \Vdash^* \phi$ and $p \Vdash^* \theta$.*
- *$p \Vdash^* \phi \vee \theta$ if $p \Vdash^* \phi$ or $p \Vdash^* \theta$.*
- *$p \Vdash^* \phi \to \theta$ if for all $p'' \leq_P p' \sim p$ if $p'' \Vdash^* \phi$ then there is a $p''' \leq_P p''$ such that $p''' \Vdash^* \theta$.*
- *$p \Vdash^* \exists x\ \phi(x)$ if for some Q-term σ $p \Vdash^* \phi(\sigma)$.*
- *$p \Vdash^* \forall x\ \phi(x)$ if for all $p'' \leq_P p' \sim p$ and Q-term σ there is a $p''' \leq_P p''$ such that $p''' \Vdash^* \phi(\sigma)$.*

Lemma 8. *If $p \sim p'$ then $p \Vdash^* \phi$ iff $p' \Vdash^* \phi$.*

Proof. For the cases $\in, =, \to$, and $\forall$, that is built right into the definition of $\Vdash^*$. The other cases are a trivial induction.

Lemma 9. *If $q \leq_P p \Vdash^* \phi$ then $q \Vdash^* \phi$.*

Proof. Inductively on ϕ. For $\in$, use that $proj_Q$ is monotone. The cases $\wedge, \vee$, and $\exists$ are trivial inductions. For the remaining cases, suppose $q'' \leq_P q' \sim q \leq_P p$. Then $q'' \leq_P q' \upharpoonright dom(p) \sim p$, and use that $p \Vdash^* \phi$.

Proposition 10. *$\bot \models \phi$ iff for some $p \in G$ $p \Vdash^* \phi$.*

Proof. Inductively on ϕ. We give only a typical case, equality.

Suppose $G \ni p \Vdash^* \sigma = \tau$. By taking p' equal to p in the definition of $\Vdash^*$, for every member ρ of either σ or τ, it is dense to $*$-force ρ to be in the other set. By the genericity of G some such p''' will be in G, and so inductively ρ will end up in the other set. This shows that σ and τ have the same members at $\bot$. Regarding a future node $f(G)_{p''}$, because $f(G)_{p''}$ is a legal weakening of $f(G)$, $p'' \upharpoonright dom(p) \sim p$, so again it is dense for any member of σ or τ to be forced into the other, so they have the same members at node $f(G)_{p''}$. Hence $\bot \Vdash^* \sigma = \tau$.

Conversely, suppose for all $p \in G$ $p \not\Vdash^* \sigma = \tau$. That means there are $p'' \leq_P p' \sim p$ and ρ forced by p'' into σ (without loss of generality), but p'' has no extension $*$-forcing ρ into τ. For every natural number n the set $D_n = \{q \mid$ for some $k > n$ $dom(q) \subseteq 2^k$ and all binary sequences of length k either are labeled ∞ by q or some initial segment is labeled IN by $q\}$ is dense. Hence cofinally many levels of G are in D_0. Observe that if q is in $D_0 \cap G$ and $q' \sim q$ then any extension of q' can be extended again to induce a legal weakening of G. In N, by overspill choose $p \in f(G)$ to be in $f(D_0)$. Choose $p'' \leq_P p' \sim p$ and ρ as given by the case hypothesis. Extend p'' to p''' so that $f(G)_{p'''}$ is a legal weakening of $f(G)$. Since p''' has no extension $*$-forcing ρ into τ, inductively at node $f(G)_{p'''}$ ρ is not a member of τ. Hence $\perp \not\models \sigma = \tau$.

Theorem 11. $\perp \models IZF$

Proof. Most axioms are easy. For ϵ-Induction, use the elementarity of the transition functions f. For Separation, given ϕ and σ, let $\mathrm{Sep}_{\phi,\sigma}$ be $\{\langle proj_Q(p), \tau\rangle \mid$ for some $\langle q, \tau\rangle \in \sigma$ with $p \leq q$ we have $p \Vdash^* \phi(\sigma)\}$. By lemmas 8 and 10, this works.

4 FAN$_\Delta$ Does Not Imply FAN$_c$

We will need a tree similar to that of the last proof. In fact, we will need two trees: the $c-$bar C, and the decidable set C' from which C is defined. (Both can be viewed as 2^* with labels or as subtrees of 2^*.) Mostly we will focus on C. Because FAN$_c$ refers to eventual membership in a tree, the difference between IN and OUT nodes is no longer relevant: the bar is uniform beneath any OUT node. So we can describe the forcing in terms similar to those before, and with some simplifications introduced. The forcing partial order P will be the set of appropriate labelings of finitely many nodes from 2^*. A labeling of nodes assigns to each one either IN or ∞, with the following restrictions. Any node labeled IN has no descendant, the idea being that once a node gets into the eventual bar so are all of its descendants automatically, so nothing more need be said. For any node labeled ∞, if both children are labeled, then at least one of them must be labeled ∞. Let G be P-generic through the condition labeling the empty sequence with ∞.

As before, we will need to look at weaker trees, ones with bigger bars.

Definition 7. *H is a legal weakening of G if H can be constructed by choosing finitely many nodes labeled ∞ by G, whose siblings are also labeled ∞ by G, and changing those labels to IN and eliminating all descendants.*

As before, each legal weakening H can be summarized by one forcing condition p, which consists of those nodes changed by H and their ancestors, labeled as in G. H is then the set of conditions in G each minimally changed to be consistent with p. Hence we refer to H as G_p.

Lemma 12. *If G_p is a legal weakening of G then G_p is generic through p.*

Proof. As in the corresponding lemma in the previous section.

Definition 8. *Terms are defined inductively (through the ordinals) as sets of the form $\{\langle B_i, \sigma_i\rangle \mid i \in I\}$, where I is any index set, σ_i a term, and B_i a finite set of truth values. A truth value is a symbol of the form b^+ or b' or $\neg b'$, for $b \in 2^*$ a finite binary sequence.*

Definition 9. *Let C be the term $\{\langle\{b^+\}, \hat{b}\rangle \mid b \in 2^*\}$, and C' be $\{\langle\{b'\}, \hat{b}\rangle \mid b \in 2^*\}$.*

In our final model, (the interpretation of) C will be the c-bar induced by (the interpretation of) C', and C will not be uniform, thereby falsifying FAN_c. Furthermore, we will show that FAN_Δ holds in this model.

We can now start to describe the ultimate Kripke model. Recall that G is generic for P over M and labels the empty sequence with ∞. The bottom node $\perp$ of the Kripke model consists of the terms. At $\perp$, b^+ counts as true iff $G(b) = \mathrm{IN}$, b' always counts as true, and $\neg b'$ never counts as true. Later nodes will have different ways of counting the various literals as true. At any node, for $\sigma = \{\langle B_i, \sigma_i\rangle \mid i \in I\}$, if each member of some B_i counts as true, then at that node $\sigma_i \in \sigma$. This induces a notion of extensional equality among the terms. One way of viewing this is at any node to remove from a term σ any pair $\langle B_i, \sigma_i\rangle$ if some member of B_i is not true at that node. Then each remaining $\langle B_i, \sigma_i\rangle$ can be replaced by σ_i. Equality is then as given by the Axiom of Extensionality as interpreted in the model.

As for what the other nodes in the model are, there are two different kinds. As in the last section, let N be an ultrapower of $M[G]$ using any non-principal ultrafilter on ω, with elementary embedding $f : M[G] \to N$. This necessarily produces non-standard integers. In N, any forcing condition p which induces a legal weakening of $f(G)$ will index a successor node to $\perp$. At the node indexed by p, the universe will be the terms of N as interpreted by $f(G)_p$. That is, b^+ is true if $f(G)_p(b) = \mathrm{IN}$, b' is always true, and $\neg b'$ never. Regarding the embeddings from $\perp$, for a term $\sigma \in M$, $f(\sigma)$ is a term in N, so send σ to $f(\sigma)$. In addition, definably over $M[G]$, any non-standard $c \in 2^*$ with $f(G)(c) = \infty$ also indexes a node. At such a node c, b' counts as true iff $b \neq c$, $\neg b'$ counts as true iff $b = c$, and b^+ counts as true iff $b \not\subseteq c$ (b is not an initial segment of c). Note that at $\perp$ any b' refers only to a standard b; for some b' to be declared false at a later node c, b would have to equal c, and c indexes a node only if c is non-standard. Hence there is no conflict with the Kripke structure: once b' is deemed true, it remains true. Similarly with b^+: G_p is a fattening of G. Hence membership, being based on finitely many truth values, is monotone.

Any node indexed by such a $c \in 2^*$ is terminal in the Kripke ordering. Also, among nodes of the other kind, there is one trivial condition p, the one with $p(\langle\rangle) = \mathrm{IN}$. This is also a terminal node, where each b^+ and each b' is true. At any other node, iterate. That is, suppose p is not the preceding condition. The model at p can be built in N. As an ultrapower of $M[G]$, N internally looks like $f(M)[f(G)]$. The structure at node p could be built in $f(M)[f(G)_p]$, where $f(G)_p$ is generic through $f(P)$ (and non-trivial). Hence the construction just described, using an ultrapower and legal weakenings and non-standard binary strings to get additional nodes, can be performed in $f(M)[f(G)_p]$ just as above. This provides immediate successors to nodes indexed by (non-trivial) p's. Iterate ω-many times.

The picture is that at $\perp$ C looks like G, that is, those nodes G assigns to be IN. This tree gets fatter at later nodes that are legal weakenings. At terminal nodes c, C is everything but the branch up to c. At most nodes C' looks like everything; at node c, where c is non-standard relative to its predecessor, we find the one thing not in C', namely c.

It is easy to see that C is the $c-$set induced by C': once b is forced into C, none of its descendants index terminal nodes, so no descendant is forced out of C'; similarly, if b is not forced into C, say at node p, then $G_p(b) = \infty$, and in N some non-standard extension c of b will also be labeled ∞ by $f(G)_p$, and that c will index a node at which c is not in C'. Clearly, C is not uniform, and C' is decidable.

What remains to show is that C is bar, that FAN$_\Delta$ holds, and that IZF holds.

Lemma 13. $\perp \models C$ *is a bar.*

Proof. Suppose σ is forced to be an infinite binary path at some node. If that node is a terminal node, C contains cofinitely many members of 2^*, and so certainly intersects σ. Else without loss of generality we can assume the node is $\perp$. Then for some $p \in G$ $p \Vdash$ "$\perp \models \sigma$ is an infinite binary path." If it is not dense beneath p to force the standard part of σ (that is, σ applied to the standard integers) to be in the ground model, then extensions q and r of p force incompatible facts about σ. The only incompatible facts about σ are of the form $b^\frown 0 \in \sigma$ and $b^\frown 1 \in \sigma$. The positive parts of q and r (that is, $q^{-1}(IN)$ and $r^{-1}(IN)$) induce a legal weakening of G. That is, there is a canonical condition inpart(q, r), with domain $dom(q) \cup dom(r)$, that returns IN on any node that either q or r returns IN on, as well as on any node if inpart(q, r) returns IN on both children, else OUT. Because terms use only positive (i.e. IN) information, at the node $f(G)_{inpart(q,r)}$, both $b^\frown 0$ and $b^\frown 1$ are in σ. (More coarsely and perhaps more simply, at the node induced by the trivial condition sending the empty sequence to IN, the same conclusion holds for the same reason.) Hence $\perp$ could not have forced σ to be a path in the first place. Therefore p forces σ on the standard binary tree to be in the ground model. It is easy to see that generically G labels some node in σ IN.

Lemma 14. $\perp \models$ *FAN*$_\Delta$

Proof. If a set of nodes B is forced by p to be decidable, then no extensions of p can force incompatible facts about B. Hence B is in the ground model. If B were not a bar in the ground model, there would be a ground model path missing B. This path would also be in the Kripke model. Hence B is a bar in the ground model, which is taken to be classical, so B is uniform.

Regarding getting IZF to be true, as above, the problem is that truth in the Kripke model is on the surface determined by forcing conditions in the ground model, to which the Kripke model has no access. The essence is to capture truth at a node using those truth values that are allowed in the building of terms.

Definition 10. *For a forcing condition p, $B_p = \{b^+ \mid$ for some initial segment c of b $p(c) = IN\}$. For a set of truth values B, $B^+ = B \cap \{b^+ \mid b \in 2^*\}$. Also, B is positive if B contains no truth value of the form $\neg b'$.*

Definition 11. *1.* $\neg b' \Vdash^* B$ *iff* $c^+ \in B \to c^+ \not\subseteq b'$, $c' \in B \to c \neq b$, *and* $\neg c' \in B \to c = b$.

2. $\sigma^{\neg b'} = \{\sigma_i^{\neg b'} \mid$ *for some* $\langle B_i, \sigma_i \rangle \in \sigma$ $\neg b' \Vdash^* B_i\}$.

3. For $\phi(\sigma_1, \ldots \sigma_n)$ *in the language of the Kripke model,* $\phi^{\neg b'} = \phi(\sigma_1^{\neg b'}, \ldots \sigma_n^{\neg b'})$.

4. $\neg b' \Vdash^* \phi$, *for* ϕ *in the language of the Kripke model, if* $\phi^{\neg b'}$ *is true (i.e. in* V.*) Note that* $\phi^{\neg b'}$ *is a formula with set parameters.*

Definition 12. $q \leq_W p$ *(q is a weakening of p as conditions) if for* $b \in dom(p)$ *either* $p(b) = \infty$ *or for some initial segment c of b* $q(c) = IN$.

The idea behind this definition is the q may change some ∞'s to IN's, as well as add other stuff (extend the domain). Notice that $\leq_W$ is a partial order, and inpart(p, q), from lemma 14, is the glb of p and q.

Definition 13. $p \Vdash^* \phi$, *for* ϕ *in the language of the Kripke model, i.e. when* ϕ*'s parameters are terms, inductively on* ϕ :

- $p \Vdash^* \sigma \in \tau$ *if for some* $\langle B_i, \tau_i \rangle \in \tau$ *with* B_i *positive,* $B_i^+ \subseteq B_p$ *and* $p \Vdash^* \sigma = \tau_i$.
- $p \Vdash^* \sigma = \tau$ *if*
  i) for all $\langle B_i, \sigma_i \rangle \in \sigma$ *(resp.* τ*) and* $q \leq_W p$ *if* B_i *is positive and* $B_i^+ \subseteq B_q$ *then there is an* $r \leq q$ *such that* $r \Vdash^* \sigma_i \in \tau$ *(resp.* σ*), and*
  ii) for all $b \notin dom(p)$, *if for no initial segment c of b is* c^+ *in* B_p, *then* $\neg b' \Vdash^* \sigma = \tau$.
- $p \Vdash^* \phi \wedge \theta$ *if* $p \Vdash^* \phi$ *and* $p \Vdash^* \theta$.
- $p \Vdash^* \phi \vee \theta$ *if* $p \Vdash^* \phi$ *or* $p \Vdash^* \theta$.
- $p \Vdash^* \phi \to \theta$ *if*
  i) for all $q \leq_W p$ *if* $q \Vdash^* \phi$ *then there is an* $r \leq q$ *such that* $r \Vdash^* \theta$, *and*
  ii) for all $b \notin dom(p)$, *if for no initial segment c of b is* c^+ *in* B_p, *then* $\neg b' \Vdash^* \phi \to \theta$.
- $p \Vdash^* \exists x\, \phi(x)$ *if for some term* σ $p \Vdash^* \phi(\sigma)$.
- $p \Vdash^* \forall x\, \phi(x)$ *if*
  i) for all terms σ *and* $q \leq_W p$ *there is an* $r \leq q$ *such that* $r \Vdash^* \phi(\sigma)$, *and*
  ii) for all $b \notin dom(p)$, *if for no initial segment c of b is* c^+ *in* B_p, *then* $\neg b' \Vdash^* \forall x\, \phi(x)$.

Lemma 15. *If* $q \leq_W p \Vdash^* \phi$ *then* $q \Vdash^* \phi$, *and* $\perp \models \phi$ *iff for some* $p \in G$ $p \Vdash^* \phi$.

Lemma 16. $\perp \models IZF$

Proof. As above, most of the axioms have soft proofs. For Separation, given ϕ and σ, let Sep$_{\phi,\sigma}$ be $\{\langle B, \tau \rangle \mid$ for some $\langle B', \tau \rangle \in \sigma$ with $B \supseteq B'$ either $B = B_p \Vdash \phi(\sigma)$ or $(\neg b' \in B$ and $\neg b' \Vdash^* \phi)\}$. By lemma 15, this works.

5 FAN$_c$ Does Not Imply FAN$_{\Pi_1^0}$

Let G be P-generic as above. By convention, we say that if $G(\alpha) = $ IN then G applied to any extension of α is also IN. Our goal is to hide G a bit better than before, so FAN$_c$ remains true, but not too well, so that FAN$_{\Pi_1^0}$ is false.

Let N be an ultrapower of $M[G]$ using a non-principal ultrafilter on ω. The Kripke model has a bottom node $\bot$, and the successors of $\bot$ are indexed by the labels $\langle n, \alpha \rangle$, where n is a non-standard integer, and $\alpha \in 2^*$ either has non-standard length or $G(\alpha) = \infty$.

Definition 14. *A truth value is a symbol of the form $\langle n, \alpha \rangle$, $\neg\langle n, \alpha \rangle$, or $\langle \forall n, \alpha \rangle$, for n a natural number (in the first two cases) and $\alpha \in 2^*$. Admittedly truth values of the first kind are also used to index nodes; whether truth values or nodes are intended in any particular case should be clear from the context. Terms are defined inductively (through the ordinals) as sets of the form $\{\langle B_i, \sigma_i \rangle \mid i \in I\}$, where I is any index set, σ_i a term, and B_i a finite set of truth values.*

The sets at $\bot$ will be the terms in M. The sets at any other node will be analogous, that is, the terms in what N thinks is the ground model, i.e. $\bigcup_{\kappa \in ORD} f(M_\kappa)$. At $\bot$, $\langle n, \alpha \rangle$ will always be true, $\neg\langle n, \alpha \rangle$ always false, and $\langle \forall n, \alpha \rangle$ true exactly when $G(\alpha) = \mathrm{IN}$. At node $\langle m, \beta \rangle$, $\langle n, \alpha \rangle$ is true exactly when $\langle n, \alpha \rangle \neq \langle m, \beta \rangle$, $\neg\langle n, \alpha \rangle$ is true exactly when $\langle n, \alpha \rangle = \langle m, \beta \rangle$, and $\langle \forall n, \alpha \rangle$ true exactly when $\alpha \neq \beta$. This interpretation of the truth values induces an interpretation of the terms at all nodes.

Let T_n be the term $\{\langle \{\langle n, \alpha \rangle\}, \hat{\alpha} \rangle \mid \alpha \in 2^*\}$. Let C be a term naming the function that on input n returns T_n. T_n at $\bot$ and at $\langle m, \alpha \rangle$, $m \neq n$, looks like the full tree 2^*, and T_n at $\langle n, \alpha \rangle$ looks like everything except α. The term for $\bigcap_n C(n)$ is given by $\{\langle \{\langle \forall n, \alpha \rangle\}, \hat{\alpha} \rangle \mid \alpha \in 2^*\}$, and is interpreted as $\{\alpha \mid G(\alpha) = \mathrm{IN}\ \}$ at $\bot$ and $2^* \backslash \{\alpha\}$ at $\langle n, \alpha \rangle$.

It is clear that T_n is decidable, and so $\bigcap_n C(n)$ is on the face of it Π^0_1. So to show that $\bigcap_n C(n)$ is a counter-example to $\mathrm{FAN}_{\Pi^0_1}$, the next lemma suffices.

Lemma 17. $\bot \Vdash$ "*$\bigcap_n C(n)$ is a bar.*"

Proof. Let $\bot \models$ "Br is a branch through 2^*." Work beneath a condition forcing that, so we can assume Br consists of sets of the form $\langle B_i, \hat{\alpha} \rangle$, for various $\alpha \in 2^*$. If the standard part of Br, the part visible at $\bot$, is in the ground model M, then by the genericity of G Br will hit G (i.e. for some $\alpha \in Br$ $G(\alpha) = \mathrm{IN}$), which is how $\bot$ interprets $\bigcap_n C(n)$. If the standard part of Br were not in M, then contradictory facts about Br would be forced by different forcing conditions. In particular, we would have p, q, and α with $p \Vdash$ "$\bot \models \alpha^\frown 0 \in Br$" and $q \Vdash$ "$\bot \models \alpha^\frown 1 \in Br$." That means there are $\langle B_p, \widehat{\alpha^\frown 0} \rangle \in Br$ and $\langle B_q, \widehat{\alpha^\frown 1} \rangle \in Br$, with B_p and B_q consisting only of truth values automatically true at $\bot$ save for some of the form $\langle \forall n, \alpha \rangle$. But at some node $\langle n, \alpha \rangle$ with α non-standard, all of those latter truth values will be true. Hence $\langle n, \alpha \rangle \models$ "$\widehat{\alpha^\frown 0}, \widehat{\alpha^\frown 1} \in Br$," so $\bot$ could not force Br to be a path.

In order to finish, we will need to deal with truth at $\bot$.

Definition 15. *For a forcing condition p, let $\mid p \mid$, the length of p, be the length of the longest $\alpha \in dom(p)$. Let B_p be $\{\langle n, \alpha \rangle \mid n,\ length(\alpha) \leq \mid p \mid\} \cup \{\langle \forall n, \alpha \rangle \mid length(\alpha) \leq \mid p \mid$ and for some initial segment β of α $p(\beta) = IN\}$.*

Definition 16. *1.* $\neg\langle n,\alpha\rangle \Vdash^* B$ *iff* $\langle n,\alpha\rangle \notin B$, $\langle\forall n,\alpha\rangle \notin B$, *and the only truth value of the form* $\neg\langle m,\beta\rangle$ *in* B *is* $\neg\langle n,\alpha\rangle$ *itself.*

2. $\sigma^{\neg\langle n,\alpha\rangle} = \{\sigma_i^{\neg\langle n,\alpha\rangle} \mid$ *for some* $\langle B_i,\sigma_i\rangle \in \sigma\ \neg\langle n,\alpha\rangle \Vdash^* B_i\}$.

3. For $\phi(\sigma_1,...\sigma_n)$ *in the language of the Kripke model, that is, with parameters (displayed) terms,* $\phi^{\neg\langle n,\alpha\rangle} = \phi(\sigma_1^{\neg\langle n,\alpha\rangle},...\sigma_n^{\neg\langle n,\alpha\rangle})$.

4. $\neg\langle n,\alpha\rangle \Vdash^* \phi$, *for* ϕ *in the language of the Kripke model, if* $\phi^{\neg\langle n,\alpha\rangle}$ *is true (i.e. in* V*.) Note that* $\phi^{\neg\langle n,\alpha\rangle}$ *is a formula with set parameters.*

Definition 17. $p \Vdash^* \phi$, *for* B *a finite set of truth values and* ϕ *in the language of the Kripke model, i.e. when* ϕ*'s parameters are terms, inductively on* ϕ *:*

- $p \Vdash^* \sigma \in \tau$ *if for some* $\langle B_i,\tau_i\rangle \in \tau$, $B_i \subseteq B_p$ *and* $p \Vdash^* \sigma = \tau_i$.
- $p \Vdash^* \sigma = \tau$ *if*
  i) for all $\langle B_i,\sigma_i\rangle \in \sigma$ *and* $q \leq p$ *if* $B_i \subseteq B_q$ *then there is an* $r \leq q$ *such that* $r \Vdash^* \sigma_i \in \tau$, *and symmetrically between* σ *and* τ, *and*
  ii) if $n >\mid p \mid$, *and if either length(*α*)* $>\mid p \mid$ *or for no initial segment* β *of* α *do we have* $p(\beta) = IN$*, then* $\neg\langle n,\alpha\rangle \Vdash^* \sigma = \tau$.
- $p \Vdash^* \phi \wedge \theta$ *if* $p \Vdash^* \phi$ *and* $p \Vdash^* \theta$.
- $p \Vdash^* \phi \vee \theta$ *if* $p \Vdash^* \phi$ *or* $p \Vdash^* \theta$.
- $p \Vdash^* \phi \to \theta$ *if*
  i) for all $q \leq p$ *if* $q \Vdash^* \phi$ *then there is an* $r \leq q$ *such that* $r \Vdash^* \theta$, *and*
  ii) if $n >\mid p \mid$, *and if either length(*α*)* $>\mid p \mid$ *or for no initial segment* β *of* α *do we have* $p(\beta) = IN$*, then* $\neg\langle n,\alpha\rangle \Vdash^* \phi \to \theta$.
- $p \Vdash^* \exists x\ \phi(x)$ *if for some term* σ $p \Vdash^* \phi(\sigma)$.
- $p \Vdash^* \forall x\ \phi(x)$ *if*
  i) for all terms σ *and* $q \leq p$ *there is an* $r \leq q$ *such that* $r \Vdash^* \phi(\sigma)$, *and*
  ii) if $n >\mid p \mid$, *and if either length(*α*)* $>\mid p \mid$ *or for no initial segment* β *of* α *do we have* $p(\beta) = IN$*, then* $\neg\langle n,\alpha\rangle \Vdash^* \forall x\ \phi(x)$.

Lemma 18. *If* $q \leq p \Vdash^* \phi$ *then* $q \Vdash^* \phi$, *and* $\perp \models \phi$ *iff for some* $p \in G$ $p \Vdash^* \phi$.

Lemma 19. $\perp \models IZF$

Proof. As before, all of the axioms have soft proofs, save for Separation. Given ϕ and σ, let $\text{Sep}_{\phi,\sigma}$ be $\{\langle B_i \cup B_p,\tau\rangle \mid \langle B_i,\tau\rangle \in \sigma$ and $p \Vdash^* \phi(\tau)\} \cup \{\langle B,\tau\rangle \mid$ for some $\neg\langle n,\alpha\rangle \in B$ and some B_i, $\langle B_i,\tau\rangle \in \sigma$, $\neg\langle n,\alpha\rangle \Vdash^* B_i$, and $\neg\langle n,\alpha\rangle \Vdash^* \phi(\tau)\}$. By the previous lemma, this works.

Lemma 20. $\perp \models FAN_c$

Proof. Suppose that at $\perp$ we have a decidable set $C \subseteq 2^*$ inducing a c-bar. We would like to show that at $\perp$ the c-bar is uniform, which means that for some k C contains every sequence of length at least k (in notation, $C \supseteq 2^{\geq k}$).

Say that $\alpha \in 2^*$ is *good* if there is a natural number k such that, whenever $n \geq k$ and $\beta \supseteq \alpha$ has length at least k, $\neg\langle n,\beta\rangle \Vdash$ "$C \supseteq 2^{\geq k}$." Observe that if $\alpha\frown 0$ and $\alpha\frown 1$ are good then so is α (by taking k sufficiently large). So if the empty sequence $\langle\rangle$ is bad (i.e. not good) then there is a branch Br_0 of bad nodes. For each $\alpha \in Br_0$, by the definition of badness, taking k to be the length $\mid \alpha \mid$ of

α, we have some $\beta \supseteq \alpha$ and $n \geq | \alpha |$ and $\gamma \in 2^{\geq k}$ such that $\neg\langle n, \beta \rangle \Vdash$ "$\gamma \notin C$." By choosing α's of increasing length, we can get infinitely many γ's of increasing length, in particular infinitely many distinct γ's. Hence there is a branch Br_1 such that each node in Br_1 has infinitely many γ's as extensions.

That was all in M. Now in N, pick some non-standard $\alpha \in Br_0$ such that with regard to the induced fact that $\neg\langle n, \beta \rangle \Vdash$ "$\gamma \notin C$," γ extends some non-standard node in Br_1. Since $\beta \supseteq \alpha$ and $n \geq | \alpha |$, and α is non-standard, $\langle n, \beta \rangle$ indexes a node in the model. But at $\perp$, C induces a c-bar, so $\perp \models$ "there is a node δ in Br_1 such that every extension of δ is in C." This contradicts the choice of γ.

We conclude from this that $\langle \rangle$ is good. Fix k witnessing this goodness. It is worth noting that, by the decidability of C, if α has standard length at least k then $\perp \models$ "$\alpha \in C$." So all that needs to be checked to show $\perp \models$ "$C \supseteq 2^{\geq k}$" is the finitely many α's not covered by $\langle \rangle$'s goodness. Namely, for those α's of length less than k such that $G(\alpha) = \infty$, we need to show that for all non-standard n $\langle n, \alpha \rangle \models$ "$C \supseteq 2^{\geq k}$". For that, it suffices to show in M that there is a finite n such that for all $m \geq n$ $\neg\langle m, \alpha \rangle \Vdash^*$ "$C \supseteq 2^{\geq k}$".

Toward that end, suppose not. Then for infinitely many m there is a γ of length at least k such that $\neg\langle m, \alpha \rangle \Vdash^*$ "$\gamma \notin C$". If those γ's are of bounded length then one occurs infinitely often. For that fixed γ, by overspill there is a non-standard m such that $\neg\langle m, \alpha \rangle \Vdash^*$ "$\gamma \notin C$". But $\langle m, \alpha \rangle$ is a Kripke node, and that contradicts the decidability of C. Hence there are infinitely many different γ's. That means there is a branch Br_2 such that every node on Br_2 has infinitely many different γ's as extensions. Pick a non-standard m such that the induced γ extends a non-standard node of Br_2. But again, at $\perp$ C induces a c-bar, so $\perp \models$ "there is a node δ in Br_2 such that every extension of δ is in C." This contradicts the choice of γ.

6 $\mathbf{FAN}_{\Pi^0_1}$ Does Not Imply $\mathbf{FAN}_{\mathbf{full}}$

Let G be generic as above. In $M[G]$, the Kripke model will have bottom node $\perp$, and successor nodes labeled by those $\alpha \in 2^*$ with $G(\alpha) = \infty$. To define the full Kripke model with nodes any partial order $\langle P, < \rangle$, at node $p \in P$ a term σ is any function of domain $P^{\geq p}$ with $\sigma(q)$ a set of terms at node q; furthermore, with transition function f_{qr} for $q < r$, if $\tau \in \sigma(q)$ then $f_{qr}(\tau) \in \sigma(r)$; finally, f_{pq} is extended to σ by restriction: $f_{pq}(\sigma) = \sigma \restriction P^{\geq q}$. For the current construction, we will take a sub-model of the full model by imposing one additional restriction: a term at any node α other than $\perp$ must be in the ground model M.

Let C be the term such that $\perp \models$ "$\hat{\beta} \in C$" iff for some initial segment $\beta \restriction n$ of β $G(\beta \restriction n) = \mathrm{IN}$, and at node $\alpha \neq \perp$ $\alpha \models$ "$\hat{\beta} \in C$" iff β is not an initial segment of α.

Lemma 21. $\perp \models FAN_{\Pi^0_1}$

Proof. If $\perp \models$ "$B \subseteq 2^*$ is decidable" then for any $\beta \in 2^*$ $\perp \models$ "$\hat{\beta} \in B$" iff for some node $\alpha \neq \perp$ $\alpha \models$ "$\hat{\beta} \in B$" iff the same holds for all $\alpha \neq \perp$. Hence

$\perp \models$ "$B = \hat{B_M}$" for some set $B_M \in M$. So if $\perp \models$ "B_n is a sequence of decidable trees," then that sequence is the image of a sequence of sets from M. Hence their intersection internally is the image of a set from M. So if $\bigcap_n B_n$ is internally a bar, it is the image of a bar, and by the Fan Theorem in M is uniform.

Lemma 22. $\perp \not\models FAN_{\text{full}}$

Proof. At $\perp$, C is not uniform, so it suffices to show $\perp \models$ "C is a bar". If $\perp \models$ "P is a path through 2^*" then $\perp \models$ "P is decidable", and as above P is then the image of a ground model path. Generically, for some β along that path, $G(\beta) = \mathrm{IN}$. For that β, $\perp \models$ "P goes through $\hat{\beta}$ and $\hat{\beta} \in C$."

Lemma 23. $\perp \models IZF$

References

[1] Beeson, M.: Foundations of Constructive Mathematics. Springer (1985)

[2] Berger, J.: The Logical Strength of the Uniform Continuity Theorem. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) CiE 2006. LNCS, vol. 3988, pp. 35–39. Springer, Heidelberg (2006)

[3] Berger, J.: A separation result for varieties of Brouwer's fan theorem. In: Proceedings of the 10th Asian Logic Conference (ALC 10), Kobe University in Kobe, Hyogo, Japan, September 1-6 (2008) (to appear)

[4] Diener, H.: Compactness under constructive scrutiny. Ph.D. Thesis (2008)

[5] Diener, H., Loeb, I.: Sequences of real functions on [0, 1] in constructive reverse mathematics. Annals of Pure and Applied Logic 157(1), 50–61 (2009)

[6] Fourman, M., Hyland, J.: Sheaf models for analysis. In: Fourman, M., Mulvey, C., Scott, D. (eds.) Applications of Sheaves. Lecture Notes in Mathematics, vol. 753, pp. 280–301. Springer, Heidelberg (1979)

[7] Julian, W., Richman, F.: A uniformly continuous function on [0,1] that is everywhere different from its infimum. Pacific Journal of Mathematics 111(2), 333–340 (1984)

The Logic of Uncertain Justifications

Bob Milnikel

Department of Mathematics
Kenyon College, Gambier, OH 43022
milnikelr@kenyon.edu

Abstract. In Artemov's Justification Logic, one can make statements interpreted as "t is evidence for the truth of formula F." We propose a variant of this logic in which one can say "I have degree r of confidence that t is evidence for the truth of formula F." After defining both an axiomatic approach and a semantics for this Logic of Uncertain Justifications, we will prove the usual soundness and completeness theorems.

Keywords: Justification Logic, Epistemic Logic, Proof Theory.

1 Introduction

Although Artemov's semantics ([1], [2]) for the Logic of Proofs, his first foray into Justification Logic, make clear its origins in Proof Theory and answering a question left open by Gödel's ([11]), the broad applicability of Justification Logic to nearly in which normal modal logics have application soon became apparent ([3], [5], [6], [7], [8], [20], [21], [23], [22]). In an epistemic context, Justification Logic takes us from simply $\Box F$, "F is known/believed," to $t : F$, "Reason t provides justification for knowledge/belief that F."

However, not all justifications for belief are equal. One might read about the president's policy speech in the *New York Times* and hear about it from an irate caller on a radio talk show. Each might provide some degree of justification for believing that the president was about to undertake a certain change in policy, but one should put much more credence in the *Times* than in a random caller. We can reflect this by noting the degree p to which a piece of evidence t can serve as justification for the belief that F, which we will notate $t :_p F$.

Note that this does not reflect my degree of belief in F. I might have heard the same thing from both the irate caller on the radio and from the *Times* and the fact that one wouldn't consider the caller a reliable source on this matter doesn't cause one to believe it any less. This distinguishes both our intention and approach from that of logics dealing with the probability that certain propositions are true. (See [12] and the many books and papers referenced in its bibliography.)

We will begin with a very short presentation of the basics of Justification Logic, then present the syntax and a Kripke-style semantics for the Logics of Uncertain Justifications, followed by a short discussion of some alternatives to the definitions proposed in the main body of the paper.

2 Justification Logic: A Brief Synopsis

Both to orient the reader and to draw a distinction between what is established and what is new to the present paper, we will begin with a brief overview of the syntax and semantics of (Basic) Justification Logic **J**, the explicit counterpart of the epistemic logic **K** of belief in the absence of introspection. The presentation of definitions and results of this section will be drawn from [4], although the syntax goes back to [2] and the semantics and associated proofs to [9].

The language of **J** is built out of:

- propositional variables P, Q,... (possibly with subscripts)
- justification variables x, y,... (possibly with subscripts)
- justification constants c_0, c_1, c_2,...
- the propositional constant $\perp$
- the logical connective $\rightarrow$
- the binary functions $\cdot$ and $+$, operating on justification terms, and
- an operator of the type $\langle term \rangle : \langle formula \rangle$, producing formulas.

Justification terms are built up from justification variables and constants by means of the functions $\cdot$ and $+$; *formulas* are built up from propositional variables and $\perp$ using $\rightarrow$ (with other connectives defined in the standard way), plus the rule that if t is a justification term and F is a formula, then $t : F$ is a formula.

Intuitively, the formula $t : F$ is to be read as "t represents a justification for believing F." The function $\cdot$ represents the believer's internal application of the justification of belief in the premise of an implication to the justification of belief in the implication itself to justify belief in the consequent of that implication. That is, if s justifies belief in $F \rightarrow G$ and t justifies belief in F, then $s \cdot t$ justifies belief in G. The function $+$ combines justifications in the sense that $s + t$ justifies belief in all the things that s justifies belief in as well as in all the things that t justifies belief in.

We will treat constants a little later, but the intuition is that constants justify belief in axiomatically true statements.

Definition 1. *The* Basic Logic of Justifications $\mathbf{J}_0$ *consists of the following axiom schemes:*

A0. Classical propositional axioms
A1. $s : (F \rightarrow G) \rightarrow (t : F \rightarrow (s \cdot t) : G)$ *(The* application *axiom scheme)*
A2. $s : F \rightarrow (s + t) : F$ *and* $t : F \rightarrow (s + t) : F$ *(The* monotonicity *axiom schemes)*

along with the rule of inference modus ponens.

Although there can be much subtlety in the treatment of constants ([4], [15], [18] and many others), they will not be central to our discussion and we will simply extend $\mathbf{J}_0$ to **J** by adding the following rule:

Definition 2. *The* Axiom Internalization Rule *states that from any axiom A which is an instance of axiom scheme A0, A1, or A2, for any $n \geq 1$ and for any constants $c_1, c_2, \ldots, c_n$ we may infer $c_n : c_{n-1} : \cdots : c_2 : c_1 : A$.*

We will state here only one important syntactic theorem:

Theorem 1. *Justification logic **J** enjoys* internalization. *That is,*

If $\vdash F$ then $\vdash t : F$ for some justification term t.

We now move on to a definition of the standard semantics, due largely to Fitting ([9]), building on concepts from Kripke ([13]) and Mkrytychev ([19]). We follow [4] very closely here.

Definition 3. *A* Kripke-Fitting **J**-Model $\mathcal{M} = (W, R, \mathcal{E}, \Vdash)$ *is a Kripke model* $(W, R, \Vdash)$ *together with an* admissible evidence function $\mathcal{E}$ *such that $\mathcal{E}(t, F) \subseteq W$ for any justification term t and any **J**-formula F. Informally, $\mathcal{E}(t, F)$ specifies the set of possible worlds where t is considered to constitute evidence for the belief in F.*

Any admissible evidence function $\mathcal{E}$ must satisfy the following closure conditions:

– Application: $\mathcal{E}(s, F \to G) \cap \mathcal{E}(t, F) \subseteq \mathcal{E}(s \cdot t, G)$
– Monotonicity: $\mathcal{E}(s, F) \cup \mathcal{E}(t, F) \subseteq \mathcal{E}(s + t, F)$.

Given our particular treatment of constants, we require further of $\mathcal{E}$ that for any instance A of an axiom scheme A0, A1, or A2, any $n \geq 1$, and any justification constants $c_1, c_2, \ldots, c_n$, $\mathcal{E}(c_n, c_{n-1} : c_{n-2} : \cdots : c_2 : c_1 : A) = W$.

Given a model $\mathcal{M} = (W, R, \mathcal{E}, \Vdash)$, we extend the forcing relation $\Vdash$ from sentence variables to formulas as follows, for each $u \in W$:

1. $u \nVdash \bot$
2. $u \Vdash F \to G$ *if and only if either* $u \nVdash F$ *or* $u \Vdash G$
3. $u \Vdash t : F$ *if and only if both* $u \in \mathcal{E}(t, F)$ *and* $v \Vdash F$ *for every* v *with* uRv.

Fitting ([9]) proved:

Theorem 2. ***J** is sound and complete for the class of all Kripke-Fitting **J**-models.*

The canonical model constructed in the standard proof of the completeness theorem satisfies two desirable properties:

– The *Strong Evidence* property: If $u \in \mathcal{E}(t, F)$ then $u \Vdash t : F$.
– The *Fully Explanatory* property: If $v \Vdash F$ for every v with uRv, then $u \Vdash t : F$ for some justification term t.

Mkrtychev semantics ([19]) are a predecessor of Kripke-Fitting semantics. *Mkrtychev **J**-models* are Kripke-Fitting **J**-models consisting of a single world and an empty accessibility relation. It turns out that **J** is also sound and complete with respect to Mkrtychev **J**-models, but since the relation R on the singleton world is empty, the Strong Evidence property is free but the Fully Explanatory property is impossible unless belief in every formula is justified.

3 Uncertain Justifications

What I propose as a potentially useful variant of **J** is the logic of uncertain justifications, wherein we are able to say that "I have at least degree r of confidence in the reliability of t as evidence for belief in F." (The reason for the "at least" will be clear, or at least clearer, shortly.)

3.1 Syntax

The only change necessary in the language will be the replacement of the : operator with a family of operators $:_r$ where r is a rational number with $0 < r \leq 1$.

We modify the deductive system as follows:

A0. Classical propositional axioms
A1. $s :_p (F \to G) \to (t :_q F \to (s \cdot t) :_{p \cdot q} G)$ (The *application* axiom scheme)
A2. $s :_r F \to (s + t) :_r F$ and $s :_r F \to (t + s) :_r F$ (The *monotonicity* axiom schemes)
A3. $s :_q F \to s :_p F$ where $p \leq q$ (The *confidence weakening* axiom)

along with the rule of inference *modus ponens* and the *Axiom Internalization Rule*: from any axiom A which is an instance of axiom scheme A0, A1, A2, or A3 for any $n \geq 1$ and for any justification constants $c_1, c_2, \ldots, c_n$ we may infer $c_n :_1 c_{n-1} :_1 \cdots : c_2 :_1 c_1 :_1 A$.

There are a few things worth noting here:

– The most substantive choice we have made in modifying our rules is to consider degrees of confidence to compound multiplicatively in axiom scheme A1. This has the advantage of working well with our intuitions about probability, although it is worth bearing in mind that we are *not* talking about the probabilities that various formulas are true, just the degrees to which certain pieces of evidence can be trusted. Nevertheless, if I have half-reliable evidence for $A \to B$ and independent half-reliable evidence for A, it makes intuitive sense that the combination would be one-quarter-reliable evidence for B. Independence is a difficult issue and I will return to it in the conclusion, though I do not promise much by way of resolution.
– The confidence weakening axiom is clearly bound up with the "at least" in the "I have at least degree r of confidence" interpretation of $:_r$. My motivation for this lies in the following example: If I have either 90% reliable evidence for A or 80% reliable evidence for B, then I'd certainly like to say that I have evidence for A or B, but without confidence weakening, we cannot deduce $(s_1 :_{0.9} A \vee s_2 :_{0.8} B) \to t :_r (A \vee B)$ for any justification term t and rational number r. In the presence of confidence weakening, we can deduce $(s_1 :_{0.9} A \vee s_2 :_{0.8} B) \to ((c_1 \cdot s_1) + (c_2 \cdot s_2)) :_{0.8} (A \vee B)$ (where $c_1 :_1 A \to A \vee B$ and $c_2 :_1 B \to A \vee B$).
– It is conceivable that under some circumstances we would want a broader Axiom Internalization Rule, allowing us to infer $c_n :_1 c_{n-1} :_{r_{n-1}} \cdots : c_2 :_{r_2} c_1 :_{r_1} A$ for $r_1, \ldots, r_{n-1} \leq 1$, but I do not see the need for such a rule at this point.

We will prove only one theorem before moving on to the semantics.

Theorem 3. *The Logic of Uncertain Justifications enjoys* internalization. *That is,*

$$\text{If} \vdash F \text{ then} \vdash t :_1 F \text{ for some justification term } t.$$

Proof. The proof is entirely standard and dates back to some of the earliest versions of justification logic (e.g. [1]). Proceed by induction on the length of the derivation. If F is an axiom itself or was deduced by the Axiom Internalization Rule, then by the Axiom Internalization Rule, $c :_1 F$ Otherwise, F was the result of an application of *modus ponens* to formulas $G \rightarrow F$ and G. In this case, we can assume by our inductive hypothesis that there are justification terms s_1 and s_2 such that $s_1 :_1 (G \rightarrow F)$ and $s_2 :_1 G$, which yield $(s_1 \cdot s_2) :_1 F$.

Note that all internalized deductions enjoy 100% confidence.

3.2 Semantics

While Mkrtychev-Fitting evidence functions $\mathcal{E}$ have domain $\{Terms\} \times \{Formulas\}$ and codomain 2^W, we could very easily reformulate them to have domain $W \times \{Terms\} \times \{Formulas\}$ and codomain $\{0, 1\}$, with $\mathcal{E}(u, t, F) = 1$ in this reformulation if and only if $u \in \mathcal{E}(t, f)$ in the more traditional formulation.

This slightly altered perspective on evidence functions, seeing them as yes/no on triples of (world, term, formula), makes the extension to uncertain justifications almost immediate. We can simply alter the codomain from the two-element set $\{0, 1\}$ to downward-closed nonempty subsets of the rational interval $[0, 1]$.

I will explain a little more about the choice of downward-closed nonempty subsets (i.e. intervals of the form $[0, r)$ or $[0, r]$ for some $r \in [0, 1]$) in Section 4.

Definition 4. *A* Kripke-Fitting Model *with uncertain justifications* $\mathcal{M} = (W, R, \mathcal{E}, \Vdash)$ *is a Kripke model* $(W, R, \Vdash)$ *together with an* admissible evidence function $\mathcal{E}$ *such that for each* $u \in W$*, any justification term* t *and any formula* F*,* $\mathcal{E}(u, t, F) = [0, r)$ *or* $[0, r]$ *for some rational number* $r \in [0, 1]$*. Informally,* $\mathcal{E}(u, t, F)$ *specifies the degree of confidence in* t *as evidence for the belief in* F *at world* u*.*

Any admissible evidence function $\mathcal{E}$ *must satisfy the following conditions:*

– Application: $\{p \cdot q | p \in \mathcal{E}(u, s, F \rightarrow G), q \in \mathcal{E}(u, t, F)\} \subseteq \mathcal{E}(u, s \cdot t, G)$
– Monotonicity: $\mathcal{E}(u, s, F) \cup \mathcal{E}(u, t, F) \subseteq \mathcal{E}(u, s + t, F)$.

Given our particular treatment of constants, we require further of $\mathcal{E}$ *that for* $u \in W$*, any instance* A *of an axiom scheme from among A0, A1, A2, and A3, any* $n \geq 1$*, and any justification constants* $c_1, c_2, \ldots, c_n$*,* $\mathcal{E}(u, c_n, c_{n-1} :_1 c_{n-2} :_1 \cdots : c_2 :_1 c_1 :_1 A) = [0, 1]$*.*

Given a model $\mathcal{M} = (W, R, \mathcal{E}, \Vdash)$*, we extend the forcing relation* $\Vdash$ *from sentence variables to formulas as follows, for each* $u \in W$*:*

1. $u \nVdash \bot$
2. $u \Vdash F \rightarrow G$ *if and only if either* $u \nVdash F$ *or* $u \Vdash G$
3. $u \Vdash t :_r F$ *if and only if both* $r \in \mathcal{E}(u, t, F)$ *and* $v \Vdash F$ *for every* v *with* uRv*.*

We now state and prove the central theorem of the paper, closely following Fitting's proof from [9] as presented in [4].

Theorem 4. ***J***^{U} *is sound and complete for Kripke-Fitting models with uncertain justifications.*

Proof. Soundness is straightforward. Let us check each of the axioms:

A0. Propositional axioms are valid due to the standard treatment of $\bot$ and $\rightarrow$ in the semantics.

A1. Application axioms of the form: $s :_p (F \rightarrow G) \rightarrow (t :_q F \rightarrow (s \cdot t) :_{p \cdot q} G)$. Assume that $u \Vdash s :_p (F \rightarrow G)$ and that $u \Vdash t :_q F$. Then we know that:
- $p \in \mathcal{E}(u, s, F \rightarrow G)$
- For each v such that uRv, $v \Vdash F \rightarrow G$
- $q \in \mathcal{E}(u, t, F)$
- For each v such that uRv, $v \Vdash F$.

Because $p \in \mathcal{E}(u, s, F \rightarrow G)$ and $q \in \mathcal{E}(u, t, F)$, our Application inequality condition on $\mathcal{E}$ guarantees that $(p \cdot q) \in \mathcal{E}(u, s \cdot t, G)$. And because for each v such that uRv, $v \Vdash F \rightarrow G$ and $v \Vdash F$, we know that $v \Vdash G$ for all such v as well. Thus, $u \Vdash (s \cdot t) :_{p \cdot q} G$.

A2. Monotonicity axioms of the form: $s :_r F \rightarrow (s+t) :_r F$ and $s :_r F \rightarrow (t+s) :_r F$. Let us treat the first form; the treatment of the second is identical. Let us assume that $u \Vdash s :_r F$. Since we know that $r \in \mathcal{E}(u, s, F)$, our Monotonicity inequality condition on $\mathcal{E}$ guarantees that $r \in \mathcal{E}(u, s + t, F)$ as well. And of course for every v such that uRv, $v \Vdash F$. Thus, $u \Vdash (s + t)_r : F$.

A3. Confidence weakening axioms of the form: $s :_q F \rightarrow s :_p F$ where $p \leq q$. If $u \Vdash s :_q F$, it is immediate that $p \in \mathcal{E}(u, s, F)$ since $p \leq q \in \mathcal{E}(u, s, F)$ and the codomain of $\mathcal{E}$ consists of downward-closed intervals. And since $u \Vdash s :_q F$, $v \Vdash F$ for any world v with uRv. So $u \Vdash s :_p F$ as well.

As for the rules of inference, the soundness of *modus ponens* is immediate from our treatment of $\rightarrow$, but the soundness of the Axiom Internalization Rule requires a very straightforward induction on n, the :-depth of the instance of the rule. Only in the base case, $n = 1$, do we use the fact that axioms are, indeed, valid in all worlds.

Completeness uses a standard canonical model construction. The canonical model $\mathcal{M} = (W, R, \mathcal{E}, \Vdash)$ for $\mathbf{J}^U$ is defined as follows:

- W is the set of all maximally consistent sets in $\mathbf{J}^U$. Tradition urges us to denote these by capital Greek letters Γ, Δ, etc.
- $\Gamma R \Delta$ iff $\{F | t :_r F \in \Gamma$ for some justification term t and some $r > 0\} \subseteq \Delta$
- $\mathcal{E}(\Gamma, t, F) = \{0\} \cup \{r | t :_r F \in \Gamma\}$.
- For propositional atom P, $\Gamma \Vdash P$ iff $P \in \Gamma$, and for all Γ, $\Gamma \nVdash \bot$.

We claim the usual Truth Lemma, that for all formulas F and for all worlds Γ, $\Gamma \Vdash F$ if and only if $F \in \Gamma$.

We prove the Truth Lemma by induction on the construction of F. If F is a propositional atom or $\bot$, the lemma follows from the definition of $\Vdash$. For cases of the form $G_1 \to G_2$, it follows from the fact that $\Vdash$ respects $\to$. The interesting case is that when F is of the form $t :_r G$ for $r > 0$.

If $t :_r G \in \Gamma$, then by the definition of R, $G \in \Delta$ for all Δ such that $\Gamma R \Delta$. By induction, $\Delta \Vdash G$. Furthermore, since $\mathcal{E}(\Gamma, t, G) = \{0\} \cup \{p | t :_p G \in \Gamma\}$, clearly $r \in \mathcal{E}(\Gamma, t, G)$. Hence, $\Gamma \Vdash t :_r G$.

If $t :_r G \notin \Gamma$, then by the definition of $\mathcal{E}$, $r \notin \mathcal{E}(\Gamma, t, G)$, which immediately implies that $\Gamma \nVdash t :_r G$.

This finishes the Truth Lemma. We must also prove that $\mathcal{E}$ as defined in the canonical model satisfies the Application and Monotonicity conditions as well as the condition on constants.

- *Application:* Let $p \in \mathcal{E}(\Gamma, s, F \to G)$ and let $q \in \mathcal{E}(\Gamma, t, F)$. We know that $s :_p (F \to G) \in \Gamma$ and $t :_q F \in \Gamma$. Since maximally closed sets are deductively closed, we use the Application Axiom Scheme to conclude that $(s \cdot t) :_{p \cdot q} G \in \Gamma$, and thus $(p \cdot q) \in \mathcal{E}(\Gamma, s \cdot t, G)$.
- *Monotonicity:* Let $p \in \mathcal{E}(\Gamma, s, F)$. We know that $s :_p F \in \Gamma$. Again, since Γ is maximally consistent and hence deductively closed, we may use the Monotonicity Axiom Scheme to conclude that $(s + t) :_p F \in \Gamma$, and thus $p \in \mathcal{E}(\Gamma, s + t, F)$. The case for $p \in \mathcal{E}(\Gamma, t, F)$ is identical.
- The presence of the Axiom Internalization Rule, and thus the presence of all of its consequences in every maximally consistent set of $\mathbf{J}^U$-formulas, guarantees that for any axiom A which is an instance of axiom scheme A0, A1, A2, or A3 for any $n \geq 1$ and for any justification constants $c_1, c_2, \ldots, c_n$, $c_n :_1 c_{n-1} :_1 \cdots : c_2 :_1 c_1 :_1 A \in \Gamma$. Thus, by the definition of $\mathcal{E}$, $\mathcal{E}(u, c_n, c_{n-1} :_1 c_{n-2} :_1 \cdots : c_2 :_1 c_1 :_1 A) = [0, 1]$.

So we know that our canonical model really is a model for $\mathbf{J}^U$.

From here, the argument is entirely standard. Let F not be derivable in $\mathbf{J}^U$, making $\{\neg F\}$ consistent. Then by Lindenbaum's Lemma ([10]) we can extend $\{\neg F\}$ to a maximally consistent set Γ such that $\neg F \in \Gamma$. Since $F \notin \Gamma$, $\Gamma \nVdash F$ and so F is not valid in $\mathbf{J}^U$.

Just as for the logic $\mathbf{J}$, the canonical model satisfies the Strong Evidence and Fully Explanatory properties, and the proof is identical to that originating in [9] as presented in [4]. (The proof involving the Fully Explanatory property uses the fact that $\mathbf{J}^U$ enjoys internalization.) Similarly, the deductive system we have defined is also sound and complete for the Mkrtychev (one-world) version of the semantics, and again the proof is identical to those found in the above-mentioned sources.

4 Variations

There are three variants worth mentioning here, of increasing intricacy. (1) We could extend our degrees of confidence to include real numbers as well as rational numbers; (2) we could limit our degrees of confidence to a finite subset of the rational numbers; (3) we could alter our semantics to assign each justification to a single rational number rather than to an interval.

The first of these does not merit much discussion in the present paper since all of the arguments presented above go through without difficulty for the real numbers, except that an extension of the Lindenbaum Lemma to uncountable languages would be needed.

4.1 Discrete Degrees of Confidence

The first and most obvious change to the system if we were to move from a dense set of degrees of confidence to a finite and discrete set would be to the Application Axiom Scheme. $s :_p (F \to G) \to (t :_q F \to (s \cdot t) :_{p \cdot q} G)$ makes no sense if the set of degrees of confidence is not closed under multiplication. The obvious choice, at least to me, is to replace $p \cdot q$ with $\min(p, q)$; "Every chain of evidence is precisely as strong as its weakest link."

We could make simplifications to the semantics in this case. Evidence functions could take on a single value for a degree of confidence (this single value representing the maximum p such that $s :_p F$ would be considered true). The Application condition on evidence functions would need to be changed to reflect the use of minima rather than products, but this is straightforward.

The great advantage to this approach is that it tames the complexity of determining the satisfiability of $\mathbf{J}^U$ formulas. I have not yet established an upper bound on the complexity of the satisfiability problem for $\mathbf{J}^U$ as defined above, but in this variation, a slight modification of the usual proof for $\mathbf{J}$ ([14], [16]) shows that the complexity stays at the Σ_2^p-level, the same as for $\mathbf{J}$. (One linear term becomes quadratic.)

4.2 Single-Valued Semantics

Another variation, one which requires no changes in the syntax, is to let $\mathcal{E}(u, s, F) = r$ instead of letting it be $[0, r]$ or $[0, r)$. Saying "I have 0.8 confidence that this piece of evidence justifies belief in F" feels much more intuitive to me than saying "I have every degree of confidence between 0 and 0.8 that this piece of evidence justifies belief in F."

However, there is a problem here: Under our chosen deductive system, $\{s :_p F | p < 0.8\} \cup \{\neg s :_{0.8} F\}$ is consistent. I can have every degree of confidence that s justifies belief in F up to but not including 0.8. What single number should I assign to this degree of confidence? The only possible answer is 0.8, but of course this creates difficulties.

This approach was my initial one, and I was loath to give it up. In fact, one need not give it up because our axiom system is also sound and complete for this semantics! However, the proof is quite awkward and inelegant, and this semantics furthermore has the quite serious flaw that not every consistent set of formulas has a model.

However, one can use a version of the Lindenbaum construction with additional constraints to show that any *finite* consistent set of formulas can be extended to a maximally consistent set of formulas which does happen to have a model. In brief, if one ever finds that $s :_p F$ is consistent with the maximally consistent set under construction, one also adds the condition, inexpressible in the language of $\mathbf{J}^U$, that for some $r \geq p$, $s :_q F$ will be in our maximally consistent set if and only if $q \leq r$. This has the effect of making sure that all maximally consistent sets have degrees of confidence living in closed intervals rather than open intervals.

Part of me still feels that this might have been the correct approach, since single-valued semantics accord so much better with my intuition than interval-valued semantics, but the fact that some consistent sets of formulas lack models persuaded me to present the interval-valued approach as primary.[1] There is clearly room for more investigation here.

4.3 Other Justification Logics

There is not space in this paper to talk with any thoroughness about other logics of justified belief for which uncertain justification is appropriate, but the obvious candidate is **J4**, which is **J** with the addition of positive introspection. I see no problem with considering positive introspection a 100% degree of confidence, so the transition from **J4** to $\mathbf{J4}^U$ would parallel exactly the transition from **J** to $\mathbf{J}^U$. As far as I can tell, there are no issues related to positive introspection that differ in this context from the standard context. (See [2], [9], [4] and many other sources for the relationship of positive introspection to justification logic.) I suspect, but have not worked out the details, that the same is true of the 5 axiom, negative introspection.

The axiom scheme D which mandates consistent belief $((t : \bot) \to \bot)$ is also unproblematic in this context, although I think that a system without this axiom scheme might be of more interest, since $t :_{0.005} \bot$ for a compound justification term t might be acceptable while $t :_{0.9} \bot$ would indicate a serious problem with one of the components of t. This opens the door to the use of the Logic of Uncertain Justifications in belief revision.

By contrast, I see the axiom scheme T $((t :_r F) \to F)$ as incompatible with the motivation behind uncertain justifications, at least if naïvely extended. If we are considering some formulas to be only weakly justified, then treating all formulas, regardless of level of justification, as true strikes me as counterintuitive.

5 Conclusions

As this paper is introducing a new extension of justification logic, there is clearly much more to be done to explore both the technicalities and potential applications of this system. Most obvious to me are questions related to the decidability

[1] Thanks to several colleagues who consulted with me on which approach to take.

and complexity of satisfiability, and if this complexity is high (as I would expect), what can be done to tame it besides limiting to a finite set of degrees of confidence.

The other central problem is dealing with independence. If a single source in which one has 50% confidence tells us both that F and $F \to G$, we find that $s :_{0.5} (F \to G)$ and $s :_{0.5} F$ together imply only $(s \cdot s) :_{0.25} G$, or 25% confidence that s combined with itself justifies belief in G. Under some circumstances, that might be appropriate. If s is designating anything I learn from Wikipedia, then two unrelated pieces of information from Wikipedia might well be independent. But in other circumstances, the use of some other s to justify belief in F and $F \to G$ would certainly not be independent.

One way around this is to rely on the source only once, for a justification of $F \wedge (F \to G)$. If $s :_{0.5} (F \wedge (F \to G))$, then there are constants $c_1, \ldots c_n$ (depending on one's propositional axiomatization) such that $((c_1 \cdot \cdots \cdot c_n) \cdot s) :_{0.5} G$. If we "go to the source" only once for all information to be taken from that source, then conclusions drawn from that source will have the same reliability as the original facts.

This is just a hint of some of the considerations which would need to be addressed in a serious treatment of the independence of justifications.

Acknowledgements. The logic of uncertain justifications was first presented at a symposium in honor of Sergei Artemov's birthday in May 2012, and I wish to thank Prof. Artemov and the organizers of that meeting for the opportunity to present these ideas in very preliminary form.

I am also indebted to several anonymous reviewers who pointed out various areas for improvement. Time permitted me to make only relatively few changes from the initial draft, but I hope to address some of their helpful suggestions in a later version of the paper.

References

1. Artëmov, S.N.: Logic of proofs. Annals of Pure and Applied Logic 67(1-3), 29–59 (1994)
2. Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
3. Artemov, S.N.: Justified common knowledge. Theoretical Computer Science 357(1-3), 4–22 (2006)
4. Artemov, S.N.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
5. Artemov, S.N.: Why do we need Justification Logic? In: van Benthem, J., Gupta, A., Pacuit, E. (eds.) Games, Norms and Reasons: Logic at the Crossroads. Synthese Library, vol. 353, ch. 2, pp. 23–38. Springer (2011)
6. Artemov, S., Kuznets, R.: Logical Omniscience Via Proof Complexity. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 135–149. Springer, Heidelberg (2006)
7. Bavera, F., Bonelli, E.: Justification Logic and History Based Computation. In: Cavalcanti, A., Deharbe, D., Gaudel, M.-C., Woodcock, J. (eds.) ICTAC 2010. LNCS, vol. 6255, pp. 337–351. Springer, Heidelberg (2010)

8. Bonelli, E., Feller, F.: The Logic of Proofs as a Foundation for Certifying Mobile Computation. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 76–91. Springer, Heidelberg (2009), Errata available at, http://www.lifia.info.unlp.edu.ar/eduardo/publications/errataLFCS09.txt
9. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
10. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Kluwer Academic Press (1998)
11. Gödel, K.: Eine interpretation des intuitionistischen aussagenkalküls. Ergebnisse Math. Colloq. 4, 39–40 (1933)
12. Halpern, J.: Reasoning About Uncertainty. MIT Press (2003)
13. Kripke, S.A.: Semantical analysis of modal logic i. normal propositional calculi. Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 9, 67–96 (1963)
14. Kuznets, R.: On the Complexity of Explicit Modal Logics. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 371–383. Springer, Heidelberg (2000); Errata concerning the explicit counterparts of $\mathcal{D}$ and $\mathcal{D}4$ are published as [17]
15. Kuznets, R.: On decidability of the logic of proofs with arbitrary constant specifications. In: 2004 Annual Meeting of the Association for Symbolic Logic, Carnegie Mellon University, Pittsburgh, PA, May 19–23. Association for Symbolic Logic, p. 111 (March 2005) (Abstract)
16. Kuznets, R.: Complexity Issues in Justification Logic. PhD thesis, CUNY Graduate Center (May 2008)
17. Kuznets, R.: Complexity through tableaux in justification logic. In: 2008 European Summer Meeting of the Association for Symbolic Logic, Logic Colloquium 2008, Bern, Switzerland, July 3-8. Bulletin of Symbolic Logic, vol. 15(1), p. 121. Association for Symbolic Logic (March 2009) (Abstract)
18. Milnikel, R.S.: Derivability in certain subsystems of the Logic of Proofs is Π_2^p-complete. Annals of Pure and Applied Logic 145(3), 223–239 (2007)
19. Mkrtychev, A.: Models for the Logic of Proofs. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)
20. Renne, B.: Propositional games with explicit strategies. Information and Computation 207(10), 1015–1043 (2009); Published online April 2009
21. Sedlár, I., Podroužek, J.: Justification logic as dynamic epistemic logic? In: Arrazola, X., Ponte, M. (eds.) Proceedings of the Second ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, LogKCA 2010, pp. 431–442. University of the Basque Country Press (2010)
22. Studer, T.: An application of justification logic to protocol verification. In: Proceedings of the 2011 Seventh International Conference on Computational Intelligence and Security, CIS 2011, pp. 779–783. IEEE (2011)
23. Studer, T.: Justification logic, inference tracking, and data privacy. Logic and Logical Philosophy 20(4), 297–306 (2011)

Justifications, Awareness and Epistemic Dynamics

Igor Sedlár

Institute of Philosophy, Slovak Academy of Sciences
Klemensova 19, 813 64 Bratislava, Slovak Republic
igor.sedlar@savba.sk

Abstract. The paper introduces a new kind of models for the logic of proofs LP, the group justification models. While being an elaboration of Fitting models, the group justification models are a special case of the models of general awareness. Soundness and completeness results of LP with respect to the new semantics are established. The paper also offers an interpretation of the group models, which pertains to awareness and group epistemic dynamics.

1 Introduction

Recent developments in justification logic include contributions to what may be called foundational research. This line of research stems from the fact that the usual kinds of semantics for justification logic [6,11,13] do not provide a specific interpretation of justification terms (let us call this 'the interpretation problem'). An interesting solution to the interpretation problem are Artemov's modular models [5], where terms are interpreted as sets of formulas.

This paper treats the issue along different lines. It offers a solution to the interpretation problem that utilises the notion of awareness and pertains to information dynamics as well. From our viewpoint, justification terms describe *epistemic actions* within specific *groups of agents*. As such, these actions result in alternations of awareness and explicit knowledge of the agents in the group. Justification formulas $[t]F$ describe outcomes of such actions: the action corresponding to t results in F being explicitly known in the group corresponding to t. More specifically, justification formulas are seen as instructions for obtaining universal explicit knowledge in a group of agents.

The connection with awareness has been there for a long time: the Fitting semantics for justification logic [11] bears a strong resemblance to the semantics of the logic of general awareness [9].[1] The connection with information dynamics stems from the fact that justification operations have an obvious dynamic flavour. You can see the operations as types of actions that *create* complex justifications from simpler ones.

[1] This fact has been pointed out many times in the justification logic literature (most notably in [12]). However, a closer analysis of the connection between these two branches of epistemic logic remains to be carried out.

The paper is organised as follows. Section 2 outlines the necessary basics of justification logic and offers a brief summary of awareness models. Section 3 builds upon the connection between justification models and awareness models. Group justification models are defined and soundness and completeness results with respect to LP are established. Section 4 gives an interpretation of the group justification models.

2 Justification Logic and General Awareness

This section provides background for the main parts of the paper, Sects. 3 and 4. We shall be dealing with the logic of proofs LP throughout the paper. Hence, the language and axiom system of LP are described first (2.1). After that, the basics of Fitting semantics for LP are outlined (2.2). Third, we sketch the essentials of the semantics of general awareness (2.3).

2.1 LP, Syntactically

Justification logic started as the logic of proofs [1,2] but it soon evolved into a broader epistemic project [3,4]. However, the original logic of proofs LP keeps its central position in the justification logic family. We shall begin by describing its syntax.

The language of justification logic extends the boolean propositional language by a set of *justification terms* Tm. These allow to express statements of the form 't is a justification of F', where t is a justification term and F is a formula. The set Tm is given by the set of basic terms bTm and a number of operations on terms. The set bTm contains countably many justification variables $x_1, \ldots, y_1, \ldots$ and constants $a_1, \ldots, b_1, \ldots$ (x, y, z etc. are used as metavariables ranging over justification variables and a, b, c etc. as metavariables ranging over constants). We shall be working with two binary operations on terms $+$, $\cdot$ and a unary operation !. The operations allow to construct structured terms, which in turn represent complex justifications.

Formally, the sets of terms Tm and formulas Fm of the justification language are defined as follows:

$$Tm ::= t_i \mid s + t \mid s \cdot t \mid !t$$
$$Fm ::= p_i \mid \neg F \mid F \to G \mid [t]F$$

Here, t_i belongs to bTm and p_i to Φ, a countable set of propositional variables. $[t]F$ is read 't justifies F' (or 't is a justification of F').

Two auxiliary notions will be useful. $STm(t)$ is a set of subterms of t and $g(t)$ is the intersection of bTm and $STm(t)$, i.e. the set of basic subterms of t.

The operation $\cdot$ (application) corresponds to applying modus ponens: it takes the justifications s and t and produces a justification $s \cdot t$ such that if $[s](F \to G)$ and $[t]F$, then $[s \cdot t]G$. The operation $+$ (sum) corresponds to a monotonic combination of justifications. Hence, if $[s]F$, then $[s + t]F$ and $[t + s]F$. The operation ! is a proof-checker (or justification-checker): if t justifies F, then $!t$ justifies $[t]F$, i.e. $!t$ justifies the fact that t justifies F.

Axiomatically, the logic LP consists of:

(A1)	Classical	Propositional tautologies
(A2)	Factivity	$[t]F \to F$
(A3)	Application	$[s](F \to G) \to ([t]F \to [s \cdot t]G)$
(A4)	Sum	$[s]F \to [s + t]F$, $[s]F \to [t + s]F$
(A5)	Checker	$[t]F \to [!t][t]F$

To provide justifications of axioms, a *constant specification* CS is usually introduced. Formally, CS is a function from the set of constants to sets of axioms. If $F \in CS(c)$ for some c, then we say that F has a constant. A CS is *axiomatically appropriate* for LP iff it is exactly the instances of axioms (A1) - (A5) that have constants.

In addition to the axioms we have the following rules:

(R1)	Modus ponens	If $\vdash F$ and $\vdash F \to G$, then $\vdash G$
(R2)	CS axiom necessitation	$\vdash [c]F$ if F is an axiom and $F \in CS(c)$ for some c and an axiomatically appropriate CS.

If CS is an axiomatically appropriate constant specification, then F is CS-provable in LP (LP $\vdash_{CS} F$) iff it is provable from (A1) - (A5) using (R1) and (R2), while CS is used in applications of (R2).

2.2 Fitting Models

A Fitting model (called a weak LP model in [11]) for LP is a quadruple

$$\mathcal{M} = (W, R, E, V)$$

where W is a non-empty set ('points' of $\mathcal{M}$), R is a reflexive transitive binary relation on W, E is a function from $W \times Tm$ to 2^{Fm} (the evidence function) and V is a function from Φ to 2^W (a valuation).

The evidence function E obeys the following conditions:

(E1)	Application	If $F \to G \in E(w, s)$ and $F \in E(w, t)$, then $G \in E(w, s \cdot t)$
(E2)	Monotonicity	If Rwv, then $E(w, s) \subseteq E(v, s)$
(E3)	Checker	If $F \in E(w, s)$, then $[s]F \in E(w, !s)$
(E4)	Sum	$E(w, s) \cup E(w, t) \subseteq E(w, s + t)$

The truth-conditions of formulas are as follows:

$w \Vdash p$	iff $w \in V(p)$
$w \Vdash \neg F$	iff $w \nVdash F$
$w \Vdash F \to G$	iff $w \nVdash F$ or $w \Vdash G$
$w \Vdash [t]F$	iff i) $v \Vdash F$, for all v s.t. Rwv and ii) $F \in E(w, t)$

F is valid in $\mathcal{M}$ iff F holds at every point of $\mathcal{M}$ ($\mathcal{M} \Vdash F$). A Fitting model $\mathcal{M}$ respects a constant specification CS iff $CS(c) \subseteq E(w, c)$ for all w. It is known [11] that if CS is an axiomatically appropriate constant specification, then F is CS-provable in LP iff it is valid in every Fitting model respecting CS.

2.3 General Awareness

The logic of general awareness [9] has been introduced to overcome the notorious problem of logical omniscience. The basic idea is this: to get rid of the fact that every valid formula is known and that knowledge is closed under consequence, one has to modify the definition of knowledge as truth in all accessible points. The logic of general awareness does this by introducing awareness: in order to know F, the agent has to be *aware* of F in addition to F being true in every accessible point.[2]

An awareness model for n agents is a quadruple

$$M = (W, \{R_i\}, \{\mathcal{A}_i\}, V)_{1 \leq i \leq n}$$

where W and V are as before, R_i is a binary relation on W and $\mathcal{A}_i$ is a function from W to sets of formulas of the awareness language. Relations R_i represent epistemic accessibility between states $w \in W$ relative to agents i and are used to define implicit knowledge. $R_i wv$ is usually interpreted as 'The information that i has at w is consistent with what holds at v'. Hence, F holds at every v that is consistent with the information i has at w if and only if i knows F 'implicitly'. It is no surprise that implicit knowledge has the omniscience properties, i.e. it is closed under consequence and every tautology is implicitly known.

But this is not the everyday notion of 'explicit' knowledge. More is needed to formalise explicit knowledge. This is provided by the functions $\mathcal{A}_i$ which give for every $w \in W$ a set of formulas of which the agent i is aware at w. Hence, i is aware of F at w iff $F \in \mathcal{A}_i(w)$. Explicit knowledge is defined as implicit knowledge plus awareness: i knows F explicitly at w iff F holds at every v such that $R_i wv$ and $F \in \mathcal{A}_i(w)$.

It is plain that the properties of explicit knowledge in awareness models depend on the assumed properties of awareness functions. For example, if we allow for states w such that a tautology is not a member of $\mathcal{A}_i(w)$, then not every tautology is explicitly known by i at w. One can avoid closure under consequence similarly.

3 Group Justification Models

This section is the technical core of the paper. First, we point out a familiar fact – Fitting models bear a striking resemblance to awareness models and, consequently, justification formulas $[t]F$ are close to statements about explicit

[2] 'Awareness' is understood in a somewhat general fashion here. For possible interpretations see [9].

knowledge (3.1). Only a minor modification of Fitting models is needed to provide awareness models for LP - group justification models are introduced (3.2) and LP is shown to be sound and complete with respect to the group justification semantics (3.3).

3.1 Justifications and Explicit Knowledge

The similarity of Fitting models to awareness models is striking. It becomes even more obvious if we replace the evidence function E by a family of functions E_s for every $s \in Tm$. This yields a modified truth-condition for $[s]F$:

$$\begin{aligned} w \Vdash [s]F \text{ iff i) } & v \Vdash F, \text{ for all } v \text{ s.t. } Rwv \text{ and} \\ \text{ii) } & F \in E_s(w) \end{aligned}$$

The functions E_s may be seen as awareness functions and, consequently, terms s as denoting agents. Hence, $[s]F$ behaves similarly to explicit knowledge in awareness models: $[s]F$ iff F is known implicitly and s is aware of F.

However, there are two problems. First, if terms are seen as denoting agents, then complex terms should denote 'complex agents'. But what are these? Second, the interpretation of R is a little unclear. If R is an epistemic accessibility relation and terms s denote agents, *'whose'* accessibility relation is R? There is no obvious answer, since R is not linked to a specific term s nor to a specific set of terms. But there might be a way to dispense with a single R. Perhaps we can assign to every s its own awareness function as well as its own accessibility relation R_s.

3.2 The Basic Definitions

A small modification of Fitting models yields a special class of awareness models:

Definition 1. *A* group justification model *for the justification language is a quadruple*

$$\mathbf{M} = (W, \{R_s\}, \{\mathcal{A}_s\}, V)_{s \in Tm}$$

where W is a non-empty set, V is a valuation, every R_s is a binary relation on W such that

- *Every R_s is reflexive and transitive*
- $R_{s+t}, R_{s \cdot t} = R_s \cap R_t$
- $R_{!t} = R_t$

and every $\mathcal{A}_s$ is an awareness function such that:

- *If $F \in \mathcal{A}_s(w)$ and $R_s wv$, then $F \in \mathcal{A}_s(v)$*
- *If $F \to G \in \mathcal{A}_s(w)$ and $F \in \mathcal{A}_t(w)$, then $G \in \mathcal{A}_{s \cdot t}(w)$*
- *If $F \in \mathcal{A}_s(w)$, then $F \in \mathcal{A}_{s+t}(w)$ and $F \in \mathcal{A}_{t+s}(w)$*
- *If $F \in \mathcal{A}_t(w)$, then $[t]F \in \mathcal{A}_{!t}(w)$*

A group justification model respects a CS iff $CS(c) \subseteq \mathcal{A}_c(w)$ for all w.

Definition 2. Truth *of formulas of the justification language in points of group justification models is defined in the usual way. Just to be sure, here is the clause for* $[t]F$*:*

– $w \Vdash [t]F$ *iff* $v \Vdash F$ *for all* v *such that* $R_t wv$ *and* $F \in \mathcal{A}_t(w)$

Validity in $\mathbf{M}$ *(*$\mathbf{M} \Vdash F$*) and* CS*-validity are defined in the usual way.*

It is plain that group justification models are special cases of awareness models for Tm as the set of agents. Hence their name. A more detailed interpretation is provided in Sect. 4.

3.3 Soundness and Completeness of LP

Every Fitting model $\mathcal{M} = (W_{\mathcal{M}}, R, E, V_{\mathcal{M}})$ can be 'transformed' into a group model $\mathbf{M}^{\mathcal{M}} = (W, \{R_s\}, \{\mathcal{A}_s\}, V)_{s \in Tm}$:

– $W_{\mathcal{M}} = W$
– $V_{\mathcal{M}} = V$
– $R = R_s$ for every s
– $F \in \mathcal{A}_s(w)$ iff $F \in E(w, s)$

(Note that $\mathbf{M}^{\mathcal{M}}$ is indeed a group justification model, since the relations R_s defined as above obey the conditions of Def. 1) In a sense, $\mathcal{M}$ and $\mathbf{M}^{\mathcal{M}}$ are the same model.

Theorem 1. *Let* CS *be an axiomatically appropriate constant specification. If* F *is not* CS*-provable in* LP*, then there is a group justification model* $\mathbf{M}$ *respecting* CS *such that* F *is not valid in* $\mathbf{M}$*.*

Proof. It is plain that if $\mathcal{M}$ respects CS then so does $\mathbf{M}^{\mathcal{M}}$. Then it is sufficient to show that for every $\mathcal{M}$ and $\mathbf{M}^{\mathcal{M}}$:

$$\mathcal{M} \Vdash F \text{ iff } \mathbf{M}^{\mathcal{M}} \Vdash F$$

This is trivial for propositional atoms and boolean combinations of formulas. The case for $[s]F$ is equally simple: $F \in E(w, s)$ may be replaced by $F \in \mathcal{A}_s(w)$ and Rwv by $R_s wv$ for every s.

Theorem 2. *Let* CS *be an axiomatically appropriate constant specification. If* F *is* CS*-provable in* LP*, then* $\mathbf{M} \Vdash F$ *for every group justification model* $\mathbf{M}$ *respecting* CS*.*

Proof. Obvious induction on the length of proofs. The only interesting cases are the specific justification axioms. (A2) is CS-valid since extended models are reflexive.

Ad (A3): assume that $w \Vdash [s](F \to G)$ and $w \Vdash [t]F$. Then $F \to G$ holds in every R_s-accessible point and F holds in every R_t-accessible point. Hence, by the definition of $R_{s \cdot t}$, G holds in every $R_{s \cdot t}$-accessible point. Moreover, by the definition of $\mathcal{A}_{s\cdot t}$, $G \in \mathcal{A}_{s\cdot t}(w)$. Hence, $w \Vdash [s\cdot t]G$. The proof for the sum axioms is similar.

Ad (A5): Assume that $w \Vdash [t]F$ and $w \nVdash [!t][t]F$. The latter means that either i) $[t]F \notin \mathcal{A}_{!t}(w)$, or ii) there is a $R_{!t}$-accessible point v such that $v \nVdash [t]F$. Case i) is ruled out by the assumption that if $F \in \mathcal{A}_t(w)$, then $[t]F \in \mathcal{A}_{!t}(w)$. Case ii) implies that either iii) $F \notin \mathcal{A}_t(v)$ or iv) there is a v' such that Rvv' and $v' \nVdash F$. Case iii) is ruled out by the definition of $R_{!t}$ and the fact that $F \in \mathcal{A}_t(w)$ and $R_t wv$ imply $F \in \mathcal{A}_t(v)$. Case iv) is ruled out by the same facts and the transitivity of R_t.

Hence, every axiom is CS-valid. Consequently, every formula derivable by (R2) is also CS-valid.

Corollary 1. LP *is sound and complete with respect to group justification models: F is CS-provable in* LP *iff F is valid in every group justification model respecting CS.*

Proof. Theorems 1 and 2.

Hence, LP is sound and complete with respect to a special class of awareness models, the group justification models. But the mathematics involved is quite simple. Group models are rather close to the original Fitting models and, therefore, Corollary 1 is not a surprising result. However, considering group models may be quite useful. They help to explain that several notions pertaining to group epistemic dynamics are at the heart of justification logic. We expand on this idea in the following section.

4 An Interpretation of the Group Justification Models

This section outlines an interpretation of group justification models. The interpretation utilises the notion of awareness and notions pertaining to epistemic dynamics. Basic justification terms are seen as denoting agents and the corresponding justification formulas as claims about explicit knowledge of the agents (4.1). Complex terms are explained in terms of groups of agents and the link between the corresponding justification formulas and distributed knowledge is made explicit (4.2). Next, it is explained that complex terms encode specific epistemic actions that result in change of awareness in the corresponding groups (4.3). In general, justification formulas are explained as specific instructions for obtaining universal explicit knowledge within groups of agents (4.4). The section concludes by pointing out that the dynamic epistemic interpretation of group justification models provides a specific angle on epistemic closure principles (4.5).

4.1 Agents, Awareness and Explicit Knowledge

First, basic terms $i \in bTm$ may be seen as denoting agents (cf. Sect. 3.1). Hence, relations R_i for $i \in bTm$ are the respective epistemic accessibility relations. These correspond to implicit knowledge: i knows F implicitly at w iff $R_i wv$

implies $v \Vdash F$ for all v. The sets $\mathcal{A}_i(w)$ for $i \in bTm$ correspond to awareness sets of agents i. Hence, i is aware of F at w iff $F \in \mathcal{A}_i(w)$. Thus, formulas $[i]F$ for $i \in bTm$ correspond to claims pertaining to explicit knowledge: $w \Vdash [i]F$ iff i explicitly knows F at w.

Note that special features of group models imply certain interesting features of awareness and implicit knowledge. For example, the condition that $F \in \mathcal{A}_t(w)$ and $R_t wv$ imply $F \in \mathcal{A}_t(v)$ entails that if an agent is aware of F, then the agent implicitly knows that she is aware of F.

A simple epistemic scenario might be helpful, cf. Fig. 1. The agent x explicitly

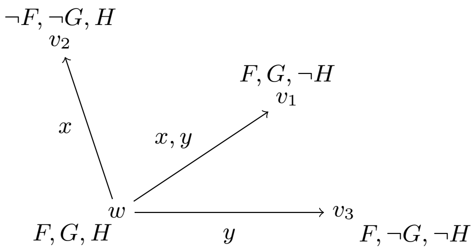


Fig. 1. A simple epistemic scenario involving agents x, y. The reflexive arrows are not drawn. Assume that $\mathcal{A}_x(w) = \mathcal{A}_x(v_1) = \mathcal{A}_x(v_2) = \{F \to G\}$ and $\mathcal{A}_y(w) = \mathcal{A}_y(v_1) = \mathcal{A}_y(v_3) = \{F\}$.

knows $F \to G$ at w, since $F \to G$ holds at w, v_1, v_2 and, importantly, $F \to G \in \mathcal{A}_x(w)$. Similarly, the agent y explicitly knows F at w since F holds at w, v_1, v_3 and $F \in \mathcal{A}_y(s)$.

4.2 Groups and Distributed Knowledge

However, the simple scenario in Fig. 1 contains more information. For example, observe that $[!y][y]F$ and $[x \cdot y]G$ hold at w. But complex terms, such as $!y$ and $x \cdot y$ might seem a bit problematic. If terms denote agents, then complex terms should denote 'complex agents'. But what are the latter?

By Def. 1, $R_{x \cdot y} = R_x \cap R_y$. Hence, the epistemic accessibility relation of the 'agent' $x \cdot y$ is $R_x \cap R_y$, the intersection of the accessibility relations of x and y. Note that, in standard epistemic logic, intersections of accessibility relations of agents in a group are used to define *distributed implicit knowledge* of the group [7,8,10]. Therefore, the following interpretation emerges quite naturally. The formula $[x \cdot y]G$ implies that G is distributed implicit knowledge in the group $\{x, y\}$.

In general, the set $g(t)$ of basic subterms of t may be seen as the group of agents corresponding to t. Accordingly, justification formulas $[t]F$ imply information about distributed knowledge in $g(t)$: $[t]F$ implies that F is distributed knowledge in $g(t)$.

There is also a dynamic angle on the matter. Distributed knowledge in a group may be seen as knowledge that results from 'information sharing' within the group. In our example scenario, relations R_x, R_y are given by the implicit information available to agents x, y. If the agents share their respective information, then no longer $R_x w v_2$ (since the information available to y excludes the point v_2 as possible), nor $R_y w v_3$ (since x excludes v_3). After sharing their information, both agents consider only w and v_1 as possible. In other words, their accessibility relations are reduced to $R_x \cap R_y$.

Hence, R_t for complex t may be seen as the epistemic accessibility relation of agents in $g(t)$ after a complete sharing of information within $g(t)$. Therefore, complex terms t, such as $x \cdot y$, are better not seen as denoting 'complex agents', but as denoting the epistemic state of the agents in $g(t)$ that results from a specific epistemic action – sharing of implicit information.

However, at least two points need further comment. First, the interpretation in terms of distributed knowledge does not seem to apply to terms of the form $!t$. The reason is that $R_{!t} = R_t$. However, $R_t = R_t \cap R_t$. Hence, for example, $[!y][y]F$ may be seen as implying that $[y]F$ is distributed knowledge in the 'group' $\{y\}$.

Second, the meaning of formulas $[t]F$ is not completely described by the above interpretation. Note that $g(x \cdot y) = g(y \cdot x)$ but, obviously, $[y \cdot x]G$ does not hold at w in our example. The structure matters.

4.3 Dynamics of Information and Awareness

Observe that $\mathcal{A}_i(w)$ for $i \in BTm$ may be seen as the *awareness set* of agent i at w. However, the interpretation of the functions $\mathcal{A}_{x \cdot y}$ and $\mathcal{A}_{!y}$ seems a bit unclear: If complex terms correspond to groups of agents, what are awareness sets of groups of agents?

However, a simple explanation may be provided. It has been argued that $x \cdot y$ and $!y$ may be seen as related to specific epistemic actions within $\{x, y\}$ and $\{y\}$ respectively. Now assume that these actions involve not only sharing of implicit information, but also specific adjustments of awareness sets of agents in the respective groups. Outputs of these actions are specific awareness sets, common for every agent in the group.

These outputs are described in Def. 1. For example, the action related to $x \cdot y$ outputs a set that contains every F' such that the awareness set of x contains $F \to F'$ and the awareness set of y contains F. Note that the terms $x \cdot y$ and $y \cdot x$ plainly correspond to different actions. Hence, they might output different awareness sets. Moreover, observe that the resultant awareness set might not contain some elements of the sets of x and y. In other words, the action related to $x \cdot y$ might involve 'forgetting'.

On the other hand, the action related to $x + y$ outputs an awareness set that contains the awareness sets of both x and y unreduced. Hence, this corresponds to a monotonic 'awareness gain'. Interestingly, the action related to $!y$ outputs an 'introspective' awareness set. Therefore, the action itself is 'introspective': if y is aware of F, then y 'becomes aware' of $[y]F$ after performing the introspective action related to $!y$.

In general, the group epistemic actions related to complex terms make the respective groups *epistemically united*. To see this, note that both the relation R_t and the awareness set given by $\mathcal{A}_t$ for complex t are seen as applying to the whole group $g(t)$ uniformly. Hence, in a sense, complex terms correspond to 'group-agents'.

4.4 Terms as Recipes for Explicit Knowledge

Hence, the following picture emerges. Basic terms denote agents and formulas $[i]F$ for $i \in BTm$ may be read in the familiar way as 'i knows explicitly that F'. Complex terms t correspond to specific epistemic actions within the group $g(t)$. Consequently, formulas $[t]F$ for complex terms t describe *hypothetical* explicit knowledge of (all) agents in the group $g(t)$. They may be read as 'every agent in $g(t)$ will explicitly know that F, *provided that* the action corresponding to t takes place'. In a sense, justification formulas $[t]F$ for complex t provide *instructions* for obtaining explicit knowledge of F within $g(t)$.[3]

The actions corresponding to terms have a twofold nature. First, they involve sharing of implicit information. Distributed implicit knowledge in a group becomes universal implicit knowledge ('every agent implicitly knows that...') if the agents share their implicit information. Second, they involve specific adjustments of awareness sets of the agents in the group. These adjustments are encoded by the structure of the respective terms.

4.5 A Note on Closure Principles

The interpretation given above provides a specific perspective on epistemic closure principles. In standard awareness models, the closure properties of explicit knowledge are given somewhat arbitrarily by the respective awareness functions. Group justification models allow for a more natural explanation.

For sake of brevity, let us consider only a single agent a, where 'a' is a constant. Epistemic closure principles of the form

$$[a]F_1 \wedge \ldots \wedge [a]F_n \rightarrow [a]G \tag{1}$$

(where $F_1 \wedge \ldots \wedge F_n \rightarrow G$ is a tautology) are in general not valid in group justification models. However, *dynamic variants* of these principles *are* valid once knowledge of the corresponding tautologies is assumed.

The simplest example is closure under known implication. The 'static' version

$$([a](F \rightarrow G) \wedge [a]F) \rightarrow [a]G \tag{2}$$

is, of course, not valid. However, a 'dynamic variant' of (2)

$$([a](F \rightarrow G) \wedge [a]F) \rightarrow [a \cdot a]G \tag{3}$$

[3] The hypothetical reading of $[t]F$ may be applied to cases where t is basic as well: the action involved is simply the null action 'do nothing'.

is valid. To obtain knowledge of G, the agent has to *do something*: she has to perform an epistemic action and extend her knowledge dynamically.

Another example is closure under $\wedge$–elimination. This closure principle holds only for agents that know that the according logical principle holds. In other words, we have to assume that the agent knows $F \wedge G \rightarrow F$. In general:

$$([a](F \wedge G \rightarrow F) \wedge [a](F \wedge G)) \rightarrow [a \cdot a]F \tag{4}$$

Again, this is a dynamic closure principle.

In general, assume that a knows that $F_1 \wedge \ldots \wedge F_n \rightarrow G$ is a tautology, i.e.

$$[a](F_1 \wedge \ldots \wedge F_n \rightarrow G) \tag{5}$$

Moreover, assume that the agent knows that (5) may be reformulated as a nested conditional, i.e.

$$[a]\left((F_1 \wedge \ldots \wedge F_n \rightarrow G) \rightarrow (F_1 \rightarrow (\ldots (F_n \rightarrow G) \ldots))\right) \tag{6}$$

Then the following general dynamic closure principle holds:

$$[a]F_1 \wedge \ldots \wedge [a]F_n \rightarrow [a^{n+2}]G \tag{7}$$

where $a^1 = a$ and $a^{n+1} = (a^n \cdot a)$.

5 Conclusion

We have defined a new version of models for justification logic, the group justification models. These are a simple modification of the usual Fitting models, while the connection with awareness models of Fagin and Halpern has been made explicit. In fact, group justification models are a specific case of awareness models. Next, soundness and completeness results for LP with respect to group justification models have been obtained. This simple technical result may be seen as a step towards a better understanding of the connections between justification logic and other branches of epistemic logic. Moreover, an interpretation of the group justification models has been provided. This interpretation explains justification terms as corresponding to structured epistemic actions within specific groups of agents. In general, justification formulas $[t]F$ may be seen as instructions for obtaining universal explicit knowledge in specific groups of agents. Consequently, the paper suggests that justification logic absorbs the usual epistemic themes of awareness, group agency and dynamics in a natural way.

Acknowledgements. This paper developed from discussions with Sergei Artemov. His encouragement and constructive comments are the *sine qua non* of its existence. Melvin Fitting provided helpful feedback on an earlier draft. The author is grateful also to three anonymous referees for their comments on the previous version of the paper.

The paper was written in the Institute of Philosophy of the Slovak Academy of Sciences as a contribution to the research project *Language and the Determination of Meaning in Communication.* The project is funded by the grant VEGA 2/0019/12.

References

1. Artemov, S.: Operational modal logic. Technical report, Cornell University (1995)
2. Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7, 1–36 (2001)
3. Artemov, S.: The logic of justification. The Review of Symbolic Logic 1, 477–513 (2008)
4. Artemov, S.: Why do we need justification logic? In: van Benthem, J., Gupta, A., Pacuit, E. (eds.) Games, Norms and Reasons: Logic at the Crossroads, pp. 23–38. Springer, Dordrecht (2011)
5. Artemov, S.: The ontology of justifications in the logical setting. Studia Logica 100, 17–30 (2012)
6. Artemov, S., Nogina, E.: The topology of justification. Logic and Logical Philosophy 17, 59–71 (2008)
7. van Benthem, J.: Logical Dynamics of Information and Interaction. Cambridge University Press, Cambridge (2011)
8. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer, Dordrecht (2008)
9. Fagin, R., Halpern, J.Y.: Belief, awareness, and limited reasoning. Artificial Intelligence 34, 39–76 (1988)
10. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
11. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132, 1–25 (2005)
12. Fitting, M.: Reasoning with Justifications. In: Makinson, D., Malinowski, J., Wansing, H. (eds.) Towards Mathematical Philosophy: Papers from the Studia Logica Conference Trends in Logic IV, pp. 107–123. Springer, Dordrecht (2009)
13. Mkrtychev, A.: Models for the Logic of Proofs. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)

Normal Forms for Multiple Context-Free Languages and Displacement Lambek Grammars

Alexey Sorokin

Moscow State University, Faculty of Mechanics and Mathematics

Abstract. We introduce a new grammar formalism, the displacement context-free grammars, which is equivalent to well-nested multiple context-free grammars. We generalize the notions of Chomsky and Greibach normal forms for these grammars and show that every language without the empty word generated by a displacement context-free grammar can be also generated by displacement Lambek grammars.

1 Introduction

It is well-known, that context-free grammars are insufficient for linguistic purposes. They are unable to capture numerous linguistic phenomena (discontinuous idioms, VP ellipsis, medial extraction, etc.) or, even when they generate correct sentences, their derivation trees do not represent the syntactic structure in a proper way. Also, in some sense, it seems unnatural that a particular grammar formalism generates the language $\{a^n b^n \mid n \in \mathbb{N}\}$, but is unable to generate $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. All that gave birth to different grammar formalisms, such as multiple context-free grammars (MCFGs)([16], [17]), tree-adjoining grammars (TAGs) ([15], [19]), coupled context-free grammars ([4]) and many others, which all generate some context-sensitive languages. They preserve basic properties of context-free grammars, such as polynomial processing and context-freeness of the derivation (that means that whether a particular nonterminal does generate a particular string, doesn't depend on the position of this nonterminal in the derivation), but extend their generative power by generalizing the notions of context, rule and string.

If we regard context-free rules of the form $A \to B_1 \dots B_n$ from the logical point of view, they can be understood as instructions of the form "if for every i from 1 to n a string w_i satisfies a condition B_i (which is expressed in the form $B_i \vdash_G w_i$), then their concatenation $w_1 \dots w_n$ satisfies some predicate A". In fact, context-free grammars can be named "the grammars of concatenation". So it is interesting to add some other operation, for example, intercalation, and investigate the properties of the resultant formalism. We take inspiration from the categorial grammars, where standard Lambek calculus (which is a "logic of concatenation") was enriched with the operation of intercalation and its residuals to obtain the displacement Lambek calculus ([11], [12]). Also our approach is a generalization of the one used by Pollard in Head Grammars ([14]).

In our work we introduce the displacement context-free grammars. They turn out to be equivalent to well-nested multiple context-free grammars ([9], [7]).

Then we prove normal form theorems for the grammars introduced and show how these normal forms can help us to connect some context-sensitive formalisms with categorial grammars and Lambek grammars (as was done in [2] and [13] for context-free grammars and Lambek grammars), proving that every displacement context-free language can be generated by some displacement Lambek grammar (we prove this result only for 1-displacement context-free grammars and 1-displacement Lambek grammars and give the idea of the proof in a general case).

2 Preliminaries

Let k be some natural number (possibly zero). We denote by Op_k the set $\{\cdot, +_1, \ldots, +_k\}$. Let N be some ranked set of variables, such that for every variable its rank is a natural number not greater than k. The notion of rank is extended from variables to terms, defining inductively $rk(\alpha \cdot \beta) = rk(\alpha) + rk(\beta)$, $rk(\alpha +_i \beta) = rk(\alpha) + rk(\beta) - 1, i \in \overline{0,k}$. We denote by $Tm_k(N)$ the set of terms, built from variables in N with the help of operations in Op_k, that do not contain subterms of rank greater than k.

We write the terms from $Tm_k(N)$ in infix notation and assume that all the connectives from Op_k are left-associative. Also we assume that $\cdot$ has greater priority than all the $+_i$ and we will sometimes omit $\cdot$-s in terms. For example $AB +_1 CD +_2 A$ means $((A \cdot B) +_1 (C \cdot D)) +_2 A$.

Definition 1. *A k-displacement context-free grammar (k-DCFG) is a quadruple $G = \langle \Sigma, N, S, P \rangle$, where Σ is a finite alphabet, N is a finite ranked set of nonterminals and $\Sigma \cap N = \emptyset, S \in N$ is a start symbol such that $rk(S) = 0$ and P is a set of rules of the form $A \to \alpha$. Here A is a nonterminal, α is a term from $Tm_K(\Sigma \cup N \cup \varepsilon, 1)$, where ε is a symbol for the empty word and 1 is a metalinguistic separator. Neither ε nor 1 belong to $\Sigma \cup N$. We set the ranks of alphabet symbols and ε to zero, and the rank of 1 to 1. For each rule in P the ranks of its left and right side must coincide.*

We denote $\Sigma' = \Sigma \cup 1$. For every word w over this alphabet we define its rank $rk(w)$ as the number of 1-s it contains. For any two words u, v and every number i not greater than the rank of u we define the operation $u +_i v$ which is merely the result of replacing the i-th separator in u by v. For example, $a1b1cd +_1 a1d = aa1db1cd$. This operation is extended from words to languages in a natural way.

Definition 2. *For every term α from $Tm_K(\Sigma \cup N \cup \varepsilon, 1)$ the set of words $W(\alpha)$ generated by α is the smallest set satisfying the following conditions:*

1) $W(a) = \{a\}$, $a \in \Sigma' \cup \{\varepsilon\}$,
2) $W(\alpha * \beta) = W(\alpha) * W(\beta)$, $* \in Op_k$,
3) If $(A \to \alpha)$ is a rule from P, then $W(\alpha) \subseteq W(A)$.

It is easy to see that nonterminals of rank i can generate only words of rank i. For every k-DCFG $G = \langle \Sigma, N, S, P \rangle$ the language $L(G)$ it generates is $W(S)$. Below we give some examples of DCFG-s.

Example 1. For every positive integer i we define i-DCFG $G_i = \langle \{a, b\}, \{S, T\}, S, P_i \rangle$. Here P_i is the following set of rules (the notation $A \to \alpha|\beta$ stands for $A \to \alpha, A \to \beta$):

$$S \to aT \underbrace{+_1 a \ldots +_1 a}_{i \text{ times}} \quad | \quad bT \underbrace{+_1 b \ldots +_1 b}_{i \text{ times}}$$
$$T \to aT +_1 1a +_2 \ldots +_i 1a \quad | \quad bT +_1 1b +_2 \ldots +_i 1b \quad | \quad 1^i$$

The grammar G_i generates the language $\{w^{i+1} \mid w \in \{a, b\}^+\}$. For example, this is the derivation of the word $(aba)^3$ in G_2: $abaabaaba = aba1ba1ba +_1 a +_1 a = a(b(a1a1a +_1 1b) +_2 1b) +_1 a +_1 a = a((b((a11 +_1 1a) +_2 1a) +_1 1b) +_2 1b) +_1 a +_1 a = a((b((aT +_1 1a) +_2 1a) +_1 1b) +_2 1b) +_1 a +_1 a \in W(a((bT +_1 1b) +_2 1b) +_1 a +_1 a) \subseteq W(a((bT +_1 1b) +_2 1b) +_1 a +_1 a) \subseteq W(aT +_1 a +_1 a) \subseteq W(S)$.

We will not recall the definition of well-nested multiple context-free grammar, because it does not play any role in the further, the interested reader may consult [7]. Well-nested MCFG-s were also studied in recent works of Kanazawa and Salvati ([6], [8]). They generate a proper subclass of multiple context-free languages and seem to be a reasonable formalization of the informal notion of "mildly context-sensitive language" ([5]).

It is not difficult to see that they are equivalent to $k-1$-displacement context-free grammars. Let G be a well-nested k-MCFG, then for every nonterminal B and every tuple of strings $w_1, \ldots w_r$ we must create a corresponding nonterminal B' of a desired $k-1$-DCFG such that $B(w_1, \ldots, w_r)$ holds if and only if the word $w_1 1 \ldots 1 w_r$ belongs to $W(B')$. It is easy because all well-nested rules can be modeled using concatenation and intercalation only (also we must be able to remove superfluous separators, but that is also simulated by intercalation). The opposite translation is also straightforward.

The main difference between DCFG-s and other similar formalisms is that it uses only unary predicates (or nonterminals), like context-free grammars do. Here we follow the Head Grammars [14], where the predicates are also unary. As we show in the next sections, DCFG-s are also convenient to use the standard machinery of context-free grammars.

3 Normal Forms for DCFGs

3.1 Chomsky Normal Form

In this section we adapt the notion of Chomsky normal form to DCFG. Also we introduce the notion of Greibach normal form for 1-DCFG and explain, how it should be modified for k-DCFG in the case of $k > 1$.

Definition 3. *A DCFG $G = \langle \Sigma, N, S, P \rangle$ is called direct, if all rules in P have the form $A \to a$, $A \to B, A \to B * C$ or $A \to \varepsilon$, where $a \in \Sigma'$, $B \in N, * \in Op_k$ and $C \in N$.*

Proposition 1. *Every k-DCFG is equivalent to some direct k-DCFG.*

We want to remove all ε-rules (recall that an ε-rule is a rule of the form $A \to \varepsilon$).

Definition 4. *A nonterminal B is called 1-generating if $1^i \in W(B)$ for some natural i. A nonterminal B is called strictly 1-generating if $W(B) = \{1^i\}$ for some natural i.*

Definition 5. *A direct DCFG is called disjoint, if it does not contain ε-rules, every 1-generating nonterminal is also strictly 1-generating and for every i there is at most one strictly 1-generating nonterminal of rank i.*

Lemma 1. *Every k-DCFG which does not generate an empty word is equivalent to some disjoint k-DCFG.*

Proof. The proof is analogous to removing ε-rules from a context-free grammar, but is a bit more complicated. Firstly, we find all 1-generating symbols the same way we do it for ε-generating symbols in the case of context-free grammars. Then for every $i \leq k$ we introduce a new nonterminal X_i and enrich the grammar with the rules $X_i \to X_1 X_{i-1}$, $i > 1$ and $X_1 \to 1, X_0 \to \varepsilon$.

We want to "divide" each 1-generating nonterminal B into two new nonterminals: B', which generates only non-empty words generated by B, and $X_{rk(B)}$, which generates only the word $1^{rk(B)}$. To achieve this goal we replace every rule $A \to B * C$, $* \in Op_k$ by the rule $A' \to B' * C'$. Also, if C is 1-generating, we add the rule $A' \to B' * X_{rk(C)}$ and if B is 1-generating, we add the rule $A' \to X_{rk(B)} * C'$. It is easy to prove that for every nonterminal B we have $W(B') = W(B) - 1^*$. So we can remove all 1-generating nonterminals except the X_i-s. If we denote the initial grammar by $G = \langle \Sigma, N, S, P \rangle$, then for the resulting grammar $G = \langle \Sigma, N_1, S', P' \rangle$ we have $W(S') = W(S) - \varepsilon = W(S)$.

Now all 1-generating symbols are strictly 1-generating and we should remove ε-generating symbols (in fact, one symbol X_0). In the rules of the form $A \to BX_0$ this is easy, but we need some more sophisticated algorithm in the case of the rules $A \to B{+_i}X_0$. For each nonterminal A we must introduce its "i-th bridge": a new nonterminal A_i such that $W(A_i) = \{w_1 w_2 \mid w_1 1 w_2 \in W(A), rk(w_1) = i{-}1\}$. This is done by tracing the position of i-th separator for A in the derivation tree: for example, if a nonterminal A is of rank 3 and occurs in the rules $A \to BC$ and $A \to D +_2 E$, $rk(B) = rk(D) = rk(E) = 2, rk(C) = 1$, then we must add the rules $A_1 \to B_1 C$, $A_1 \to D_1 +_1 E$, $A_2 \to B_2 C$, $A_2 \to D +_2 E_1$, $A_3 \to BC_1$, $A_3 \to D +_2 E_2$. We must also take into account that $(X_i)_j = X_{i-1}$. After repeating this procedure $k-1$ times (it is also applied to new nonterminals) with every nonterminal the set N of nonterminals also contains all its bridges.

Now we replace all the rules $A \to B +_i X_0$ by the rules $A \to B_i$. All this changes do not affect the language generated by nonterminals, so they do not affect the language generated by the whole grammar. The lemma is proved.

Since this moment we will assume that for every disjoint k-DCFG its set of nonterminals contains the symbols $X_1, \ldots X_k$, which occur in the left side only in rules of the form $X_i \to X_1 X_{i-1}$ and $X_1 \to 1$.

Definition 6. *A disjoint k-DCFG is called specialized if the symbols X_j can occur in the right side only in the rules of the form $A \to B \cdot X_j$ or $A \to X_j \cdot C$.*

Proposition 2. *Every k-DCFG which does not generate the empty word is equivalent to some specialized k-DCFG.*

Proof. We can assume that the original grammar $G = \langle \Sigma, N, S, P \rangle$ is already disjoint. Then for every nonterminal B of rank $i > 0, j \leq i$ and $l \leq (k-i)+1$ we introduce a new nonterminal $B^{j,l}$. We want to augment the grammar with additional rules so that $W(B^{j,l}) = \{w_1 1^l w_2 \mid w_1 1 w_2 \in W(B), rk(w_1) = j-1\}$ (the words in $W(B^{j,l})$ are exactly the words in $W(B)$ after replacing the i-th separator by l consecutive separators). The procedure is analogous to removing ε-rules, but instead of removing the separator we replace it by l new separators. For example, $X_i^{j,l} = X_{i+l-1}$. After repeating this procedure $k-1$ times with every nonterminal B the set of nonterminals also contains all the nonterminals $A^{i,l}$.

Now we can replace every rule of the form $A \to X_i +_j B$ with the pair of rules $A \to X_{j-1}A'$, $A' \to BX_{i-j}$ (A'-s are different for different rules) and every rule of the form $A \to B +_j X_i$ with the rule $A \to B^{i-j+1}$ (if $i = 1$ we just add the rule $A \to B$). So the resulting grammar is specialized and equivalent to initial grammar.

Definition 7. *A k-DCFG grammar $G = \langle \Sigma, N, S, P \rangle$ is said to be in Chomsky normal form, when all the rules in P are in one of the following forms (here $\mathbf{X} = \{X_i \mid i \leq k\}$):*

1) $A \to B \cdot C, A \in N - \mathbf{X}, B \in N - \{S\}, C \in N - \{S\}$,
2) $A \to B +_j C, j \leq k, A \in N - \{S\} - \mathbf{X}, B \in N - \{S\} - \mathbf{X}, C \in N - \{S\} - \mathbf{X}$,
3) $A \to a$, $A \in N - \{X_i | i \leq rk(G)\}, a \in \Sigma$,
4) $S \to \varepsilon$,
5) $X_1 \to 1$,
6) $X_i \to X_1 \cdot X_{i-1}$, $i \geq 2$.

Theorem 1. *Every k-DCFG grammar $G = \langle \Sigma, N, S, P \rangle$ is equivalent to some k-DCFG grammar in Chomsky normal form.*

Proof. As in the case of context-free grammars, the case where G generates the empty word obviously reduces to the case where it does not generate such a word. Then we can assume that G is specialized. To reduce it to Chomsky normal form we should only remove the starting nonterminal from the right sides of the rules and remove all productions of the form $A \to B$. This is done in exactly the same way as in the case of context-free grammars. The theorem is proved.

The existence of Chomsky normal form for DCFG allows us to use the obvious CYK-style algorithm for their processing. For fixed k-DCFG, $k > 1$, this algorithm works in time $O(n^{2k+4})$ in the worst case.

3.2 Greibach Normal Form

Greibach normal form ([3]) is very important for the theory of formal grammars. It provides a weak lexicalization of the corresponding grammar formalism and shows the connection between generative grammars and categorial grammar formalisms. For example, Greibach normal form for context-free grammars offers a natural proof of the fact that every context-free language can be generated by a basic categorial grammar and, hence, by a Lambek grammar. In this section we introduce Greibach normal form for k-DCFG and formulate the reduction theorem. We give a detailed proof for 1-DCFG and draw a sketch of it for $k > 1$.

Let us concern ourselves only with 1-displacement context-free grammars. In this case the set of connectives equals $Op_1 = \{\cdot, +_1\}$, so we will omit the subscript of the intercalation operation and write $A + B$ for $A +_1 B$. Also we will denote the separated 1-generating nonterminal X_1 by X. We assume that all grammars in this chapter are in Chomsky normal form if the opposite is not explicitly postulated. To prove the Greibach normal form theorem we need some auxiliary definitions.

Definition 8. *A* 1*-DCFG* $G = \langle \Sigma, N, S, P \rangle$ *is called a fixed-intercalation-position-1-DCFG(fip-1-DCFG), if for every nonterminal* B *of rank* 1 *one of the following facts holds:* 1)$W(B) \subseteq 1\Sigma^+$, 2)$W(B) \subseteq \Sigma^+ 1 \Sigma^+$, 3)$W(B) \subseteq \Sigma^+ 1$, 4)$W(B) = \{1\}$ *and* $B = X$.

Proposition 3. *Every* 1*-DCFG is equivalent to some fip-1-DCFG.*

Proof. For every nonterminal $B \neq X$ of rank 1 we introduce three new nonterminals B_l, B_c, B_r, where the subscript shows the position of the separator. Then we trace this position through the derivation (for example, the rule $B \to CD$ in the case of $rk(C) = 1$ transforms to rules $B_l \to C_l D, B_c \to C_c D, B_c \to C_r D,$ etc.).

For every fip-1-DCFG we denote by N_l the set of nonterminals generating the words from $1\Sigma^+$. The denotations N_c, N_r are understood in the same way.

Definition 9. *A fip-1-DCFG* $G = \langle \Sigma, N, S, P \rangle$ *is called central if in the rules of the form* $B \to C + D$ *either* $rk(B) = 1$ *or* $C \in N_c$ *and for all the rules of the form* ($B \to C \cdot D$) C *does not belong to* N_l *and* D *does not belong to* N_r.

Lemma 2. *Every 1-DCFG* $G = \langle \Sigma, N, S, P \rangle$ *is equivalent to some central 1-DCFG.*

Proof. We may assume that G is already a fip-1-DCFG. For every nonterminal $B \in N_l^1$ we introduce a new nonterminal $X \backslash B$. For every nonterminal D and every term α, such that the rules $B \to X \cdot D$ and $D \to \alpha$ are in P we add a new rule $(X\backslash B) \to \alpha$. We add the rule $(X\backslash B) \to (X\backslash C) \cdot D$ for every rule $B \to C \cdot D$ and the rule $(X\backslash B) \to (X\backslash D) \cdot (X\backslash C)$ for every rule $B \to C \cdot D$. Now we remove from P all the rules with B in left side and add the rule $B \to X \cdot (X\backslash B)$. For the nonterminals from N_r^1 the procedure is symmetrical (we introduce new nonterminals B/X and so on). The equivalence between the grammar built and the initial grammar follows from the proposition below (here $G' = \langle \Sigma, N', S, P' \rangle$ is a new grammar):

Proposition 4
1) For every nonterminal $B \in N_l^1$ and every word $w \in \Sigma^+$ the word $1w$ belongs to the set $W_G(B)$ iff w is in the set $W_{G'}(X\backslash B)$.
2) For every nonterminal $B \in N_r^1$ and every word $w \in \Sigma^+$ the word $w1$ belongs to the set $W_G(B)$ iff w is in the set $W_{G'}(B/X)$.
3) For every nonterminal $B \in N$ $W_G(B) = W_{G'}(B)$.

Definition 10. *A 1-DCFG is said to be in Greibach normal form if all its rules have one of the following forms:*

1) $B \to a$, where $B \in N - \{X\}, a \in \Sigma$,
2) $B \to aD$ or $B \to Ca$, where $B \in N - \{X\}, C \in N, D \in N, a \in \Sigma$,
3) $B \to CaD, B \to CDa$ or $B \to aCD$, where $B \in N - \{X\}, C \in N, D \in N, a \in \Sigma$,
4) $B \to C+(Da), B \to C+(aD)$ or $B \to C+a$, where $B \in N-\{X\}, C \in N-\{X\}, D \in N, a \in \Sigma$,
5) $B \to CX$ or $B \to XD$, where $B \in N, a \in \Sigma$,
6) $S \to \varepsilon$ or $X \to 1$.

Theorem 2. *Every 1-DCFG $G = \langle \Sigma, N, S, P \rangle$ is equivalent to some 1-DCFG in Greibach normal form.*

Proof. We may assume that G is central. We may also assume that every nonterminal A from the rules $A \to a$, where $a \in \Sigma$, does not stand in the left sides of other rules (we call such nonterminals alphabetical and denote the set of all alphabetical symbols by N_Σ). We use denote a rank of a nonterminal by the superscript(for example, B^1), if there is no superscript, then the rank is arbitrary or is recovered from the context, for the symbols of rank 1 a subscript shows, whether they belong to the set N_l^1, N_c^1 or N_r^1. Alphabet letters will be denoted by symbols a and e, which are in the right side only in the rules $A \to a$ and $E \to e$.

At first we add to N all the nonterminals $A\backslash B$ for $B \in N - N_l^1$ and B/A for $B \in N - N_r^1$ and define a new set P' of rules as a minimal set satisfying the following conditions:

1) For each rule $(A \to a) \in P$ the rule $(A \to a)$ belongs to P'.
2) For all nonterminals $A \in N_\Sigma$ and $B \in N - N_l^1$ the rule $B \to A(A\backslash B)$ is in P'.
For all nonterminals $A \in N_\Sigma$ and $B \in N - N_r^1$ the rule $B \to (B/A)A$ is in P'.
3) For each nonterminal $A \in N_\Sigma$ the rules $A\backslash A \to \varepsilon$ and $A/A \to \varepsilon$ belong to P'.
4) For each pair of letters a, e and each rule $(B^0 \to CD) \in P$ the rules $(A\backslash B) \to (A\backslash C)(D/E)e$ and $(B/A) \to e(E\backslash C)(D/A)$ belong to P'.
5) For each pair of letters a, e and each rule $(B^1 \to C^1 D) \in P$ the rules $(A\backslash B) \to (A\backslash C)e(E\backslash D)$ and $(B/A) \to e(E\backslash C)(D/A)$ belong to P'.
6) For each pair of letters a, e and each rule $(B^1 \to C^0 D^1) \in P$ the rules $(A\backslash B) \to (A\backslash C)(D/E)e$ and $(B/A) \to e(E\backslash C)(D/A)$ belong to P'.
7) For each pair of letters a, e and each rule $(B^0 \to C^1 + D^0) \in P$ the rules $(A\backslash B) \to (A\backslash C) + (e(E\backslash D))$ and $(B/A) \to (C/A) + (e(E\backslash D))$ belong to P'.
8) For each letter a and each rule $(B_l^1 \to XD) \in P$ the rule $(B/A) \to X(D/A)$ is in P'.

9) For each letter a and each rule $(B^1_r \to CX) \in P$ the rule $(A\backslash B) \to (A\backslash C)X$ is in P'.
10) For each pair of letters a, e and each rule $(B^1 \to C^1 + D^1_l) \in P$ the rules $(A\backslash B) \to (A\backslash C) + ((D/E)e)$ and $(B/A) \to (C/A) + ((D/E)e)$ belongs to P'.
11) For each pair of letters a, e and each rule of the form $(B^1 \to C^1 + D^1_c)$ or $(B^1 \to C^1 + D^1_r)$ fr the rules $(A\backslash B) \to (A\backslash C) + (e(E\backslash D))$ and $(B/A) \to (C/A) + (e(E\backslash D))$ belong to P'.
12) The rule $X \to 1$ belongs to P.
13) If the rule $S \to \varepsilon$ belongs to P, it also belongs to P'.

Let us mention that removing ε-rules from the grammar built leads to a grammar in Greibach normal form. Hence to prove the theorem it is sufficient to show that the grammar $G' = \langle \Sigma, N \cup \{A\backslash B \mid A \in N_\Sigma, B \in (N - N^1_l - \{X\})\} \cup \{B/A \mid A \in N_\Sigma, B \in (N - N^1_r - \{X\})\}, 1, P', S\rangle$ is equivalent to the initial grammar G. Let us prove some auxiliary propositions.

Proposition 5. *1) For every letter a, every nonterminal $B \in N - N^1_l - \{X\}$ and every word $w \in \Sigma'^*$ the fact that w is in $W_{G'}(A\backslash B)$ implies that aw is in $W_G(B)$. 2) For every letter a, every nonterminal $B \in N - N^1_r - \{X\}$ and every word $w \in \Sigma'^*$ the fact that w is in $W_{G'}(B/A)$ implies that wa is in $W_G(B)$.*

Proof. It suffices to prove the first proposition. We use induction on derivation length. In the base this length equals 1, so the only possible case is $w = \varepsilon$ and $A = B$. In this case $(A \to a) \in P$ and $a \in W_G(A)$. The basis is proved.

In the induction step we should consider all the possible rules $A\backslash B \to \alpha$ in P' which can start the derivation of the word w from $A\backslash B$. We should also consider all original rules from P which were transformed to place the rule $A\backslash B \to \alpha$ to P'. For example, let $\alpha = (A\backslash C) \cdot e \cdot (E\ D)$ and this rule was created from the rule $B \to C \cdot D$. Then $w = w_1 e w_2$, $w_1 \in W_{G'}(A\backslash C)$, $w_2 \in W_{G'}(E\backslash D)$. By the induction hypothesis $aw_1 \in W_G(C)$, $ew_2 \in W_G(D)$, hence $aw_1ew_2 = aw \in W_G(C \cdot D) \subseteq W_G(B)$, which was required.

Let us consider the variant $\alpha = A\backslash C + (e \cdot (E\backslash D))$ and this rule was created from the rule $B \to C + D$. In this case $w = w_1ew_2w_3$, $w_1 1 w_3 \in W_{G'}(A\backslash C)$, $w_2 \in W_{G'}(E\backslash D)$. By the induction hypothesis $aw_1 1 w_3 \in W_G(C), ew_2 \in W_G(D)$, hence $aw_1ew_2w_3 = aw \in W_G(C+D) \subseteq W_G(B)$. Other cases are analogous. The proposition is proved.

Proposition 6. *1) For any two letters a, e, every nonterminal $B \in N - N^1_l - N^1_r - \{X\}$ and every word $w \in \Sigma'^+$ the fact, that $w = aw_1 = w_2e \in W(B)$ implies, that $w_1 \in W_{G'}(A\backslash B)$ and $w_2 \in W_{G'}(B/E)$.*
2) For any two letters a, e, every nonterminal $B \in N^1_l$ and every word $w \in \Sigma'^+$ the fact, that $w = ua \in W(B)$ implies that $u \in W_{G'}(B/A)$.
3) For any two letters a, every nonterminal $B \in N^1_r$ and every word $w \in \Sigma'^+$ the fact, that $w = au \in W(B)$ implies that $u \in W_{G'}(A\backslash B)$.

Proof. It suffices to prove the first statement, the two others are analogous. We use induction on derivation length (and suppose the induction hypothesis is proved for all the three cases). In the base the length equals 1. It is possible only in the case of the rule $A \to a$ (in this case $A = B$). Because the rules $(A\backslash A \to \varepsilon)$ and $(A/A \to \varepsilon)$ are in P', it holds, that $\varepsilon \in W(A\backslash A), \varepsilon \in W(A/A)$. The base is proved.

In the induction step we should consider all the rules $B \to \alpha$, which start the derivation of the word w from B. Let $\alpha = C \cdot D, C \neq X, D \neq X$. Then $w = w_1 w_2$, $w_1 = au_1 \in W_G(C)$, $w_2 = u_2 e \in W_G(D)$, then by induction hypothesis $u_1 \in W_{G'}(A\backslash C)$, $u_2 \in W_{G'}(D/E)$ (and $a \neq 1, e \neq 1$ because G is central). Then $u_1 u_2 e \in W_{G'}(A\backslash C \cdot D/E \cdot E) \subseteq W_{G'}(A\backslash B)$, which was required. The case of the right slashed nonterminal B/E is analogous.

Now let $\alpha = C^1 + D^1_c$. Then $w = w_1 w_2 w_3$, $w_1 1 w_3 \in W_G(C)$, $w_2 \in W_G(D)$. Let $w_1 = au_1, w_2 = eu_2$, then by induction hypothesis $u_1 1 w_3 \in W_{G'}(A\backslash C), u_2 \in W_{G'}(E\backslash D)$. Hence, $u_1 e u_2 w_3 \in W_{G'}(A\backslash C + (e \cdot (E\backslash D))) \subseteq W_{G'}(A\backslash B)$, and this was required. The case of the right slashed nonterminal B/E is analogous. Other variants are proved in a similar way. The proposition is proved.

It remains to prove $W_G(S) = W_{G'}(S)$. Indeed, let $w = au \in W_G(S)$, then by the proposition 6 $u \in W_{G'}(A\backslash S)$, consequently $au \in W_{G'}(A \cdot (A\backslash S)) \subseteq W_{G'}(S)$. One inclusion is proved. The proof of the opposite one: let $w = au = ve \in W_{G'}(S)$, then by the construction either $u \in W_{G'}(A\backslash S)$, or $v \in W_{G'}(S/E)$. Then by the proposition 5 either $au \in W_{G'}(S)$, or $ve \in W_{G'}(S)$, which was required. The case $\varepsilon \in W_G(S)$ is trivial. The theorem is completely proved.

The notion of Greibach normal form can be modified for k-DCFG in the case of arbitrary k. We will not prove the equivalence theorem, because it requires some technical constructions, but only formulate the result.

Definition 11. *A k-DCFG is said to be in modified Greibach normal form if all its rules have one of the following forms:*

1) $B \to X^j a, B \in N - \{X\}, a \in \Sigma, j \in \mathbb{N}$,
2) $B \to X^j a D$ or $B \to C X^j a$, $B \in N - \{X\}, C \in N, D \in N, a \in \Sigma, j \in \mathbb{N}$,
3) $B \to C X^j a D$, $B \in N - \{X\}, C \in N, D \in N, a \in \Sigma, j \in \mathbb{N}$,
4) $B \to C +_i (X^j a D)$ or $B \to C + (X^j a), B \in N - \{X\}, C \in N - \{X\}, D \in N, a \in \Sigma, i \in \mathbb{N}, j \in \mathbb{N}$,
5) $B \to C X^j$ or $B \to X^j D$, $B \in N, j \in \mathbb{N}$,
6) $S \to \varepsilon$ or $X \to 1$.

Theorem 3. *Every k-DCFG is equivalent to some k-DCFG in modified Greibach normal form.*

4 Displacement Lambek Calculus and Lambek Grammars

4.1 Discontinuous Lambek Calculus

The basic Lambek calculus was introduced in [10]. It is a pure logic of concatenation, which possesses many useful properties, such as decidability, cut-elimination, the subformula property, etc. But, as proved in [13], Lambek grammars cannot generate non-context free languages (the opposite inclusion was proved in [2]) and therefore are insufficient to capture linguistic phenomena. There were several attempts to enrich Lambek calculus with additional connectives. The most interesting and powerful approach is the displacement calculus of Morrill. He adds to the standard product operation and its residuation laws the discontinuous product operation (or intercalation), with the corresponding family of residuation laws. The resulting calculus inherits useful properties of Lambek calculus and allows one to generate some non-context-free languages, as proved in [11]. We strengthen this result by proving that every k-DCFG-language is generated by some displacement Lambek grammar (in fact, k-displacement Lambek grammar).

The types of discontinuous Lambek calculus are made from the basic types with the help of continuous connectives $\backslash, /, \cdot$ and discontinuous connectives $\downarrow_k, \uparrow_k, \odot_k, k \in \mathbb{N}^+$. For every type A we define its sort sA which means that the language of this type can contain only strings of sort sA (the sort of a string is what we previously called its rank: the number of separators it contains). The complex types and their sorts are defined in the following way:

1) If A and B are types and $sA \geq sB$ then A/B and $B\backslash A$ are types and $s(A/B) = s(B\backslash A) = sA - sB$.
2) For all types A, B their product $A \cdot B$ is a type and $s(A \cdot B) = sA + sB$.
3) For all types A, B such that $sA \geq sB - 1$ $B \downarrow_k A$ is a type and $s(B \downarrow_k A) = sA - sB + 1$.
4) If A and B are types and $sA \geq sB$ then for every $k \leq sA - sB + 1$ $A \uparrow_k B$ is a type of the sort $sA - sB + 1$.
5) For all types A, B such that $sA \geq 1$ and for every $k \leq sA$ $A \odot_k B$ is a type and $s(A \odot B) = sA + sB - 1$.
6) I is a type called product unit and $sI = 0$, J is a type called discontinuous product unit and $sJ = 0$.

Let $\mathcal{F}_i$ denote the set of the types of the sort i, Λ be the empty string and [] be a metalinguistic separator. Then the set of hyperconfigurations is defined by the following grammar:
$\mathcal{O} ::== \Lambda|[]|\mathcal{F}_0|\mathcal{F}_i\{\underbrace{\mathcal{O}, \ldots, \mathcal{O}}_{i\mathcal{O}'s}\}|\mathcal{O}, \mathcal{O}.$

The sort of a hyperconfiguration $s(\Gamma)$ is the number of separators it contains and is defined inductively: $s(\Lambda) = 0$, $s([]) = 1$; $s(A) = i$ for $A \in \mathcal{F}_i$; $s(A\{\Gamma_1, \ldots, \Gamma_{s(A)}\} = s(\Gamma_1) + \ldots + s(\Gamma_{s(A)})$; $s(\Gamma, \Delta) = s(\Gamma) + s(\Delta)$. The sequents are of the form $\Gamma \to A$, where $s(\Gamma) = s(A)$. For every type A we define its vector $\overrightarrow{A}$, which is A if $sA = 0$ and $A\{\underbrace{[], \ldots, []}_{sA[]'s}\}$ in the other case.

For any two configurations Γ, Δ we define by $\Gamma|_k\Delta$ the result of replacing the k-th separator in Γ by Δ (it is valid only if $k \leq s(\Gamma)$). If Γ is a configuration of sort i then we denote by $\Gamma \otimes \langle \Delta_1, \ldots, \Delta_i \rangle$ the result of simultaneous replacement of all the separators in Γ by the hyperconfigurations $\Delta_1, \ldots, \Delta_i$. If $\Delta, \Delta_1, \ldots, \Delta_i$ are continuous hyperconfigurations and Γ is a hyperconfiguration of sort i, then a distinguished hyperoccurence $\Delta\langle\Gamma\rangle$ abbreviates a distinguished occurrence $\Delta(\Gamma(\Delta_1, \ldots, \Delta_i))$. Now we can formulate the rules of discontinuous Lambek calculus (we denote it by $\mathbf{D}$):

$$\frac{}{A \to A}(ax) \qquad \frac{\Gamma \to A \quad \Delta\langle\overrightarrow{A}\rangle \to B}{\Delta\langle\Gamma\rangle \to B}(cut)$$

$$\frac{\overrightarrow{A}, \Gamma \to C}{\Gamma \to A \backslash C}(\to \backslash) \qquad \frac{\Gamma \to A \quad \Delta\langle\overrightarrow{C}\rangle \to D}{\Delta\langle\Gamma, \overrightarrow{A \backslash C}\rangle \to D}(\backslash \to)$$

$$\frac{\Gamma, \overrightarrow{A} \to C}{\Gamma \to C/A}(\to /) \qquad \frac{\Gamma \to A \quad \Delta\langle\overrightarrow{C}\rangle \to D}{\Delta\langle\overrightarrow{C/A}, \Gamma\rangle \to D}(/ \to)$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \cdot B}(\to \cdot) \qquad \frac{\Delta\langle\overrightarrow{A}, \overrightarrow{B}\rangle \to D}{\Delta\langle\overrightarrow{A \cdot B}\rangle \to D}(\cdot \to)$$

$$\frac{}{\Lambda \to I}(\to I) \qquad \frac{\Delta\langle\Lambda\rangle \to A}{\Delta\langle I\rangle \to A}(I \to)$$

$$\frac{\overrightarrow{A}|_k \Gamma \to C}{\Gamma \to A \downarrow_k C}(\to\downarrow) \qquad \frac{\Gamma \to A \quad \Delta\langle\overrightarrow{C}\rangle \to D}{\Delta\langle\Gamma|_k \overrightarrow{A \downarrow_k C}\rangle \to D}(\downarrow\to)$$

$$\frac{\Gamma|_k \overrightarrow{A} \to C}{\Gamma \to C \uparrow_k A}(\to\uparrow) \qquad \frac{\Gamma \to A \quad \Delta\langle\overrightarrow{C}\rangle \to D}{\Delta\langle\overrightarrow{C \uparrow_k A}|_k \Gamma\rangle \to D}(\uparrow\to)$$

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma|_k \Delta \to A \odot_k B}(\to \odot) \qquad \frac{\Delta\langle\overrightarrow{A}|_k \overrightarrow{B}\rangle \to D}{\Delta\langle\overrightarrow{A \odot_k B}\rangle \to D}(\odot \to)$$

$$\frac{}{[] \to J}(\to J) \qquad \frac{\Delta\langle[]\rangle \to A}{\Delta\langle J\rangle \to A}(I \to)$$

4.2 Discontinuous Lambek Grammars

Discontinuous Lambek grammars are categorial grammars based on discontinuous Lambek calculus. Here we follow [11]. A lexical assignment α is a pair, comprising a type A and a finite language over the alphabet $V = \Sigma \cup 1$ (the language should not contain 1^{sA}), containing some words of the sort sA. A lexicon is a finite set of lexical assignments. A lexicon Lex and a hyperconfiguration Δ given, we define a correct labelling σ as a function sending each occurrence of a type A in Δ to some word in $Lex(A)$ (the words may differ for different occurrences). When a lexicon is not clear from the context we add "correct w.r.t. Lex". Then we continue the labelling from type occurrences to hyperconfigurations and denote the extended labelling by $yield(\Delta^\sigma)$:

1) $yield(\Lambda^\sigma) = \Lambda$, $yield([]^\sigma) = 1$, $yield(A^\sigma) = \sigma(A), A$ is a type of sort 0.
2) $yield((\Gamma_1, \Gamma_2)^\sigma) = yield(\Gamma^1)^\sigma \cdot yield(\Gamma^2)^\sigma$.
3) $yield(A\{\Gamma_1, \ldots, \Gamma_{sA}\}) = a_0 \cdot yield(\Gamma_1)^\sigma \cdot a_1 \cdot yield(\Gamma_1)^\sigma \cdot \ldots \cdot a_{sA}$, where $\sigma(A) = a_0 1 a_1 1 \ldots a_{sA}$.

For a lexicon Lex and a type A the language $L(Lex, A)$ is defined as $L(Lex, A) = \{yield(\Delta)^\sigma \mid \Delta \to A$ is a theorem of $\mathbf{D}$ and σ is a correct labelling w.r.t $Lex\}$. A grammar $\mathcal{G}$ is a pair of a lexicon Lex and a distinguished type B called the target type. Its language $L(G)$ is defined as $L(G) = L(Lex, B)$. Sometimes we will define displacement Lambek grammars not in terms of a lexicon Lex and a labelling σ but in terms of a relation $\triangleright$ which sends every element of an alphabet Σ to a finite set of zero-sorted types.

Example 2 (Morrill, Valentín). Let the assignment $\triangleright$ be $a \triangleright A\backslash C$, $b \triangleright J\backslash B, J\backslash(A \downarrow_1 B)$, $c \triangleright B\backslash C$, where $sA = sB = sC = 1$ and the distinguished type be $A \odot_1 I$. Then this grammar generates the language $\{a^n b^n c^n \mid n > 0\}$.

A displacement Lambek grammar (DLG) is called a k-DLG if the sort of all types and hyperconfigurations used in its derivations cannot be greater than k. As proved in [11], 1-DLGs generate all permutation closures of regular languages. The applications of DLG-s to natural language syntax can be found in [12].

For our purposes we need to consider some subclass of displacement Lambek grammars, which corresponds to the subclass of AB-grammars in the class of usual Lambek grammars. We define a set of simple types as a smallest set satisfying the conditions below. For every simple type we also define its head.

1) All basic types and the type 1 are simple types (called basic simple types) The head of such a type is the type itself.
2) If $A \neq 1$ is a simple type and B is a basic simple type, then $B\backslash A, A/B, B \downarrow_k A, A \uparrow_K B$ are simple types (if their sort is correctly defined). The head of such types coincides with the head of type A.

The notion of a simple hyperconfiguration is analogous to the notion of a hyperconfiguration except we use only simple types. We denote by $\mathbf{D}_L$ the calculus which considers only simple hyperconfigurations and types and is obtained from D by omitting all product and right-hand rules. We define $\mathbf{D}_L$-grammars exactly as $\mathbf{D}$-grammars are defined except for the basic calculus and the fact that the target type should be basic and of the sort 0. The lemma below easily follows from the subformula property, which implies that righthand and product rules are useless for simple hyperconfigurations.

Lemma 3. *Given a $\mathbf{D}_L$-grammar (G), the languages $L_{\mathbf{D}}(G)$ and $L_{\mathbf{D}_L}(G)$ coincide.*

So it suffices to show that every DCFG-language without the empty word is generated by some $\mathbf{D}_L$-grammar to prove that every such language is also generated by some displacement Lambek grammar. The statements below are proved by induction on derivation and play a basic role in proving the main theorem.

Proposition 7. *Let $\Delta \to A$ be a theorem of $\mathbf{D}_L$. Then A is a subtype of some type occurrence in Δ.*

Lemma 4 (Unfolding lemma). *Let $\Delta \to A$ be a theorem of $\mathbf{D}_L$. Then one of the following properties holds:*

1) $\Delta = \overrightarrow{A}$.
2) $\Delta = \Delta_1, \Delta_2$ and there are simple types A/B and B such that $\Delta_1 \to A/B$ and $\Delta_2 \to B$ are theorems of $\mathbf{D}_L$.
3) $\Delta = \Delta_1, \Delta_2$ and there are simple types B and $B\backslash A$ such that $\Delta_1 \to B$ and $\Delta_2 \to B\backslash A$ are theorems of $\mathbf{D}_L$.
4) $\Delta = \Delta_1|_k\Delta_2$ and there are simple types $A \uparrow_k B$ and B such that $\Delta_1 \to A \uparrow_k B$ and $\Delta_2 \to B$ are theorems of $\mathbf{D}_L$.
5) $\Delta = \Delta_1|_k\Delta_2$ and there are simple types B and $B \downarrow_k A$ such that $\Delta_1 \to B$ and $\Delta_2 \to B \downarrow_k A$ are theorems of $\mathbf{D}_L$.

5 Greibach Normal Form and Displacement Lambek Grammars

In this section we prove that every 1-MCFG which does not generate the empty word is equivalent to some $\mathbf{D}_L$-grammar. Also we formulate the corresponding theorem for k-MCFG but do not prove it because of the lack of space and its technical difficulty. We will omit the subscript 1 near the connectives $\downarrow, \uparrow$ and $|$.

Definition 12. *Let $G = \langle \Sigma, N, S, P \rangle$ be a 1-DCFG in Greibach normal form. Then its corresponding $\mathbf{D}_L$-grammar $\mathcal{G} = \langle \triangleright, S \rangle$ is defined as follows (we identify the nonterminal X and the type 1 because they play the same role):*

1) The set of basic types is N. The rank of each type is inherited from G.
2) For every rule $(B \to a) \in P$ it holds that $a \triangleright B$.
3) For every rule $(B \to Ca) \in P$ it holds that $a \triangleright C \backslash B$. For every rule $(B \to aD) \in P$ it holds that $a \triangleright B/D$.
4) For every rule $(B \to CDa) \in P$ it holds that $a \triangleright D\backslash(C\backslash B)$. For every rule $(B \to CaD) \in P$ it holds that $a \triangleright C\backslash(B/D)$. For every rule $(B \to aCD) \in P$ it holds that $a \triangleright C\backslash(D\backslash B)$.
5) For every rule $(B \to C + Da) \in P$ it holds that $a \triangleright D\backslash(C \downarrow B)$. For every rule $(B \to C + aD) \in P$ it holds that $a \triangleright (C \downarrow B)/D$. For every rule $(B \to C + a) \in P$ it holds that $a \triangleright C \downarrow B$.
6) For every rule $B \to CX$ and for every assignment $a \triangleright A$, where C is a head of A we add to the lexicon the assignment $a \triangleright C'$, where C' is the result of replacing the head of A to $1\backslash B$.
7) For every rule $B \to XD$ and for every assignment $a \triangleright A$, where D is a head of A we add to the lexicon the assignment $a \triangleright C'$, where B' is the result of replacing the head of A to $D/1$.

Note that only the types of rank 1 can be replaced in the head, so in every type the replacement took place at most once. That means that all the types in the lexicon contain at most four basic types.

Lemma 5. *Let G be a 1-DCFG in Greibach normal form and $\mathcal{G}$ be the corresponding $\mathbf{D}_L$-grammar. Then $L(G) \subseteq L(\mathcal{G})$.*

Proof. We prove a stronger statement: for every word in Σ' and every nonterminal B the fact that $w \in W(B)$ implies that $w \in L(Lex, B)$, where Lex is the lexicon of $\mathcal{G}$. This is proved by induction on derivation, the basis is obvious. In the induction step we consider all the rules that could start the derivation.

Let the derivation start with the rule $B \to C{+}Da$. Then if $w = w_1 uaw_2$, where $w_1 1 w_2 \in W(C)$ and $u \in W(D)$. By induction hypothesis $w_1 1 w_2 \in L(Lex, C)$ and $u \in L(Lex, D)$, but the sequent $C|(D, D\backslash(C \downarrow B)) \to B$ is derivable and yields exactly the word $w = w_1 uaw_2$.

Let the derivation start with the rule $B \to CX$. Then $w = u1$, by induction hypothesis $u \in L(Lex, C)$. Replacing the head occurrences of C by $B/1$ (we can do it by the construction), we obtain that also $u \in L(Lex, B/1)$. But the sequent $(B/1), 1 \to B$ is derivable, so $w \in L(Lex, B)$. All other cases are similar.

The proof of the converse inclusion extensively uses the unfolding lemma.

Lemma 6. *Let G be a* 1*-DCFG in Greibach normal form and $\mathcal{G}$ be the corresponding $\mathbf{D}_L$- grammar. Then $L(\mathcal{G}) \subseteq L(G)$.*

Proof. We prove a stronger statement: for every word in $(\Sigma \cup 1)^+$ and every nonterminal B the fact that $w \in L(Lex, B)$ implies that $w \in W(B)$, where Lex is the lexicon of $\mathcal{G}$. The proof uses induction on derivation length. The basis is the case of axiom, it is obvious. In the inductive step we examine which unfolding starts the derivation.

Let it be the unfolding $C|(C \downarrow B) \to B$. Then $w = u_1 v u_2$, where $u = u_1 1 u_2 \in L(Lex, C)$. By the induction hypothesis $u \in W(C)$, but we cannot apply the hypothesis to the type $(C \downarrow B)$, because it is not from N, so we continue the unfolding. Let the next unfolding be $((C \downarrow B)/D)D \to (C \downarrow B)$. But by proposition 7 the type $(C \downarrow B)/D$ must be a subtype of some type from the lexicon. It could be so only if $a \triangleright (C \downarrow B)/D$ for some letter a. Applying the induction hypothesis to the type D we deduce that $v = av', v' \in W(D)$. By construction there is a rule $B \to C + aD$ in P, so the word $u + av' = u_1 a v' u_2 = w$ belongs to $W(B)$. Other cases are similar, because all the unfoldings should reach the type from the lexicon in a bounded number of steps. The lemma is proved.

It is not hard to see that the grammar $\mathcal{G}$ is in fact a 1-DLG, so we have proved the following theorem.

Theorem 4. *Every* 1*-DCFG language that does not contain the empty word is generated by some* 1*-displacement Lambek grammar.*

Applying an analogous construction to the modified Greibach normal form, we obtain the general theorem:

Theorem 5. *Every k-DCFG language that does not contain the empty word is generated by some k-displacement Lambek grammar.*

6 Further Research

We have adapted the notion of Greibach normal form to displacement context-free grammars and proved, that every language that does not contain the empty word and is generated by some k-displacement context-free grammar is also generated by some k-displacement Lambek grammar. On the contrary to usual Lambek grammars, the converse is not true. As proved in [8], the language $MIX = \{w \| w|_a = |w|_b = |w|_c\}$ cannot be generated by well-nested 2-MCFG and, hence, by 1-DCFG. But in [12] MIX is proved to be generated by 1-DLG as a permutation closure of a regular language. If the conjecture of Kanazawa and Salvati that MIX is not a well-nested multiple context-free language holds, than the class of displacement context-free languages is properly included to the class of the languages generated by some displacement Lambek grammar. Since the proof of the equivalence of context-free languages and Lambek grammars is based on interpolation and conjoinability properties of Lambek calculus, it is also interesting to study the corresponding properties of displacement Lambek calculus. Also it seems interesting to establish some nontrivial upper bounds for the generative power of displacement Lambek grammars and their variants, for example, if a displacement Lambek grammar generates non-multiple context-free languages.

References

1. Kracht, M.: The mathematics of language. Mouton de Gruyter, Berlin (2003)
2. Bar-Hillel, Y., Gaifman, H., Shamir, E.: On categorial and phrase-structure grammars. Bull. Res. Council Israel Sect. F 9F, 1–16 (1960)
3. Greibach, S.: A new normal form theorem for context-free structure grammars. Journal of the ACM 12(1), 42–52 (1965)
4. Hotz, G., Pitsch, G.: On parsing coupled context-free languages. Theoretical Computer Science 161, 205–233 (1996)
5. Joshi, A.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press, New York (1985)
6. Kanazawa, M., Salvati, S.: The Copying Power of Well-Nested Multiple Context-Free Grammars. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 344–355. Springer, Heidelberg (2010)
7. Kanazawa, M.: The Pumping Lemma for Well-Nested Multiple Context-Free Languages. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583, pp. 312–325. Springer, Heidelberg (2009)
8. Kanazawa, M., Salvati, S.: MIX is not a tree-adjoining language. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pp. 666–674 (2012)
9. Kuhlman, M., Möhl, M.: Mildly context-sensitive dependency languages. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, pp. 160–167 (2007)
10. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65, 154–170 (1958)
11. Morrill, G., Valentín, O.: On calculus of displacement. In: Proceedings of the 10th International Workshop on Tree Adjoining Grammars and Related Formalisms, pp. 45–52 (2010)
12. Morrill, G., Valentín, O., Fadda, M.: The displacement calculus. Journal of Logic, Language and Information 20(1), 1–48 (2011)
13. Pentus, M.: Lambek grammars are context-free. In: Proceedings of the Logic in Computer Science, LICS 1993, pp. 429–433 (1993)
14. Pollard, C.: Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages. Ph.D. thesis, Stanford University (1984)
15. Joshi, A., Schabes, Y.: Tree-adjoining grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Beyond Words, vol. 3, pp. 69–123. Springer, Berlin (1997)
16. Seki, H., Matsumoto, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88(2), 191–229 (1991)
17. Seki, H., Kato, Y.: On the generative power of multiple context-free grammars and macro grammars. IEICE Transactions on Information and Systems E91-D(2), 209–221 (2008)

18. Valentín, O.: 1-discontinuous Lambek calculus: Type logical grammar and discontinuity in natural language. Master's thesis. Universitat Autònoma de Barcelona (2006)
19. Vijay-Shanker, K., Weir, D.: The equivalence of four extensions of context-free grammar. Mathematical Systems Theory 27, 511–546 (1994)
20. Weir, D.: Linear context-free rewriting systems and deterministic tree-walking transducers. In: Proceedings of the 30th Annual Meeting on Association for Computational Linguistics, pp. 136–143 (1992)

Constructive Polychronous Systems

Jean-Pierre Talpin[1], Jens Brandt[2], Mike Gemünde[2], Klaus Schneider[2], and Sandeep Shukla[3]

[1] INRIA Rennes-Bretagne-Atlantique, France
[2] Department of Computer Science, University of Kaiserslautern, Germany
[3] Department of Electrical and Computer Engineering, Virginia Tech, USA

Abstract. The synchronous paradigm provides a logical abstraction of time for reactive system design which allows automatic synthesis of embedded programs that behave in a predictable, timely and reactive manner. According to the synchrony hypothesis, a synchronous model reacts to input events and generates outputs that are immediately made available. But even though synchrony greatly simplifies design of complex systems, it often leads to rejecting models when data dependencies within a reaction are ill-specified, leading to causal cycles. Constructivity is a key property to guarantee that the output during each reaction can be algorithmically determined. Polychrony deviates from perfect synchrony by using a partially ordered or relational model of time. It captures the behaviors of (implicitly) multi-clocked data-flow networks and can analyze and synthesize them to GALS systems or to Kahn process networks (KPNs). In this paper, we provide a unified constructive semantic framework, using structural operational semantics, which captures the behavior of both synchronous modules and multi-clocked polychronous processes. Along the way, we define the very first operational semantics of Signal.

1 Introduction

Languages such as Esterel [1], Quartz [2] or Lustre [3] are based on the *synchrony hypothesis*. Synchrony is a logical abstraction of time which greatly facilitates verification and synthesis of safety-critical embedded systems. In particular, it enforces deterministic concurrency, which has many advantages in system design, e.g. avoiding Heisenbugs (i.e. bugs that disappear when one tries to simulate/test them), predictability of real-time behavior, as well as provably correct-by-construction software synthesis [4].

It is also the key to generate *deterministic* single-threaded code from multi-threaded synchronous programs so that synchronous programs can be directly executed on simple micro-controllers without using complex operating systems. Another advantage is the straightforward translation of synchronous programs to hardware circuits [5,6]. Furthermore, the concise formal semantics of synchronous languages allows one to formally reason about program properties [7], compiler correctness and worst-case execution time [8,9].

Under the synchrony hypothesis, computation progresses through totally ordered synchronized execution steps called reactions. The computation involved in reacting to a particular input combination starts by reading the inputs, computing the intermediate values as well as the outputs, and the next state of the system. Each reaction is referred to as a *macro-step*. Computations during the reaction are called *micro-steps*. A reaction is said to happen at a *logical instant*. A logical instant abstracts the duration of a reaction to a single point in a discrete totally ordered timeline.

Consequently, and from a semantic point of view which postulates that a reaction is atomic, neither communication nor computation therefore takes any time in a synchronous instant. Even though this zero-time assumption does not correspond to reality, it is where the power of the synchronous abstraction lies – zero delay is compatible to predictability. If (1) the minimum arrival time of two consecutive values on all inputs is long enough, and if (2) all micro-steps in a reaction (macro-step) are executed according to their data dependencies, then the behaviors under the zero-time assumption are the same as behaviors of the same system in reality.

However, the synchronous abstraction of time also has a drawback. Since outputs are generated in zero-time, and since synchronous systems can typically read their own outputs, there may be cyclic dependencies due to actions modifying their own causes within the same reaction. These issues lead to programs having inconsistent or ambiguous behaviors. In the context of synchronous programs, they are known as *causality problems*, and various solutions have been proposed over the years to tackle them. The most obvious and pragmatic one is to syntactically forbid cyclic data dependencies. It is simple to check but rules out many valid programs. For example, the synchronous language LUSTRE follows this approach.

Clearly, this is a conservative approach that hardly scales to mapping models on platforms or composing models as this often introduces cycles [10]. Therefore, other synchronous languages, such as ESTEREL [1], opted for a more sophisticated but costly solution. Their semantics is given in terms of a constructive logic, and compilers perform a causality analysis [11,12,13,14] based on the computation of fixpoints in a three-valued logic similar to Brzozowski and Seger's ternary simulation of asynchronous circuits [15]. Thereby, cyclic dependencies are allowed as long as they can be constructively resolved. This definition of causality does not only show parallels to hardware circuit analysis but also to many other areas.

In contrast to synchronous languages, the polychronous language SIGNAL [16] follows a different model of computation. Execution is not aligned to a totally ordered set of logical instants but to a partial order. This allows one to directly express (abstractions of) asynchronous computations which possibly synchronize intermittently. The lack of a global reference of time offers many advantages for the design of embedded software architectures. First, it is closer to reality since at the system level, integrated components are typically designed based on different clock domains or different paces, which is a desirable feature especially with the advent of, e.g., multi-core embedded processors. Second, polychrony avoids unnecessary synchronization, thereby offering additional optimization opportunities.

Polychrony gives developers the possibility to refine the system in different ways, and compilers can choose from different schedules according to non-functional mapping constraints, which are ubiquitous in embedded systems design. Due to these advantages, SIGNAL is particularly suited as a coordination layer on top of synchronous components to describe a globally asynchronous locally synchronous (GALS) network.

As SIGNAL makes use of the synchronous abstraction of time, it faces the same problems as other synchronous languages. One way to overcome the causality problem is to syntactically forbid cyclic dependencies but as stated before that is not always possible, especially when composing separately specified processes. The SIGNAL compiler uses a so-called *conditional dependence graph* [17,18] to model dependencies between equations and check that all equations in a syntactic cycle cannot happen at the same logical instant. As discussed above, the synchronous languages are all based on slightly varying notions of causality.

This mismatch makes it unnecessarily hard (if not impossible) to integrate, e.g., a set of reactive QUARTZ modules with a SIGNAL data-flow network: should the integration of modules and processes be limited/approximated by syntactically causal data-flow networks, instead of constructive ones ? There is no fundamental reason why a common notion of constructivity should not exist for these languages. So, instead of an approach to causality analysis based on cycle detection, we want to endow SIGNAL with a constructive semantics compatible to that of languages like QUARTZ, which is exactly what this paper presents.

Contribution. Our work is rooted in a collaborative project, Polycore, in which we aim at combining the expressive capabilities of the imperative synchronous language QUARTZ and the data-flow polychronous language SIGNAL towards the goal of synthesizing executable GALS systems from the specification of polychronous networks of synchronous modules. This goal demands a common understanding of (i) constructivity and synchronous determinism, best known and studied in the context of imperative synchronous languages, and (ii) endochrony and asynchronous determinism, better developed in the context of relational synchronous languages (yet applicable to imperative ones). This paper tries to bridge the mathematical gap between constructivity and clock/causality analysis in order to show to what extent the former can be explained with the latter.

Our approach consists in the definition of a constructive semantics for polychronous processes that works for synchronous modules as well, so as to share existing notions, theorems and methods. This semantics is of interest on its own: it allows us to better understand the relationship between synchrony and polychrony, between constructivity and endochrony, and to model causality as a formal verification problem. We provide a unified structured operational semantics framework which both captures the synchronous behavior of reactive modules and the multi-clocked behavior of polychronous data-flow networks. This framework allows us to formulate a constructivity theory which captures determinism for both synchronous modules and asynchronous networks. Its expressive capability defines an effective framework in which embedded systems can be designed by combining the best of both styles: imperative modules to describe system functions and polychronous data-flow networks to describe high-level abstractions of their software architecture. Along the way, we give the very first executable semantics (i.e. an interpreter) of Signal.

Related Work. Causal cycles may be real or may be induced by the synchronous abstraction of time. They were first considered in hardware circuits in the early seventies [19,20]. However, causality issues are not only related to the stability analysis of hardware circuits. Berry pointed out that causality analysis is equivalent to theorem proving in intuitionistic (constructive) propositional logic and introduced the notion of constructive programs [11]. Finally, Edwards reformulates the problem such that the existence of dynamic schedules must be guaranteed for the execution of mutually dependent microsteps [13]. Hence, causality analysis is a fundamental algorithm that has already found many applications in computer science. Malik [21] was first to show that this problem in general is NP-hard [21] and used the embedding of Boolean algebra in ternary algebra as proposed by Bryant for the simulation of switch-level circuits [22]. More details about causality analysis for synchronous programs may be found in [21,12,14,23]. In the domain of polychronous programs, causal cycle detection using SMT solvers has been reported in [24,25].

2 Constructive Synchronous Systems

In general, cyclic systems might have no behavior (loss of reactivity), more than one behavior (loss of determinism) or a unique behavior. However, having a unique behavior is generally not sufficient, since there are programs whose unique behavior can only be found by trial and error (or large lookup-tables for all inputs and states, alternatively) – which obviously does not lead to an efficient computation. For this reason, one is usually interested in whether a program has a unique behavior that can be *constructively determined* by the operational semantics. Such a *constructive* semantics is based on fixpoint iteration, which repeatedly execute iterations in order to infer the clocks and values of all signals at every logical instant, i.e., during every reaction.

Embedding Clocks and Values in a Complete Lattice. We shall first define some of the essential concepts of fix-point theory used in this paper [26].

Definition 1 (Complete Lattice). *A partial order $(\mathcal{D}, \sqsubseteq)$ is a* lattice*, if every pair $\{x, y\} \subseteq \mathcal{D}$ has a supremum $\bigsqcup\{x, y\}$ and an infimum $\bigsqcap\{x, y\}$ in $\mathcal{D}$. It is* complete *if $\bigsqcup M$ and $\bigsqcap M$ exist for all $M \subseteq \mathcal{D}$. A function $f : \mathcal{D} \to \mathcal{D}$ is monotonic, if for all $x, y \in \mathcal{D}$ s.t. $x \sqsubseteq y$, $f(x) \sqsubseteq f(y)$. It is continuous, if $f(\bigsqcup M) = \bigsqcup f(M)$ and $f(\bigsqcap M) = \bigsqcap f(M)$ holds for all directed sets $M \subseteq \mathcal{D}$.*

To define a constructive semantics, we need to embed the set of data values into a lattice. We achieve this by adding new elements $\mathcal{D}' = \mathcal{D} \cup \{?, \bot, \top, \lightning\}$ to the domain $\mathcal{D}$, the domain of values taken by a variable (see Figure 1).

Starting from $\mathcal{D}'$, we define a partial order $\sqsubseteq\ \subseteq \mathcal{D}' \times \mathcal{D}'$. Intuitively, the *greater* a value is, the more information we have about it. The error value $\lightning$ is the greatest element since inconsistent values should never become consistent, while the opposite may occur. We can lift every operator $g : \mathcal{D}^n \to \mathcal{D}^m$ to $g' : \mathcal{D}'^n \to \mathcal{D}'^m$ to evaluate g' in the lattice $\mathcal{D}'$ under the conditions (1) the extended operator g' comply with the original one on $\mathcal{D}$, i. e. $\forall \boldsymbol{x} \in \mathcal{D}^n.\ g'(\boldsymbol{x}) = g(\boldsymbol{x})$ and (2) g' is monotonic w. r. t. $\sqsubseteq$ (for finite $\mathcal{D}'$ this also implies continuity of g').

? signal status is unknown
$\bot$ signal is absent or inhibited
$\top$ signal is present or activated
$\lightning$ signal is inconsistent (e. g. runtime error)

$$\begin{tikzcd}
 & & \bot \arrow[dr] & \\
? \arrow[urr] \arrow[r] & \top \arrow[r] \arrow[dr] & 0 \arrow[r] & \lightning \\
 & & \vdots \arrow[ur] &
\end{tikzcd}$$

Fig. 1. Embedding a value domain $\mathcal{D}$ in a complete lattice

These conditions predetermine some values of the extension. For the remaining ones, we prefer values $x \in \mathcal{D}$ in order to accept as many programs as possible, and hence choose to use the maximal extension of g w. r. t. $\sqsubseteq$ [23]. As an example, Figure 2, consider Boolean conjunction $\wedge : \mathbb{B}^2 \to \mathbb{B}$. Its extension $\wedge'$ is extended from the original function $\wedge$ on Booleans by choosing the greatest value that satisfies condition 2 above. For example, $\top \wedge' 0$ must be less than or equal $0 \wedge' 0 = 0$ and $1 \wedge' 0 = 0$ (since $\top \sqsubseteq 0$ and $\top \sqsubseteq 1$). As both results are 0, we can also set $\top \wedge' 0 = 0$. Now consider $\top \wedge' 1$, it must be less than or equal to $0 \wedge' 1 = 0$ and $1 \wedge' 1 = 1$. Again, we must set $\top \wedge' 1 = \top$. All the other values in the table can be determined in the same way.

$\wedge'$	?	$\bot$	$\top$	0	1	$\lightning$
?	?	$\bot$	$\top$	0	$\top$	$\lightning$
$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$	$\lightning$

$\wedge'$	?	$\bot$	$\top$	0	1	$\lightning$
$\bot$	$\bot$	$\bot$	$\lightning$	$\lightning$	$\lightning$	$\lightning$
$\top$	$\top$	$\lightning$	$\top$	0	$\top$	$\lightning$

$\wedge'$	?	$\bot$	$\top$	0	1	$\lightning$
0	0	$\lightning$	0	0	0	$\lightning$
1	$\top$	$\lightning$	$\top$	0	1	$\lightning$

Fig. 2. Embedding conjunction into $\mathbb{B}'$

We obtain an embedding of all operators into continuous functions over our extended domain $\mathcal{D}'$. As monotonic functions are closed under function composition, the entire system model also yields a monotonic function. Hence, from the Tarski-Knaster theorem [1], it follows that the extensions of all monotonic operators in lattice $\mathcal{D}'$ have fixpoints and, more interestingly, that they have uniquely defined least and a greatest fixpoints. Our framework only needs one half of this theorem, namely the existence and computability of a least fix-point, which requires a complete semi-lattice with an infimum but not necessarily a supremum. Hence, our constructive semantics start with known input variables and local/output variables. If the least fix-point does no longer have unknown values, the program has a unique behavior, it is constructive.

Theorem 1 (Fixpoints in Lattices [26]). *Let $(\mathcal{D}, \sqsubseteq)$ be a complete lattice and $f : \mathcal{D} \to \mathcal{D}$ be a monotonic function. Then, f has fixpoints and the set of fixpoints even has a minimum and a maximum. If f is moreover continuous, then the least fixpoint of f can be computed by the iteration $p_0 := \sqcap\mathcal{D}$, $p_{i+1} := f(p_i)$, and the greatest fixpoint of f by $q_0 := \sqcup\mathcal{D}$, $q_{i+1} := f(q_i)$.*

Constructive Synchronous Guarded Actions. In this article, we represent imperative QUARTZ modules using synchronous guarded actions [27,28], as defined in Figure 3 and in the spirit of *guarded commands* [29,30,31], a well-established formalism for the description of concurrent systems. However, our guarded actions follow the *synchronous abstraction of time.* A reactive system is represented by a set of synchronous guarded actions of the form $\langle\gamma \Rightarrow \alpha\rangle$ defined over a set of variables $\mathcal{V}$. The Boolean condition γ is called the guard and α is called the action of the guarded action. An immediate assignment $x = \tau$ writes the evaluated value of τ immediately to the variable x. A delayed assignment $\mathsf{next}(x) = \tau$ stores it until the next execution step.

$$\begin{array}{llllll} p,q ::= \mathsf{init}(x)=\tau & \text{(initial)} & \mid \gamma \Rightarrow x=\tau & \text{(immediate)} & \mid p \parallel q & \text{(compose)}\\ \quad\mid \gamma\Rightarrow \mathsf{next}(x)=\tau & \text{(delayed)} & \mid \mathsf{var}\, x\colon p\,\mathsf{default}\, v & \text{(block)} & \mid \mathsf{done}\ p & \text{(done)} \end{array}$$

Fig. 3. Synchronous Guarded Actions

Immediate assignments define a causal dependency within the instant from all the variables read (i. e. variables in the guard γ and on the right-hand side τ) to the written variable x. In contrast, a delayed assignment does not, because they set a values for the following instant. Guarded actions are composed by the operator $p \| q$ and grouped by the operator $\mathsf{var}\, x\colon p\, \mathsf{default}\, v$ to locate all actions defining a variable x. If none of these guarded actions apply, x is set to its *default value* v. Immediate variables $\mathcal{E} \subseteq \mathcal{V}$ are reset to their default values (like wires in hardware circuits), while delayed variables $\mathcal{M} \subseteq \mathcal{V}$ are reset to their previous value (like registers in hardware circuits).

As guarded actions manipulate signal (timed) values, we have to keep track of the status and value of each signal. To this end, we use a store $s \in S = X \rightarrow \mathcal{D}'$, defined by a function from signal names to status, to evaluate the program expression ϕ with respect to the values stored in s as $s, \phi \rightharpoonup v$. For $\star \in \{\mathsf{and}, \mathsf{or}\}$, we assume that $s, x \star y \rightharpoonup v$ is defined iff $v = s(x) \star s(y) \in \mathbb{B}$ and, for $\star \in \{\mathsf{not}, \mathsf{id}\}$, that $s, \star x \rightharpoonup v$ iff $v = \star s(x) \in \mathbb{B}$. Notice that the relation $s, \phi \rightharpoonup v$ evaluates ϕ to the value $v \in \mathcal{D}$ only if all its free variables are defined on $\mathcal{D}$ in s: it is a synchronous expression (which explains why $\bot$ and $\top$ will not be needed in Figure 4). Additionally, an update operation $\uplus$ sets the status of x in s as follows: $s \uplus (x, v) = s \cup \{(x, \sup(s(x), v))\}$. This definition sets the status of x to $\lightning$ if a status v is assigned that conflicts with that stored in s.

Figure 4 defines transition rules $s, p \rightharpoonup s', q$ for synchronous guarded actions (we assume $v, w, u \in \mathcal{D}$). If the guard γ of an immediate action $\gamma \Rightarrow x = \tau$ evaluates to 1 and its action τ to a value $v \in \mathcal{D}$, the store s is updated with $s \uplus (x, v)$. In the case of a delayed action, the additional initial action $\mathsf{init}(x) = v$ is added to be applied in the following instant. The rule for composition $p \| q$ defines the possible evaluation schedules of concurrent statements. The marker done indicates that a guarded action has been executed and is propagated by composition and blocks $\mathsf{var}\, x\colon p\, \mathsf{default}\, v$, allowing one to reset the default value of a variable x in that block.

$$\frac{s' = (s(x) \in \mathcal{D})?s, s \uplus (x,v) \quad w = (x \in \mathcal{M})?s'(x), v}{s, \mathsf{var}\, x\colon \mathsf{done}\ (p)\ \mathsf{default}\, v \rightharpoonup s', \mathsf{done}\ (\mathsf{var}\, x\colon p\, \mathsf{default}\, w)} \qquad \frac{s,p \rightharpoonup s',p'}{s,\ q\|p \rightharpoonup s',\ q\|p'}$$

$$\frac{s,\gamma \rightharpoonup 1 \quad s,\tau \rightharpoonup v}{s,\gamma \Rightarrow \mathsf{next}(x)=\tau \rightharpoonup s, \mathsf{done}\ (\mathsf{init}(x) = v \,\|\, \gamma \Rightarrow \mathsf{next}(x)=\tau)} \qquad \frac{s,p\rightharpoonup s',p'}{s,\ p\|q \rightharpoonup s',\ p'\|q}$$

$$\frac{s,\gamma\rightharpoonup 1\quad s,\tau\rightharpoonup v}{s,\gamma\Rightarrow x=\tau \rightharpoonup s\uplus(x,v), \mathsf{done}\ (\gamma\Rightarrow x=\tau)} \qquad \frac{}{s,\ \mathsf{done}\ p \,\|\, \mathsf{done}\ q \rightharpoonup s, \mathsf{done}\ (p\|q)}$$

$$\frac{s,\gamma\rightharpoonup 0}{s,\gamma\Rightarrow \mathsf{next}(x)=\tau\rightharpoonup s,\mathsf{done}\ (\gamma\Rightarrow\mathsf{next}(x)=\tau)}$$

$$\frac{s,\gamma\rightharpoonup 0}{s,\gamma\Rightarrow x=\tau\rightharpoonup s,\mathsf{done}\ (\gamma\Rightarrow x=\tau)} \qquad \frac{}{s,\ \mathsf{init}(x)=v \rightharpoonup s\uplus(x,v), ()}$$

Fig. 4. Rules for Synchronous Guarded Actions

3 Polychronous Systems

In contrast to synchronous systems, polychronous specifications [32,16] are based on a partially ordered model of time to express asynchronous computations which possibly need to synchronize sporadically. As the name suggests, polychronous systems use multiple clocks, which means that signals do not need to be present in all instants. Here, the clock of a signal is encoded by its status, present or absent. The value of a signal can only be computed if it is known to be present.

Signal Specifications. In the remainder, we use Signal programs to represent polychronous specifications. A Signal program, Figure 5, consists of the composition of several nodes. Each node has an interface consisting of input and output signals and, possibly, local signals. Its body is the composition of equations built from primitive $\star \in \{\mathsf{and}, \mathsf{or}, \mathsf{not}, \$\mathsf{init}, \mathsf{when}, \mathsf{default}\}$ operators.

$$\begin{aligned} p,q,r &::= \cdots \\ &\mid x := y \star z \text{ (equation)} \end{aligned} \qquad \frac{s(x,y,z) \rightharpoonup_\star (a,b,c)}{s,\ x := y \star z \rightharpoonup s \uplus (x,a)(y,b), (z,c),\ x := y \star z}\ \text{(op)}$$

$$\frac{s(x,y), a \rightharpoonup_{\$\mathsf{init}} (b,c,d)}{s,\ x := y\, \$\mathsf{init}\, a \rightharpoonup s \uplus (x,b)(y,c),\ x := y\, \$\mathsf{init}\, d}\ \text{(pre)}$$

Fig. 5. Small-step operational semantics of polychronous equations

Figure 5 gives the main transition rule for a polychronous equation $x := y \star z$. It relies on the relation $\rightharpoonup_\star$ to check progress from the current status $s(x,y,z)$ of its inputs and output to an hypothetical triple (a,b,c). If so, a transition occurs and the status of (x,y,z) signals is updated. Let us first consider the case of a functional equation such as $x := y\, \mathsf{and}\, z$, Figure 6. From an initial status $s(x,y,z)$, there are three possible ways to progress. First, (1) is when one of the inputs or the outputs is known to be absent, and the others are either unknown or absent (written $?/\bot$). In that case, all three parameters can be deemed absent altogether (right) as they need to occur synchronously. As a result, information on absence may flow backwards from outputs to inputs and possibly inhibits further signals in the environment. A second case is (2) when one of the inputs or the outputs is known to be present, and others either unknown or present (written $?/\top$). In this case all three parameters can be deemed present (right). As a result information on presence may again flow from outputs to inputs and possibly trigger further input signals in the environment. At last, (3), the values $a, b \in \mathbb{B}$ of inputs are known and the result $a \wedge b$ of the equation is computed.

$\rightharpoonup_{\mathsf{and}}$						
$s(x)$	$s(y)$	$s(z)$	a	b	c	
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	
$?/\bot$	$\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	
$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$	(1)
$\top$	$?/\top$	$?/\top$	$\top$	$\top$	$\top$	
$?/\top$	$\top$	$?/\top$	$\top$	$\top$	$\top$	(2)
$?/\top$	$?/\top$	$\top$	$\top$	$\top$	$\top$	
$?/\top$	a	$?/\top$	$\top$	a	$\top$	(3)
$?/\top$	$?/\top$	a	$\top$	$\top$	a	
$?/\top$	b	a	$a \wedge b$	b	a	

$\rightharpoonup_{\mathsf{when}}$						
$s(x)$	$s(y)$	$s(z)$	a	b	c	
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	
$?/\bot$	$\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	(1)
$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$	
$?/\bot$	x	0	$\bot$	x	0	
$\top$	$?/\top$	a	$\top$	$\top$	$a \sqcup 1$	(2)
$?/\top$	a	1	a	a	1	

Fig. 6. Small-step relations of polychronous equations

The same evaluation principle can be applied to the downsampling operator $x := y\,\mathsf{when}\,z$, Figure 6, the rules are the same for propagating absence (1) and there is only one possible way to propagate presence: when that of the output is already known, for any positive progress $a, b \in \{?, \top, 0, 1\}$ of the inputs. When $z = 0$, we "don't care" the first input: $x \in \mathcal{D}'$, and issue the output as absent. Again, there is only one way to propagate presence (2): when that of the output is already known and for any positive progress $a \in \{?, \top, 1\}$ of the inputs. Also, there is only one case where the output can possibly have a value $a \in \mathbb{B}$, when $z = 1$ (supremum $a \sqcup 1$).

$\rightharpoonup_{\mathsf{default}}$						
$s(x)$	$s(y)$	$s(z)$	a	b	c	
$\bot$	$?/\bot$	$?/\bot$	$\bot$	$\bot$	$\bot$	(1)
$?/\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	
$?$	$\top$	x	$\top$	$\top$	x	(2)
$?$	x	$\top$	$\top$	x	$\top$	
$?/\top$	a	x	a	a	x	(3)
$?/\top$	$\bot$	a	a	$\bot$	a	

$\rightharpoonup_{\mathsf{\$init}}$						
$s(x)$	$s(y)$	a	b	c	d	
$\bot$	$?/\bot$	a	$\bot$	$\bot$	a	(1)
$?/\bot$	$\bot$	a	$\bot$	$\bot$	a	
$\top$	$?/\top$	a	a	$\top$	a	(2)
$?/\top$	$\top$	a	a	$\top$	a	
$?/\bot$	b	a	a	b	b	(3)
a	b	a	a	b	b	

Fig. 7. Small-step relations of polychronous equations

The merge operator $x := y\,\mathsf{default}\,z$ works in a way opposite to sampling, Figure 6, there is only one way its result (or its inputs) can be ruled absent (1). There are many ways merge can make positive progress from the knowledge of either of its inputs, "regardless" of the (don't care) value $x \in \mathcal{D}'$ of the other (2). Finally, merge gives a value $a \in \mathcal{D}$ to its output if y holds a or when it is absent and z does (3). The case of the delay equation $x := y\,\mathsf{\$init}\,a$ is a little trickier and first requires a specific rule (pre) to take care of the fact that its third argument, the state, is a value $a, d \in \mathcal{D}$. Apart from that, it mainly acts as a synchronous operation between its input and output signals. Once again, absence can be propagated both ways in the data-flow to either inhibit or trigger signals in the environment (1). However, as soon as one of the input or output is known to be present, delay forwards its stored value $a \in \mathcal{D}$ to the output (2). Last, once the input value $b \in \mathcal{D}$ has been calculated (3), it can be stored in place of the old one (here a, the output).

Example. We consider the specification of a polychronous counter of input n and output o. Every time its execution is triggered, its purpose is to provide the value of a count along with its output signal o. When that count reaches 0, the counter synchronizes with the input signal n to reset the count with a new value. Clock synchronization $n \,\mathsf{sync}\, y$ is rewritten by an equation $(z := ((y = y) = (n = n)))/z$ that forces y and n to have the same status.

$$\begin{array}{l} \mathit{counter}(n, o) \overset{\Delta}{=} \\ \qquad (c := o \,\mathsf{\$init}\, 0 \,\|\, o := n \,\mathsf{default}\, x \,\|\, x := (c - 1) \,\|\, n \,\mathsf{sync}\, y \,\|\, y := 1 \,\mathsf{when}\, (c = 0)))\, /cxy \end{array}$$

In the remainder, and referring to both a Quartz module or a Signal process $\langle s, p \rangle$, we note $V(p)$ for all signals of p, $O(p)$ for its output signals, $I(p)$ for its inputs and $L(p)$ for its locals. The counter has input n, output o and locals $\{c, x, y\}$. The local signals c defines the current count, x its decrement, and y the reset condition. The synchronous execution of the counter is modeled by a series of steps (changes are marked with $\mathrm{a}_\bullet$). First, the environment of the counter triggers execution by setting the output signal o to present ("I want a count"). This forces the current count c evaluate ($\rightharpoonup_{\mathsf{\$init}}$). Then, its decrement x and the reset condition y can both be evaluated ($\rightharpoonup_{sub}$ and $\rightharpoonup_{eq}$). Next, the status of the input n can be defined from that of y ($\rightharpoonup_{\mathsf{sync}}$). Since it is absent and c is present with the count, the output o can now be output ($\rightharpoonup_{\mathsf{default}}$) and its value is stored in place of the previous one ($\rightharpoonup_{\mathsf{\$init}}$). As the example shows, we obtain a constructive and executable operational semantics that captures the behavior of synchronous Quartz modules, as well as of polychronous Signal processes and for the first time in the literature.

$$\begin{array}{rlllll l} (c,?)(n,?)(o,\top_\bullet)(x,?)(y,?) \rightharpoonup & (c,1_\bullet) & (n,?) & (o,\top) & (x,?) & (y,?) & \text{from } \rightharpoonup_{\mathsf{\$init}} \\ \rightharpoonup & (c,1) & (n,?) & (o,\top) & (x,0_\bullet) & (y,?) & \text{from } \rightharpoonup_{\mathsf{sub}} \\ \rightharpoonup & (c,1) & (n,?) & (o,\top) & (x,0) & (y,\bot_\bullet) & \text{from } \rightharpoonup_{\mathsf{when}} \\ \rightharpoonup & (c,1) & (n,\bot_\bullet) & (o,\top) & (x,0) & (y,\bot) & \text{from } \rightharpoonup_{\mathsf{sync}} \\ \rightharpoonup & (c,1) & (n,\bot) & (o,0_\bullet) & (x,0) & (y,\bot) & \text{from } \rightharpoonup_{\mathsf{default}} \\ \rightharpoonup & (c,1) & (n,\bot) & (o,0) & (x,0) & (y,\bot) & \text{from } \rightharpoonup_{\mathsf{\$init}} \end{array}$$

From Synchrony to Asynchrony. Our next step is to embed this semantics in its execution environment of asynchronous streams in order to reason about networked processes. Towards this goal, we first need to interface the small step operational semantics with an environment of asynchronous streams. We represent a stream by a word $w \in \mathbb{S} = \mathcal{D}^*$ of values a and define the operation of reading a from a stream as $a.w$ and writing onto it as $w.a$ in order to reflect a first-in-first-out protocol. The environment or trace of a process p in a network is represented by a finite map $E \in \mathbb{T} = X \mapsto \mathbb{S}$ that associates signal names x

and streams w. Since a module p owns a local store s, we shall note $\langle s, p\rangle$ its embedding in a network P constructed by asynchronous composition.

$$w ::= \epsilon \mid a.w \mid w.a \quad \text{(FIFO)} \qquad P, Q ::= \langle s, p\rangle \mid P \parallel Q \quad \text{(network)}$$

Then, we simply lift the transition relation $s, p \rightharpoonup t, q$ to further account for the way a process p interacts with asynchronous environment and construct the small-step semantics $E, P \rightharpoonup F, Q$ to define this interaction between local, synchronous, steps of execution and shared, asynchronous, FIFO streams, i.e.,

$$\frac{s, p \rightharpoonup t, q}{E, \langle s, p\rangle \rightharpoonup E, \langle t, q\rangle}.$$

Interfacing Polychronous Processes. The execution of a SIGNAL process p is locally triggered by activating the status of one or several of its signals, by setting them present (Figure 8, top-left). We call these signals $T(p)$ – the triggers of p. Once a trigger is enabled, other signals can be and, when an input x is, its value can be loaded from the environment E (right). The output of a process onto streams is performed when computation within the process is completed (Figure 8, bottom) in order to respect the synchronous step paradigm. The "flush" relation $E, \langle s, p\rangle \rightharpoondown F, \langle s, p\rangle$ models this very step. It is defined by pulling computed output values from the local store to the shared streams. It delimits the temporal barriers of a synchronous instant or reaction and filters inputs which have been read and signals which are absent.

$$\frac{x \in T(p)}{E, \langle s \uplus (x, ?), p\rangle \rightharpoonup E, \langle s \uplus (x, \top), p\rangle} \qquad \frac{x \in I(p)}{E \uplus (x, a.w), \langle s \uplus (x, \top), p\rangle \rightharpoonup E \uplus (x, w), \langle s \uplus (x, a), p\rangle}$$

$$\frac{x \notin O(p) \vee a \notin \mathcal{D}}{E, \langle s \uplus (x, a), p\rangle \rightharpoondown E, \langle s \uplus (x, ?), p\rangle} \qquad \frac{x \in O(p) \wedge a \in \mathcal{D}}{E \uplus (x, w), \langle s \uplus (x, a), p\rangle \rightharpoondown E \uplus (x, w.a), \langle s \uplus (x, ?), p\rangle}$$

Fig. 8. Interface semantics of polychronous equations p

Interfacing Synchronous Modules. The asynchronous interface of a synchronous QUARTZ module is simpler. It only requires rules for the $\rightharpoondown$ relation to handle writing the computed output values and reading the next values of inputs (Figure 9, s_x stands for s without x).

$$\begin{array}{rl}
E \uplus (x, w), \langle s \uplus (x, a), p\rangle \rightharpoondown E \uplus (x, w.a), \langle s_x \uplus (x, ?), p\rangle & x \in O(p) \\
E \uplus (x, a.w), \langle s, p\rangle \rightharpoondown E \uplus (x, w), \langle s_x \uplus (x, a), p\rangle & x \in I(p) \\
E, \langle s, p\rangle \rightharpoondown E, \langle s_x \uplus (x, ?), p\rangle & x \in L(p) \\
E, \langle s, \mathsf{done}\ p\rangle \rightharpoondown E, \langle s, p\rangle &
\end{array}$$

Fig. 9. Interface semantics of synchronous modules p

$$\frac{E, P \rightharpoonup F, Q}{E, P \rightarrow E, Q} \quad \frac{E, \langle s, p\rangle \rightharpoondown^* F, \langle s', q\rangle}{E, \langle s, p\rangle \rightarrow F, \langle s', q\rangle} \quad \frac{E, P \rightarrow E', P'}{E, P \parallel Q \rightarrow E', P' \parallel Q} \quad \frac{E, P \rightarrow E', P'}{E, P \parallel Q \rightarrow E', P' \parallel Q}$$

Fig. 10. Interface semantics of synchronous modules p

A Constructive GALS Semantics. Finally, the GALS semantics $E, P \rightarrow E, Q$, Figure 10, can be defined for synchronous modules and polychronous processes. It comprises of local synchronous execution steps $E, P \rightharpoonup E, Q$ (left) during which inputs are read, outputs are computed, and a synchronization step $E, P \rightharpoondown^* F, Q$ (middle), during which outputs are registered in the environment. and the local store is reset to start a new step of execution. Interleaving or scheduling is again defined by parallel composition (right).

Communication from a process to another is assumed to be point-to-point. Multiple writers on a single stream are not allowed as per the synchronous paradigm. Broadcast communication can be simulated by multiplexing the output of the writer to multiple readers: for any $x \in dom(E)$ such that $x \in O(P)$ and $x \in I(Q)$ and $x \in I(R)$, we can rewrite $P \parallel Q \parallel R$ as $P \parallel (Q[y/x] \parallel R[z/x] \parallel \langle(), y := x \,\|\, z := x\rangle) \,/yz$ using any y, z not in Q, R.

4 Determinism and Constructivity

The layered constructive semantics allows us to formulate classical properties of polychronous processes, as stated in the trace or logical settings of [33], yet in a constructive operational semantics framework. We first state the property of reactivity pertaining to the correctness of Quartz programs. A synchronous module p is reactive iff for any combination of input values, its transition function always terminates by producing a combination of output values.

Definition 2 (Reactivity). *A module p is reactive iff, for all valuation $s_0 = \{(x, a) \,|\, x \in I(p), a \in \mathcal{D}\} \cup \{(y, ?) \,|\, y \in V(p) \setminus I(p)\}$, there exists $s_0, p \rightharpoonup^* s, q$ with s defined on $\mathcal{D}$.*

Synchronous determinism relates to reactivity. However, while reactivity assumes that the values of all inputs are known, synchronous determinism also applies to the case of a Signal process, whose inputs may not all be read: a process p is synchronously deterministic iff for any initial status s, its step relation always converges to a unique fixpoint.

Definition 3 (Synchronous Determinism). *A process p is synchronously deterministic iff for all store s and derivations $s, p \rightharpoonup^* t, q$ and $s, p \rightharpoonup^* u, r$ we have $t = u$ and $q = r$.*

Now, once embedded in an asynchronous environment of streams E, determinism becomes endochrony or asynchronous determinism. A process P is asynchronously deterministic iff for every input trace E, it yields a unique output trace F and unique final state Q.

Definition 4 (Asynchronous Determinism). *A process p is asynchronously deterministic iff for all input traces E and derivations $E, \langle s, p\rangle \rightharpoonup^* F, \langle t, q\rangle \rightharpoondown^* G, \langle u, r\rangle$ and $E, \langle s, p\rangle \rightharpoonup^* F', \langle t', q'\rangle \rightharpoondown^* G', \langle u', r'\rangle$, $G = G'$ and $r = r'$.*

Notice that, in the case of a deterministic process, equality $G = G'$ is defined over finite maps between variables and values and that, in $r = r'$, syntactic congruence is taken care of by the rules of synchronous composition i.e. $G = G'$ and $r = r'$ denote canonical configurations).

Constructivity. So far, and thanks to the definition of a complete domain of clocked signals, we have defined the very first, executable, structured operational semantics for the polychronous data-flow language SIGNAL. Its purpose starts to unveil as we consider the fixpoint theoretic implication of its definition on that continuous domain and try to formulate the property of constructivity. Originally, constructivity was defined as a property of an imperative synchronous module that pertains to the reachability of a stable state in its electrical semantics [12]. In the operational setting of the QUARTZ language, this means that, given any combination of input values, a synchronous module p should always define unique output values.

Definition 5 (Synchronous constructivity). *A module p is synchronous constructive iff for all initial valuations $s_0 = \{(x, a) \mid x \in I(p), a \in \mathcal{D}\} \cup \{(y, ?) \mid y \in V(p) \setminus I(p)\}$, $s_0, p \rightharpoonup^* s, q$ and s is defined on $\mathcal{D}$ (i.e. $s = lfp{\rightharpoonup_p}(s_0)$).*

Thanks to the fidelity level of our small-step operational framework, the formulation of constructivity matches that of reactivity and determinism as stated in the previous section.

Proposition 1. *If p is synchronously constructive then p is reactive and synchronously deterministic*

We shall now formulate constructivity in the context of a polychronous process p. However, it is clear from the example of the counter that a polychronous process not necessarily is constructive in the sense as QUARTZ: it may not be reactive.

$$\begin{aligned}&counter(n, o) \overset{\Delta}{=}\\ &\quad (c := o \,\mathsf{\$init}\, 0 \,\|\, o := n \,\mathsf{default}\, x \,\|\, x := (c - 1) \,\|\, n \,\mathsf{sync}\, y \,\|\, y := 1 \,\mathsf{when}\, (c = 0))) \,/cxy\end{aligned}$$

Unlike a QUARTZ module, the counter triggers a sequence of execution steps every time its trigger o is activated. It only loads an input from n when c is 0. Hence, it is not reactive w. r. t. its input signals, but it is reactive w. r. t. its input streams. This observation yields a more general (asynchronous) formulation of constructivity: a process p is asynchronously constructive iff for any combination of values available from its input streams, it always produces an output.

Definition 6 (Asynchronous constructivity). *A process p is asynchronously constructive iff for any environment E of non-empty streams defined on $V(p)$ and $s_0 = \{(x, ?) \mid x \in V(p)\}$, we have $E, \langle s_0, p\rangle \rightharpoonup^* F, \langle s, q\rangle$ with s defined on $\mathcal{D}^\perp$ (i.e. $(F, s) = lfp{\rightharpoonup_p^*}(E, s_0)$).*

Notice that the transitive closure $\rightharpoonup^* F$ of a constructive process always yields a unique valuation on $\mathcal{D}^\perp$ (it is a continuous domain). If a process isn't constructive, it either blocks (e.g. $x = y \,\|\, y = x$) and yields values below $\mathcal{D}^\perp$ or conflicts (e.g. $x = 0 \,\|\, x = 1$) and yields value $\lightning$. Again, asynchronous constructivity corresponds to the property of asynchronous determinism in the logical and denotational frameworks of SIGNAL [33,34].

Proposition 2. *If p is asynchronously constructive then p is asynchronously deterministic*

Case of Isochronous Systems. Definition 6 of asynchronous constructivity is formulated in a way that may seem to coincide with a class of systems which can deterministically be executed starting from a singleton trigger $T(p)$. This case indeed characterizes so called endochronous systems [33] whose input/output signal status can all be decided from that of a single one, "the master clock". A larger class of deterministic systems can be captured by considering processes p that deterministically execute from several, independent, concurrent triggers: so-called weakly endochronous systems [34]. In our framework, a weakly endochronous process p is characterized by a set of multiple, independent triggers $T(p)$, each of them triggers execution of independent computations and yields a confluent state. To allow this, however, we additionally need to allow some of these triggers to possibly (non-deterministically) be chosen to be absent during a given execution step (all computations depending of that trigger would then evaluate to absent). This can be done by choosing the following inhibition rule, instead of the triggering one:

$$\frac{x \in T(p)}{E, \langle s \uplus (x, ?), p\rangle \rightharpoonup E, \langle s \uplus (x, \perp), p\rangle}$$

While a weakly endochronous process may have several triggers, it is additionally required to be stuttering invariant: an inhibited process shall not react to absence.

Definition 7 (Stuttering). *p is stuttering iff for $s = \{(x, \perp) \,|\, x \in T(p)\} \cup \{(x, ?) \,|\, x \in V(p) \setminus T(p)\}$, we have $\langle s, p\rangle \rightharpoonup^* \langle s', p\rangle$ and $s' = \{(x, \perp) \,|\, x \in V(p)\}$*

Notice that Definition 6 accommodates the case of weakly endochronous systems with the above additions of a rule and of a definition for stuttering.

5 Summary

In this article, we defined a constructive operational semantics to unite the synchronous imperative language QUARTZ and the polychronous data-flow language SIGNAL in a common framework. This model is of interest on its own, since it allows us to better understand the relationship between synchrony and polychrony, between constructivity and endochrony. It additionally allows us to model the causality problem as a formal verification problem. We formulated a constructivity theory which captures the behavior of correct synchronous modules and deterministic asynchronous/polychronous networks. Along the way, we provided the very first truly executable operational semantics of SIGNAL.

Acknowledgement. This work is partially supported by INRIA under associate-project Polycore, by the US Air Force Research Laboratory, grant FA8750-11-1-0042, and the German Research Foundation (DFG).

References

1. Berry, G.: The foundations of Esterel. In: Plotkin, G., Stirling, C., Tofte, M. (eds.) Proof, Language and Interaction: Essays in Honour of Robin Milner, pp. 425–454. MIT Press (1998)
2. Schneider, K.: The synchronous programming language Quartz. Internal Report 375, Department of Computer Science, University of Kaiserslautern (December 2009)
3. Halbwachs, N.: A synchronous language at work: the story of Lustre. In: Formal Methods and Models for Codesign (MEMOCODE), pp. 3–11. IEEE Computer Society (2005)
4. Titzer, B., Palsberg, J.: Nonintrusive precision instrumentation of microcontroller software. In: Paek, Y., Gupta, R. (eds.) Languages, Compilers, and Tools for Embedded Systems (LCTES), Chicago, Illinois, USA, pp. 59–68. ACM (2005)
5. Berry, G.: A hardware implementation of pure Esterel. Sadhana 17(1), 95–130 (1992)
6. Rocheteau, F., Halbwachs, N.: Pollux: A Lustre-based hardware design environment. In: Quinton, P., Robert, Y. (eds.) Algorithms and Parallel VLSI Architectures II, Gers, France, pp. 335–346. Elsevier (1992)
7. Schneider, K., Brandt, J., Schuele, T.: A verified compiler for synchronous programs with local declarations. Electronic Notes in Theoretical Computer Science 153, 71–97 (2006)
8. Logothetis, G., Schneider, K.: Exact high level WCET analysis of synchronous programs by symbolic state space exploration. In: Design, Automation and Test in Europe (DATE), pp. 10196–10203. IEEE Computer Society (2003)
9. Boldt, M., Traulsen, C., von Hanxleden, R.: Compilation and worst-case reaction time analysis for multithreaded Esterel processing. EURASIP Journal on Embedded Systems (2008)
10. Stok, L.: False loops through resource sharing. In: International Conference on Computer-Aided Design (ICCAD), pp. 345–348. IEEE Computer Society (1992)
11. Berry, G.: The constructive semantics of pure Esterel (July 1996), http://www-sop.inria.fr/meije/esterel/esterel-eng.html
12. Shiple, T., Berry, G., Touati, H.: Constructive analysis of cyclic circuits. In: European Design Automation Conference (EDAC), Paris, France, pp. 328–333. IEEE Computer Society (1996)
13. Edwards, S.: Making cyclic circuits acyclic. In: Design Automation Conference (DAC), Anaheim, California, USA, pp. 159–162. ACM (2003)
14. Schneider, K., Brandt, J., Schuele, T.: Causality analysis of synchronous programs with delayed actions. In: Compilers, Architecture, and Synthesis for Embedded Systems (CASES), Washington, District of Columbia, USA, pp. 179–189. ACM (2004)
15. Brzozowski, J., Seger, C.J.: Asynchronous Circuits. Springer (1995)
16. Le Guernic, P., Gauthier, T., Le Borgne, M., Le Maire, C.: Programming real-time applications with SIGNAL. Proceedings of the IEEE 79(9), 1321–1336 (1991)

17. Le Guernic, P., Benveniste, A.: Real-time, synchronous, data-flow programming: The language SIGNAL and its mathematical semantics. Research Report 533, INRIA (June 1986)
18. Besnard, L., Gautier, T., Le Guernic, P., Talpin, J.P.: Compilation of polychronous data flow equations. In: Shukla, S., Talpin, J.P. (eds.) Synthesis of Embedded Software – Frameworks and Methodologies for Correctness by Construction. Springer (2010)
19. Kautz, W.: The necessity of closed circuit loops in minimal combinational circuits. IEEE Transactions on Computers (T-C) C-19(2), 162–166 (1970)
20. Rivest, R.: The necessity of feedback in minimal monotone combinational circuits. IEEE Transactions on Computers (T-C) C-26(6), 606–607 (1977)
21. Malik, S.: Analysis of cycle combinational circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (T-CAD) 13(7), 950–956 (1994)
22. Bryant, R.: A switch level model and simulator for MOS digital systems. IEEE Transactions on Computers (T-C) C-33(2), 160–177 (1984)
23. Schneider, K., Brandt, J., Schuele, T., Tuerk, T.: Maximal causality analysis. In: Desel, J., Watanabe, Y. (eds.) Application of Concurrency to System Design (ACSD), Saint-Malo, France, pp. 106–115. IEEE Computer Society (2005)
24. Jose, B., Gamatie, A., Ouy, J., Shukla, S.: SMT based false causal loop detection during code synthesis from polychronous specifications. In: Singh, S. (ed.) Formal Methods and Models for Codesign (MEMOCODE), Cambridge, UK, pp. 109–118. IEEE Computer Society (2011)
25. Nanjundappa, M., Kracht, M., Ouy, J., Shukla, S.: Synthesizing embedded software with safety wrappers through polyhedral analysis in a polychronous framework. In: Electronic System Level Synthesis Conference (ESLsyn), pp. 1–6 (2012)
26. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5(2), 285–309 (1955)
27. Brandt, J., Schneider, K.: Separate translation of synchronous programs to guarded actions. Internal Report 382/11, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany (March 2011)
28. SYNCHRON: The common format of synchronous languages - the declarative code DC. Technical report, C2A-SYNCHRON project (1998)
29. Dijkstra, E.: Guarded commands, nondeterminacy and formal derivation of programs. Communications of the ACM (CACM) 18(8), 453–457 (1975)
30. Chandy, K., Misra, J.: Parallel Program Design. Addison-Wesley (May 1989)
31. Dill, D.: The Murϕ Verification System. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 390–393. Springer, Heidelberg (1996)
32. Gamatié, A., Gautier, T., Le Guernic, P., Talpin, J.: Polychronous design of embedded real-time applications. ACM Transactions on Software Engineering and Methodology (TOSEM) 16(2) (2007)
33. Le Guernic, P., Talpin, J.P., Le Lann, J.C.: Polychrony for system design. Journal of Circuits, Systems, and Computers (JCSC) 12(3), 261–304 (2003)
34. Potop-Butucaru, D., Sorel, Y., de Simone, R., Talpin, J.P.: Correct-by-construction asynchronous implementation of modular synchronous specifications. Fundamenta Informaticae 108(1-2), 91–118 (2011)

On Tolerance Analysis of Games with Belief Revision

Çağıl Taşdemir

The Graduate Center, CUNY, 365 Fifth Avenue
New York, NY 10016 USA
ctasdemir@gc.cuny.edu

Abstract. Aumann's Rationality Theorem claims that in perfect information games, common knowledge of rationality yields backward induction (BI). Stalnaker argued that in the belief revision setting, BI does not follow from Aumann's assumptions. However, as shown by Artemov, if common knowledge of rationality is understood in the robust sense, i.e., if players do not forfeit their knowledge of rationality even hypothetically, then BI follows. A more realistic model would bound the number of hypothetical non-rational moves by player i that can be tolerated without revising the belief in i's rationality on future moves. We show that in the presence of common knowledge of rationality, if n hypothetical non-rational moves by any player are tolerated, then each game of length less than $2n+3$ yields BI, and that this bound on the length of model is tight for each n. In particular, if one error per player is tolerated, i.e., $n = 1$, then games of length up to 4 are BI games, whereas there is a game of length 5 with a non-BI solution.

1 Introduction

Aumann proved that in games of perfect information, common knowledge of rationality yields backward induction [3]. Stalnaker showed that if players are allowed to revise their beliefs in each other's rationality in response to surprising information, this is not the case [10]. In [7], Halpern showed that the difference between the two lies in how they interpret the following counterfactual statement: "If the player were to reach vertex v, then she would be rational at vertex v."

Let us consider the game in Figure 1 which is due to Stalnaker and which Halpern uses to point out the difference in Aumann's and Stalnaker's arguments. Assume that it is common knowledge that the actual state is (dda). This means that Ann plays *down* (d) in vertex v_1, Bob plays *down* (d) in vertex v_2, and Ann plays *across* (a) in vertex v_3, and that all of this is common knowledge between Ann and Bob. To say that some fact F is common knowledge between Ann and Bob means that Ann knows F, Bob knows F, Ann knows that Bob knows F, Bob knows that Ann knows F and so on. So if we assume that the state (dda) is common knowledge, this means that all moves are known right at the beginning of the game. The question is whether (dda), which is different than the backward induction solution (aaa), can be the solution of the game in the presence of common knowledge of rationality.

Here it should also be noted that Stalnaker has no problem with the formal correctness of Aumann's proof. Aumann's framework does not allow belief revision. Stalnaker, on the other hand, allows players to revise their beliefs after a non-rational move by another player, even if that mentioned non-rational move is hypothetical.

Let us look at the game in Figure 1 from Stalnaker's perspective: At (dda), Ann is rational at v_1, because Bob is playing d at v_2. At v_2, Bob revises his belief on Ann's rationality, due to her hypothetical non-rational move (or the surprising information) a at v_1, and considers Ann's playing d at v_3 also possible as a result of this belief revision. He plays d and he is rational. Ann is rational at v_3 by playing a.

While Halpern layed out the differences in Aumann's and Stalnaker's arguments, Artemov in [2] showed that in perfect information games with Stalnaker-style belief revision setting, if players maintain their beliefs in each other's rationality in all, even hypothetical situations, i.e., if there is so-called *robust knowledge of rationality* in the game, then the only solution of the game is the backward induction. That is, if Bob does not revise his beliefs on Ann's rationality at v_2, (dda) cannot be the solution of the game in the presence of common knowledge of rationality.

Other works on epistemic foundations for backward induction include [1], [4], [5], [8] and [9].

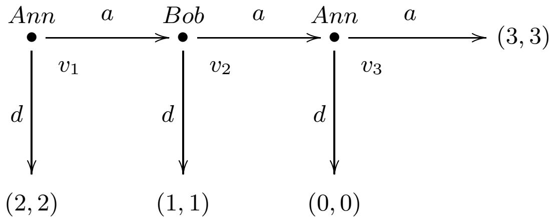


Fig. 1. 3-move game

In this paper, we will investigate the case where it is common knowledge that players are rational (in Stalnaker's sense) at all vertices of the game tree, and they tolerate n hypothetical non-rational moves of other players. If $n = 0$, we end up with Stalnaker's framework where after 1 error, players revise their beliefs.

2 Game Models and Rationality

Halpern extends Aumann models to represent N-player extensive form games with perfect information where players can revise their beliefs [7]. An extended model is a tuple

$$M = (\Omega, K_1, ..., K_N, s, f)$$

where Ω is a set of states of the world, K_i is the information partition of player i, and s maps each state $\omega \in \Omega$ to a strategy profile $s(\omega) = (s_1, ..., s_N)$ where s_i is player i's strategy at state ω. Function f, called *selection function*, maps state-vertex pairs to states. Informally, $f(\omega, v) = \omega'$ means that ω' is the closest state to ω where vertex v is reached. Let $h_i^v(s)$ denote player i's payoff if strategy profile s is played starting at vertex v. Let P be the function that maps non-terminal nodes to players to indicate the player moving at a given node.

Definition 1. Player i is *Aumann-rational*, or A-rational, at vertex v in state ω if for all strategies s^i such that $s^i \neq s_i(\omega)$, $h_i^v(s(\omega')) \geq h_i^v(s_{-i}(\omega'), s^i)$ for some $\omega' \in K_i(\omega)$ where $s_{-i}(\omega')$ denotes the strategy profile of the players other than i at state ω'.

Note that according to this definition, a player is rational as long as her strategy in the current state ω yields her a payoff at least as good as any of her other strategies in *some* state that she considers possible at ω.

Definition 2. Player i is *Stalnaker-rational*, or *S-rational*, at vertex v in state ω if i is A-rational at v in state $f(\omega, v)$.

Substantive rationality is rationality (A-rationality or S-rationality, depending on which framework we are working with) at all vertices of the game tree.

The formalization of selection functions is due to Halpern [7], and the main idea of a selection function f is for each state ω and vertex v to indicate the epistemically closest state $f(\omega, v)$ to ω in which v is reached. Halpern assumes that the selection function f satisfies the following requirements:

- F1. Vertex v is reached in $f(\omega, v)$.
- F2. If v is reached in ω, then $f(\omega, v) = \omega$.
- F3. $s(f(\omega, v))$ and $s(\omega)$ agree on the subtree below v.

3 Tolerating Hypothetical Errors

Our model extends Halpern's so that the selection function now satisfies an additional requirement F4$_n$ (given below) in order to represent the n-tolerance of the players. We will give the definitions of an error-vertex and the condition F4$_n$ simultaneously.

The following definition extends Aumann's rationality to hypothetical moves.

Definition 3. We say that a *move m at v in ω is rational* if player $i = P(v)$ is Aumann-rational at v in some state ω' which has the same profile as some $\widetilde{\omega} \in K_i(\omega)$ except, possibly, for the move m which is plugged into v.

Definition 4. Given a state ω, a vertex v is an *n-error vertex*, if n is the least natural number ≥ 0 such that each player makes not more than n non-rational moves (possibly hypothetical) at vertices v' from the root to v in states $f(\omega, v')$. Obviously, the root vertex is always 0-error. If there are $2k$ moves from the root to v, then each player makes $\leq k$ moves there and v is at most a k-error vertex. Other examples will be discussed later in this section.

The following condition reflects the idea of n-tolerance which is built-in into the selection function: sets of future scenarios are not revised after $\leq n$ (possibly hypothetical) non-rational moves of each player.

Condition F4$_n$. For each state ω and k-error vertex v with $k \leq n$ and $i = P(v)$, if $\omega' \in K_i(f(\omega, v))$, then there exists a state $\omega'' \in K_i(\omega)$ such that $s(\omega')$ and $s(\omega'')$ agree on the subtree below v.

Halpern uses a similar condition to model Aumann's framework, which says that players consider at least as many strategies possible at ω as at $f(\omega, v)$; and this applies to all vertices in the game tree. Our condition F4$_n$ says the same thing for $\leq n$-error vertices, hence limiting the tolerance level in the game to n (possibly hypothetical) non-rational moves per player. In other words, in an n-tolerance game, players will not revise their beliefs about rationality for the first n hypothetical non-rational moves of those players.

Example 1. Consider the game in Figure 1. The following extended model is from [7]

The strategy profiles are as follows:

- $s^1 = (dda)$
- $s^2 = (ada)$
- $s^3 = (add)$
- $s^4 = (aaa)$: this is the BI solution.
- $s^5 = (aad)$

The extended model is $M_1 = (\Omega, K_{Ann}, K_{Bob}, s, f)$ where

- $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$
- $K_{Ann} = \{\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4\}, \{\omega_5\}\}$
- $K_{Bob} = \{\{\omega_1\}, \{\omega_2, \omega_3\}, \{\omega_4\}, \{\omega_5\}\}$
- $s(\omega_j) = s^j$ for $j = 1 - 5$
- $f(\omega_1, v_2) = \omega_2$, $f(\omega_1, v_3) = \omega_4$, $f(\omega_2, v_3) = \omega_4$, $f(\omega_3, v_3) = \omega_5$, and $f(\omega, v) = \omega$ for all other ω and v.

It is assumed that the actual state is ω_1 with $s(\omega_1) = (dda)$, and this is commonly known to players. Let us check which vertices are erroneous:

- v_1 is a 0-error vertex.
- v_2 is a 1-error vertex since Ann's move from v_1 to v_2 is not rational in ω_1. She only considers (dda) possible at this node and changing the move at v_1 to a results in (ada), which would make Ann non-rational at v_1.
- v_3 is a 1-error vertex, by an easy combinatorial argument. Ann was not rational at v_1, so v_3 is at least 1-error vertex. However, it is at most 1-error, since the move at v_2 is made by Bob, and it cannot change the maximum of error counts at v_3. However, let us check that Bob is rational in moving from v_2 to v_3 at $f(\omega_1, v_2) = \omega_2 = ada$. In (ada), Bob considers both (ada) and

(add) possible, and plugging the hypothetical move a into vertex v_2 would result in strategy profile (aaa) that corresponds to state ω_4 in which Bob is rational at v_2.

4 Belief Revision with Tolerance

Example 2. Let us consider the game in Figure 1 again. Note that in this game, with the model M_1, in the presence of common knowledge of substantive rationality the realized strategy profile, i.e., (dda), is different than the backward induction solution (aaa). We will also assume common knowledge of substantive rationality, and show that (dda) cannot be the solution of the 1-tolerant version of this game.

Since players are 1-tolerant, the selection function f should satisfy the condition F4_n with $n = 1$. This means that the first hypothetical error for each player is tolerated. In particular, even if Ann and Bob make one hypothetical error each, those will be tolerated and beliefs in rationality will not be revised.

Therefore in a 1-tolerance game, if we assume that the state (dda) is common knowledge, we need to consider only three strategy profiles:

- $s^1 = (dda)$: This is the original strategy profile which is commonly known.
- $s^2 = (ada)$: This is the revised state at v_2.
- $s^3 = (aaa)$: This is the revised state at v_3.

The extended 1-tolerance game model is $M_2 = (\Omega, K_{Ann}, K_{Bob}, s, f)$ where

- $\Omega = \{\omega_1, \omega_2, \omega_3\}$
- $K_{Ann} = K_{Bob} = \{\{\omega_1\}, \{\omega_2\}, \{\omega_3\}\}$
- $s(\omega_j) = s^j$ for $j = 1 - 3$
- $f(\omega_1, v_2) = \omega_2$, $f(\omega_1, v_3) = \omega_3$, $f(\omega_2, v_3) = \omega_3$.

The actual state is ω_1 with $s(\omega_1) = (dda)$. Let us count the number of errors in this model.

- v_1 is 0-error.
- v_2 is 1-error, since in order to (hypothetically) get from v_1 to v_2, Ann has to make a non-rational move, by the same reasoning as in Example 1.
- v_3 is again 1-error by trivial combinatorial reasons, as before. Moreover, Bob's move from v_2 to v_3 is rational at ω_2, by the same reasoning as before.

Condition F4_1 is obviously met, so this is a 1-tolerant model in which strategy profile (dda) is common knowledge. We'll see, however, that the substantive S-rationality condition is violated in this model, namely, Bob is not rational at v_2. Indeed, S-rationality in ω_1 at v_2 reduces to (Aumann-)rationality in $f(\omega_1, v_2)$ at v_2, i.e., in ω_2 at v_2. Since $s(\omega_2) = (ada)$, the real move at v_2 is *down* which is not rational because of the better alternative *across*.

Example 3. Figure 2 shows an extensive form 1-tolerance game of length 5. Assuming common knowledge of substantive rationality, we will show that there exists a non-BI solution, namely ($dddda$).

The strategy profiles are as follows:

$$\begin{tikzcd}
\overset{Ann}{\bullet} \arrow[r, "a", "v_1"'] \arrow[d, "d"'] & \overset{Bob}{\bullet} \arrow[r, "a", "v_2"'] \arrow[d, "d"'] & \overset{Ann}{\bullet} \arrow[r, "a", "v_3"'] \arrow[d, "d"'] & \overset{Bob}{\bullet} \arrow[r, "a", "v_4"'] \arrow[d, "d"'] & \overset{Ann}{\bullet} \arrow[r, "a", "v_5"'] \arrow[d, "d"'] & (5,5) \\
(4,4) & (3,3) & (2,2) & (1,1) & (0,0) &
\end{tikzcd}$$

Fig. 2. 5-move game

- s^1 is the strategy profile $(dddda)$
- s^2 is the strategy profile $(addda)$
- s^3 is the strategy profile $(aadda)$
- s^4 is the strategy profile $(aaada)$
- s^5 is the strategy profile $(aaaaa)$
- s^6 is the strategy profile $(aaaad)$
- s^7 is the strategy profile $(aaadd)$.

Consider the extended model $A = (\Omega, K_{Ann}, K_{Bob}, s, f)$ where

- $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7\}$
- $K_{Ann} = \{\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4\}, \{\omega_5\}, \{\omega_6\}, \{\omega_7\}\}$
- $K_{Bob} = \{\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4, \omega_7\}, \{\omega_5\}, \{\omega_6\}\}$
- $s(\omega_j) = s^j$ for $j = 1 - 7$
- $f(\omega_1, v_2) = \omega_2,\ f(\omega_1, v_3) = \omega_3,\ f(\omega_1, v_4) = \omega_4,\ f(\omega_1, v_5) = \omega_5,$
  $f(\omega_2, v_3) = \omega_3,\ f(\omega_2, v_4) = \omega_4,\ f(\omega_2, v_5) = \omega_5,$
  $f(\omega_3, v_4) = \omega_4,\ f(\omega_3, v_5) = \omega_5,$
  $f(\omega_4, v_5) = \omega_5,$
  $f(\omega_7, v_5) = \omega_6.$

The actual state is $(dddda)$. Let us count the number of errors.

- v_1 is 0-error.
- Ann is not rational moving from v_1 to v_2 in state $f(\omega_1, v_2) = \omega_2$. Indeed, $s(\omega_2) = (addda)$, hence Ann moves *across* at v_1 while knowing that Bob will play *down* at v_2. Hence v_2 is 1-error.
- Bob, is not rational when moving from v_2 to v_3 in state $f(\omega_1, v_3) = \omega_3$. Indeed, $s(\omega_3) = (aadda)$, hence Bob moves *across* at v_2 while knowing that Ann will play *down* at v_3. Hence v_3 is 1-error by both Ann's and Bob's accounts.
- Ann is not rational moving from v_3 to v_4 in state $f(\omega_1, v_4) = \omega_4$. Indeed, $s(\omega_4) = (aaada)$, hence Ann moves *across* at v_3 while knowing that Bob will play *down* at v_4. Hence v_4 is 2-error on Ann's account.
- v_5 is 2-error by trivial combinatorial reasons. However, it is worth mentioning that Bob is rational when moving *across* from v_4 to v_5.

To secure 1-tolerance, we have to check the conclusion of $\mathrm{F4}_1$ at all 0-error and 1-error vertices, in this case at vertices v_1, v_2, and v_3 which is quite straightforward. Indeed, selection function f does not add new indistinguishable states at these vertices, but just makes the corresponding vertex accessible in the revised state.

Since $K_{Ann}(\omega_1) = K_{Bob}(\omega_1) = \{\omega_1\}$, everything that is true at ω_1 will be common knowledge to Ann and Bob at that state. To check substantive rationality at ω_1, we need to check players' rationality in the following situations:
$S = \{(\omega_1, v_1), (\omega_2, v_2), (\omega_3, v_3), (\omega_4, v_4), (\omega_5, v_5)\}$

– Ann is rational at (ω_1, v_1). Since Bob plays d at vertex v_2, d is the rational move for Ann at (ω_1, v_1).
– Bob is rational at (ω_2, v_2). At (ω_2, v_2), Bob thinks Ann was not rational at v_1 but since we assume that each player tolerates one error, he does not revise his beliefs on her future rationality. So he looks at node v_3. Seeing that Ann is playing d at that node, he himself chooses to play d at v_2, which is the rational thing to do. So we can conclude that Bob is rational at (ω_2, v_2).
– Ann is rational at (ω_3, v_3). At (ω_3, v_3), Ann thinks Bob was not rational at node v_2. This time Ann tolerates Bob's error and does not revise her beliefs about his future rationality. She looks at node v_4. Seeing that Bob is playing d at that node, she chooses to play d at v_3, which is the rational thing to do.
– Bob is rational at (ω_4, v_4). At (ω_4, v_4), Bob thinks that Ann was not rational at node v_3. Since he will not tolerate one more error, he revises his beliefs and takes into the account the possibility of Ann's playing d at v_5. In this case, it is rational for him to play d.
– Ann is rational at (ω_5, v_5) regardless of her beliefs about Bob.

If we count the length of the game as the number of moves in its longest path in the game tree, this example shows that, assuming common knowledge of rationality and 1-tolerance, there exists a game of length 5, where a non-BI solution is realized.

Theorem 1. *In perfect information games with common knowledge of rationality and of n-tolerance, each game of length less than $2n + 3$ yields BI.*

Sketch of Proof: Let $m \leq 2n + 2$. We will show that all m-tolerant games are BI-games. At a vertex that at which the last move of a given path is made (such vertex is reachable from the root in $\leq 2n + 1$), Aumann-rationality yields the move that is dictated by the backward induction solution. Any other vertex v is reachable from the root in $\leq 2n$ steps. So there are at most $2n$ previous nodes prior to reaching v. Since no player makes two moves in a row, each player makes at most n moves prior to v, and even if all of these moves were erroneous, player $i = P(v)$ will tolerate them and do not revise his assumption of the common belief of rationality till the end of the game. By Artemov's argument in [2], this yields BI solution for the rest of the game.

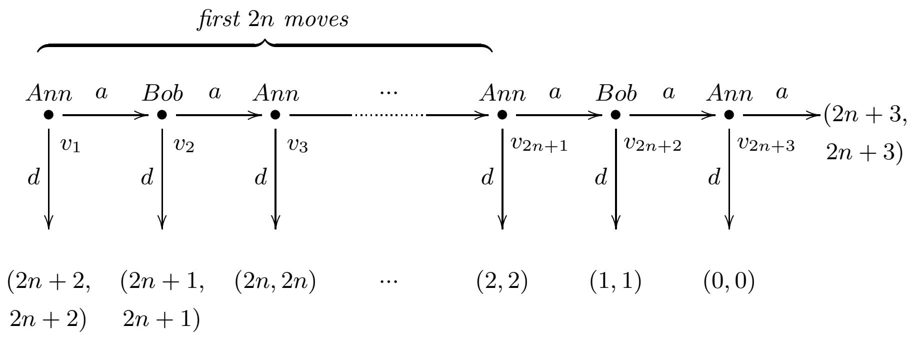
Theorem 2. *The upper bound $2n+3$ from Theorem 1 is tight. Namely, for each n, there exists a perfect information game with common knowledge of rationality and of n-tolerance of length $2n+3$ which does not yield BI.*

Proof: Consider the straightforward generalization of Example 3 (which has length $5 = 2 \times 1 + 3$, i.e., corresponds to $n = 1$) to an arbitrary n in Figure 3. In particular, the profile

$$(dd \dots da)$$

is assumed to be commonly known and players to be n-tolerant.

The same reasoning as in Example 3 shows that this profile $(dd \dots da)$ is both rational and not BI. Since the strategy profile $(dd \dots da)$ is commonly known, and players are n-tolerant, Ann and Bob will not revise their beliefs in each other's rationality during the first $2n$ moves. The move at v_{2n+1} belongs to Ann. She is playing d according to the strategy profile $(dd \dots da)$ and she is rational (because Bob is playing d at v_{2n+2}). However, in order to decide whether Bob is rational at v_{2n+2}, we need to take into account Ann's hypothetical non-rational moves a to reach v_{2n+2}, and there are $n+1$ such moves. Therefore Bob may revise his beliefs on her rationality, consider Ann's playing d possible at v_{2n+3}, and this makes Bob's move d at v_{2n+2} rational. At the very last vertex, Ann is also rational since she plays a.



first 2n moves
Ann a Bob a Ann ··· Ann a Bob a Ann a
v1 v2 v3 v2n+1 v2n+2 v2n+3
d d d d d d
(2n + 3, 2n + 3)
(2n + 2, 2n + 2) (2n + 1, 2n + 1) (2n, 2n) ··· (2, 2) (1, 1) (0, 0)

Fig. 3. Game of length $2n + 3$

5 Conclusion and Future Work

We see a conceptual contribution of this work in stressing the role of tolerance in the analysis of perfect information games in the belief revision setting that accommodates both Aumann's and Stalnaker's paradigms. Stalnaker's players are zero-tolerant and give up their "knowledge of rationality" in hypothetical reasoning after the first hypothetical non-rational move of other players. Aumann's players are infinitely tolerant, and never give up their knowledge of rationality even when confronted, hypothetically, with strong evidence of the contrary. A natural problem of what happens in between, when the level of tolerance to hypothetical errors is a parameter of the game is addressed is this paper. Our findings indicate that for a given tolerance level n, short games, up to length $2n+2$ are Aumann's games, i.e., yield backward induction solutions only. Longer games of length $2n+3$ and greater can show Stalnaker's behavior based on the revision of player's belief of each other's rationality.

What does it say about games with human players who can be tolerant to some limited degree? One more parameter intervenes here: the nested epistemic depth of reasoning, which is remarkably limited for humans ([6]) to small numbers like one – two. In order to calculate BI, players have to possess the power of nested epistemic reasoning of the order of the length of the game. So, realistically, the BI analysis of human players applies to rather short games, say, of length three – four. According to Theorem 1, assuming 1-tolerance of players (which we regard as a meaningful assumption for humans) the only solution is backward induction.

In this game, with the given definition of rationality of hypothetical moves and common knowledge of a non-BI strategy profile as the actual state, we see no way to interpret the hypothetical errors as a move (signal) where the player is trying to reach the pareto-optimal payoff pair, which is the BI solution. A next logical step could be to look into this direction.

Acknowledgments. The author is deeply indebted to Sergei Artemov for his guidance, stimulating suggestions and encouragement.

References

1. Asheim, G.B.: On the epistemic foundation for backward induction. Mathematical Social Sciences 44, 121–144 (2002)
2. Artemov, S.: Robust knowledge and rationality. Technical Report TR-2010010, CUNY Ph.D. Program in Computer Science (2010)
3. Aumann, R.: Backward induction and common knowledge of rationality. Games and Economic Behavior 8, 6–19 (1995)
4. Baltag, A., Smets, S., Zvesper, J.A.: Keep "hoping" for rationality: A solution to the backward induction paradox. Synthese 169, 301–333 (2009); Knowledge, Rationality and Action, 705–737
5. Feinberg, Y.: Subjective reasoning: Dynamic games. Games and Economic Behavior 52, 54–93 (2005)
6. Ghosh, S., Meijering, B., Verbrugge, R.: Empirical reasoning in games: Logic meets cognition. In: Ågotnes, T., Alechina, N., Logan, B. (eds.) Proceedings Third Logics for Resource Bounded Agents Workshop (LRBA 2010), pp. 15–34 (2010)
7. Halpern, J.: Substantive rationality and backward induction. Games and Economic Behavior 37, 425–435 (2001)
8. Perea, A.: Epistemic game theory: Reasoning and choice. Cambridge University Press (2012)
9. Samet, D.: Hypothetical knowledge and games with perfect information. Games and Economic Behavior 17, 230–251 (1996)
10. Stalnaker, R.: Belief revision in games: forward and backward induction. Mathematical Social Sciences 36, 31–56 (1998)

Temporalizing Modal Epistemic Logic

Ren-June Wang

Philosophy Department, National Chung-Cheng University,
No.168, Sec. 1, University Rd., Min-Hsiung Township
Chia-yi County 62102, Taiwa

Abstract. Timed Modal Epistemic Logic, tMEL, is a newly introduced logical framework for reasoning about the modeled agent's knowledge. The framework, derived from the study of Justification Logic, is adapted from the traditional Modal Epistemic Logic, MEL, to serve as a logically non-omniscient epistemic logic and dealing with problems where the temporal constraint is an unavoidable factor. In this paper we will give a semantic proof for the formal connection between MEL and tMEL, the Temporalization Theorem, which states that every MEL theorem can be turned into a tMEL theorem if suitable time labels can be found for each knowledge statement involved in the MEL theorem. As a result, the proof also gives us a better understanding of the semantics on the both sides of the theorem.

Keywords: Epistemic Logic, Agent Theory, Modal Logic, Reasoning Time, Timed Modal Epistemic Logic, Temporalization Theorem, Justification Logic, Realization Theoremn.

1 Introduction

Contemporarily, the modal approach of epistemic logic, MEL, with its possible world semantics is the standard logical framework for reasoning about the mental qualities of agents [11, 8, 20]. But it is also well-noticed that the setting is defective. The modeled agents are able to know, say, all the logical consequences of their knowledge, a reasoning ability categorizing only ideal agents. Not surprisingly, approaches has been proposed to deal with the problem (e.g. [16, 13, 12, 7]). Based on the analysis that the problem is due to the agents' knowing too much, different apparatuses have been suggested to restrict the modeled agent's reasoning ability (*awareness function*, *impossible world*, *incomplete set of rules*, etc.). However, these restrictions have been argued to be ad hoc, and the rationality of the agency appears absent in these approaches [4]. Alternatively, it has been suggested that the problem can't be solved by proposing weaker logical frameworks, as what these approaches are trying to do. Instead, to solve the problem, a logical framework toward the epistemic foundation of agent theory should reveal the dynamic feature of agents's reasoning such that the propositions that are hard to know can be distinguished from the ones that are easy to.

Several logical setting can be counted as falling into this later group of approaches, which include Step Logic (later Active Logic) [6, 15], Algorithmic Logic [4, 5], Justification Logic [1, 2, 9], and timed Modal Epistemic Logic tMEL [17, 18], to name some of them. Among them, tMEL is distinct in the way that it is built up on the foundation of the original MEL logical framework, so keeping the flavor of possible world semantics, serves as a logically non-omniscient epistemic logic, so the modeled agents don't assume to possess the reasoning ability beyond the reach of human beings, and is capable of dealing with problems where deadline constraint is a factor, such as *The Nell & Dudley Problem* [14, 10, 3]. Roughly speaking, syntactically, each MEL formula of the form $\mathbf{K}\phi$ will be accompanied with a natural number i to form a formula $\mathbf{K}^i\phi$ in tMEL, purported to mean ϕ is known at the time i; and semantically, each world of a tMEL structure will be equipped with a syntactical device, called an awareness function, to capture when the agent is aware of a formula by the deductive procedure that the model designers assume the agent to possess. Then for example the MEL theorem $\mathbf{K}(\phi{\to}\psi){\to}(\mathbf{K}\phi{\to}\mathbf{K}\psi)$, which can be interpreted as saying that the agent is able to perform *modus ponens*, has temporal counterparts in tMEL, $\mathbf{K}^i(\phi{\to}\psi){\to}(\mathbf{K}^j\phi{\to}\mathbf{K}^k\psi)$, for numbers $i, j{<}k$, saying further that the rule takes time to apply. This temporalization of an MEL theorem into a tMEL theorem is in fact not an isolated result. One of the important metatheorems concerning tMEL, the *Temporalization Theorem*, making the formal connection between MEL and tMEL, states that every MEL theorem can be turned into a tMEL theorem if suitable time labels can be found for each knowledge statement (formula of the form $\mathbf{K}\phi$) involved in the MEL theorem.

This Temporalization Theorem renders such a fact that there is indeed a temporal aspect hidden in our all familiar Modal Epistemic Logic, with its temporal information revealed in the setting of tMEL. And with such a connection result, it moreover suggests that it is possible for the future study to turn whatever technical results established based on MEL in the context that temporal relations are not relevant, to more refined consequences in which the time that the modeled agents is taken for reasoning plays an essential role. The goal of this paper is thus to supply a semantic proof for such a connection metatheorem. Although a syntactic proof of the Temporalization Theorem has been given in the context of studying the proof relations between MEL, tMEL, and Justification Logic [19], a semantic proof of a logical result is always of its own interests. In particular, the generalization of the proof-theoretical method is restricted, for it needs to take cut-free Gentzen style proofs into consideration, and, as we can see, besides its promise of generalization, the semantic proof provided here shed light on both the tMEL and MEL semantics.

2 MEL and tMEL Logics

2.1 Modal Epistemic Logic

We begin with a presentation of the semantics, together with the axiom systems for reference, of the logics on the both sides of the main result. We first review the possible world semantics, which is the foundation of tMEL semantics. The language of MEL is built up from a nonempty set of primitive propositions $\mathcal{P}$, boolean connectives, and a modal operator $\mathbf{K}$. To simplify the arguments, only cases of boolean connectives negation ($\sim$) and implication ($\rightarrow$) will be explicitly discussed. A well-formed MEL formula is defined according to the following grammar $\phi := p|{\sim}\phi|\phi{\rightarrow}\phi|\mathbf{K}\phi$, where $p \in \mathcal{P}$.

A *structure* or *a model* for MEL is a tuple $\langle W, R, \mathcal{V}\rangle$, where W is a set of worlds or epistemic alternatives, R is a binary relation defined on W, normally called accessibility relation, and $\mathcal{V}$ is a function assigning possible worlds to primitive propositions. The *satisfaction relation in a structure M* is recursively defined as follows:

$M, w \Vdash p \Leftrightarrow w \in \mathcal{V}(p)$,
$M, w \Vdash {\sim}\phi \Leftrightarrow M, w \nVdash \phi$,
$M, w \Vdash \phi \rightarrow \psi \Leftrightarrow M, w \nVdash \phi$ or $M, w \Vdash \psi$,
$M, w \Vdash \mathbf{K}\phi \Leftrightarrow M, w' \Vdash \phi$ for all $w' \in W$ with wRw'.

A formula is *valid in a structure* if it is satisfied in every world of the structure. Formulas which are valid in all structures compose the smallest MEL logic K, and its corresponding complete and sound axiom system is:

Axioms
Classical propositional axiom schemes
$\mathbf{K}(\phi \rightarrow \psi) \rightarrow (\mathbf{K}\phi \rightarrow \mathbf{K}\psi)$
Inference Rules
if $\vdash \phi \rightarrow \psi$ and $\vdash \phi$, then $\vdash \psi$
if $\vdash \phi$, then $\vdash \mathbf{K}\phi$

Several extensions of K are often discussed in the literature. The following is a table of some well studied modal logical axioms, especially in the epistemic context, and their correpsponding conditions on the binary relation R:

	Axiom		R
T	$\mathbf{K}\phi{\rightarrow}\phi$	Truth Axiom	reflexive
4	$\mathbf{K}\phi{\rightarrow}\mathbf{K}(\mathbf{K}\phi)$	Positive Introspection Axiom	transitive
5	${\sim}\mathbf{K}\phi{\rightarrow}\mathbf{K}({\sim}\mathbf{K}\phi)$	Negative Introspection Axiom	euclidean

Let Λ be a subset of $\{T, 4, 5\}$. A $K\Lambda$-*structure* is a structure whose binary relation satisfies conditions corresponding to the axioms mentioned in Λ. We call a formula $K\Lambda$-*valid* if it is valid in all $K\Lambda$ structures. $K\Lambda$ *logic* contains all $K\Lambda$-valid formulas, and its complete and sound axiomatic counterpart is exactly the axiom system K plus axioms in Λ. For example $K45$ is the logic of all formulas valid in transitive and euclidean structures, and $K45$ axiom system is K plus axioms 4 and 5. Notice that the $KT4$ logic in our terminology is the familiar $S4$, and $KT45$ is $S5$. All the logics listed here are the targets of this paper. We will show at once that theorems in these logics can be temporalized into their counterparts in tMEL logics.

2.2 Timed Modal Epistemic Logic

Semantics Basics. The language of tMEL is similar to the language of MEL except that the natural numbers are now part of the formula constructors. Natural Numbers are used to denote the passage of time, that is, in tMEL a simple structure of time, discrete, linear, with a beginning point, is considered. The grammar of well-formed tMEL formulas is: $\phi := p|{\sim}\phi|\phi{\rightarrow}\phi|\mathbf{K}^i\phi$, where $p \in \mathcal{P}$ and $i \in \mathbb{N}$ a natural number. $\mathbf{K}^i\phi$ is read as the formula ϕ is known at time i.

A *tMEL base* is a tuple $\mathcal{A} = \langle \mathbf{A}, \mathbf{f} \rangle$, where $\mathbf{A}$ is a set of tMEL formulas and $\mathbf{f} \colon \mathbf{A} \rightarrow \mathbb{N}$. Given a base $\mathcal{A} = \langle \mathbf{A}, \mathbf{f} \rangle$, we call a partial function α that associates tMEL formulas with natural numbers an $\mathcal{A}$-*awareness function*, if it satisfies the following condition:

If $A \in \mathbf{A}$, then $\alpha(A) \leq \mathbf{f}(A)$. (*Initial Condition*)

Furthermore, we will call an $\mathcal{A}$-awareness function *normal* if it satisfies two more conditions ($\alpha(\phi){\downarrow}$ means $\alpha(\phi)$ is defined):

If $\alpha(\phi{\rightarrow}\psi){\downarrow}$ and $\alpha(\phi){\downarrow}$, then
$\alpha(\psi){\leq}\max(\alpha(\phi{\rightarrow}\psi), \alpha(\phi)){+}1$. (*Deduction by Modus Ponens*)

If $A \in \mathbf{A}$ and $\mathbf{f}(A) \leq i$, then
$\alpha(\mathbf{K}^i A) \leq i + 1$. (*Deduction by $\mathcal{A}$-Epistemization*)

Basically, an agent modeled by a tMEL logic is assumed to employ some kind of axiomatic method for reasoning, and the aim of an awareness function is to record the reasoning process of the modeled agent. Formulas in the set $\mathbf{A}$ of a base are supposed to be the formulas of which the truths are acceptable by the agent through non-deductive methods, such as perceiving some self-evident logical truths inherently or conveyed by others, and $\mathbf{f}$ indicate when these non-deductive methods takes place. Then those conditions for awareness function just reflect the rules that the agent can apply, and for an awareness function α, $\alpha(\phi) = i$ indicates that the first time when the agent accepts the truth of ϕ is i.

Given a base $\mathcal{A} = \langle \mathbf{A}, \mathbf{f} \rangle$, a tMEL $\mathcal{A}$-structure is a tuple $M{=}\langle W, R, \mathfrak{A}, \mathcal{V} \rangle$, where $\langle W, R, \mathcal{V} \rangle$ is an MEL structure and $\mathfrak{A} = \{\alpha_w\}$ is a collection of $\mathcal{A}$-awareness functions with one for each world $w \in W$. Then the *satisfaction relation in a tMEL structure M* is the same as that in MEL structure except that the rule for modal formulas is changed to the following:

$$M, w \Vdash \mathbf{K}^i\phi \Leftrightarrow M, w' \Vdash \phi \text{ for all } w' \in W \text{ with } wRw', \\ \text{and } \alpha_w(\phi) \leq i.$$

It says that in the world $w \in W$ the agent knows a formula at the time i if and only if the formula is true in all possible worlds accessible from w and the agent accepts the truth of the formula before or equal to i.

A formula is *valid* in a tMEL structure if the formula is satisfied at all worlds in the structure. Given a base $\mathcal{A}$, a structure $M{=}\langle W, R, \mathfrak{A}, \mathcal{V} \rangle$ is a $tK(\mathcal{A})$-*structure* if $\mathfrak{A}$ consists of normal $\mathcal{A}$-awareness functions, and the *logic of* $tK(\mathcal{A})$ is the set of formulas valid in all $tK(\mathcal{A})$-structures.

Similar to MEL logics, several extensions of $tK(\mathcal{A})$ are defined based on subclasses of $tK(\mathcal{A})$ structures. But now subclasses are determined not only by the binary relation R but also by the collection of awareness functions $\mathfrak{A}$, and its relation to the structure.

Given two awareness functions α, β, we write $\beta \leq \alpha$ to mean that $\beta(\phi) \leq \alpha(\phi)$ for every formula ϕ with $\alpha(\phi)\downarrow$. Let $M = \langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ be a tMEL structure, and here are some more conditions on awareness functions:

If $\alpha_w(\phi) \leq i$, then $\alpha_w(\mathbf{K}^i\phi) \leq i+1$ (*Inner Positive Introspection*)
If $\alpha_w(\phi) \not\leq i$, then $\alpha_w(\mathbf{K}^i\phi) \not\leq i+1$ (*Inner Negative Introspection*)

For any wRw', $\alpha_{w'} \leq \alpha_w$ (Monotonicity)
For any wRw', $\alpha_w \leq \alpha_{w'}$ (Anti-Monotonicity)

If $M, w \Vdash \mathbf{K}^i\phi$, then $\alpha_w(\mathbf{K}^i\phi) \leq i+1$. (*Outer Positive Introspection*)
If $M, w \nVdash \mathbf{K}^i\phi$, then $\alpha_w(\mathbf{K}^i\phi) \not\leq i+1$. (*Outer Negative Introspection*)

Within a given structure, an awareness function is *positive regular (with respect to the structure)* if it satisfies the monotonicity and both inner and outer positive introspection, and *negative regular (with respect to the structure)* if it satisfies the anti-monotonicity and both inner and outer negative introspection. Some tMEL axioms and their corresponding conditions on the awareness functions in $\mathfrak{A}$ are listed in the following table:

	Axiom		$\mathfrak{A}$
tT	$\mathbf{K}^i\phi \to \phi$		none
$t4$	$\mathbf{K}^i\phi \to \mathbf{K}^j(\mathbf{K}^i\phi)$	$i < j$	positive regular
$t5$	$\sim\mathbf{K}^i\phi \to \mathbf{K}^j(\sim\mathbf{K}^i\phi)$	$i < j$	negative regular

Let Λ be a subset of $\{T, 4, 5\}$, and $\mathcal{A}$ be a base. A $tK(\mathcal{A})$-structure $\langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ is a *$tK\Lambda(\mathcal{A})$-structure* if $\langle W, R, \mathcal{V}\rangle$ is a $K\Lambda$-structure and every awareness function in $\mathfrak{A}$ also satisfies the conditions corresponding the tMEL axioms in Λ. A formula is *$tK\Lambda(\mathcal{A})$ valid* if it is valid in all $tK\Lambda(\mathcal{A})$-structures. *$tK\Lambda(\mathcal{A})$ logic* contains all $tK\Lambda(\mathcal{A})$ valid formulas. So a $tK45(\mathcal{A})$ valid formula is valid in all $tK(\mathcal{A})$-structures whose binary relation is transitive and euclidean and its awareness functions are all both positive and negative regular.

Logical Bases and Axiomatization. Till now, there is no restriction on the base that is employed for the definition of tMEL semantics. But it will be more interesting if a base consists of *logical truths*, since it means the agent modeled by a tMEL logic with respect to the base have basic logical knowledge which will function like axioms in axiom systems in the agent's reasoning. Given bases $\mathcal{A} = \langle \mathbf{A}, \mathbf{f}\rangle$ and $\mathcal{B} = \langle \mathbf{B}, \mathbf{g}\rangle$, we will write $\mathcal{B} \subseteq \mathcal{A}$ to mean $\mathbf{B} \subseteq \mathbf{A}$ and $\mathbf{f}(B) \leq \mathbf{g}(B)$ for all $B \in \mathbf{B}$. A set of bases $\{\mathcal{A}_i (= \langle \mathbf{A}_i, \mathbf{f}_i\rangle)\}_{i\in\mathbb{N}}$ is an *ascending chain* if $\mathcal{A}_1 \subseteq \mathcal{A}_2 \subseteq \ldots$, and a base $\mathcal{A}$ is the *limit* of the chain if $\mathcal{A} = \bigcup \mathcal{A}_i$, i.e., $\mathbf{A} = \bigcup \mathbf{A}_i$ and $\mathbf{f}(A) = \min\{\mathbf{f}_i(A) : \mathbf{f}_i(A)\downarrow\}$. The following is the definition of such bases:

Definition 1. *A base $\mathcal{A}$ is $tK\Lambda$-*logical *if one of following is true:*
(1) $\mathcal{A}$ is empty,
(2) $\mathcal{A}$ consists of $tK\Lambda(\mathcal{B})$-valid formulas with $\mathcal{B}$ $tK\Lambda$-logical,
(3) $\mathcal{A}$ is the limit of an ascending chain of $tK\Lambda$-logical bases $\{\mathcal{A}_i\}_{i\in\mathbb{N}}$ where $\mathcal{A}_{i+1}$ consists of $tK\Lambda(\mathcal{A}_i)$-valid formulas for every $i \in \mathbb{N}$.

Lemma 1. *If $\mathcal{A}= \langle \mathbf{A}, \mathbf{f}\rangle$ is a $tK\Lambda$ logical base, every formula in $\mathbf{A}$ is $tK\Lambda(\mathcal{A})$ valid.*

Given a tK-logical base $\mathcal{A}= \langle \mathbf{A}, \mathbf{f}\rangle$, there is a corresponding axiom system of the logic of $tK(\mathcal{A})$:

Axioms

Classical propositional axiom schemes
$\mathbf{K}^i(\phi \to \psi) \to (\mathbf{K}^j\phi \to \mathbf{K}^k\psi) \quad i, j < k$ (*Deduction by Modus Ponens*)
$\mathbf{K}^i A \to \mathbf{K}^j(\mathbf{K}^i A) \;\; i < j$ if $A \in \mathbf{A}$ and $\mathbf{f}(A) \leq i$ (*Deduction by $\mathcal{A}$-Epistemization*)
$\mathbf{K}^i\phi \to \mathbf{K}^j\phi \quad i < j$ (*Monotonicity*)

Inference Rules

if $\vdash \phi \to \psi$ and $\vdash \phi$, then $\vdash \psi$ (*Modus Ponens*)
if $A \in \mathbf{A}$ and $\mathbf{f}(A) \leq i$, then $\vdash \mathbf{K}^i A$ (*$\mathcal{A}$-Epistemization*)

Let $\mathcal{A}$ be a $tK\Lambda$-logical base. For the logic of $tK\Lambda(\mathcal{A})$, the sound and complete corresponding axiom system is tK plus the tMEL axioms in Λ (more precisely, tK plus tX axioms with $X\in\Lambda$).

Theorem 1. *Given a $tK\Lambda$-logical base $\mathcal{A}$, a tMEL formula ϕ is $tK\Lambda(\mathcal{A})$-valid if and only if it is provable in the $tK\Lambda(\mathcal{A})$ axiom system.*

So for a given $tK\Lambda$ logic, there is actually a collection of corresponding $tK\Lambda(\mathcal{A})$ logics introduced. The logical bases $\mathcal{A}$ are capturing the basic logical truths that agents are assumed to be aware of, and hence different $tK\Lambda(\mathcal{A})$ logics manifest different logical strengths. For example, if $\mathcal{A}$ is the empty base, then no formula of form $\mathbf{K}^i\phi$ is $tK\Lambda(\mathcal{A})$ valid. In [18], it is shown that there exists comprehensive $tK\Lambda$-logical bases $\mathcal{A}$ such that every $tK\Lambda(\mathcal{A})$ valid formula is in the base $\mathcal{A}$. One type of logical bases lying between the above two deserves additional attention. A *full $tK\Lambda$-logical base* $\mathcal{A}$ is such that for any $tK\Lambda$ valid formula ϕ, though ϕ might not be in the base, $\mathbf{K}^i\phi$ is $tK\Lambda$ valid for some i. This feature of a full logical base is a desirable one, since it indicates that the agent modeled by a $tK\Lambda$ logic with respect to a full logical base has enough basic logical knowledge to derive to know all valid formulas.

Another good and natural feature that we would like a logical base to possess is *schematic*. By a *schematic logical base* $\mathcal{A}= \langle \mathbf{A}, \mathbf{f}\rangle$, we mean that suppose $\phi \in \mathbf{A}$ and $\mathbf{f}(\phi) = i$, then if we add a fixed number n to every number labels in ϕ to form a new tMEL formula ψ, then $\psi \in \mathbf{A}$ and $\mathbf{f}(\psi) = i$, too. This property suggests that the agent modeled by a $tK\Lambda$ logic with respect to a schematic logical bases is aware of the formula in $\mathbf{A}$ by schema, and hence for formulas falling into the same schema the agent is aware of them at the same time.

Fixing a $\Lambda \subseteq \{T, 4, 5\}$, our main goal is just to show that every $K\Lambda$ theorem can be *temporalized* to a theorem of a $tK\Lambda$ logic with respect to a schematic full logical base $\mathcal{A}$, that is, every $K\Lambda$ theorem can be turned into a $tK\Lambda(\mathcal{A})$ theorem by finding suitable number labels for knowledge statements involved in the $K\Lambda$ theorem, and equivalently, due to the completeness theorem, to show that every $K\Lambda$ valid formula can be temporalized to a $tK\Lambda(\mathcal{A})$ valid formula.

However, the route we take to prove the result will take several stages. First we need the following lemma proved in [18] (for simplication, all the logical bases are taken to be schematic in the following discussions):

Lemma 2. *A $tK\Lambda$-logical base $\mathcal{A}$ is full if and only if there is a comprehensive $tK\Lambda$-logical base $\mathcal{B}$ such that for every tMEL formula ϕ, ϕ is $tK\Lambda(\mathcal{A})$ valid if and only if ϕ is $tK\Lambda(\mathcal{B})$ valid.*

We call a logical base $\mathcal{A}= \langle \mathbf{A}, \mathbf{f} \rangle$ *principal* if $\mathbf{f}$ is the constant function $\mathbf{0}$, that is, $\mathbf{f}(\phi) = 0$ for all $\phi \in \mathbf{A}$. Let $\mathcal{A}$ be a principal comprehensive $K\Lambda$-logical base, and $\mathcal{B}$ an arbitrary comprehensive $K\Lambda$-logical base. In the next section we will show that a $K\Lambda$ valid formula can be temporalized to a $tK\Lambda(\mathcal{A})$ valid formula if and only if it can be temporalized to a $tK\Lambda(\mathcal{B})$ valid formula. This result together with the previous lemma shows that a $tK\Lambda$ logic with the principal comprehensive logical base and that with a full logical base have the same logical strength to temporalize $K\Lambda$ valid formulas. Then in the section after, we will prove that a $tK\Lambda$ logic with the principal comprehensive logical base indeed can temporalize every $K\Lambda$ valid formula to conclude our main theorem.

3 Comprehensive Bases

Before continuing, we need some notations and terminology for our discussions in this and the next section. For simplicity, we will use *subformulas* to mean *subformula occurrences* of a formula in this paper. According to their positions in a formula, subformulas can be categorized either *positive* or *negative*: for a formula ϕ, ϕ is a positive subformula of itself and if $\theta \to \psi$, $\mathbf{K}\psi$ or $\sim\theta$ is a positive subformula, or if $\psi \to \theta$, $\mathbf{K}\theta$, or $\sim\psi$ is a negative subformula, ψ and θ are positive and negative subformula of ϕ respectively. Let ϕ be an MEL formula. We use $\mathcal{O}(\phi)$ to denote the set of all subformulas of ϕ with the form $\mathbf{K}\psi$, and $\mathcal{O}^+(\phi)$ and $\mathcal{O}^-(\phi)$ to denote the subsets of $\mathcal{O}(\phi)$ of positive and negative subformulas respectively. Given a function $\tau : \mathcal{O}(\phi) \to \mathbb{N}$, it will induce a natural translation, also denoted as τ, on ϕ such that ϕ^τ is a tMEL formula and τ fixes the primitive propositions, commutes with boolean connectives, and $(\mathbf{K}\psi)^\tau = \mathbf{K}^i(\psi^\tau)$ with $i = \tau(\mathbf{K}\psi)$. We will call $\tau : \mathcal{O}(\phi) \to \mathbb{N}$ a *temporalization function* on ϕ or a *t-function* on ϕ. For each MEL formula ϕ and a t-function τ on ϕ there is a corresponding tMEL formula ϕ^τ, and for each tMEL formula ψ there is a corresponding unique MEL formula ϕ (the resulting formula from removing number labels from ψ) and a unique t-function on ϕ such that $\psi = \phi^\tau$. So in the following we will simply write a tMEL formula as ϕ^τ with ϕ an MEL formula and τ a t-function on ϕ. Given a t-function τ on ϕ, $\tau + n$ is the t-function on ϕ such that $(\tau + n)(\mathbf{K}\psi) = \tau(\mathbf{K}\psi) + n$ for every subformula $\mathbf{K}\psi \in \mathcal{O}(\phi)$, and we will call $\phi^{\tau+n}$ the *n-shift* of ϕ^τ.

With these notations we can define a *schematic logical base* $\mathcal{A}(=\langle \mathbf{A}, \mathbf{f}\rangle)$ as that if $\phi^\tau \in \mathbf{A}$, $\phi^{\tau+n} \in \mathbf{A}$ for every n and $\mathbf{f}(\phi^{\tau+n}) = \mathbf{f}(\phi^\tau)$, and we can define a *temporalization* of a $K\Lambda$ valid formula ϕ as that there is a t-function τ on ϕ such that ϕ^τ is a $tK\Lambda$ valid formula with respect to a logical base.

We call a set S of tMEL formulas $tK\Lambda(\mathcal{A})$*-satisfiable* if all the formulas are satisfiable in a world of a $tK\Lambda(\mathcal{A})$ structure, and $tK\Lambda(\mathcal{A})$*-finitely satisfiable* if every finite subset of S is satisfiable. The compactness theorem holds for all $tK\Lambda(\mathcal{A})$ with $\mathcal{A}$ a logical base. The proof is basically by constructing the $tK\Lambda(\mathcal{A})$-structure $M=\langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ composed of all maximal $tK\Lambda(\mathcal{A})$-finitely satisfiable set Γ of formulas, where $\Gamma R \Gamma'$ if and only if $\Gamma^\sharp \subseteq \Gamma'$ for $\Gamma^\sharp = \{\psi \mid \mathbf{K}\psi^i \in \Gamma\}$, and α_Γ and $\mathcal{V}$ are defined as $\alpha_\Gamma(\psi) = \min\{i \mid \mathbf{K}\psi^i \in \Gamma\}$ and $\mathcal{V}(P) = \{\Gamma \mid P \in \Gamma\}$, respectively. In the following discussions, this model will be referred to as the *canonical* $tK\Lambda(\mathcal{A})$ *model.* We will leave the qualification of this structure as a $tK\Lambda(\mathcal{A})$-structure and the *Truth Lemma*: $M, \Gamma \Vdash \phi$ if and only if $\phi \in \Gamma$, for the readers to check.

Theorem 2 (Compactness Theorem). *Given a* $tK\Lambda$*-logical base* $\mathcal{A}$*, a set of tMEL formulas is* $tK\Lambda(\mathcal{A})$*-satisfiable if and only if it is* $tK\Lambda(\mathcal{A})$*-finitely satisfiable.*

For a function $\mathbf{f}\colon \mathbf{A} \to \mathbb{N}$, we write $\mathbf{f}|_\mathbf{B}$ to mean the restriction of $\mathbf{f}$ to the subset $\mathbf{B}$ of $\mathbf{A}$.

Corollary 1. *Given a* $tK\Lambda$*-logical base* $\mathcal{A} = \langle \mathbf{A}, \mathbf{f}\rangle$*, a tMEL formula* ϕ^τ *is* $tK\Lambda(\mathcal{A})$ *valid if and only if there is a* $\mathcal{A}' = \langle \mathbf{A}', \mathbf{f}'\rangle$ *where* $\mathbf{A}'$ *is a finite subset of* $\mathbf{A}$ *and* $\mathbf{f}'$ *is* $\mathbf{f}|_{\mathbf{A}'}$ *such that* ϕ^τ *is* $tK\Lambda(\mathcal{B})$ *valid.*

We call an awareness function β an n*-backshift* of an awareness function α providing for every ϕ if $\alpha(\phi^{\tau+n})\downarrow$, then $\beta(\phi^\tau)\downarrow$, and $\beta(\phi^\tau) = \max\{0, \alpha(\phi^{\tau+n}) - n\}$. A structure $M' = \langle W', R', \mathfrak{A}', \mathcal{V}'\rangle$ is an n*-backshift* of $M = \langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ if $\langle W, R, \mathcal{V}\rangle = \langle W', R', \mathcal{V}'\rangle$ and every awareness function β_w in $\mathfrak{A}'$ is an n-backshift of α_w in $\mathfrak{A}$. We have the following lemma.

Lemma 3. *If a tMEL structure* $M'=\langle W', R', \mathfrak{A}', \mathcal{V}'\rangle$ *is an* n*-backshift of* $M=\langle W, R, \mathfrak{A}, \mathcal{V}\rangle$*,* $M, w \vDash \phi^{\tau+n}$ *if and only if* $M', w \vDash \phi^\tau$ *for any tMEL formulas* ϕ^τ*.*

The proof is simply by induction on the complexity of formulas. Finally, we also need the following:

Lemma 4. *For any* $tK\Lambda$*-logical bases* $\mathcal{A}$*, if* ϕ^τ *is* $tK\Lambda(\mathcal{A})$ *valid, then for any number* n*,* $\phi^{\tau+n}$ *is also* $tK\Lambda(\mathcal{A})$ *valid.*

Proof. The syntactical proof of this lemma is straightforward (take a look at the axiom systems); however we give a semantic proof here to investigate the tMEL semantics and to render some skills and techniques that might be useful for the future work. Suppose that there is an n and a $tK\Lambda(\mathcal{A})$-structure $M=\langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ such that $M, w \Vdash (\sim\phi)^{\tau+n}$. Let M' be the n-backshift of M, then by Lemma 3, $M', w \Vdash (\sim\phi)^\tau$. Now we have to prove that M' is also a $tK\Lambda(\mathcal{A})$-structure to finish the proof. That is, we have to show whatever conditions that are listed above are satisfied by $\alpha_w \in M$, are also satisfied by $\beta_w \in M'$.

The proof is straightforward. We only check the case for the initial condition. Since $\phi^\tau \in \mathcal{A}$, $\phi^{\tau+n} \in \mathcal{A}$ (logical bases are supposed to be schematic). Then $\beta(\phi^\tau) = \max\{0, \alpha(\phi^{\tau+n}) - n\} \leq f(\phi^{\tau+n}) = f(\phi^\tau)$. So β_w is also an $\mathcal{A}$-awareness function.

Theorem 3. *A tMEL formula ϕ^τ is a valid formula of a tKΛ logic with respect to a comprehensive logical base if and only if there is a t-function τ' on ϕ such that $\phi^{\tau'}$ is a valid formula of a tKΛ logic with respect to the principal comprehensive logical base.*

Proof. Given a logical base $\mathcal{A} = \langle \mathbf{A}, \mathbf{f} \rangle$, we first assign a *rank* to every $tK\Lambda(\mathcal{A})$ valid formula. Let S be the set of all $tK\Lambda(\mathcal{A})$ valid formulas, and $\emptyset$ be the empty logical base. If $\phi^\tau \in S$ is $tK\Lambda(\emptyset)$ valid, the rank is 1. Suppose we have assigned formulas in S with rank less than k. Let $\mathbf{B}$ be the set of formulas in $\mathbf{A}$ whose ranks less than k, and $\mathcal{B} = \langle \mathbf{B}, \mathbf{f}|_{\mathbf{B}} \rangle$. Then for formulas in S whose ranks are undefined and which are also $tK\Lambda(\mathcal{B})$ valid, their ranks are k. Now according to Corollary 1, every formula in S is of a finite rank. Since for a comprehensive logical base every valid formula of the $tK\Lambda$ logic with the base is in the base, so every formula in the base has a rank.

We first prove the *if-part* of the theorem. Let $\mathcal{A}(= \langle \mathbf{A}, 0 \rangle)$ be the principal comprehensive $tK\Lambda$-logical base, and $\mathcal{B}(= \langle \mathbf{B}, \mathbf{g} \rangle)$ be a comprehensive $tK\Lambda$-logical base. We will actually show that if ϕ^τ is $tK\Lambda(\mathcal{A})$ valid, then there is a $n \in \mathbb{N}$ such that for any $m \geq n$, $\phi^{\tau+m}$ is $tK\Lambda(\mathcal{B})$ valid, by induction on the rank of $\phi^\tau \in \mathbf{A}$. When ϕ^τ's rank is 1, which means ϕ is $tK\Lambda(\emptyset)$ valid, so ϕ is $tK\Lambda(\mathcal{B})$ valid. Then by Lemma 4, $\phi^{\tau+n}$ is $tK\Lambda(\mathcal{B})$ valid, too, for every n. So $\phi^{\tau+n} \in \mathcal{B}$. The base case holds. Now suppose ϕ^τ's rank is $k>0$, then there is a finite base $\mathcal{A}' = \langle \mathbf{A}', 0 \rangle$ such that ϕ^τ is $tK\Lambda(\mathcal{A}')$-valid, where $\mathbf{A}' = \{\phi_1^{\tau_1}, \ldots, \phi_s^{\tau_s}\}$ and for each i the rank of $\phi_i^{\tau_i}$ is less than k. Then by IH, there is an n_i for each i such that $\phi_i^{\tau_i+n_i}$ is $tK\Lambda(\mathcal{B})$-valid, and hence $\mathbf{g}(\phi_i^{\tau_i+n_i}) \downarrow$.

Now picking m large enough such that $m > n_i$ and $m > \mathbf{g}(\phi_i^{\tau_i+n_i})$ for each i, we are going to show $\phi^{\tau+m}$ is $tK\Lambda(\mathcal{B})$-valid, and then finish the proof. Suppose $\phi^{\tau+m}$ is not $tK\Lambda(\mathcal{B})$-valid, $\sim\phi^{\tau+m}$ is satisfiable in a $tK\Lambda(\mathcal{B})$-structure M. Let M' be the m-backshift of M. Then $\sim\phi^\tau$ is satisfiable in M', by Lemma 3. So all we need to show is that M' is a $tK\Lambda(\mathcal{A}')$-structure. Everything is similar to the proof in the previous Lemma 4, except that we have to show that every m-backshift awareness function β_w in M' of the awareness function α_w in M is an $\mathcal{A}'$-awareness function. Since $\phi_i^{\tau_i+m}$ is in $\mathcal{B}$, $\alpha_w(\phi_i^{\tau_i+m})\downarrow$ and hence $\beta_w(\phi_i^{\tau_i})$ is defined. By the definition of m-backshift $\beta_w(\phi_i^{\tau_i}) = \max\{0, \alpha_w(\phi_i^{\tau_i+m}) - m\} = 0$. Hence every β_w in M' is $tK\Lambda(\mathcal{A}')$ valid and M' is a $tK\Lambda(\mathcal{A}')$-structure.

For the *only-if-part*, let $\mathcal{A}_i = \langle \mathbf{A}_i, 0 \rangle$, where $\mathbf{A}_i = \{\phi^\tau \in \mathbf{A} \mid$ the rank of ϕ is equal to or less than $i\}$, and $\mathcal{B}_i = \langle \mathbf{B}_i, \mathbf{g}_i \rangle$, where $\mathbf{B}_i = \{\phi^\tau \in \mathbf{B} \mid$ the rank of ϕ is equal to or less than $i\}$ and $\mathbf{g}_i = \mathbf{g}|_{\mathbf{B_i}}$. We will prove that for every i, $\mathcal{B}_i \subseteq \mathcal{A}_i$. and hence $\mathcal{B} \subseteq \mathcal{A}$. We prove it by induction on the index. For the base case, both $\mathbf{A}_1$ and $\mathbf{B}_1$ are the collection of $tK\Lambda(\emptyset)$ valid formulas, so $\mathcal{B}_1 \subseteq \mathcal{A}_1$. Suppose $\mathcal{B}_i \subseteq \mathcal{A}_i$, then every $tK\Lambda(\mathcal{A}_i)$-awareness function is a $tK\Lambda(\mathcal{B}_i)$-awareness function, and hence every $tK\Lambda(\mathcal{A}_i)$-structure is a $tK\Lambda(\mathcal{B}_i)$-structure, so every $tK\Lambda(\mathcal{B}_i)$ valid formula is a $tK\Lambda(\mathcal{A}_i)$ valid formula. This completes the proof.

4 Temporalization Theorem

So in this section we will complete the semantic proof of Temporalization Theorem. We first prove several interesting theorems about the tMEL semantics, which will lead us to the main result. In this section we fix an MEL logic $K\Lambda$ and its tMEL counterpart $tK\Lambda(\mathcal{A})$ with $\mathcal{A}$ the principal comprehensive logical base. All discussions will be relative to these fixed logics. We will omit the logic name. From the context it should be clear which logic ($K\Lambda$ or $tK\Lambda(\mathcal{A})$) is under discussion. We write $\models \phi^\tau$ to mean ϕ^τ is $tK\Lambda(\mathcal{A})$ valid. Notice that the most important feature of the principal comprehensive logical base is that if $\models \phi^\tau$ then $\models \mathbf{K}^0(\phi^\tau)$.

Definition 2. *Let ϕ be an MEL formula and τ and τ' two t-functions on ϕ,*

1. $\tau < \tau'$ if for any $\mathbf{K}\psi \in \mathcal{O}(\phi)$, $\tau(\mathbf{K}\psi) < \tau'(\mathbf{K}\psi)$.

2. $\tau \prec \tau'$ if for any $\mathbf{K}\psi \in \mathcal{O}^+(\phi)$, $\tau(\mathbf{K}\psi) < \tau'(\mathbf{K}\psi)$, and for any $\mathbf{K}\psi \in \mathcal{O}^-(\phi)$, $\tau'(\mathbf{K}\psi) < \tau(\mathbf{K}\psi)$.

Lemma 5. *If $\tau \prec \tau'$ on ϕ, then $\models \phi^\tau \to \phi^{\tau'}$.*

Proof. The proof is by induction on the complexity of formula ϕ. The base case is trivial. Suppose $\phi \equiv \sim\psi$, then $\tau' \prec \tau$ on ψ, and by the Induction Hypothesis (IH), $\models \psi^{\tau'} \to \psi^\tau$, so $\models \sim(\psi^\tau) \to \sim(\psi^{\tau'})$, and hence $\models (\sim\psi)^\tau \to (\sim\psi)^{\tau'}$. We skip to check the case for $\phi \equiv \psi \to \theta$. Suppose $\phi \equiv \mathbf{K}\psi$, then $\tau \prec \tau'$ on ψ. By IH, $\models \psi^\tau \to \psi^{\tau'}$. Since $\models \mathbf{K}^0(\psi^\tau \to \psi^{\tau'})$, $\models \mathbf{K}^i(\psi^\tau) \to \mathbf{K}^j(\psi^{\tau'})$ for $i < j$ and hence $\models (\mathbf{K}\psi)^\tau \to (\mathbf{K}\psi)^{\tau'}$. The case for implication is similar. This completes the proof.

Theorem 4. *Let ϕ be an MEL formula. If for every t-function τ, ϕ^τ is satisfiable, then the set S composed of formulas ϕ^τ for all τ is satisfiable.*

Proof. Suppose the set S is not satisfiable, then there is a finite subset $\{\phi^{\tau_1}, \ldots, \phi^{\tau_s}\}$ of S which is not satisfiable. So $\models \sim(\phi_1^{\tau_1} \wedge \ldots \wedge \phi_s^{\tau_s})$. By Lemma 4, for any $n \in \mathbb{N}$, $\models \sim(\phi_1^{\tau_1+n} \wedge \ldots \wedge \phi_s^{\tau_s+n})$. Hence we can pick a larger number n and a τ such that $\tau \prec \tau_i + n$ for each i. Then $\models \sim\phi^\tau$. A contradiction. So S is satisfiable.

Definition 3. *Let ϕ be an MEL formula.*

ϕ is t-satisfiable *if there is a τ on ϕ such that ϕ^τ is satisfiable, and* t-refutable *if there is a τ on ϕ such that $\sim\phi^\tau$ is satisfiable.*

ϕ is unboundedly *t-satisfiable (t-refutable) if there is a τ on ϕ such that for every $\tau' > \tau$ on ϕ there is a $\tau'' > \tau'$ on ϕ such that $\phi^{\tau''}$ is satisfiable (refutable).*

ϕ is upward-closedly *t-satisfiable (t-refutable) if there is a τ on ϕ such that for every $\tau' > \tau$, $\phi^{\tau'}$ is satisfiable (refutable).*

Definition 4. *A* configurational *structure $M = \langle W, R, \mathfrak{A}, \mathcal{V} \rangle$ is such that $\mathbf{K}\phi$ is unboundedly t-satisfiable at w, provided for every w' with wRw', ϕ is unboundedly t-satisfiable at w' then.*

Theorem 5. *Every satisfiable set is satisfied in a configurational structure.*

Proof. We will prove the canonical structure is configurational, and it is sufficient to prove the following lemma. Let $[\phi^\tau] = \{\phi^{\tau'} | \tau' > \tau\}$.

Lemma 6. *Let Γ be a maximal finitely-satisfiable set of tMEL formulas. If for an MEL formula $\sim\mathbf{K}\phi$ there is a τ on $\sim\mathbf{K}\phi$ such that $[(\sim\mathbf{K}\phi)^\tau] \subseteq \Gamma$, then $\Gamma^\sharp \cup [(\sim\phi)^{\tau'}]$ is satisfiable, for $\tau' > \tau$ on $\sim\phi$.*

Proof. We will prove this theorem by contraposition. Suppose for an $\tau' > \tau$, $\Gamma^\sharp \cup [(\sim\phi)^{\tau'}]$ is not satisfiable. Then there are finitely many formulas $F_i \in \Gamma^\sharp$, and finitely many formulas $(\sim\phi_j)^{\tau_j} \in [(\sim\phi)^{\tau'}]$ such that the set $\{F_i\} \cup \{(\sim\phi_j)^{\tau_j}\}$ is not satisfiable, and hence $\vDash \sim((\bigwedge F_i) \wedge (\bigwedge(\sim\phi_j)^{\tau_j}))$ (i and j belong to some finite index sets which we do not mention here). It follows that $\vDash (\bigwedge F_i) \rightarrow (\bigvee \phi_j^{\tau_j})$. Define τ'' on ϕ such that for any $\mathbf{K}\psi \in O^+(\phi)$, $\tau'' = \max\{\tau_j(\mathbf{K}\psi)\} + 1$, and for any $\mathbf{K}\psi \in O^-(\phi)$, $\tau''(\mathbf{K}\psi) = \tau'(\mathbf{K}\psi)$. Then $\tau_j \prec \tau''$ and $\tau < \tau''$ on ϕ. Since $\vDash (\bigwedge F_i) \rightarrow (\bigvee \phi_j^{\tau_j})$ and $\vDash \phi^{\tau_j} \rightarrow \phi^{\tau''}$ for all j, then $\vDash (\bigwedge F_i) \rightarrow \phi^{\tau''}$. Since $F_i \in \Gamma^\sharp$, then $\bigwedge F_i \in \Gamma^\sharp$ and $\mathbf{K}^k(\bigwedge F_i) \in \Gamma$ for some k. Since $\vDash (\bigwedge F_i) \rightarrow \phi^{\tau''}$, then $\vDash \mathbf{K}^0((\bigwedge F_i) \rightarrow \phi^{\tau''})$, so $\vDash \mathbf{K}(\bigwedge F_i)^k \rightarrow \mathbf{K}(\phi^{\tau''})^l$, for $l > k$, and hence $\vDash \mathbf{K}(\phi^{\tau''})^l$. Then let τ''' on $\mathbf{K}\phi$ such that $\tau'''(\mathbf{K}\psi) = \tau''(\mathbf{K}\psi)$ for all $\mathbf{K}\psi \in O(\phi)$, and $\tau'''(\mathbf{K}\phi) = \max\{\tau(\mathbf{K}\phi), l\} + 1$, so $\tau''' > \tau$ on $\mathbf{K}\phi$ and $\vDash (\mathbf{K}\phi)^{\tau'''}$. It follows $(\mathbf{K}\phi)^{\tau'''} \in \Gamma$. So the lemma holds.

(*Proof of Theorem 5 Continued*) Suppose the canonical structure is not configurational, then there is a formula $\mathbf{K}\phi$, and a maximal finitely satisfiable set Γ such that $\mathbf{K}\phi$ is not unboundedly t-satisfiable at Γ but ϕ unboundedly t-satisfiable at all maximal finitely satisfiable set $\Gamma' \supseteq \Gamma^\sharp$. Then $\sim\mathbf{K}\phi$ is upward-closedly t-satisfiable at Γ, and hence there is a τ on $\sim\mathbf{K}\phi$ such that $[(\sim\mathbf{K}\phi)^\tau] \subseteq \Gamma$, by Truth Lemma. By the previous lemma, there is a $\tau' > \tau$ on $\sim\phi$ such that $\Gamma^\sharp \cup [(\sim\phi)^{\tau'}]$ is satisfiable. Then ϕ is not unboundedly t-satisfiable at some Γ'. A contradiction. So the canonical structure is configurational.

Given a *tKΛ* structure $M = \langle W, R, \mathfrak{A}, \mathcal{V}\rangle$, we will call $\langle W, R, \mathcal{V}\rangle$ *the underlying KΛ structure of M* and denote it as M°.

Theorem 6. *Let ϕ be an MEL formula, $M = \langle W, R, \mathfrak{A}, \mathcal{V}\rangle$ be a configurational structure and $w \in W$.*

1. *If ϕ is upward-closedly t-satisfiable at (M, w) then ϕ is satisfiable at (M°, w),*
2. *If ϕ is upward-closedly t-refutable at (M, w) then ϕ is refutable at (M°, w),*

Proof. We will prove this by induction on the complexity of ϕ. The basic case is trivial. Suppose $\phi \equiv \sim\psi$ is upward-closedly t-satisfiable at (M, w), then ψ is upward-closedly t-refutable at (M, w). So ψ is refutable at (M°, w), and ϕ is satisfiable at (M°, w). The proof for the second condition is similar.

Suppose at (M, w), $\phi \equiv \psi \rightarrow \theta$ is upward-closedly t-satisfiable, we will show that either ψ is upward-closedly refutable or θ is upward-closedly satisfiable at (M, w). Suppose not, then for every τ on ψ there is a $\tau' > \tau$ on ψ such that

ψ^{τ} is satisfiable, and for every τ on θ there is a $\tau' > \tau$ on θ such that $\theta^{\tau'}$ is refutable. Then for every τ on ϕ, we can always establish a $\tau' > \tau$ on ϕ such that $\psi^{\tau'}$ is satisfiable and $\theta^{\tau'}$ is refutable, and hence $\phi \equiv \psi \rightarrow \theta$ is not upward-closedly satisfiable. A contradiction. Then either ψ is upward-closedly refutable or θ is upward-closedly satisfiable at (M, w). So, by IH, ψ is refutable at (M, w) or θ is satisfiable at (M, w), and hence $\psi \rightarrow \theta$ is satisfiable at (M, w). The other part for this case is straightforward. We skip the proof here.

Finally we deal with the modal case. Suppose $\phi \equiv \mathbf{K}\psi$, and ϕ is upward-closedly satisfiable at (M, w), then for every wRw', ψ is upward-closedly satisfiable at (M, w'), so ψ is satisfiable at (M°, w'), and hence $\mathbf{K}\psi$ is satisfied at (M°, w). Now we suppose ϕ is upward-closedly refutable at (M, w), then suppose for every wRw', ψ is unboundedly satisfiable at (M, w'), then $\mathbf{K}\psi$ is unboundedly satisfiable at (M, w), since M is configurational. Then $\mathbf{K}\psi$ is not upward-closedly refutable at (M, w). This contradicts to our assumption. So there is a w' with wRw' such that ψ is upward-closedly refutable at (M, w'), and hence ψ is not satisfiable at (M°, w'), so ϕ is not satisfiable at (M°, w). This completes the proof.

Let Λ be a subset of $\{T, 4, 5\}$. Here is our main result:

Theorem 7. *Given the principal comprehensive $tK\Lambda$-logical base $\mathcal{A}$, ϕ is $K\Lambda$ valid if and only if there is a temporalization function τ on ϕ such that ϕ^{τ} is $tK\Lambda(\mathcal{A})$ valid.*

Proof. We first prove the direction from right to left. Suppose ϕ is satisfiable in a world w of a $K\Lambda$ structure $M = \langle W, R, \mathcal{V}\rangle$. Let $M^{+} = \langle W, R, \mathfrak{A}, \mathcal{V}\rangle$, where $\mathfrak{A}$ is the collection of awareness functions α_w such that $\alpha_w(\phi^{\tau}) = 0$ for all tMEL formulas ϕ^{τ}. Then all α_w are $tK\Lambda(\mathcal{A})$-awareness function, and hence M^{+} is a $tK\Lambda(\mathcal{A})$-structure. It can then be checked by induction that for any MEL formula ϕ and any t-function τ on ϕ, $M, w \Vdash \phi$ if and only if $M^{+}, w \Vdash \phi^{\tau}$ for any $w \in M$. This completes the proof. Now suppose there is no τ such that ϕ^{τ} is $tK\Lambda$-valid, then for all τ on ϕ, $\sim\phi^{\tau}$ is satisfiable. By Theorem 4, there is a structure M and a world w of the structure such that for all τ, $\sim\phi^{\tau}$ is satisfiable at (M, w). And then by Theorem 5, we can assume M is configurational. At last, by Theorem 6, $\sim\phi$ is satisfied at (M°, w). A contradiction. So the other direction and the whole theorem is proved.

Corollary 2. *Given a full $tK\Lambda$-logical base $\mathcal{A}$, ϕ is $K\Lambda$ valid if and only if there is temporalization function τ on ϕ such that ϕ^{τ} is $tK\Lambda(\mathcal{A})$ valid.*

5 Conclusion

In this paper we render a semantic proof to the Temporalization Themorem, and the proof itself, which includes several lemmas and theorems, also gives us a closer look of the tMEL secmantics and its relation to the MEL semantics. Taken as an example, the Theorem 6 gives us the idea that the pattern of the truth-values of the formulas in the tail of such a sequence: $\{\phi^{\tau_i}\}_{i \in \mathbb{N}}$ with

$\tau_i < \tau_{i+1}$ in a world of a tMEL structure is an indication of the truth value of ϕ in the same world of the underlying MEL structure. For the future work, we hope we can extend the method used here to provide a semantic proof for the Realization Theorem in Justification Logic, which is varied from tMEL in the way that, briefly, it is the proof terms, which enjoy a more complicated structure, that plays the roles in Justification Logic as that played by natural numbers in tMEL.

References

[1] Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)

[2] Artemov, S.N.: The Logic of Justification. The Review of Symbolic Logic 1(4), 477–513 (2008)

[3] Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial intelligence 42(2-3), 213–261 (1990)

[4] Duc, H.N.: Reasoning about rational, but not logically omniscient, agent. Journal of Logic and Computation 5, 633–648 (1997)

[5] Duc, H.N.: On the Epistemic Foundations of Agent Theories. In: Rao, A., Singh, M.P., Wooldridge, M.J. (eds.) ATAL 1997. LNCS, vol. 1365, pp. 275–279. Springer, Heidelberg (1998)

[6] Elgot-drapkin, J.J., Perlis, D.: Reasoning situation in time I: Basic concepts. Journal of Experimental and Theoretical Artificial Intelligence 2(1), 75–98 (1990)

[7] Fagin, R., Halpern, J.Y.: Belief, awareness, and limited reasoning. Artificial Intelligence 34(1), 39–76 (1988)

[8] Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.: Reasoning About Knowldge. MIT Press (1995)

[9] Fitting, M.C.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132, 1–25 (2005)

[10] Haas, A.: Possible events, actual events, and robots. Computational Intelligence 1(1), 59–70 (1985)

[11] Hintikka, J.: Knowledge and belief. Cornell University Press (1962)

[12] Konolige, K.: A Deduction Model of Belief. Morgan Kaufmann, San Mateo (1986)

[13] Levesque, H.J.: A logic of implicit and explicit belief. In: Proceedings of the National Conference on Artificial Intelligence, Austin, TX, pp. 198–202 (1984)

[14] McDermott, D.: Planning and Acting. Cognitive Science 2(2), 71–109 (1978)

[15] Nirkhe, M., Kraus, S., Miller, M.J., Perlis, D.: How to (Plan to) Meet a Deadline Between Now and Then. Journal of Logic and Computation 7(1), 109–156 (1997)

[16] Rantala, V.: Impossible worlds semantics and logical omniscience. Acta Philosophica Fennica 35, 18–24 (1982)

[17] Wang, R.-J.: Knowledge, Time, and Logical Omniscience. In: Ono, H., Kanazawa, M., de Queiroz, R. (eds.) WoLLIC 2009. LNCS (LNAI), vol. 5514, pp. 394–407. Springer, Heidelberg (2009)

[18] Wang, R.-J.: Knowledge, Time, and the Problem of Logical Omniscience. Fundamenta Informaticae 106(2-4), 321–338 (2011)

[19] Wang, R.-J.: On Non-Circular Proofs and Proof Realization in Modal logic. Technical Report TR-2011012, CUNY PhD Program in Computer Science (2011)

[20] Wooldridge, M.: An introduction to multiagent systems. Wiley (2009)

Contextual Natural Deduction

Bruno Woltzenlogel Paleo[1,2]

[1] INRIA, Nancy, France
[2] Theory and Logic Group, Vienna University of Technology, Vienna, Austria
bruno@logic.at

Abstract. This paper defines the *contextual* natural deduction calculus $\mathbf{ND^c}$ for the implicational fragment of intuitionistic logic. $\mathbf{ND^c}$ extends the usual natural deduction calculus (here called **ND**) by allowing the implication introduction and elimination rules to operate on formulas that occur inside contexts. In analogy to the Curry-Howard isomorphism between **ND** and the simply-typed λ-calculus, an extension of the λ-calculus, here called λ^c-calculus, is defined in order to provide compact proof-terms for $\mathbf{ND^c}$ proofs. Soundness and completeness of $\mathbf{ND^c}$ with respect to **ND** are proven by defining translations of proofs between these calculi. Furthermore, some $\mathbf{ND^c}$-proofs are shown to be quadratically smaller than the smallest **ND**-proofs of the same theorems.

1 Introduction

Natural deduction was introduced by Gentzen in [10] and one of its distinguishing features is that the meaning of a logical connective is determined by elimination and introduction rules, and not by axioms. As a result, formal natural deduction proofs are considered to be similar in structure to their informal counterparts and hence more *natural*. This subjective claim is corroborated by the observation that widely used proof assistants[1] follow a natural deduction style.

However, as exemplified in Section 2.1, the inference rules of natural deduction style calculi can be inconvenient, lengthy and ultimately unnatural for formalizing reasoning steps that modify a deeply located subformula of a formula, such as: skolemization, double negation elimination, quantifier shifting, prenexification... Because these deep reasoning steps are commonly used by automated deduction tools during preprocessing of the theorem to be proved, the resulting proofs may contain deep inferences [8]. Therefore, automatically replaying (i.e. reproving) these proofs in proof assistants (e.g. when an automated deduction tool is integrated within a proof assistant [2]) can be inefficient in terms of proving time and size of the generated shallow proof.

These issues serve as long-term practical and technological motivations for the *contextual natural deduction calculus* presented here, which is a simple extension of the usual natural deduction calculus allowing introduction and elimination rules to operate on formulas occurring inside contexts. The goals of the current paper, however, are purely theoretical. In particular, it is shown that the

[1] e.g. Isabelle (www.cl.cam.ac.uk/research/hvg/Isabelle/) and Coq (http://coq.inria.fr).

proposed calculus is sound and complete, that proofs can be normalized, and that some proofs can be quadratically smaller than corresponding proofs in the usual natural deduction calculus. For the sake of succintness, only minimal logic (intuitionistic logic with only the implication connective) is considered, since it is straightforward to extend the techniques described here to natural deduction calculi containing inference rules for other connectives as well.

Related Work. The related idea of *deep inference* has been intensively investigated in the last decade, especially for classical logic (e.g. [3,6,5,12]). More recently, the techniques and calculi originally developed for classical logic have been adapted for intuitionisitc logic. In Tiu's calculus [16], inferences are not only deep but also local (i.e. only a fixed amount of information needs to be checked when applying a rule). Brünnler and McKinley [4] proposed another calculus with an algorithmic interpretation inspired by the Curry-Howard isomorphism. And in Guenot's calculus [11], it is possible to encode lambda terms with explicit substitutions as formulas in such a way that computation corresponds to proof-search. Although these calculi differ significantly from each other, they adher to the original deep inference methodology of avoiding branching inference rules. The calculus presented here, on the other hand, does utilize branching, since the aim is a minimal modification of the usual *branching* natural deduction calculus.

2 Deep Natural Deduction

Figure 1 presents the inference rules of a standard natural deduction calculus **ND** for the implicational fragment of intuitionistic logic. An **ND**-*derivation* is a tree of inferences (instances of the inference rules) and a derivation ψ is an **ND**-*proof* of a theorem T if and only if its leaves are axiom inferences and it ends in $\vdash t : T$, for some term t of type T of the simply typed λ-calculus. This term t is called the *proof-term* of ψ and denoted $\mathcal{I}(\psi)$. As indicated in the figure, the implication introduction rule corresponds to abstraction while the implication elimination rule corresponds to application in the λ-calculus. $\mathcal{I}$ is thus an isomorphism (Curry-Howard [7]) between **ND**-proofs and simply typed λ-terms and between (implicational intuitionistic) theorems and types.

Figure 2 shows the inference rules of $\mathbf{ND^c}$, the *contextual natural deduction calculus* that aims at extending **ND** in a simple, minimal, straightforward and yet general way by allowing the inference rules to operate on subformulas located deeply inside the premises. The notation $\mathcal{C}_\pi[F]$ indicates a formula that has the subformula F in position π. $\mathcal{C}_\pi[\_]$ is called the *context* of F in the formula $\mathcal{C}_\pi[F]$. Concretely in this paper, a *position* π is encoded as a binary string indicating the path from the root of $\mathcal{C}_\pi[F]$ to F in the tree structure of $\mathcal{C}_\pi[F]$; thus, a subformula at position π of a formula P, denoted $\mathrm{At}_\pi(P)$, can be retrieved by traversing the formula according to the following inductive definition:

$$\mathrm{At}_\epsilon(A) = A \qquad \mathrm{At}_{0\pi}(A \to B) = \mathrm{At}_\pi(B) \qquad \mathrm{At}_{1\pi}(A \to B) = \mathrm{At}_\pi(A)$$

$$\frac{}{\Gamma, a : A \vdash a : A}$$

$$\frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash \lambda a^A.b : A \to B} \to_I$$

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash (f\ a) : B} \to_E$$

Fig. 1. The natural deduction calculus **ND**

A position is said to be *positive* (*negative*) if and only if it contains an even (odd) number of digits 1. In other words, in the tree structure of a formula, a node and its left (right) child always occupy positions with opposite (same) polarities, and the root position is positive. Moreover, a position is *strongly positive* if and only if it does not contain any digit 1.

Note: π, π_1 and π_2 must be positive positions.

$$\frac{}{\Gamma, a : A \vdash a : A}$$

$$\frac{\Gamma, a : A \vdash b : \mathcal{C}_\pi[B]}{\Gamma \vdash \lambda_\pi a^A.b : \mathcal{C}_\pi[A \to B]} \to_I (\pi)$$

Intuitionistic Contextual Soundness Condition:
a is allowed to occur in b only if π is strongly positive.

$$\frac{\Gamma \vdash f : \mathcal{C}^1_{\pi_1}[A \to B] \qquad \Gamma \vdash a : \mathcal{C}^2_{\pi_2}[A]}{\Gamma \vdash (f\ a)^{\rightharpoonup}_{(\pi_1;\pi_2)} : \mathcal{C}^1_{\pi_1}[\mathcal{C}^2_{\pi_2}[B]]} \to_E^{\rightharpoonup} (\pi_1;\pi_2)$$

$$\frac{\Gamma \vdash f : \mathcal{C}^1_{\pi_1}[A \to B] \qquad \Gamma \vdash a : \mathcal{C}^2_{\pi_2}[A]}{\Gamma \vdash (f\ a)^{\leftharpoonup}_{(\pi_1;\pi_2)} : \mathcal{C}^2_{\pi_2}[\mathcal{C}^1_{\pi_1}[B]]} \to_E^{\leftharpoonup} (\pi_1;\pi_2)$$

Fig. 2. The contextual natural deduction calculus $\mathbf{ND^c}$

The contextual natural deduction calculus $\mathbf{ND^c}$ has two implication elimination rules, because the implication elimination rule of **ND** is extendable in two different ways, depending on the order in which the contexts are combined.

The proof-term of an $\mathbf{ND^c}$-proof ψ is denoted $\mathcal{I}_c(\psi)$. As shown in Figure 2, proof terms for $\mathbf{ND^c}$-proofs, here called λ^c-*terms* (*contextual* λ-*terms*), must be modified accordingly: the positions in which applications and abstractions are performed are indicated as subscripts, and the superscript arrows on applications inform the order in which the contexts are combined. In case at least one of the contexts is empty, the order does not matter, and hence the superscript arrow can be omitted. Likewise, if all positions are empty (ϵ), the subscript can be omitted. By these omission conventions, contextual applications and abstractions with empty positions can be written like ordinary ones.

No special parenthesis or precedence convention is used for λ- and λ^c-terms in this paper. As indicated in the construction rules in Figures 1 and 2, applications are always surrounded by parentheses, and parentheses are not used for abstractions. It follows that a term of the form $(\lambda x.t\ u)$ should be understood as the function $\lambda x.t$ applied to the argument u.

2.1 Comparing ND and $\mathbf{ND^c}$

Many preprocessing techniques change deeply occurring subformulas. In classical first-order resolution theorem proving, for example, the formula

$$\neg\neg\exists x.P(x) \vee \forall y.Q(y)$$

would be first transformed into the following formula in clause form

$$P(c) \vee Q(v)$$

where c is a new Skolem constant, and v is a free-variable.

All three steps used in this transformation (double negation elimination, skolemization, and the dropping of universal quantifiers) perform deep modifications in the formula. As the following examples show, formalizing these deep steps in **ND** is lengthy, inefficient and unnatural. The reason is that many introduction and elimination rules must be performed in order to decompose a formula until the subformula that has to be modified is reached. In contrast, in $\mathbf{ND^c}$ a deep modification can be done with a single inference, independently of the depth at which the target subformula occurs.

Example 1. Skolemization can be simulated in **ND** and $\mathbf{ND^c}$ by assuming the axiom schema

$$sk : \exists x.F[x] \rightarrow F[f_{sk}(x_1, \ldots, x_n)]$$

where $x_1, \ldots, x_n$ are the free-variables of F and f_{sk} must be a globally new[2] (Skolem) function symbol.

Given the formula $(A \rightarrow B) \rightarrow \exists x.P(x)$, Skolemization replaces the subformula $\exists x.P(x)$ at position 0 in the given formula by $P(c)$, for some Skolem constant c. In $\mathbf{ND^c}$, this can be formalized with a single implication elimination inference deriving $(A \rightarrow B) \rightarrow P(c)$ from $(A \rightarrow B) \rightarrow \exists x.P(x)$ and an instance of the Skolemization axiom schema:

$$\frac{\vdash sk : \exists x.P(x) \rightarrow P(c) \qquad a : \ldots \vdash a : (A \rightarrow B) \rightarrow \exists x.P(x)}{a : (A \rightarrow B) \rightarrow \exists x.P(x) \vdash (sk\ a)^{\leftarrow}_{(\epsilon;0)} : (A \rightarrow B) \rightarrow P(c)} \rightarrow^{\leftarrow}_{E}\ (\epsilon; 0)$$

[2] The only leaf of the proof in which f_{sk} should occur is the leaf where f_{sk} is introduced by a skolem axiom instance.

In **ND**, on the other hand, the proof is larger and arguably less natural:

$$\dfrac{\vdash sk : \exists x.P(x) \rightarrow P(c) \qquad \dfrac{c : \ldots \vdash c : A \rightarrow B \qquad a : \ldots \vdash a : (A \rightarrow B) \rightarrow \exists x.P(x)}{a : (A \rightarrow B) \rightarrow \exists x.P(x), c : A \rightarrow B \vdash (a\ c) : \exists x.P(x)}\rightarrow_E}{\dfrac{c : A \rightarrow B, a : (A \rightarrow B) \rightarrow \exists x.P(x) \vdash (sk\ (ac)) : P(c)}{a : (A \rightarrow B) \rightarrow \exists x.P(x) \vdash \lambda c^{A \rightarrow B}.(sk\ (a\ c)) : (A \rightarrow B) \rightarrow P(c)}\rightarrow_I}\rightarrow_E$$

□

Example 2. Double negation elimination can be simulated in **ND** and **ND^c** by assuming the axiom schema

$$dne : \neg\neg F \rightarrow F$$

Given the formula $(\neg\neg A \rightarrow B) \rightarrow C$, double negation elimination replaces the subformula $\neg\neg A$ at position 11 in the given formula by A. In $\mathbf{ND^c}$, this can be formalized with a single implication elimination inference deriving $(A \rightarrow B) \rightarrow C$ from the given formula and an instance of the double negation elimination axiom schema:

$$\dfrac{\vdash dne : \neg\neg A \rightarrow A \qquad a : \ldots \vdash a : (\neg\neg A \rightarrow B) \rightarrow C}{a : (\neg\neg A \rightarrow B) \rightarrow C \vdash (dne\ a)^{\leftarrow}_{(\epsilon;11)} : (A \rightarrow B) \rightarrow C} \rightarrow^{\leftarrow}_{E}\ (\epsilon; 11)$$

In **ND**, on the other hand, the proof is again larger and less natural:

$$\dfrac{\dfrac{a : \ldots \vdash a : (\neg\neg A \rightarrow B) \rightarrow C \qquad \dfrac{\dfrac{c : \ldots \vdash c : A \rightarrow B \qquad \dfrac{\vdash dne : \neg\neg A \rightarrow A \qquad d : \ldots \vdash d : \neg\neg A}{d : \ldots \vdash (dne\ d) : A}\rightarrow_E}{d : \ldots, c : \ldots \vdash (c\ (dne\ d)) : B}\rightarrow_E}{c : \ldots \vdash \lambda d^{\neg\neg A}.(c\ (dne\ d)) : \neg\neg A \rightarrow B}\rightarrow_I}{a : \ldots, c : \ldots \vdash (a\ \lambda d.(c\ (dne\ d))) : C}\rightarrow_E}{a : (\neg\neg A \rightarrow B) \rightarrow C \vdash \lambda c^{A \rightarrow B}.(a\ \lambda d.(c\ (dne\ d))) : (A \rightarrow B) \rightarrow C}\rightarrow_I$$

□

3 Soundness and Completeness

In order to prove that $\mathbf{ND^c}$ is sound and complete with respect to **ND**, it must be shown that a formula is provable in $\mathbf{ND^c}$ iff it is provable in **ND**. This can be done by defining a translation of proofs from one calculus into the other. In order to do so more compactly, the next two subsections will show how to translate not proofs themselves but rather their proof terms.

3.1 Translating $\boldsymbol{\lambda}$-terms into $\boldsymbol{\lambda^c}$-terms

The translation of λ-terms into λ^c-terms of the same type is trivial, since λ-terms can be regarded as λ^c-terms with empty positions.

Definition 1. *The translation function ζ, from λ-terms to λ^c-terms of the same type, is defined inductively on the structure of λ-terms:*

- $\zeta[v] = v$ *(for a variable v).*
- $\zeta[\lambda v^T.t] = \lambda_\epsilon v^T.\zeta[t]$
- $\zeta[(m\ n)] = (\zeta[m]\ \zeta[n])_{(\epsilon;\epsilon)}$

3.2 Translating λ^c-terms into λ-terms

The translation of λ^c-terms into λ-terms of the same type can be done by means of the function ξ defined below.

Definition 2. *The translation function ξ, which maps a λ^c-term t into a shallow λ-term of the same type, is defined inductively on the structure of t:*

- *If t is a variable, then $\xi[t] = t$*
- *If t is an abstraction of the form $\lambda_\pi a^A.b$, the translation is defined by induction on the position π, according to the cases below:*
  - *If $\pi = 0\pi'$, it is the case that t matches $\lambda_{0\pi'} a^A.b^{C\to D}$, and then*

$$\xi[t] = \lambda c^C.\xi[\lambda_{\pi'} a^A.(bc)]$$

  - *If $\pi = 1\pi'$, then there is at least one occurrence of the digit 1 in π', since π is positive and π' is negative. Therefore, π is necessarily of the form $10\dots01\pi''$ and t matches $\lambda_{10\dots01\pi''} a^A.b^{(C_1\to\dots C_n\to(T_{\pi''}[B]\to D_1))\to D_2}$. Then*

$$\xi[t] = \lambda k^{C_1\to\dots C_n\to(T_{\pi''}[A\to B]\to D_1)}.(b\ \lambda c_1^{C_1}\dots c_n^{C_n}.\lambda h^{T_{\pi''}[B]}.(k\ c_1\dots c_n\ \xi[\lambda_{\pi''} a^A.h])$$

    Note that, if a occurred in b, then this occurrence would become unbound after the translation. This undesirable effect, which would jeopardize soundness as shown in Example 3, is prevented by the intuitionistic contextual soundness condition *described in Figure 2.*
  - *If $\pi = \epsilon$, it is the case that t matches $\lambda_\epsilon a.b$, and then*

$$\xi[t] = \lambda a.\xi[b]$$

- *If t is an $(\rightharpoonup)$-application of the form $(f\ a)^{\rightharpoonup}_{(\pi_1;\pi_2)}$, the translation is defined by two successive inductions, firstly on the position π_1 and then (when $\pi_1 = \epsilon$) on π_2, according to the cases below:*
  - *If $\pi_1 = 0\pi$, it is the case that t matches $(f^{C\to D}\ a)^{\rightharpoonup}_{(0\pi;\pi_2)}$, and then*

$$\xi[t] = \lambda c^C.\xi[((f\ c)\ a))^{\rightharpoonup}_{(\pi;\pi_2)}]$$

  - *If $\pi_1 = 1\pi'$, then there is at least one occurrence of the digit 1 in π', since π_1 is positive and π' is negative. Therefore, π_1 is necessarily of the form $10\dots01\pi$ and t matches $(f^{(C_1\to\dots C_n\to(T_\pi[A\to B]\to D_1))\to D_2}\ a)^{\rightharpoonup}_{(10\dots01\pi;\pi_2)}$. Then*

$$\xi[t] = \lambda k^{C_1\to\dots C_n\to(T_\pi[B]\to D_1)}.(f\ \lambda c_1^{C_1}\dots c_n^{C_n}.\lambda h^{T_\pi[A\to B]}.(k\ c_1\dots c_n\ \xi[(h\ a)^{\rightharpoonup}_{(\pi;\pi_2)}])$$

  - *If $\pi_1 = \epsilon$ and $\pi_2 = 0\pi$, it is the case that t matches $(f\ a^{C\to D})^{\rightharpoonup}_{(\epsilon;0\pi)}$, and then*

$$\xi[t] = \lambda c^C.\xi[(f\ (a\ c))^{\rightharpoonup}_{(\epsilon;\pi)}]$$

  - *If $\pi_1 = \epsilon$ and $\pi_2 = 1\pi'$, then there is at least one occurrence of the digit 1 in π', since π_2 is positive and π' is negative. Therefore, π_2 is of the form $10\dots01\pi$ and t matches $(f^{A\to B}\ a^{(C_1\to\dots C_n\to(T_\pi[A]\to D_1))\to D_2})^{\rightharpoonup}_{(\epsilon;10\dots01\pi)}$. Then*

$$\xi[t] = \lambda k^{C_1\to\dots C_n\to(T_\pi[B]\to D_1)}.(a\ \lambda c_1^{C_1}\dots c_n^{C_n}.\lambda h^{T_\pi[A]}.(k\ c_1\dots c_n\ \xi[(f\ h)^{\rightharpoonup}_{(\epsilon;\pi)}]))$$

- *If $\pi_1 = \pi_2 = \epsilon$, it is the case that t matches $(f\ a)_{(\epsilon;\epsilon)}^{\rightharpoonup}$, and then*

$$\xi[t] = (\xi[f]\ \xi[a])$$

– *If t is an $(\leftharpoonup)$-application of the form $(f\ a)_{(\pi_1;\pi_2)}^{\leftharpoonup}$, the translation is analogous to the previous case for $(f\ a)_{(\pi_1;\pi_2)}^{\rightharpoonup}$, but the induction is made firstly on the position π_2 and only then (when $\pi_2 = \epsilon$) on π_1.*

3.3 Soundness and Completeness of $\mathbf{ND^c}$ Relative to ND

With the translation functions defined in the previous subsections, soundness and completeness can be proved shortly, as shown in Theorems 1 and 2 below.

Theorem 1 (Completeness).
*If T is provable in **ND**, then T is provable in $\boldsymbol{ND^c}$.*

Proof. Let ψ be an **ND**-proof of T, then $\mathcal{I}_c^{-1}(\zeta[\mathcal{I}(\psi)])$ is an $\mathbf{ND^c}$-proof of the same theorem T. □

Theorem 2 (Soundness).
*If T is provable in $\boldsymbol{ND^c}$, then T is provable in **ND**.*

Proof. Let ψ be an $\mathbf{ND^c}$-proof of T, then $\mathcal{I}^{-1}(\xi[\mathcal{I}_c(\psi)])$ is an **ND**-proof of the same theorem T. □

Example 3. The importance of the intuitionistic contextual soundness condition can be illustrated with the following incorrect $\mathbf{ND^c}$-proof. Its lowermost inference violates the condition in order to derive Peirce's Law, which should not be intuitionistically derivable.

$$\dfrac{\dfrac{g : (B \to A), a : A \vdash a : A}{a : A \vdash \lambda g^{(B\to A)}.a : (B \to A) \to A} \to_I (\epsilon)}{\vdash \lambda_{11} a^A.\lambda g^{(B\to A)}.a : ((A \to B) \to A) \to A} \to_I (11)$$

Translating the proof-term using ξ results in the following λ-term, where a has become unbound: $\lambda k^{((A\to B)\to A)}.((\lambda g^{(B\to A)}.a)\ \ (\lambda h^B.(k\ \lambda a^A.h)))$. As a is unbound, there is no closed **ND**-proof corresponding to this λ-term. The intuitionistic contextual soundness condition guarantees that there will be no variable a becoming unbound in this manner. □

4 Normalization

In λ-calculus, a term t of the form $(\lambda a^A.f\ t')$ is a *redex* with respect to β-reduction, and is β-reducible to $f[a\backslash t']$. Correspondingly, the proof $\mathcal{I}^{-1}(t)$ is said to contain a *detour* (*cut*). The normalization rules for eliminating detours correspond to β-reduction and the **ND**-proof $\mathcal{I}^{-1}(t)$ can be rewritten to $\mathcal{I}^{-1}(f[a\backslash t'])$.

The purpose of this section is to extend this notion of *normalization* to $\mathbf{ND^c}$ and to the corresponding λ^c-calculus. As the relation between $\mathbf{ND^c}$-proofs and λ^c-terms is given by the bijection $\mathcal{I}_c$, it suffices to consider the λ^c-calculus.

Unfortunately, the notion of β-redex is more complicated in λ^c-calculus. Consider, for example, the following λ^c-term:

$$(\lambda_0 a^A . f\ t')_{(0;\epsilon)}$$

Its form is very similar to the redex form described above, except for the fact that abstraction and application are now *contextual*. Therefore, it is intuitively clear (and it could be shown by using the translation function ξ) that this term could be reduced to $f[a \backslash t']$, and hence ought to be considered a redex. Nevertheless, it would not be sufficient to extend β-reduction to λ^c-terms of this form. The λ^c-term below, for example, is not of this form, but it is also clear (again by using ξ) that it could be reduced to $\lambda b^B.(f[a \backslash t']\ b)$:

$$(\lambda b^B.\lambda a^A.(f\ b)\ t')_{(0;\epsilon)}$$

As the term above exemplifies, the issue is that the abstraction (in this case, λa^A) that ought to be eliminated together with the outermost contextual application can occur arbitrarily deep inside the left subterm of the application (in this case, $\lambda b^B.\lambda a^A.(f\ b)$). Generalizing β-reduction so that it could directly handle all such potential redexes formed by pairs of contextual applications and deeply occurring abstractions would not be easy. Therefore, instead of generalizing the β-reduction rule, an alternative approach is considered in the next subsection.

4.1 Unfolding

For redexes made of contextual abstractions and applications with empty positions (which, by the position omission convention, can be written like ordinary abstractions and applications), β-reduction can be applied normally. The problem lies only in the presence of contextual applications and abstractions with non-empty positions, which may block β-reduction. This problem can be solved by translating the λ^c-term only partially, in order to remove the blocking contextual applications and abstractions. To this aim, a term rewriting system based on the translation function ξ is defined in Figure 3. The rewriting of a term t to a term t' using one of the rules of this term rewriting system is denoted $t \rightsquigarrow_\delta t'$; and it is said that the term t *unfolds* to t'.

Example 4. Consider the term $(\lambda_0 a^A.(\lambda_0 b^B.h\ a)\ t')_{(0;\epsilon)}$. Even though it is clear that this term is a deep redex, β-reduction alone is not capable of producing the desired result $(\lambda_0 b^B.h\ t')$. As shown in the rewriting sequence below, the unfolding of the outermost contextual application and of the outermost contextual abstraction enables the use of β-reduction.

$$\frac{\lambda_{0\pi} a^A . b^{C\to D}}{\lambda c^C . \lambda_\pi a^A . (bc)}$$

$$\frac{\lambda_{10\ldots01\pi} a^A . f^{(C_1\to\ldots C_n\to(T_\pi[B]\to D_1))\to D_2}}{\lambda k^{C_1\to\ldots C_n\to(T_\pi[A\to B]\to D_1)} . (f\ \lambda c_1^{C_1}\ldots c_n^{C_n} . \lambda h^{T_\pi[B]} . (k\ c_1\ldots c_n\ \lambda_\pi a^A . h)}$$

$$\frac{(f^{C\to D}\ a)^{\rightharpoonup}_{(0\pi;\pi_2)}}{\lambda c^C . ((f\ c)\ a))^{\rightharpoonup}_{(\pi;\pi_2)}} \qquad \frac{(f\ a^{C\to D})^{\leftharpoonup}_{(\pi_1;0\pi)}}{\lambda c^C . (f\ (a\ c))^{\leftharpoonup}_{(\pi_1;\pi)}}$$

$$\frac{(f^{(C_1\to\ldots C_n\to(T_\pi[A\to B]\to D_1))\to D_2}\ a)^{\rightharpoonup}_{(10\ldots01\pi;\pi_2)}}{\lambda k^{C_1\to\ldots C_n\to(T_\pi[B]\to D_1)} . (f\ \lambda c_1^{C_1}\ldots c_n^{C_n} . \lambda h^{T_\pi[A\to B]} . (k\ c_1\ldots c_n\ (h\ a)^{\rightharpoonup}_{(\pi;\pi_2)})}$$

$$\frac{(f\ a^{(C_1\to\ldots C_n\to(T_\pi[A]\to D_1))\to D_2})^{\leftharpoonup}_{(\pi_1;10\ldots01\pi)}}{\lambda k^{C_1\to\ldots C_n\to(T_\pi[B]\to D_1)} . (a\ \lambda c_1^{C_1}\ldots c_n^{C_n} . \lambda h^{T_\pi[A]} . (k\ c_1\ldots c_n\ (f\ h)^{\leftharpoonup}_{(\pi_1;\pi)}))}$$

$$\frac{(f^{A\to B}\ a^{(C_1\to\ldots C_n\to(T_\pi[A]\to D_1))\to D_2})^{\rightharpoonup}_{(\epsilon;10\ldots01\pi)}}{\lambda k^{C_1\to\ldots C_n\to(T_\pi[B]\to D_1)} . (a\ \lambda c_1^{C_1}\ldots c_n^{C_n} . \lambda h^{T_\pi[A]} . (k\ c_1\ldots c_n\ (f\ h)^{\rightharpoonup}_{(\epsilon;\pi)}))}$$

$$\frac{(f^{(C_1\to\ldots C_n\to(T_\pi[A\to B]\to D_1))\to D_2}\ a)^{\leftharpoonup}_{(10\ldots01\pi;\epsilon)}}{\lambda k^{C_1\to\ldots C_n\to(T_\pi[B]\to D_1)} . (f\ \lambda c_1^{C_1}\ldots c_n^{C_n} . \lambda h^{T_\pi[A\to B]} . (k\ c_1\ldots c_n\ (h\ a)^{\leftharpoonup}_{(\pi;\epsilon)})}$$

$$\frac{(f\ a^{C\to D})^{\rightharpoonup}_{(\epsilon;0\pi)}}{\lambda c^C . (f\ (a\ c))^{\rightharpoonup}_{(\epsilon;\pi)}} \qquad \frac{(f^{C\to D}\ a)^{\leftharpoonup}_{(0\pi;\epsilon)}}{\lambda c^C . ((f\ c)\ a)^{\leftharpoonup}_{(\pi;\epsilon)}}$$

Fig. 3. Term Rewriting Rules for Unfolding

$$\begin{aligned}
(\lambda_0 a^A . (\lambda_0 b^B . h\ a)\ t')_{(0;\epsilon)} &\rightsquigarrow_\delta \quad (\lambda b_1^B . \lambda a^A . ((\lambda_0 b^B . h\ a)\ b_1)\ t')_{(0;\epsilon)} \\
&\rightsquigarrow_\delta \quad \lambda b_2^B . ((\lambda b_1^B . \lambda a^A . ((\lambda_0 b^B . h\ a)\ b_1)\ b_2)\ t') \\
&\rightsquigarrow_\beta \quad \lambda b_2^B . (\lambda a^A . ((\lambda_0 b^B . h\ a)\ b_2)\ t') \\
&\rightsquigarrow_\beta \quad \lambda b_2^B . ((\lambda_0 b^B . h\ t')\ b_2) \\
&=_\eta \quad (\lambda_0 b^B . h\ t')
\end{aligned}$$ □

The following theorems show that unfolding is terminating and confluent.

Definition 3. *The* size $s(\pi)$ *of a position* π *is the number of digits in* π.

Theorem 3. $\rightsquigarrow_\delta$ *is strongly terminating.*

Proof. Let μ be a function that counts the total size of all positions in an λ^c-term, inductively defined as follows:

- $\mu(v) = 0$ (for a variable v)
- $\mu(\lambda_\pi a.b) = s(\pi) + \mu(b)$
- $\mu((f\ a)^{\rightharpoonup}_{(\pi_1;\pi_2)}) = s(\pi_1) + s(\pi_2) + \mu(f) + \mu(a)$
- $\mu((f\ a)^{\leftharpoonup}_{(\pi_1;\pi_2)}) = s(\pi_1) + s(\pi_2) + \mu(f) + \mu(a)$

Note that, for all rules in Figure 3, if $t \rightsquigarrow_\delta t'$, then $\mu(t') < \mu(t)$. Since there can be no infinite decreasing chain of natural numbers, there cannot be an infinite $\rightsquigarrow_\delta$ reduction sequence either. Therefore $\rightsquigarrow_\delta$ is strongly terminating. □

Theorem 4. $\rightsquigarrow_\delta$ *is confluent.*

Proof. (Sketch) Note that there are no critical pairs. Therefore, by the *critical pair theorem* (Theorem 6.2.4 in [1]), $\rightsquigarrow_\delta$ is locally confluent. By Newman's Lemma (Lemma 2.7.2 in [1]), a terminating and locally confluent term rewriting system must be confluent.

If the shallow β-reduction rule is added to the term rewriting rules shown in Figure 3, a new *higher-order* term rewriting system is obtained. The rewriting of a term t to a term t' according to this term rewriting system is denoted $t \rightsquigarrow_{\beta\delta} t'$; and it is said that the term t *beta-unfolds* to t'. Formally:

$$\rightsquigarrow_{\beta\delta} = \rightsquigarrow_\delta \cup \rightsquigarrow_\beta$$

The combined term-rewriting system also enjoys termination and confluence.

Theorem 5. $\rightsquigarrow_{\beta\delta}$ *is terminating.*

Proof. (Sketch) The traditional approach of proving termination of combined terminating term rewriting systems via commutation lemmas [9] does not work here, because the conditions required by those lemmas do not hold for $\rightsquigarrow_\beta$ and $\rightsquigarrow_\delta$. Their combination is nevertheless terminating. For the sake of contradiction, assume that there could be an infinite $\rightsquigarrow_{\beta\delta}$ reduction sequence. Since $\rightsquigarrow_\beta$ and $\rightsquigarrow_\delta$ are both terminating, this infinite sequence would need to contain alternations of $\rightsquigarrow_\beta$ and $\rightsquigarrow_\delta$ steps in such a way that $\rightsquigarrow_\delta$ generates β-redexes and $\rightsquigarrow_\beta$ generates δ-redexes. However, by inspection of the unfolding and β-reduction rules, this is not possible. β-reduction never creates new δ-redexes (although it can duplicate existing ones). Unfolding never duplicates existing β-redexes. And although it can create new β-redexes, reducing them does not cause any duplication, because the abstractions introduced by unfolding are all linear (i.e. the abstracted variables k, c, c_1,…, c_n occur only once in the respective term's body). Hence, in any $\rightsquigarrow_{\beta\delta}$ reduction sequence, duplications of $\rightsquigarrow_\delta$-redexes are finite, limited by the non-linear abstractions (which may participate in duplicating β-reductions) already occuring in the initial term. Therefore, no infinite reduction sequence is possible and thus $\rightsquigarrow_{\beta\delta}$ is terminating. □

Theorem 6. *$\rightsquigarrow_{\beta\delta}$ is confluent.*

Proof. The term rewriting system for unfolding is a first-order term rewriting system. β-reduction is a particular kind of higher-order rewriting rule known as *pattern rewriting rule* [14]. Therefore, their combination is a *pattern rewriting system (PRS).* Since there are no critical pairs in the combined system, a generalization of the critical pair theorem for PRSs (Theorem 4.7 in [14]) can be used to conclude that $\rightsquigarrow_{\beta\delta}$ is locally confluent. Therefore, as $\rightsquigarrow_{\beta\delta}$ is terminating, Newman's Lemma allows to conclude that $\rightsquigarrow_{\beta\delta}$ is confluent. □

5 Sizes of Proofs in ND and ND$^\mathbf{c}$

As seen in Examples 1 and 2, **ND$^\mathbf{c}$**-proofs can be significantly smaller than **ND**-proofs. In order to measure how much smaller they can be, the notion of *size* must first be precisely defined. This is done below for types and terms, essentially by counting the number of symbols they contain. The size $s(\psi)$ of a **ND**-proof ψ is simply defined as the size $s(\mathcal{I}(\psi))$ of its corresponding proof-term $\mathcal{I}(\psi)$. Likewise, the *size* $s(\psi)$ of a **ND$^\mathbf{c}$**-proof ψ is $s(\mathcal{I}_c(\psi))$.

Definition 4. *The* size $s(T)$ *of a type T is inductively defined as follows:*

– $s(A) = 1$ *(if A is an atomic type)*
– $s(T_1 \to T_2) = 1 + s(T_1) + s(T_2)$

Definition 5. *The* size $s(t)$ *of a λ-term t is inductively defined as follows:*

– $s(v) = 1$ *(if v is a variable)*
– $s(\lambda v^T.t') = 1 + s(v) + s(T) + s(t') = 2 + s(T) + s(t')$
– $s((m\ n)) = 1 + s(m) + s(n)$

Definition 6. *The* size $s(t)$ *of a λ^c-term t is inductively defined as follows:*

– $s(v) = 1$ *(if v is a variable)*
– $s(\lambda_\pi v^T.t') = 1 + s(\pi) + s(v) + s(T) + s(t') = 2 + s(\pi) + s(T) + s(t')$
– $s((m\ n)^{\rightharpoonup}_{\pi_1;\pi_2}) = 1 + s(m) + s(n) + s(\pi_1) + s(\pi_2)$
– $s((m\ n)^{\leftharpoonup}_{\pi_1;\pi_2}) = 1 + s(m) + s(n) + s(\pi_1) + s(\pi_2)$

The next theorem shows that, in the best cases, quadratic compression can in principle be achieved by translating **ND**-proofs to **ND$^\mathbf{c}$**-proofs.

Theorem 7. *There is a sequence of theorems F_n whose smallest **ND**-proofs ψ_n are such that $s(\psi_n) \in \Omega(n^2)$, while there are **ND^c**-proofs ψ_n^c of F_n such that $s(\psi_n^c) \in O(n)$.*

Proof. Let $F_n = T^n(A \to B) \to (A \to T^n(B))$ where:

$$T^0(F) = F \qquad\qquad T^n(F) = (T^{n-1}(F) \to D_{2n-1}) \to D_{2n}$$

and $D_1, \ldots, D_{2n}$ are distinct atomic formulas.

Let $\psi_n^c = \mathcal{I}_c^{-1}(t_n^c)$ and $\psi_n = \mathcal{I}^{-1}(t_n)$ where:

$$t_n^c = \lambda f^{T^n(A\to B)}.\lambda a^A.(f\ a)_{(\underbrace{11\ldots1}_{2n};\epsilon)} \qquad\qquad t_n = \xi(t_n^c)$$

Any **ND**-proof of F_k must decompose F_k until the subformulas $A \to B$ and A are obtained and then apply $A \to B$ to A. ψ_k does exactly this and nothing more. ψ_k could not be further compressed by introducing detours, because $D_1, \ldots, D_{2k}$ are all distinct and hence ψ_k does not contain repeated subproofs[3]. Therefore, ψ_k is a smallest **ND**-proof of F_k.

By definition, $s(\psi_n^c) = s(t_n^c)$, and as shown below, $s(t_n^c) \in O(n)$.

$$s(t_n^c) = s(\lambda f^{T^n(A\to B)}.\lambda a^A.(f\ a)_{(\underbrace{11\ldots1}_{2n};\epsilon)}) = 11 + 6n$$

By definition, $s(\psi_n) = s(t_n)$, and $s(t_n)$ is computed below:

$$\begin{aligned} s(t_n) &= s(\xi(\lambda f^{T^n(A\to B)}.\lambda a^A.(f\ a)_{(\underbrace{11\ldots1}_{2n};\epsilon)})) \\ &= s(\lambda f^{T^n(A\to B)}.\lambda a^A.\xi((f\ a)_{(11\ldots1;\epsilon)})) \\ &= 8 + 4n + s(\xi((f\ a)_{(\underbrace{11\ldots1}_{2n};\epsilon)})) \end{aligned}$$

Let $q(n) = s(\xi((f\ a)_{(\underbrace{11\ldots1}_{2n};\epsilon)}))$. Then:

$$q(0) = s(\xi((f\ a)_{(\epsilon;\epsilon)})) = 3$$

$$\begin{aligned} q(n) &= s(\xi((f\ a)_{(\underbrace{1111\ldots1}_{2n};\epsilon)})) \\ &= s(\lambda k^{T^{n-1}(B)\to D_{2n-1}}.(f\ \lambda h^{T^{n-1}(A\to B)}.\xi((h\ a)_{(\underbrace{11\ldots1}_{2n-2};\epsilon)}))) \\ &= 4 + 8n + s(\xi((h\ a)_{(\underbrace{11\ldots1}_{2n-2};\epsilon)})) = 4 + 8n + q(n-1) \end{aligned}$$

Solving the recurrence relation above gives the following closed-form for q:

$$q(n) = 4n^2 + 8n + 3$$

Therefore, $s(t_n) = 8 + 4n + q(n) = 4n^2 + 12n + 11$ and hence $s(\psi_n) \in \Omega(n^2)$. □

[3] For a term to be compressible by the introduction of detours, it must contain repetitions. An example is $((f\ t)\ t)$, having size $3 + 2s(t)$ and compressible (by a detour with a non-linear abstraction) to $(\lambda x.((f\ x)\ x)\ t)$, whose size $(8 + s(t))$ is smaller when $s(t) > 5$). On the other hand, a detour with a linear abstraction always just adds 4 to a term's size.

6 Proof Compression

The fact that $\mathbf{ND^c}$-proofs can be quadratically smaller than the smallest **ND**-proofs suggests that a **ND**-proof ψ could be compressed by first translating it to the $\mathbf{ND^c}$-proof $\zeta[\psi]$ and then *folding* it (i.e. applying the rewriting rules for unfolding in the opposite direction). The purpose of this section is to remark a few obstacles that make the use of this idea non-trivial in practice.

The first obstacle is the fact that, even though $\rightsquigarrow_\delta^{-1}$ is clearly terminating (since the size of terms decreases), it is not confluent. Indeed, assuming $f : A \to B$ and $a : (A \to D) \to E$, the term $\lambda k^{B \to D}.(a\ \lambda h^A.(k\ (f\ h)))$ can be rewritten to both terms below:

$$(f\ a)_{(\epsilon;11)} \qquad\qquad \lambda k^{B\to D}.(a\ (k\ f)_{(\epsilon;0)})$$

and this critical pair is clearly not joinable, since both terms are already normal forms with respect to $\rightsquigarrow_\delta^{-1}$. As a consequence of the non-confluence, the choice of which rewriting rule to apply matters: bad choices may lead to compressed proof-terms that are not optimally small (e.g. the second term above).

The second, and more problematic, obstacle can be illustrated with the following term t, where $h_{ab} : A \to B$, $h_{bc} : B \to C$ and $h_{ade} : (A \to D) \to E$:

$$\lambda h_{cd}^{C\to D}.(h_{ade}\ \lambda h_a^A.(h_{cd}\ (h_{bc}\ (h_{ab}\ h_a)))) : (C \to D) \to E$$

It would be very desirable for a proof compression method to be able to obtain one of the smaller λ^c-terms (of the same type and also constructed using h_{ab}, h_{bc} and h_{ade}) below:

$$(h_{bc}\ (h_{ab}\ h_{ade})_{(\epsilon;11)})_{(\epsilon;11)} \qquad\qquad ((h_{bc}\ h_{ab})_{(\epsilon;0)}\ h_{ade})_{(\epsilon;11)}$$

However these terms cannot be obtained by folding t, simply because t is already in normal form w.r.t. $\rightsquigarrow_\delta^{-1}$ (it is easy to verify that no rewriting rule from $\rightsquigarrow_\delta^{-1}$ is applicable to t). As shown below, in order to be able to obtain the terms above from t, not only folding but also β-expansion must be used:

$$\begin{aligned}
t &= \lambda h_{cd}^{C\to D}.(h_{ade}\ \lambda h_a^A.(h_{cd}\ (h_{bc}\ (h_{ab}\ h_a))))\\
&\rightsquigarrow_\beta^{-1} \lambda h_{cd}^{C\to D}.(h_{ade}\ \lambda h_a^A.(\lambda k_b^B.(h_{cd}\ (h_{bc}\ k_b))\ (h_{ab}\ h_a)))\\
&\rightsquigarrow_\beta^{-1} \lambda h_{cd}^{C\to D}.(\lambda k_{bd}^{B\to D}.(h_{ade}\ \lambda h_a^A.(k_{bd}\ (h_{ab}\ h_a)))\ \lambda k_b^B.(h_{cd}\ (h_{bc}\ k_b)))\\
&\rightsquigarrow_\delta^{-1} (h_{bc}\ \lambda k_{bd}^{B\to D}.(h_{ade}\ \lambda h_a^A.(k_{bd}\ (h_{ab}\ h_a))))_{(\epsilon;11)}\\
&\rightsquigarrow_\delta^{-1} (h_{bc}\ (h_{ab}\ h_{ade})_{(\epsilon;11)})_{(\epsilon;11)}
\end{aligned}$$

$$\begin{aligned}
t &= \lambda h_{cd}^{C\to D}.(h_{ade}\ \lambda h_a^A.(h_{cd}\ (h_{bc}\ (h_{ab}\ h_a))))\\
&\rightsquigarrow_\beta^{-1} \lambda h_{cd}^{C\to D}.(h_{ade}\ \lambda h_a^A.(h_{cd}\ (\lambda k_a^A.(h_{bc}\ (h_{ab}\ k_a))\ h_a)))\\
&\rightsquigarrow_\delta^{-1} (\lambda k_a^A.(h_{bc}\ (h_{ab}\ k_a))\ h_{ade})_{(\epsilon;11)}\\
&\rightsquigarrow_\delta^{-1} ((h_{bc}\ h_{ab})_{(\epsilon;0)}\ h_{ade})_{(\epsilon;11)}
\end{aligned}$$

The introduction of detours by β-expansion brings the term to a form to which folding rules can be applied. Unfortunately, it is generally unclear which detours need to be introduced and where they should be introduced; the possibilities are numerous, since β-expansion is non-terminating and can be applied to almost any sub-term, thus greatly enlarging the search space. Furthermore, as noticeable in the examples above, β-expansion can momentarily increase the size of the term, rendering strategies that greedily minimize the size ineffective.

It is also interesting to note that the introduction of detours in natural deduction calculi is closely related to the introduction of cuts in sequent calculi, another notoriously difficult problem in proof compression [17,13].

7 Conclusions

The main contribution of this paper was the development of $\mathbf{ND^c}$, a sound and complete simple extension of natural deduction that allows inference rules to operate on deeply occurring subformulas. Formalizations of intrinsically deep reasoning steps then become more convenient and more natural, and the resulting proofs can be significantly smaller. Indeed, it has been shown (in Theorem 7) that $\mathbf{ND^c}$-proofs can be quadratically smaller than $\mathbf{ND}$-proofs in the best cases. Therefore, $\mathbf{ND^c}$ might be especially useful in application scenarios where the size of proofs and the proof-checking time are important. $\mathbf{ND^c}$ has been implemented in the *Skeptik* system (`https://github.com/Paradoxika/Skeptik`), and current research is focusing on designing an experiment to complement the best-case asymptotic analysis of Theorem 7 with empirical data about the average-case.

Surprisingly, if the intuitionistic contextual soundness condition is dropped, a natural deduction calculus for classical logic is obtained. This is particularly interesting because classical principles (e.g. Peirce's law) do not need to be assumed - they can be derived; and because the calculus uses single conclusion sequents, and thus is distinct from other assumption-free natural deduction calculi for classical logic, such as Parigot's [15]. A deeper investigation of such a classical contextual natural deduction calculus will be pursued in the near future.

Acknowledgements. I would like to especially thank: the deep inference community, for inspiring me to develop the calculus presented here; Bernhard Gramlich, for his help in finding useful term rewriting lemmas; Pascal Fontaine and David Deharbe, for discussions on proof production and proof checking for skolemization in SMT-solving, which served as a motivation for this work.

References

1. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press (1998)
2. Böhme, S., Nipkow, T.: Sledgehammer: Judgement Day. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 107–121. Springer, Heidelberg (2010)

3. Brünnler, K.: Atomic Cut Elimination for Classical Logic. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 86–97. Springer, Heidelberg (2003)
4. Brünnler, K., McKinley, R.: An Algorithmic Interpretation of a Deep Inference System. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 482–496. Springer, Heidelberg (2008)
5. Bruscoli, P., Guglielmi, A.: On the proof complexity of deep inference. ACM Transactions on Computational Logic 10, 1–34 (2009)
6. Bruscoli, P., Guglielmi, A., Gundersen, T., Parigot, M.: A Quasipolynomial Cut-Elimination Procedure in Deep Inference via Atomic Flows and Threshold Formulae. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16 2010. LNCS, vol. 6355, pp. 136–153. Springer, Heidelberg (2010)
7. de Groote, P. (ed.): The Curry-Howard Isomorphism. Cahiers du Centre de Logique, vol. 8. Academia, Universite Catholique de Louvain (1995)
8. Deharbe, D., Fontaine, P., Paleo, B.W.: Quantifier inference rules in the proof format of verit. In: 1st International Workshop on Proof Exchange for Theorem Proving (2011)
9. Dershowitz, N.: On Lazy Commutation. In: Grumberg, O., Kaminski, M., Katz, S., Wintner, S. (eds.) Languages: From Formal to Natural. LNCS, vol. 5533, pp. 59–82. Springer, Heidelberg (2009)
10. Gentzen, G.: Untersuchungen über das logische Schließen. Mathematische Zeitschrift 39, 176–210, 405–431 (1934-1935)
11. Guenot, N.: Nested proof search as reduction in the lambda-calculus. In: Schneider-Kamp, P., Hanus, M. (eds.) PPDP, pp. 183–194. ACM (2011)
12. Guglielmi, A.: A system of interaction and structure. CoRR, cs.LO/9910023 (1999)
13. Hetzl, S., Leitsch, A., Weller, D.: Towards Algorithmic Cut-Introduction. In: Bjørner, N., Voronkov, A. (eds.) LPAR-18 2012. LNCS, vol. 7180, pp. 228–242. Springer, Heidelberg (2012)
14. Mayr, R., Nipkow, T.: Higher-order rewrite systems and their confluence. Theor. Comput. Sci. 192(1), 3–29 (1998)
15. 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)
16. Tiu, A.F.: A Local System for Intuitionistic Logic. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 242–256. Springer, Heidelberg (2006)
17. Woltzenlogel Paleo, B.: Atomic Cut Introduction by Resolution: Proof Structuring and Compression. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16 2010. LNCS (LNAI), vol. 6355, pp. 463–480. Springer, Heidelberg (2010)

Conservatively Approximable Functions

Sebastian Wyman

Department of Mathematics, University of Florida,
P.O. Box 118105, Gainesville, Florida 32611
swyman@ufl.edu

Abstract. In the study of computable functions on the Cantor space $2^{\mathbb{N}}$, it is well-known that the image of such a function is an effectively closed set, or Π^0_1 class and in fact a *decidable* closed set. Here a closed subset Q of the Cantor space is decidable if the set of finite strings w which have an extension in Q is a computable set. It was shown recently by Cenzer, Dashti and King that the set of itineraries of a computable function is also a decidable closed set. Now the set of itineraries of a continuous function is always a *subshift*, meaning that it is closed under prefixes. It was also shown that any decidable shift is the set of itineraries of some computable function on $2^{\mathbb{N}}$. This paper defines the new notion of a *conservatively approximable* function on the Cantor space in order to have a family of functions whose images, and itineraries, can be arbitrary Π^0_1 classes.

Keywords: computability, computable analysis, effectively closed sets, effective symbolic dynamics.

1 Introduction

There is a long history of interaction between computability and dynamical systems. A Turing machine may be viewed as a dynamical system which produces a sequence of configurations or words before possibly halting. The reverse notion of using an arbitrary dynamical system for general computation has generated much interesting work. See for example [2,11]. In this paper we will consider computable aspects of certain dynamical systems over the Cantor space $2^{\mathbb{N}}$ and the related space $2^{\mathbb{Z}}$.

The study of computable dynamical systems is part of the Nerode program to study the effective content of theorems and constructions in analysis. Weihrauch [21] has provided a comprehensive foundation for computability theory on various spaces, including the space of compact sets and the space of continuous real functions.

Computable analysis is related as well to the so-called reverse mathematics of Friedman and Simpson [18], where one studies the proof-theoretic content of various mathematical results. The study of reverse mathematics is related in turn to the concept of *degrees of difficulty*. Here we say that $P \leq_M Q$ if there is a Turing computable functional F which maps Q into P; thus the problem of finding an element of P can be uniformly reduced to that of finding an element of Q, so that P is *less difficult than* Q. See Medvedev [14] and Sorbi [20] for details. The degrees of difficulty of effectively closed sets (also known as Π^0_1 classes) have been intensively investigated in several recent papers, for example Cenzer and Hinman [8] and Simpson [17].

The computability of Julia sets in the reals has been studied by Cenzer [4] and Ko [13]. The computability of complex dynamical systems has been investigated by Rettinger and Weihrauch [16] and by Braverman and Yampolsky [3]. The study of the computability of dynamical systems has received increasing attention in recent years; see for example papers of Delvenne et al [11], Hochman [12], Miller [15] and Simpson [19].

The connection between dynamical systems and subshifts is the following. Certain dynamical systems may be given by a continuous function F on a *symbolic space* $\mathcal{X}$ (one with a basis of clopen sets). For each $X \in \mathcal{X}$, the sequence $(X, F(X), F(F(X)), \dots)$ is the *trajectory* of X. Given a fixed partition $U_0, \dots, U_{k-1}$ of $\mathcal{X}$ into clopen sets, the *itinerary* $\mathrm{IT}_F(X)$ of a point X is the sequence $(a_0, a_1, \dots) \in k^{\mathbb{N}}$ where $a_n = i$ iff $F^n(X) \in U_i$. Let $\mathrm{IT}[F] = \{\mathrm{IT}_F(X) : X \in \mathcal{X}\}$. Note that $\mathrm{IT}[F]$ will be a closed set. We observe that, for each point X with itinerary $(a_0, a_1, \dots)$, the point $F(X)$ has itinerary $(a_1, a_2, \dots)$. Now the shift operator σ on $k^{\mathbb{N}}$ is defined by $\sigma(a_0, a_1, \dots) = (a_1, a_2, \dots)$. It follows that $\mathrm{IT}[F]$ is closed under the shift operator, that is, $\mathrm{IT}[F]$ is a *subshift.*

Computable subshifts and the connection with effective symbolic dynamics were investigated by Cenzer, Dashti and King [5] in a recent paper. A total, Turing computable functional $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ is always continuous and thus will be termed *computably continuous* or just computable. Effectively closed sets (also known as Π^0_1 classes) are a central topic in computability theory and they arise in many applications of computability theory. For example, the set of zeroes of a computable function $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$ is always a Π^0_1 class, and conversely every Π^0_1 class is the set of zeroes of some computable function F. Nonempty decidable Π^0_1 classes always have a computable member and are often of less interest than arbitrary Π^0_1 classes, but they play an important role in effective symbolic dynamics. See [9,10] for many other applications of Π^0_1 classes.

It was shown in [5] that, for any computably continuous function $F : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$, $\mathrm{IT}[F]$ is a decidable Π^0_1 class and, conversely, any decidable Π^0_1 subshift P is $\mathrm{IT}[F]$ for some computable map F. In [6], Π^0_1 subshifts are constructed in $2^{\mathbb{N}}$ and in $2^{\mathbb{Z}}$ which have no computable elements and are not decidable. Thus there is a Π^0_1 subshift with non-trivial Medvedev degree. J. Miller [15] showed that *every* Π^0_1 Medvedev degree contains a Π^0_1 subshift. Simpson [19] studied Π^0_1 subshifts in two dimensions and showed that every Π^0_1 Medvedev degree contains a Π^0_1 subshift of finite type which is a stronger result than just containing a Π^0_1 subshift.

Now every nonempty countable Π^0_1 class contains a computable element, so that all countable Π^0_1 classes have Medvedev degree $\mathbf{0}$, and many uncountable classes also have degree $\mathbf{0}$. The paper [7] considers more closely the structure of countable subshifts, using the Cantor-Bendixson derivative to compare and contrast countable subshifts of finite rank with Π^0_1 subshifts of finite rank as well as with arbitrary Π^0_1 classes of finite rank. It is shown that if Q is a subshift of rank two, then every member of Q is eventually periodic (and therefore computable) and furthermore if $Q \subseteq 2^{\mathbb{Z}}$, then the members of rank two are periodic and Q is a decidable closed set. However, there are rank two subshifts in $2^{\mathbb{N}}$ of arbitrary Turing degree and rank two Π^0_1 subshifts of arbitrary c. e. degree, so that rank two undecidable Π^0_1 subshifts exist in $2^{\mathbb{N}}$. Π^0_1 subshifts of rank three contain only computable elements, but Π^0_1 subshifts of rank four may contain members of arbitrary c. e. degree. For any given Π^0_1 class P of rank two, there is a subshift Q of rank four such that the degrees of the members of P and the degrees of the members of Q are identical. More generally, for any Π^0_1 class $P \subseteq 2^{\mathbb{N}}$, there is a Π^0_1 subshift $Q \subseteq 2^{\mathbb{N}}$ such that the non-computable degrees of the members of Q are identical with the non-computable degrees of the member of P.

In the present paper[1], we investigate the correspondence between continuous functions and Π^0_1 classes. We first give definitions and preliminary material in section 2. In section 3, we give the definition for and investigate properties of conservatively approximable functions. We demonstrate some properties of this class of functions and show that the Π^0_1 classes are exactly the images of these functions. In section 4, we refine the notion of conservatively approximable to give a characterization of the Π^0_1 subshifts. We give a general lemma which aids in such characterizations and investigate other related notions.

2 Preliminaries

We begin with some basic definitions. Let $\mathbb{N} = \{0, 1, 2, ...\}$ denote the set of natural numbers. Let Σ be any finite alphabet (typically an initial segment of ω). Then $\Sigma^{\mathbb{N}}$ denotes the set of countable sequences from Σ and Σ^* denotes the set of finite sequences or words from Σ. For a string $w = (w(0), w(1), \ldots, w(n-1))$, $|w|$ denotes the length n of w. The *reverse* of a string $w = (w(0), \ldots, w(n-1))$ is the string $w^- = (w(n-1), \ldots, w(0))$. The shift function on strings is defined by $\sigma(w) = (w(1), \ldots, w(|w|-1))$. For $X \in 2^{\mathbb{N}}$, $\sigma(X) = (X(1), X(2), \ldots)$–the result of deleting the initial entry of X. For $Z \in 2^{\mathbb{Z}}$, $Y = \sigma(Z)$ is defined so that $Y(i) = Z(i+1)$. The empty string has length 0 and will be denoted by ϵ. For $n \in \mathbb{N}$ and $X \in \Sigma^{\mathbb{N}}$, let $X\lceil_n = X(0)X(1)\cdots X(n-1)$ and for any word $w \in \Sigma^*$ let $w\lceil_n = w$ if $|w| < n$ and $w\lceil_n = w(0)w(1)\cdots w(n-1)$ otherwise.

The space $\Sigma^{\mathbb{N}}$ is a compact metric space with a basis of clopen sets of the form $I[\sigma] = [\sigma] = \{X \in \Sigma^{\mathbb{N}} : \sigma \sqsubset X\}$. For a sequence $(w_n)_{n\in\omega}$ of unbounded length from Σ^* and $X \in \Sigma^{\mathbb{N}}$, we say that $\lim_n w_n = X$ if, for every M there is an N so that if $n \geq N$ then $w_n\lceil_M = X\lceil_M$. We define the (length-)lexicographic ordering $<_L$ on Σ^N (Σ^*). For $u, v \in \Sigma^*$, $u <_L v$ if $|u| < |v|$ or $u(n) < v(n)$ where $n \in \mathbb{N}$ is the least such that $u(n) \neq v(n)$. For $X, Y \in \Sigma^{\mathbb{N}}$, $X <_L Y$ if $X(n) < Y(n)$ where $n \in \mathbb{N}$ is the least such that $X(n) \neq Y(n)$.

Given two strings v and w, the *concatenation* $v^\frown w$ is defined by

$$v^\frown w = (v(0), v(1), \ldots, v(m-1), w(0), w(1), \ldots, w(n-1)),$$

[1] This work was done under the supervision of my advisor Douglas Cenzer.

where $|v| = m$ and $|w| = n$. For $a \in \Sigma$, we write $w^\frown a$ (or just wa) for $w^\frown(a)$ and we write $a^\frown w$ (or just aw) for $(a)^\frown w$. We say w is an *initial segment* or *prefix* of v (written $w \sqsubset v$) if $v = w^\frown x$ for some x; this is equivalent to saying that $w = v\lceil_m$ for some m; w is a *suffix* of v if $v = x^\frown w$ for some x; w is a *factor* of v if $v = x^\frown w^\frown y$ for some x and y.

A *tree* T over Σ^* is a set of finite strings from Σ^* which contains the empty string ϵ and which is closed under initial segments. We say that $w \in T$ is an *immediate successor* of $v \in T$ if $w = va$ for some $a \in \Sigma$.

For any tree T, $(X(0), X(1), \dots)$ is said to be an infinite path through T if $X\lceil_n \in T$ for all n. We let $[T]$ denote the set of infinite paths through T. It is well-known that a subset Q of $2^{\mathbb{N}}$ is closed if and only if $Q = [T]$ for some tree T. A subset P of $2^{\mathbb{N}}$ is a *Π^0_1 class* (or *effectively closed set*) if $P = [T]$ for some computable tree T. For any closed set P, define the tree T_P to be $\{w \in \mathbb{N}^*$ such that $P \cap [w] \neq \emptyset\}$. For any tree T, we say that a node $w \in T$ is *extendible* if there exists $X \in [T]$ such that $w \sqsubset X$. If $P = [T]$, then T_P will equal the set of extendible nodes of T and will not depend on T. If T is computable, then the set of extendible nodes is a tree which is a co-c. e. subset of Σ^* but is not in general computable. P is said to be *decidable* (or *computable*) if T_P is a computable set.

The closed set P is subsimilar (or a subshift) if T_P is subsimilar. (Thus being closed is a part of our definition of a subshift.)

Define the coding $\langle X_0, X_1, X_2, \dots\rangle = X_0(0)X_0(1)X_1(0)X_0(2)X_1(1)X_2(0)\dots$. Note that if $X = \langle X_0, X_1, X_2, \dots\rangle$ and $u \sqsubset X$ then $C_i(X) = X_i$ and $C_i(u)$,the largest prefix of X_i encoded in u, are computable. On the other hand, $h(n, i)$ is a strictly increasing computable function giving the bit of X corresponding to $X_n(i)$.

Given $F : A \to A$ and a partition $\{I_i\}$ of A, the itinerary of X, $\mathrm{IT}_F(X) = a_0a_1a_2\dots$ where $a_n = i$ if and only if $F^n(X) \in I_i$ and define $\mathrm{IT}[F] = \{\mathrm{IT}_F(X) : X \in A\}$.

3 Conservatively Approximable Functions and Images on $\Sigma^{\mathbb{N}}$

In this section, we define the notion of a conservatively approximable function on $\Sigma^{\mathbb{N}}$ for a finite set Σ and prove the first main theorem, that Q is a Π^0_1 class if and only if Q is the image of a conservatively approximable function.

For comparison, computable functions on $\Sigma^{\mathbb{N}}$ can be characgterized as follows.

Definition 1. *A function $F : \Sigma^{\mathbb{N}} \to \Sigma^{\mathbb{N}}$ is computable iff there is a computable function $f : \Sigma^* \to \Sigma^*$ so that*

1. *For every $X \in \Sigma^{\mathbb{N}}$, $\big(|f(X\lceil_n)|\big)$ is unbounded and $\bigcup_n f(X\lceil_n) = F(X)$.*
2. *If $w \sqsubset u \in \Sigma^*$, then $f(w) \sqsubset f(u)$.*

We say that f as described above is a *representation* of F. We may assume without loss of generality that $|f(w)| \leq |w|$ for all w. It follows from (1) and compactness that, for a computable function F with representation f there is a computable function g such that, for every n and every w with length $|w| \geq g(n)$, $|f(w)| \geq n$. It then follows that the image $\operatorname{Im} F$ is a decidable Π^0_1 class, since to see whether a word w of length n has an extension in $\operatorname{Im} F$, we simply compute $g(n)$, and then compute $f(u)$ for all u of length $g(n)$; w will have an extension in $\operatorname{Im} F$ if and only for one of those u of length $g(n)$, $w \sqsubset f(u)$. This proves one direction of the following fact.

Proposition 1. *A subset Q of $\Sigma^{\mathbb{N}}$ is a decidable Π^0_1 class if and only if it is the image of some computable function $F : \Sigma^{\mathbb{N}} \to \Sigma^{\mathbb{N}}$.*

Proof. One direction is shown above. We will sketch the other direction since our result for computably approximable functions will use a modification of this construction. Let $Q = [T]$ where T is a computable tree with no dead ends. We recursively define a computable representing function $f : \Sigma^* \to T$ so that $Q = \operatorname{Im} F$ for the computable function represented by f. Let $f(\emptyset) = \emptyset$ and for each $w \in \Sigma^*$ and each $a \in \Sigma$, define $f(w^\frown a)$ in two cases. If $f(w)^\frown a \in T$, then $f(w^\frown a) = f(w)^\frown a$. If $f(w)^\frown a \notin T$, then let i be the least in Σ such that $f(w)^\frown i \in T$ and let $f(w^\frown a) = f(w)^\frown i$. Observe that $|f(w)| = |w|$ for all w, that $f(ew) \in T$ for all w, and that for all $w \in T$, $f(w) = w$. It follows that $\operatorname{Im} F = [T] = Q$.

The general notion of an *approximable* function is a limit of computable functions. Such a function will have a Π^0_2 image.

Definition 2. *A function $F : \Sigma^{\mathbb{N}} \to \Sigma^{\mathbb{N}}$ is* conservatively approximable *if there is a computable function $f : \Sigma^* \to \Sigma^*$ and a function $g : \mathbb{N} \to \mathbb{N}$ so that*

1. *For every $X \in \Sigma^{\mathbb{N}}$, $\big(|f(X\lceil_n)|\big)$ is unbounded and $\lim_n f(X\lceil_n) = F(X)$.*
2. *If $w \sqsubset u \in \Sigma^*$ and $|w| \geq g(n)$, then $f(u)\lceil_n = f(w)\lceil_n$.*
3. *For every $w \in \Sigma^*$ and $i \in \Sigma$, there is a $u \in \Sigma^*$ with $|u| = |w|$ so that $f(u) \sqsubset f(w^\frown i)$.*

Note that, without loss of generality, we may (and will) always assume $g(n) > n$. Also, note that if $|w| \geq g(m)$ then $|f(w)| \geq m$. For let $Y \sqsupset w$. Then since $|f(Y\lceil_n)|$ is unbounded, there is $n > |w|$ with $|f(Y\lceil_n)| > m$ but by (2), $f(w)\lceil_m = f(Y\lceil_n)\lceil_m$.

It follows from this definition that any conservatively approximable function will be approximable.

Proposition 2. *If F is conservatively approximable then F is continuous.*

Proof. Let f and g be as in definition 2, $w \in \Sigma^*$ and $X \in F^{-1}(I[w])$. Then if $u \sqsupset X\lceil_{g(|w|)}$, $f(u)\lceil_{|w|} = w$ hence if $Y \sqsupset X\lceil_{g(|w|)}$ then $F(Y) \sqsupset w$. Thus $F(I[X\lceil_{g(|w|)}]) \subset I[w]$ so $X \in I[X\lceil_{g(|w|)}] \subset F^{-1}(I[w])$ hence F is continuous.

The following property of conservatively approximable functions will be useful.

Proposition 3. *If F is conservatively approximable and H is computable, then $F' = H \circ F$ is conservatively approximable.*

Proof. Let f and g be approximation functions for F as in definition 2 and h be an approximation function for H. We define f' and g' for F' by $f' = h \circ f$ and $g' = g \circ h'$ where $h'(n) = \min\{m : (\forall w \in \Sigma^m)(|h(w)| \geq n)\}$. Note that h' is well defined by compactness. We need to show that f' and g' satisfy definition 2

2. Let $u \sqsupset w$ and n be given. Where $|w| \geq g'(n)$. Since $|w| \geq g(h'(n))$, we have $v = f(w)\lceil_{h'(n)} = f(u)\lceil_{h'(n)}$. So, $|h(v)| \geq n$, $h(f(w)) \sqsupset h(v)$ and $h(f(u)) \sqsupset h(v)$. Thus, $h(f(w))\lceil_n = h(v)\lceil_n = h(f(u))\lceil_n$.
1. Clearly $\big(|f'(X\lceil_n)|\big)_n$ is unbounded as both $\big(|f(X\lceil_n)|\big)_n$ and $\big(|h(X\lceil_n)|\big)_n$ are. Further, (2) shows that $\big(|f'(X\lceil_n)|\big)_n$ converges and the continuity of F and H shows that the limit is $F'(X)$.
3. Let $w^\frown i$ be given. There is a u so that $f(u) \sqsubset f(w^\frown i)$. Therefore, $h(f(u)) \sqsubset h(f(w^\frown i))$ so we have $f'(u) \sqsubset f'(w^\frown i)$ as desired.

Part (3) in the definition of a computably approximable function is designed to make the image a Π^0_1 class and indeed this is true.

Proposition 4. *If F is conservatively approximable then* $\operatorname{Im} F$ *is a Π^0_1 class.*

Proof. Let f and g be as in definition 2 and let $U_n = \bigcup\{I[f(w)\lceil_n] : w \in \Sigma^n\}$. Then (U_n) is a computable sequence of clopen sets and so $P = \bigcap_n U_n$ is a Π^0_1 class. We will show that $\operatorname{Im} F = P$.

Suppose $X \in P$. Then there is a sequence (w_n) with $|w_n| = n$ and $f(w_n)\lceil_n \sqsubset X$. Choose $Y_n \sqsupset w_n$ and consider the sequence $(Y_{g(n)})$. Then $F(Y_{g(n)})\lceil_n = f(w_{g(n)})\lceil_n = X\lceil_n$ since $g(n) > n$ and so $\lim_n F(Y_n) = X$. Since F is continuous and $\Sigma^{\mathbb{N}}$ is compact, $\operatorname{Im} F$ is compact. In particular, $\operatorname{Im} F$ is closed, thus $X \in \operatorname{Im} F$.

Supose $X \in \operatorname{Im} F$ so there is a Y with $F(Y) = X$ and so $\lim_n f(Y\lceil_n) = X$. Now, given k, $f(Y\lceil_{g(k)})\lceil_k = X\lceil_k$. We show by induction $n - k$ that for $w \in \Sigma^*$ with $|w| = n > k$ there is a $u \in \Sigma^*$ with $|u| = k$ and $f(u) \sqsubset f(w)$. Thus $f(u)\lceil_k \sqsubset f(Y\lceil_{g(n)})\lceil_n \sqsubset X$ and so, $X \in U_k$ for any k.

The induction proceeds as follows: If $n - k = 1$, then the statement is clear by property (3) in definition 2. If $n - k = i + 1$ and the statement holds when $n - k = i$, then there is a u with $|u| = k + 1$ and $f(u) \sqsubset f(w)$. Again, property (3) gives a u' with $|u'| = k$ and $f(u') \sqsubset f(u)$ and so $f(u') \sqsubset f(w)$.

Next we see that the image of a conservatively approximable fuction need not be decidable.

Proposition 5. *For any non-empty Π^0_1 class P there is a conservatively approximable function F with $P = \operatorname{Im} F$.*

Proof. Let L_P be the leftmost element in P and for any X let X_P be the nearest element of P left of X. *I.e.* $X_P \in P$, $X \leq_L X_P$, and if $X \leq_L Y \leq_L X_P$ then $Y \notin P$. Note that if $X \in P$ then $X_P = X$ and otherwise, X_P exists since P is closed. Then the function given by

$$F(X) = \begin{cases} L_P & X \leq_L L_P \\ X_P & X \not\leq_L L_P \end{cases}$$

is such a function. Clearly, $\operatorname{Im} F = P$ since $F(X) = X$ for $X \in P$ and $L_P, X_P \in P$ for every X. To see that F is conservatively approximable, we must find f and g satisfying the three properties in definition 2. First let T be a computable tree so that $P = [T]$ and define f as follows. Given w, if possible, find $u \in T \cap \Sigma^{|w|}$ nearest to w so that also $u \leq_L w$; $f(w) = u$. Otherwise $f(w)$ is the leftmost in $T \cap \Sigma^{|w|}$. Using the above notation, we might write this as

$$f(w) = \begin{cases} l_{T\lceil_{|w|}} & w \leq_L l_{T\lceil_{|w|}} \\ w_{T\lceil_{|w|}} & w \not\leq_L l_{T\lceil_{|w|}} \end{cases}.$$

Then f is computable, since T is computable. Also, let $g(n)$ be the least k so that any u of length k which has an extension of length n in T has an infinite extension in $[T]$.

1. For any X, $\left(|f(X\lceil_n)|\right)$ is unbounded since $|f(X\lceil_n)| = n$. Additionally, $F(X) = \lim_n f(X\lceil_n)$.
2. If $w \sqsubset u$ and $|w| \geq g(n)$, then $f(u)\lceil_n = f(w)\lceil_n$.
3. Finally, since for $w \in T$, $f(w) = w$ and $f(w) \in T$ for all w, $f(f(w)\lceil_{|w|-1}) \sqsubset f(w)$.

And so F is conservatively approximable.

The function defined above has the property that $f(f(w)) = f(w)$ for every w. This property implies (3) in definition 2. Thus, we can restrict the class of functions using this stronger property and still have every image be a Π^0_1 class and every Π^0_1 class be an image.

We now have our first main result.

Theorem 1. *For any finite set Σ and any subset Q of $\Sigma^{\mathbb{N}}$, Q is a Π^0_1 class if and only if $Q = \operatorname{Im} F$ for some conservatively approximable function.*

While this theorem shows that the notion of conservatively approximable is in some sense the right notion for images and Proposition 3 is nice, the following example shows that the notion of conservatively approximable is not entirely satisfactory.

Example 1. There is a conservatively approximable function F so that $\operatorname{Im} F^2$ is not a Π^0_1 class. Thus F^2 is not conservatively approximable.

Let $\left(T_e\right)_e$ be a computable enumeration of primitive recursive trees so that $\left([T_e]\right)_e$ enumerates all Π^0_1 classes (see [10]). We will focus on prefixes of four types of sequences:

$$\begin{aligned} H_e &= 0^{e+1}10^{\omega}, \\ I_e &= 10^{e+1}10^{\omega}, \\ J_e &= 110^{e+1}10^{\omega}, \text{ and} \\ X_e &= 110^{e+1}110^{\omega}. \end{aligned}$$

Our goal is that if $X_e \notin [T_e]$ then $F(I_e) = J_e$ and $F(J_e) = X_e$ hence $X_e \in \operatorname{Im} F^2$; otherwise $X_e \notin \operatorname{Im} F^2$. We use H_e to seemingly remove X_e from $\operatorname{Im} F^2$ while preserving the conservative approximability of F and the ability to add X_e to $\operatorname{Im} F^2$ when it leaves $[T_e]$. Prefixes of H_e are never in $\operatorname{Im} f$, however, they always map to prefixes of X_e. We define f inductively on words of length n. First define $f(0) = f(1) = 1$. If $|w^\frown i| = n$, define $f(w^\frown i)$ as follows:

$$f(w^\frown i) = \begin{cases} X_e\lceil_n & w^\frown i = H_e\lceil_n \\ J_e\lceil_n & w^\frown i = I_e\lceil_n \vee w^\frown i = X_e\lceil_n \\ J_e\lceil_n & w^\frown i = J_e\lceil_n \wedge (X\lceil_n \in T_e \cap \{0,1\}^n \vee e+5 > n) \\ X_e\lceil_n & w^\frown i = J_e\lceil_n \wedge X\lceil_n \notin T_e \cap \{0,1\}^n \wedge e+5 \leq n \\ f(w)^\frown i & \text{otherwise} \end{cases}$$

We first check that there is a g satisfying (2) in definition 2. We claim that for every X and n there is an $M_{X,n}$ so that if $u \sqsupset X\lceil_{M_{X,n}}$ then $f(u)\lceil_n = f(X\lceil_{M_{X,n}})\lceil_n$.

Let X and n be given and note that $|f(u)| = |u|$. There several cases:

1. If $X \sqsupset w$ where

$$w \in \{0, 10, 111, 110^k 1, 110^{e+1}111, 110^{e+1}10^k 1, 110^{e+1}111, 110^{e+1}110 : e, k \in N\}.$$

   Then if $u \sqsupset v \sqsupset w$ we have $f(u) \sqsupset f(v)$. Hence, if $M = \max\{n, |w|\}$, we have that $u \sqsupset X\lceil_M$ implies $f(u)\lceil_n = f(X\lceil_M)\lceil_n$.
2. If $X = J_e$ and $X_e \in [T_e]$, then for every m, $f(J_e\lceil_m) = J_e\lceil_m$. So for $M = \max\{n, e+5\}$ we have $u \sqsupset v \sqsupset X\lceil_M$ then $f(u) \sqsupset f(v)$. Hence, we have that $u \sqsupset X\lceil_M$ implies $f(u)\lceil_n = f(X\lceil_M)\lceil_n$.
3. If $X = J_e$ and $X_e \notin [T_e]$ then for some k, $X\lceil_k \notin T_e \cap \{0,1\}^k$. So for $M = \max\{k, e+5, n\}$ we have $u \sqsupset v \sqsupset X\lceil_M$ then $f(u) \sqsupset f(v)$. Hence, we have that $u \sqsupset X\lceil_M$ implies $f(u)\lceil_n = f(X\lceil_M)\lceil_n$.
4. If $X = \lim_e X_e = \lim_e J_e = 110^\omega$ then for every m and for $e > m$ $X\lceil_m = X_e\lceil_m = J_e\lceil_m$ so $f(X\lceil_m) = X\lceil_m$. Let $M = n$. If $u \sqsupset X\lceil_M$ either $u \sqsubset X_e, J_e$ in which case $f(u)\lceil_n = X_e\lceil_n = J_e\lceil_n$ or for $v \sqsupset u$, $f(v) \sqsupset f(u)$ and so $f(u)\lceil_n = f(X\lceil_M)\lceil_n$.

Now, consider the open cover $\{[X\lceil_{M_{X,n}}] : X \in 2^{\mathbb{N}}\}$. By compactness, there is a finite cover $\{[X_i\lceil_{M_{X_i,n}}] : i = 1, \ldots, k_n\}$. Define $g(n) = \max\{M_{X_i,n} : i = 1, \ldots, k_n\}$. Then if $|w| > g(n)$ and $u \sqsupset w$, we have $u \sqsupset w \sqsupset X_i\lceil_{M_{X_i,n}}$ for some i hence $f(u)\lceil_n = f(w)\lceil_n = f(X_i\lceil_{M_{X_i,n}})\lceil_n$.

It is clear that f satisfies (1) and (3) in definition 2 and is computable. Thus F defined by $F(X) = \lim_n f(X\lceil_n)$ is conservatively approximable. We need to show that $[T_e] \neq \operatorname{Im} F^2$ for any e so $\operatorname{Im} F^2$ is not a Π^0_1 class and F^2 is not conservatively approximable. To see this, suppose $X_e \notin [T_e]$. Then for some N, $f(I_e\lceil_n) = J_e\lceil_n$ and $f(J_e)\lceil_n) = X_e\lceil_n$ for every $n > N$. Thus $F^2(I_e) = F(J_e) = X_e$ so $X_e \in \operatorname{Im} F^2$. On the other hand if $X_e \in [T_e]$, then $F(Y) = X_e$ only if $Y = H_e$ and $H_e \notin \operatorname{Im} F$ so $X_e \notin \operatorname{Im} F^2$.

For computable functions, the image of a decidable Π^0_1 class is still decidable. We can use the example above to show that a similar result does not hold for conservatively approximable functions.

Example 2. There is a decidable Π^0_1 class P and conservatively approximable function F so that $F[P]$ is not Π^0_1.

In the notation of the previous example, let F be as above, and let $P = \{110^\omega\} \cup \{J_e : e \in \mathbb{N}\}$. Then $X_e \in F[P]$ if and only if $e \notin [T_e]$ and so $F[P] \neq [T_e]$ for any e, hence $F[P]$ is not a Π^0_1 class.

4 Symbolic Dynamics for Conservatively Approximable Functions

In this section, we consider the set of itineraries $IT[F]$ associated with a conservatively approximable function. We see that every Π^0_1 class will equal $IT[F]$ for some conservatively approximable function. We present a converse result with some additional restrictions on the function F.

Lemma 1. *If $Q \subset \Sigma^{\mathbb{N}}$ is a subshift and G is a function with $Q = \operatorname{Im} G$ such that $G\lceil_Q$ is the identity on Q and $G(X)(0) = X(0)$ then, for $F = \sigma \circ G$, we have that $\mathrm{IT}[F] = Q$, where the itineraries are with respect to the natural partition of $\Sigma^{\mathbb{N}}$ into $\{I[a] : a \in \Sigma\}$.*

Proof. Let $X \in Q$. Then $F(X) = \sigma(G(X)) = \sigma(X) \in Q$ since G is fixed on Q and Q is a subshift. Thus, $F^n(X) = \sigma^n(X)$ and so $F^n(X)(0) = X(n)$. Thus we have $X = \mathrm{IT}_F(X) \in \mathrm{IT}[F]$.

Let $Y \in \mathrm{IT}[F]$ so $Y = \mathrm{IT}_F(X)$ for some X. By induction, $F^n(X) = \sigma^n(G(X))$ for $n \geq 1$. Since $F(X) = \sigma(G(X))$ and using the inductive hypothesis, $F^{n+1}(X) = \sigma(G(\sigma^n(G(X)))) = \sigma^{n+1}(G(X))$ since $\sigma^n(G(X)) \in Q$. So for $n \geq 1$, $Y(n) = F^n(X)(0) = G(X)(n)$. Now, since $G(X)(0) = X(0) = Y(0)$, $Y = G(X) \in Q$.

Theorem 2. *If Q is a Π^0_1 subshift with $Q \cap I[i] \neq \emptyset$ for every $i \in \Sigma$ then there is a conservatively approximable function F and a partition I_i so that $\mathrm{IT}[F] = Q$.*

Proof. Fix $I_i = I[i]$. Since $Q \cap I[i] \neq \emptyset$ for every $i \in \Sigma$, the function

$$G_i(X) = \begin{cases} L_{Q_i} & X \leq_L L_{Q_i} \\ X_{Q_i} & X \not\leq_L L_{Q_i} \end{cases}$$

where $Q_i = \{X : i^\frown X \in Q\}$ is a Π^0_1 class since Q is. Since $Q \cap I[i] \neq \emptyset$ for every $i \in \Sigma$, the function $G(i^\frown X) = G_i(X)$ has $G(X)(0) = X(0)$ and satisfies all the above properties so, by Lemma [1], $F = \sigma \circ G$ has $\mathrm{IT}\, F = Q$. F is also conservatively approximable with approximations $f_F(w) = \sigma(f_G(Ww))$ and $g_F(n) = g_G(n+1)$.

When trying to show the converse, the proof runs into trouble in the same place as it does when trying to demonstrate closure under composition. Consider the following example.

Example 3. There is a conservatively approximable function F so that $\mathrm{IT}[F]$ is not a Π^0_1 class.

Let $I_0 = [0]$ and $I_1 = [1]$ and let $\left(T_e\right)_e$ be a computable enumeration of primitive recursive trees so that $\left([T_e]\right)_e$ enumerates all Π^0_1 classes (see [10]). Consider $X_e = 10^{e+1}1^\omega$ and $Y_e = 0^{e+1}10^\omega$. Our goal is to define f approximating F so that if $X_e \notin [T_e]$ then $F(\sigma^n(X_e)) = \sigma^{n+1}(X_e)$ and so $X_e \in \mathrm{IT}[F]$. If $X_e \in [T_e]$, then for no X is $\mathrm{IT}_F(X) = X_e$. Y_e will be used as a place holder. We do this by emulating σ as much as possible. That is, for most X, $F(X) = \sigma(X)$, but $F(X_e) = \sigma(Y_e)$ when ever $X_e \in [T_e]$.

Define f inductively on words of length n. If $n \leq 2$ then $f(w) = \sigma(w)$. If $|w^\frown i| = n$, define f by:

$$f(w^\frown i) = \begin{cases} \sigma(Y_e\lceil_n) & w^\frown i = X_e\lceil_n \wedge (X_e\lceil_n \in T_e \cap \{0,1\}^n \vee e+3 \geq n) \\ \sigma(X_e\lceil_n) & w^\frown i = X_e\lceil_n \wedge X\lceil_n \notin T_e \cap \{0,1\}^n \wedge e+3 < n \\ f(w)^\frown 0 & w^\frown i \sqsupset u10^{e+1}1 \wedge |u| > 0, \text{ minimal} \\ f(w)^\frown i & \text{otherwise} \end{cases}$$

To show that there is a g satisfying (2) in definition 2, we claim, as in the previous proof, that for every X and n there is an $M_{X,n}$ so that if $u \sqsupset X\lceil_{M_{X,n}}$ then $f(u)\lceil_n = f(X\lceil_{M_{X,n}})\lceil_n$.

Let X and n be given. Note that for all u, $|f(u)| = |u| - 1$. There are two cases.

1. If $X\lceil_{n+1} \neq X_e\lceil_{n+1}$ for all $e < n+4$, then for all $u \sqsupset X\lceil_{n+1}$, $f(u) \sqsupset f(X\lceil_{n+1})$ by the second two cases in the definition of f so let $M = n+1$.
2. Suppose $X\lceil_{n+1} = X_e\lceil_{n+1}$ for some $e < n+4$. If $X_e \in [T_e]$ define $k_e = 0$, otherwise, $k_e = (\mu k)(X_e\lceil_k \notin T_e \cap \{0,1\}^k)$. Let $M = \max\{k_e, n : e < n+4\}$, then for every $u \sqsupset X\lceil_M$, either $u\lceil_m = X_e\lceil_m$ for some m and some $e \geq n+3$ or not. In the first case, $u\lceil_n = X\lceil_n = 0^n$ so $f(u)\lceil_n = f(X\lceil_M)\lceil_n = 0^n$. In the second case, if $u \sqsupset X\lceil_M$ then $f(u) \sqsupset f(X\lceil_M)$.

As in the previous argument, compactness allows us to define $g(n)$ and to see that if we let $F(X) = \lim_n f(X\lceil_n)$ then F is conservatively approximable. Now, we can compute $F(X)$ and $\mathrm{IT}_F(X)$.

1. If $X \in \{0^\omega, 0^{n+1}1^\omega, 0^{n+1}1^{m+1}0^\omega, 1^{n+2}0^\omega, 1^\omega : n, m \in \mathbb{N}\}$, then $F(X) = \sigma(X)$ and so $\mathrm{IT}_F(X) = X$.
2. If $X \in \{10^{n+1}1^\omega, 10^{n+1}1^{m+1}0^\omega, 10^{n+1}1^{m+1}0^{k+1}1Y : n, m, k \in \mathbb{N}, Y \in 2^{\mathbb{N}}\}$ and $X_n \in [T_n]$, then $F(X) = \sigma(0^{n+1}10^\omega)$ and so $\mathrm{IT}_F(X) = 10^n10^\omega$.
3. If $X \in \{10^{n+1}1^\omega = X_n, 10^{n+1}1^{m+1}0^\omega : n, m \in \mathbb{N}\}$ and $X_n \notin [T_n]$, then $F(X) = \sigma(X)$ and $\mathrm{IT}_F(X) = X$.
4. If $X \sqsupset w0^{k+1}1$ and $w \in \{0^{n+1}1^{m+1}, 1^{n+2}, 10^{e+1}1^{m+1}, n, m \in \mathbb{N}, X_e \notin [T_e]\}$, then $F(X) = \sigma(w0^\omega)$ and so $\mathrm{IT}_F(X) = w0^\omega$.

In each case, we can calculate $\mathrm{IT}_F(X)$ by noting that for every X, $F(X)$ falls under the first case. Thus for $n > 2$, we have $F^n(X) = \sigma^{n-1}(F(X))$.

From this we can see that $X_e \in \mathrm{IT}[F]$ if and only if $X_e \notin [T_e]$. Since $[T_e]$ is the e-th Π^0_1 class, $\mathrm{IT}[F]$ is not a Π^0_1 class.

In the example above, we have that $F^{n+1}(X) = \sigma^n(F(X))$. This rules out at least one possible strengthening of the notion of conservative approximability. Here is another possible notion.

Definition 3. *A function $F : \Sigma^{\mathbb{N}} \to \Sigma^{\mathbb{N}}$ is* strongly conservatively approximable *if the function F^* defined by $F^*(X) = \left\langle X, F(X), F^2(X), F^3(X), \ldots \right\rangle$ is conservatively approximable.*

Unfortunately we have the following.

Proposition 6. *If F is strongly conservatively approximable, then F is computable.*

Proof. Since F^* is conservatively approximable, $F' : X \mapsto \langle X, F(X) \rangle$ is conservatively approximable. Thus, $\operatorname{Im} F'$ is a Π^0_1 class. However, $\operatorname{Im} F' = \operatorname{Graph} F$ hence, F is computable.

Here is a better idea.

Definition 4. *A function $F : \Sigma^{\mathbb{N}} \to \Sigma^{\mathbb{N}}$ is* locally conservatively approximable *if, for every $w \in \Sigma^*$, the function F_w, defined by $F_w(X) = F(w^\frown X)$, is conservatively approximable.*

Combining this idea with Lemma 1 above, we have the following partial converse to Theorem 2.

Theorem 3. *Suppose F is a locally conservatively approximable function such that $F^2 = H \circ F$ for some computable function H. Then $\mathrm{IT}[F]$ is a Π^0_1 subshift.*

Proof. Let I_i be a partition of $\Sigma^{\mathbb{N}}$ into finitely many clopen sets and let A_i the the corresponding partition of Σ^m for some appropriate m. We first note that $F^{n+1} = H \circ F^n$ so by induction, $F^{n+1} = H^n \circ F$. So

$$\mathrm{IT}[F] = \bigcup_i \bigcup_{\substack{w \in A_i \\ [w] \cap \operatorname{Im} F \neq \emptyset}} i^\frown \mathrm{IT}_H[\operatorname{Im} F_w].$$

Now, since H is computable, so is IT_H and since F_w is conservatively approximable, $\operatorname{Im} F_w$ is a Π^0_1 class. Thus, $\mathrm{IT}_H[\operatorname{Im} F_w]$ is a Π^0_1 class and so the above finite union is also a Π^0_1 class. Notice that we use the local condition for only finitely many intervals determined from the partition given.

Conversely:

Proposition 7. *Fix the partion $I_i = [i]$ of $2^{\mathbb{N}}$ and let Q be Π^0_1 subshift Q so that $[i] \cap Q \neq \emptyset$ for each i. There is a locally conservatively approximable function F so that $F^2 = H \circ F$ for some computable function H with $\mathrm{IT}[F] = Q$.*

Proof. By Lemma 1 and Proposition 3 we need only find a suitable function G with $\operatorname{Im} G = Q$ and G_w conservatively approximable for each w. Now, let $Q = [T]$ for some computable tree T and define $f(w)$ inductively as follows:

$$f(w)(0) = w(0)$$

$$f(w)(n+1) = \begin{cases} w(n+1) & (\exists v \in T \cap 2^{|w|})(f(w)\lceil_{n+1}{}^\frown w(n+1) \sqsubset v) \\ 1 - w(n+1) & \text{otherwise} \end{cases}.$$

Note the for each $n < |w|$, $f(w)\lceil_{n+1}$ has an extension in $T \cap 2^{|w|}$.

We now define $G(X) = \lim_n f(X\lceil_n)$. We know $\lim_n f(X\lceil_n)$ exists as: if $v \sqsubset f(w)$, $v \in \operatorname{ext}(T)$, and $u \sqsupset w$ then $v \sqsubset f(u)$ and, by compactness, for each n there is an m so that if $|u| = n$ and $u \sqsubset v \in T \cap 2^m$ for some v, then $u \in \operatorname{ext}(T)$. Additionally, notice that if $w \in T$, $f(w) = w$ and so clearly, G is conservative.

To show that G_w is conservatively approximable, we define $f_w(u)$ in a similar way. First, find $v_w \in \operatorname{ext}(T)$, $|v_w| = |w|$:

$$v_w(0) = w(0)$$

$$v_w(n+1) = \begin{cases} w(n+1) & (\exists v \in \operatorname{ext}(T) \cap 2^{|w|})(v_w\lceil_{n+1}{}^\frown w(n+1) \sqsubset v) \\ 1 - w(n+1) & \text{otherwise} \end{cases}.$$

Of course v_w is not computable from w. Then, we can compute $f_w(u)$ by:

$$f_w(u)\lceil_{|w|} = v_w$$

$$f_w(u)(n+|w|) = \begin{cases} u(n) & \exists v \in [v_w] \cap T \cap 2^{|w|}(f_w(u)\lceil_{n+|w|}{}^\frown u(n) \sqsubset v) \\ 1 - u(n) & \text{otherwise} \end{cases}.$$

Again, by compactness, there is an m so that if $|u| > m$, $f(wu) \sqsupset v_w$ hence $f_w(u) = f(wu)$ and so $F_w(X) = F(wX) = \lim_n f(wX\lceil_n) = \lim_n f_w(X\lceil_n)$ and $F_w(X)$ is conservative since, if $v_w u \in T$ we have $f_w(u) = v_w u$.

The following example shows that not every conservatively approximable function is locally conservatively approximable.

Example 4. There is a conservatively approximable function F so that $F_0(X) := F(0^\frown X)$ is not conservatively approximable.

Let $X_e = 1^{e+1}0^{e+1}1^\omega$. We construct F so that $X_e \in \operatorname{Im} F_0$ if and only if $e \in \emptyset'$. Thus, $\emptyset' \leq_1 T_{\operatorname{Im} F_0}$ so F_0 is not conservatively approximable, as its image is not a Π^0_1 class. To do this, let A_n be a computable approximation of $\emptyset'$ and, for $|u^\frown i| = n$, define

$$f(u^\frown i) = \begin{cases} X_e\lceil_n & u^\frown i \sqsubset 0^\frown X_e \wedge e \in A_n \\ \sigma(u^\frown i) & 1 \sqsubset u^\frown i \\ f(u)^\frown 1 & \text{otherwise} \end{cases}$$

Thus, $F(X) = \lim_n f(X\lceil_n)$ is conservatively approximable and has

$$F(0^\frown 1^{e+1}0^{e+1}X) = \begin{cases} X_e & e \in \emptyset' \\ 1^\omega & \text{otherwise} \end{cases},$$

while $F(00X) = 1^\omega$ and $F(1X) = X$ for every X in $2^{\mathbb{N}}$. Thus, F has the property desired.

Nor can we ask that F_w be approximable uniformly, for then F is computable.

Proposition 8. *If F is conservatively approximable and $F_w(X) = F(wX)$ is also conservatively approximable via $f_w(u) = f(wu)$ for every w, then F is computable.*

Proof. By assumption, we have that for every w and every $u^\frown i$, there is a v with $|u| = |v|$ and $f_w(v) \sqsubset f_w(u^\frown i)$ and so $f(wv) \sqsubset f(wu^\frown i)$. In particular, if $u = \epsilon$, the empty string, then also $v = \epsilon$ and we have $f(w) \sqsubset f(w^\frown i)$. Thus, F is computable.

5 Conclusion and Future Research

In this paper, we defined the conservatively approximable functions. We showed that images of conservatively computable functions are exactly the Π^0_1 classes but that images of decidable Π^0_1 classes need not be Π^0_1 in general. We showed that the class of conservatively approximable functions is closed under composition with computable function, but not under composition in general. Moving to symbolic dynamics, we constructed a conservatively approximable function whose itineraries do not form a Π^0_1 class. However, locally conservative function whose iterates are given by composition with computable functions have exactly the Π^0_1 subshifts for itineraries.

Possible extensions for this work include classifying the complexity of images of decidable and arbitrary Π^0_1 classes under conservatively approximable functions. General properties of locally conservative functions should be studied. We are also interested in finding an analog in functions on Cantor space for the semi-computable reals, studying their images and symbolic dynamics.

The notion of conservatively computable also translates to the reals. Here images of continuous functions are intervals, disallowing much of the diagonalization used in this paper. We would like to study the symbolic dynamics of these and computable functions of the unit interval, as well as other weakenings of the computability notion proposed by Bauer, Weihrauch, Zheng ([1], [22]).

References

1. Bauer, M., Zheng, X.: On the Weak Computability of Continuous Real Functions. In: CCA 2010, pp. 29–40 (2010)
2. Bournez, O., Cosnard, M.: On the computational power of dynamical systems and hybrid systems. Theoretical Computer Science 168, 417–459 (1996)

3. Braverman, M., Yampolsky, M.: Non-computable Julia sets. J. Amer. Math. Soc. 19, 551–578 (2006)
4. Cenzer, D.: Effective dynamics. In: Crossley, J., Remmel, J., Shore, R., Sweedler, M. (eds.) Logical Methods in Honor of Anil Nerode's Sixtieth Birthday, pp. 162–177. Birkhauser (1993)
5. Cenzer, D., Dashti, A., King, J.L.F.: Computable Symbolic Dynamics. Math. Logic Quarterly 54, 524–533 (2008)
6. Cenzer, D., Dashti, A., Toska, F., Wyman, S.: Computability of Countable Subshifts. In: Ferreira, F., Löwe, B., Mayordomo, E., Mendes Gomes, L. (eds.) CiE 2010. LNCS, vol. 6158, pp. 88–97. Springer, Heidelberg (2010)
7. Cenzer, D., Dashti, A., Toska, F., Wyman, S.: Computability of countablesub shifts in one dimension. Theory of Computing Systems (2012)
8. Cenzer, D., Hinman, P.G.: Degrees of difficulty of generalized r. e. separating classes. Arch. for Math. Logic 45, 629–647 (2008)
9. Cenzer, D., Remmel, J.B.: Π^0_1 classes. In: Ersov, Y., Goncharov, S., Marek, V., Nerode, A., Remmel, J. (eds.) Handbook of Recursive Mathematics, Vol. 2: Recursive Algebra, Analysis and Combinatorics. Elsevier Studies in Logic and the Foundations of Mathematics, vol. 139, pp. 623–821 (1998)
10. Cenzer, D., Remmel, J.B.: Effectively Closed Sets, Perspectives in Mathematical Logic. Cambridge University Press (2013)
11. Delvenne, J.-C., Kurka, P., Blondel, V.: Decidability and Universality in Symbolic Dynamical Systems. Fund. Informaticae (2005)
12. Hochman, M.: On the dynamics and recursive properties of multidimensional symbolic systems. Invent. Math. 176, 131–167 (2009)
13. Ko, K.: On the computability of fractal dimensions and Julia sets. Ann. Pure Appl. Logic 93, 195–216 (1998)
14. Medvedev, Y.: Degrees of difficulty of the mass problem. Dokl. Akad. Nauk SSSR 104, 501–504 (1955)
15. Miller, J.: Two notes on subshifts. Proc. Amer. Math. Soc. (to appear)
16. Rettinger, R., Weihrauch, K.: The computational complexity of some Julia sets, in. In: Goemans, M.X. (ed.) Proc. 35th ACM Symposium on Theory of Computing, San Diego, pp. 177–185. ACM Press, New York (2003)
17. Simpson, S.G.: Mass problems and randomness. Bull. Symbolic Logic 11, 1–27 (2005)
18. Simpson, S.G.: Subsystems of Second Order Arithmetic, 2nd edn. Cambridge U. Press (2009)
19. Simpson, S.G.: Medvedev degrees of two-dimensional subshifts of finite type. Ergodic Theory and Dynamical Systems (to appear)
20. Sorbi, A.: The Medvedev lattice of degrees of difficulty. In: Cooper, S.B., et al. (eds.) Computability, Enumerability, Unsolvability: Directions in Recursion Theory. London Mathematical Society Lecture Notes, vol. 224, pp. 289–312. Cambridge University Press (1996) ISBN 0-521-55736-4
21. Weihrauch, K.: Computable Analysis. Springer (2000)
22. Weihrauch, K., Zheng, X.: Computability on continuous, lower semi-continuous and upper semi-continuous real functions. Theoretical Computer Science 234, 109–133 (2000)

Self-referentiality in the Brouwer–Heyting–Kolmogorov Semantics of Intuitionistic Logic

Junhua Yu[⋆]

Doctoral Program in Computer Science
The Graduate Center, City University of New York
jyu1@gc.cuny.edu

Abstract. The intended provability semantics of Intuitionistic Propositional Logic IPC (also called Brouwer–Heyting–Kolmogorov semantics) has been formalized within Gödel-Artemov's framework. According to this approach, IPC is embedded in modal logic S4 by Gödel embedding and S4 is realized in Artemov's Logic of Proofs LP which has a provability interpretation in Peano Arithmetic. Artemov's realization of S4 in LP uses self-referential LP-formulas of the form $t{:}\phi(t)$, namely, 't is a proof of a formula ϕ containing t itself.' Kuznets showed that this is not avoidable by offering S4-theorems that cannot be realized without using self-referential LP-formulas. This paper extends Kuznets' method to find IPC-theorems that call for direct self-referentiality in LP. Roughly speaking, examples include double-negations of tautologies that are not IPC-theorems, e.g., $\neg\neg(\neg\neg p \to p)$, and there are also examples in the purely implicational fragment $\mathsf{IPC}_{\to}$. This suggests that the Brouwer–Heyting–Kolmogorov semantics of intuitionistic logic is intrinsically self-referential.

1 Introduction

The Brouwer–Heyting–Kolmogorov (BHK) semantics of Intuitionistic Propositional Logic IPC follows the reading of intuitionistic truth as provability. This was initially suggested by Brouwer and then stipulated informally by Heyting and Kolmogorov. Gödel [7] contributed to this by introducing a modal calculus of provability that is essentially S4, and suggesting a possible embedding of IPC in his calculus. The suggested embedding, prefixing each subformula with the provability modality $\Box$, [1] reflects the intuitionistic view of logical truth as

⋆ The author has been supported by a Presidential Research Fellowship of the CUNY Graduate Center and NSF award 0830450 Justification Logic and Applications.

[1] To be exact, in [7], Gödel suggested the embedding "prefixing each subformula with a $\Box$" along with some of its S4-equivalent simplifications. We will consider Gödel-style modal embeddings in a general setting and get results that are not sensitive to the choice of embedding.

provability. Gödel's embedding of IPC in S4 was shown to be faithful by McKinsey and Tarski [9] and hence transformed the problem of finding a provability semantics for IPC to finding a provability semantics for S4. Artemov [1,2] filled the gap by introducing the Logic of Proofs LP with completeness w.r.t. formal arithmetical provability and establishing an embedding, called *realization*, of S4 in LP. A detailed description of the approach of formalizing BHK semantics of IPC can be found in [2].

Definition 1 (The Logic of Proofs LP [2]). *In the language of* LP*:* formulas *are defined by* $\phi ::= \bot \,|\, p \,|\, \phi \to \phi \,|\, t : \phi$*, where* $t ::= c \,|\, x \,|\, t \cdot t \,|\, t{+}t \,|\, !t$ *is called a* term*,* c *is a* constant*,* x *is a (proof)* variable*, and* p *is a propositional letter.*

LP *has the following axiom schemes*

$$\begin{array}{ll} (A0) & \textit{A finite set of classical propositional axiom schemes} \\ (A1) & t{:}\phi \to \phi \\ (A2) & t_1{:}(\phi \to \psi) \to (t_2{:}\phi \to t_1{\cdot}t_2{:}\psi) \\ (A3) & t{:}\phi \to !t{:}t{:}\phi \\ (A4) & t_1{:}\phi \to t_1{+}t_2{:}\phi \quad \textit{and} \quad t_2{:}\phi \to t_1{+}t_2{:}\phi \end{array}$$

and rules

$$\begin{array}{ll} (MP) & \textit{Modus Ponens} \\ (AN) & \dfrac{}{c{:}A} \ \textit{where } c \textit{ is a constant and } A \textit{ is an axiom.} \end{array}$$

A constant specification*, denoted by* $\mathcal{CS}$*, is a set of formulas of the form* $c : A$ *where* c *is a constant and* A *is an axiom. A constant specification* $\mathcal{CS}$ *is* injective *if for each constant* c*, at most one axiom* A *satisfies* $c : A \in \mathcal{CS}$*.* LP$(\mathcal{CS})$ *is the fragment of* LP *with only formulas from* $\mathcal{CS}$ *being allowed by rule* (AN)*. Each* LP*-proof* calls for *a constant specification, namely the set of formulas introduced by rule* (AN) *in this proof. Clearly, an* LP*-proof that calls for* $\mathcal{CS}$ *is an* LP$(\mathcal{CS})$*-proof.*

LP can be seen as the "explicit counterpart" of S4, as indicated by the following definition and theorem from [2].

Definition 2 (Realization [2]). *For a modal formula* ϕ*, a* realization *of* ϕ *consists of an assignment of* LP*-terms to all* $\Box$*-occurrences*[2] *in* ϕ *and a constant specification which indicates axioms associated to constants that occur in those terms. We denote the image of* ϕ *under a realization* r *by* ϕ^r*.*

A realization is normal *if it assigns to each negative* $\Box$ *a distinct variable and the constant specification associated to this realization is injective.*

Theorem 1 (Realization Theorem [2]). S4 $\vdash \phi$ *iff* LP$(\mathcal{CS}) \vdash \phi^r$ *for some normal realization* r *with an injective constant specification* $\mathcal{CS}$*.*

[2] We assume that $\Diamond$ is defined as $\neg\Box\neg$.

Artemov's proof of Theorem 1 in [2] employs a cut-free Gentzen-style formulation of $\mathsf{S4}$, along with the "Lifting Lemma" of LP, which actually displays the ability of LP to internalize its own proofs. This realization theorem, together with the arithmetical completeness of LP in [2], gives an explicit provability semantics of $\mathsf{S4}$, and then of IPC via Gödel embedding. Artemov mentioned in [2] that the realization procedure used there may involve constant specifications that contain self-referential formulas of the form $c{:}A(c)$, and asked whether this kind of formulas are avoidable. Other proofs of the realization theorem (e.g., [5], [6]) offer possibilities of assigning terms in other ways, but Kuznets showed in [3] that self-referentiality is intrinsic in the realization of $\mathsf{S4}$ in LP, as presented below.

Definition 3 (Self-referentiality of Constant Specification [8]). *A constant specification is* self-referential*, if it has a subset of the form*

$$\{\, c_1{:}A_1(c_2), \cdots, c_{n-1}{:}A_{n-1}(c_n), c_n{:}A_n(c_1)\,\}.$$

If $n = 1$, then this constant specification is directly self-referential*.*

Theorem 2 (Direct Self-referentiality of $\mathsf{S4}$ [3]). *Any realization of the $\mathsf{S4}$-theorem*

$$\neg\Box\neg(p \to \Box p) \tag{1}$$

calls for a directly self-referential constant specification.

By Theorem 2, we know that the provability semantics of $\mathsf{S4}$ is intrinsically self-referential. The proof of Theorem 2 in [3] employs the Mkrtychev model (M-model) of LP from [10]. The idea is, for any possible realization of (1), we construct a counter M-model that admits the largest non-directly self-referential constant specification.[3] We do not present Kuznets' proof used in [3], but will employ a similar proof for Theorem 4 in Section 2.[4]

Theorem 2 answered Artemov's question on the $\mathsf{S4}$ layer negatively, but left this question open on the IPC layer, since (1) is not the image of an IPC-theorem under known Gödel-style modal embeddings of IPC in $\mathsf{S4}$ (cf. [7], [4], and [12]). In Section 2, we will answer Artemov's question on the IPC layer negatively.

At the end of this introduction section, we present the definition and some facts about M-models. We generally follow [10], with some modifications.

Definition 4 (Mkrtychev Model [10]). *A function $*(\cdot)$ from LP-terms to the power set of LP-formulas is called an* evidence function *if it satisfies:*

$$\begin{cases} \psi \in *(t) \text{ implies } (t{:}\psi) \in *(!t), \\ \tau \to \phi \in *(t_1) \text{ and } \tau \in *(t_2) \text{ implies } \phi \in *(t_1 \cdot t_2), \\ *(t_1) \cup *(t_2) \subseteq *(t_1 + t_2). \end{cases} \tag{2}$$

[3] In this paper, by a "non-directly self-referential constant specification," we mean a constant specification that is not directly self-referential.

[4] The idea of Kuznets' proof used in [3] was extended to the "$*$-calculus" in [8], where M-model was replaced by its multi-state version, the Fitting model from [5]. We will follow the approach in [3], which is sufficient for our goal.

*An evidence function $*$ is called a $\mathcal{CS}$-*evidence function *if it satisfies $A \in *(c)$ for any $c{:}A \in \mathcal{CS}$.*

A Mkrtychev model *is $\mathcal{M} = (*, v)$ where $*$ is an evidence function and v is a propositional valuation (from the set of propositional letters to the set $\{0, 1\}$). If $*$ is a $\mathcal{CS}$-evidence function, then $\mathcal{M} = (*, v)$ is called a $\mathcal{CS}$-*model.

For $\mathcal{M} = (*, v)$, $\mathcal{M} \vDash \phi$ means $\mathcal{M}$ satisfies ϕ. This is defined by

$$\begin{cases} \mathcal{M} \nvDash \bot, \\ \mathcal{M} \vDash p \text{ iff } v(p) = 1, \\ \mathcal{M} \vDash \phi \to \psi \text{ iff } \mathcal{M} \nvDash \phi \text{ or } \mathcal{M} \vDash \psi, \\ \mathcal{M} \vDash t{:}\phi \text{ iff } \phi \in *(t) \text{ and } \mathcal{M} \vDash \phi. \end{cases}$$

Lemma 1 ([10]). *For any set X of formulas of form $t{:}\phi$, there is a smallest[5] evidence function $*_X$ that satisfies*

$$t{:}\phi \in X \Rightarrow \phi \in *_X(t). \tag{3}$$

Moreover, $*_X$ can be constructed from $\{\phi \in *_X(t) \,|\, t{:}\phi \in X\}$ by using conditions in (2) of Definition 4.

We call X the *initial set* of $*_X$.

Theorem 3 (Completeness of LP w.r.t. M-models [10]). $\mathsf{LP}(\mathcal{CS}) \vdash \phi$ *iff $\mathcal{M} \vDash \phi$ for any $\mathcal{CS}$-model $\mathcal{M}$.*

2 Self-referentiality of IPC

In this section, we have the following notations and conventions.

Convention 1. *We take $\bot, \to$ as primitive connectives in classical logic; other connectives like $\top, \neg, \wedge, \vee, \leftrightarrow$ are considered as abbreviations defined in the standard way.*

By $\phi = \psi$, we mean ϕ and ψ are syntactically identical. For instance, we may write $\neg\phi = \phi \to \bot$, but $\bot \neq \bot \wedge \bot$.

By $s \rhd \phi$ (or $s \rhd t$), we mean term s occurs in formula ϕ (or term t, resp.).

By $\Box^i\phi$, we mean $\underbrace{\Box \cdots \Box}_{i}\phi$ for any natural number i. $\Box^0\phi = \phi$.

By $\mathcal{CS}^{\varnothing}$, we mean the set $\{c : A \,|\, A$ is an axiom and $c \not\rhd A\}$, which is the largest non-directly self-referential constant specification.

In the following Theorem, we adapt Kuznets' method from [3], and find a natural class of S4-theorems that call for direct self-referentiality in LP.

[5] Smallest in the sense that $*_X(s) \subseteq *'_X(s)$ for any evidence function $*'_X$ that satisfies (3), and any term s.

Theorem 4. *For any modal formula σ s.t.* $\mathsf{S4} \nvdash \sigma$*:*

(1) For any modal formula θ, any propositional formula ζ s.t. $\mathsf{S4} \nvdash \zeta$*, any realization of*

$$\Box^u(\Box^w(\Box^x\sigma\to\theta)\to\Box^z\zeta) \qquad (w,x>0)$$

has a counter $\mathcal{CS}^{\varnothing}$-model.[6]

(2) If $\mathsf{S4}\vdash\Diamond\Box\sigma$*, then any realization of* $\mathsf{S4}$*-theorem*

$$\Box^u(\Box^w(\Box^x\sigma\to\Box^y\bot)\to\Box^z\bot) \qquad (w,x>0)$$

calls for a directly self-referential constant specification.

(3) If $\mathsf{S4}\vdash\Diamond\Box\sigma$*, then any realization of $\Diamond\Box\sigma$ calls for a directly self-referential constant specification.*

Proof. (1) We first show this with a further assumption that σ is a prime formula[7] or an implication, and will eliminate this assumption afterwards.

Any possible realization of $\Box^u(\Box^w(\Box^x\sigma\to\theta)\to\Box^z\zeta)$ has the form (note that ζ is $\Box$-free, which implies $\zeta^r=\zeta$)

$$t^0_u{:}\cdots{:}t^0_1{:}(t^1_w{:}\cdots{:}t^1_1{:}(t^2_x{:}\cdots{:}t^2_1{:}\sigma^r\to\theta^r)\to t^4_z{:}\cdots{:}t^4_1{:}\zeta).$$

We have the following abbreviations:

$$\begin{array}{rll} \delta & \text{stands for} & t^2_x{:}\cdots{:}t^2_1{:}\sigma^r\to\theta^r \ , \\ X & \text{stands for} & \{t^1_1{:}\delta, t^1_2{:}t^1_1{:}\delta, \ \dots, t^1_w{:}\cdots{:}t^1_1{:}\delta\} \ , \\ X^- & \text{stands for} & \{\delta, t^1_1{:}\delta, \ \dots, t^1_{w-1}{:}\cdots{:}t^1_1{:}\delta\} \ , \\ *' & \text{stands for} & *_{\mathcal{CS}^{\varnothing}} \ , \\ * & \text{stands for} & *_{\mathcal{CS}^{\varnothing}\cup X} \ . \end{array}$$

Some remarks here: $w>0$ ensures that $X\neq\varnothing$, and hence by the definition of X^-, we have:

$$\text{for any } \psi,\ t{:}\psi\in X \text{ for some } t \text{ iff } \psi\in X^-. \tag{4}$$

We also have

$$\delta \text{ is a subformula of } \psi \text{ for any } \psi\in X\cup X^-. \tag{5}$$

The existence of $*'$ and $*$ is guaranteed by Lemma 1.

Since $\mathsf{S4}\nvdash\zeta$ and ζ is $\Box$-free, there is some propositional valuation that falsifies ζ. Take such a valuation v, the desired counter model is $\mathcal{M}^r=(*,v)$. For any constant c, axiom A s.t. $c:A\in\mathcal{CS}^{\varnothing}$, we have $c:A\in\mathcal{CS}^{\varnothing}\cup X$, and then by Lemma 1, $A\in *_{\mathcal{CS}^{\varnothing}\cup X}(c)=*(c)$. So $\mathcal{M}^r$ is indeed a $\mathcal{CS}^{\varnothing}$-model. It is now sufficient to show that

$$\mathcal{M}^r\nvDash t^0_u{:}\cdots{:}t^0_1{:}(t^1_w{:}\cdots{:}t^1_1{:}\delta\to t^4_z{:}\cdots{:}t^4_1{:}\zeta).$$

[6] The counter model we will construct also works when $\mathsf{S4}\nvdash\Box^u(\Box^w(\Box^x\sigma\to\theta)\to\Box^z\zeta)$.

[7] By a prime formula, we mean $\bot$, or a propositional letter (p, q, etc.). Note that $\top$ is defined as $\bot\to\bot$, and is not a prime formula.

Since $x > 0$, the term t_1^2 does exist. Suppose that $\sigma^r \in *(t_1^2)$. If $\sigma^r \in *'(t_1^2)$, then by Lemma 2 (forthcoming), $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \sigma^r$, hence $\mathsf{S4} \vdash \sigma$. This contradicts our assumption. If $\sigma^r \in *(t_1^2) \setminus *'(t_1^2)$, then by Lemma 2, σ^r is not a prime formula, and hence σ is not a prime formula. By our further assumption, σ is an implication, and hence so is σ^r. By Lemma 2,

$$\sigma^r = \delta = t_x^2 : \cdots : t_1^2 : \sigma^r \to \theta^r,$$

a contradiction. Neither case is possible, hence $\sigma^r \notin *(t_1^2)$.

Thus, we have $\mathcal{M}^r \nvDash t_1^2 : \sigma^r$, hence $\mathcal{M}^r \nvDash t_x^2 : \cdots : t_1^2 : \sigma^r$. Then $\mathcal{M}^r \vDash \delta$. By the definition of $*$ and of X, $\delta \in *(t_1^1)$, $t_1^1 : \delta \in *(t_2^1)$, ... , $t_{w-1}^1 : \cdots : t_1^1 : \delta \in *(t_w^1)$. Thus $\mathcal{M}^r \vDash t_w^1 : \cdots : t_1^1 : \delta$.

Since ζ is $\Box$-free and v falsifies ζ, $\mathcal{M}^r \nvDash \zeta$, which implies $\mathcal{M}^r \nvDash t_z^4 : \cdots : t_1^4 : \zeta$. By the observation in last paragraph, $\mathcal{M}^r \nvDash t_w^1 : \cdots : t_1^1 : \delta \to t_z^4 : \cdots : t_1^4 : \zeta$. Therefore, $\mathcal{M}^r \nvDash t_u^0 : \cdots : t_1^0 : (t_w^1 : \cdots : t_1^1 : \delta \to t_z^4 : \cdots : t_1^4 : \zeta)$, and what is constructed is actually a counter $\mathcal{CS}^{\varnothing}$-model.

We eliminate the further assumption by induction on the structure of σ. If σ is a prime formula or an implication, then our further assumption is true, and the claimed result is verified by what presented above. Suppose $\sigma = \Box\sigma_0$ for some σ_0. Since $\mathsf{S4} \nvdash \sigma$, $\mathsf{S4} \nvdash \sigma_0$. Since

$$\Box^u(\Box^w(\Box^x\sigma \to \theta) \to \Box^z\zeta) = \Box^u(\Box^w(\Box^{x+1}\sigma_0 \to \theta) \to \Box^z\zeta),$$

it is sufficient to show that any realization of $\Box^u(\Box^w(\Box^{x+1}\sigma_0 \to \theta) \to \Box^z\zeta)$ has a counter $\mathcal{CS}^{\varnothing}$-model. This model is given by applying IH on σ_0.

(2) Since $\mathsf{S4} \vdash \Diamond\Box\sigma = \neg\Box\neg\Box\sigma = \Box(\Box\sigma \to \bot) \to \bot$ and $\mathsf{S4} \vdash \Box^i\bot \leftrightarrow \bot$ for any i, we have $\mathsf{S4} \vdash \Box(\Box\sigma \to \Box^y\bot) \to \Box^z\bot$. By $\mathsf{S4} \vdash \Box\phi \leftrightarrow \Box\Box\phi$ for any ϕ, we have $\mathsf{S4} \vdash \Box^w(\Box^x\sigma \to \Box^y\bot) \to \Box^z\bot$ since $w, x > 0$. By applying necessitation u-times, $\mathsf{S4} \vdash \Box^u(\Box^w(\Box^x\sigma \to \Box^y\bot) \to \Box^z\bot)$.

Employing (1) in the sense that $\theta = \Box^y\bot$ and $\zeta = \bot$ gives a counter $\mathcal{CS}^{\varnothing}$-model $\mathcal{M}^r$ for any possible realization of $\Box^u(\Box^w(\Box^x\sigma \to \Box^y\bot) \to \Box^z\bot)$.

To see that any realization of $\Box^u(\Box^w(\Box^x\sigma \to \Box^y\bot) \to \Box^z\bot)$ calls for direct self-referentiality, suppose that we were able to realize it with a constant specification $\mathcal{CS}'$ that is not directly self-referential. That is, we have an $\mathsf{LP}(\mathcal{CS}')$-proof of the realized formula, denoted temporally by κ. By Definition 3, for any $c : A \in \mathcal{CS}'$, $c \not\trianglerighteq A$. That is to say, $\mathcal{CS}' \subseteq \mathcal{CS}^{\varnothing}$ and hence the proof is also an $\mathsf{LP}(\mathcal{CS}^{\varnothing})$-proof. By Theorem 3, $\mathcal{M} \vDash \kappa$ for any $\mathcal{CS}^{\varnothing}$-model $\mathcal{M}$, which contradicts the existence of the counter $\mathcal{CS}^{\varnothing}$-model $\mathcal{M}^r$.

(3) A trivial case of (2) where $w = x = 1$ and $u = y = z = 0$. $\Box$

Now we state and prove Lemma 2, with all notations and abbreviations from Theorem 4.

Lemma 2. *For any subterm s of t_1^2, any modal formula ϕ:*

(1) If $\phi \in *'(s)$, then

(i) $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \phi$,

(ii) $t_1^2 \not\trianglerighteq \phi$;

(2) If $\phi \in *(s) \setminus *'(s)$, then

(iii) $t_1^2 \rhd \phi$,

(iv) ϕ is not prime, and $\phi = \delta$ if ϕ is an implication,

(v) $\phi \notin \{t_1^2{:}\sigma^r, t_2^2{:}t_1^2{:}\sigma^r, \ldots, t_x^2{:}\cdots{:}t_1^2{:}\sigma^r\}$.

Proof. Induction on s. We need to consider the reason why a term-formula pair (e.g., t-ψ) is, or is not, added to an evidence function. By Lemma 1, aside from the initial set $\mathcal{CS}^{\varnothing}$ or $\mathcal{CS}^{\varnothing} \cup X$, all possibilities are given by closure conditions in (2) of Definition 4. In each possibility, the corresponding condition sets restrictions on forms of t and ψ. Based on these restrictions on forms, we get subterm-subformula pairs which enjoy useful properties by IH.

Case 1. s is a constant c:

(1) If $\phi \in *'(c)$, then $c : \phi \in \mathcal{CS}^{\varnothing}$ (closure conditions in (2) of Definition 4 only add formulas for composite terms). So ϕ is an axiom, hence (i) holds. Suppose that $t_1^2 \rhd \phi$, then $c \rhd \phi$ since c is a subterm of t_1^2, hence $\mathcal{CS}^{\varnothing}$ is directly self-referential, a contradiction. Thus, (ii) holds.

(2) If $\phi \in *(c) \setminus *'(c)$, then $c{:}\phi \in X$ (precisely, $(\mathcal{CS}^{\varnothing} \cup X) \setminus \mathcal{CS}^{\varnothing}$, which is a subset of X), which implies $\phi \in X^-$ by (4) in the proof of Theorem 4. By (5) in the proof of Theorem 4, δ occurs in each formula in X^-. Since $t_1^2 \rhd \delta$, we have $t_1^2 \rhd \phi$, (iii) holds. There is no prime formula in X^-, and δ is the only implication in X^-, therefore (iv) holds. To see that (v) holds, suppose that $\phi \in \{t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$, which implies ϕ is a proper subformula of δ. Since $\phi \in X^-$ and δ is a subformula of any element of X^-, δ is a subformula of ϕ. Thus we have δ as a proper subformula of δ itself, a contradiction. Note that independently of assumptions of this case,

$$\text{all of (iii,iv,v) follow from } \phi \in X^-, \tag{6}$$

which will be useful in later cases.

Case 2. s is a variable x:

(1) If $\phi \in *'(x)$, then $x : \phi \in \mathcal{CS}^{\varnothing}$, which is impossible, since x is not a constant.

(2) If $\phi \in *(x) \setminus *'(x)$, then $x{:}\phi \in X$, which implies $\phi \in X^-$ (by (4)). From (6), all of (iii,iv,v) hold.

Case 3. s is $!s_1$ for some term s_1:

(1) Suppose $\phi \in *'(!s_1)$. Since $!s_1$ is not a constant, $!s_1 : \phi \notin \mathcal{CS}^{\varnothing}$, thus this pair is added by the closure condition for ! . According to (2) of Definition 4, $\phi = s_1{:}\psi$ for some $\psi \in *'(s_1)$. Since s_1 is a subterm of t_1^2, by IH, $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \psi$. For any $\mathcal{CS}^{\varnothing}$-model $\mathcal{M}$, by Theorem 3, $\mathcal{M} \vDash \psi$. Since $\psi \in *'(s_1)$ where $*'$ is the smallest $\mathcal{CS}^{\varnothing}$-evidence function, $\mathcal{M} \vDash s_1{:}\psi$. By Theorem 3, $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash s_1{:}\psi$, (i) holds. Also by IH, $t_1^2 \not\rhd \psi$. Since s_1 is a proper subterm of t_1^2, we know $t_1^2 \not\rhd s_1$. Hence, $t_1^2 \not\rhd s_1{:}\psi$, (ii) holds.

(2) Suppose $\phi \in *(!s_1) \setminus *'(!s_1)$. Subcase (a), via the initial set. Then we have $!s_1 : \phi \in X$, which implies $\phi \in X^-$ with the help of (4). Now (6) ensures all

of (iii,iv,v). Subcase (b), via the closure condition. Then $\phi = s_1 : \psi$ for some $\psi \in *(s_1) \setminus *'(s_1)$. By IH, $t_1^2 \rhd \psi$, hence $t_1^2 \rhd s_1 : \psi$, (iii) holds. Since $s_1 : \psi$ is neither a prime formula nor an implication, (iv) holds in a vacuous way. By IH, $\psi \notin \{t_1^2 : \sigma^r, t_2^2 : t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$, from which we can derive that $s_1 : \psi \notin \{t_2^2 : t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$. Moreover, s_1 is a proper subterm of t_1^2, which implies $s_1 : \psi \neq t_1^2 : \sigma^r$. Therefore, we know that (v) holds, since $s_1 : \psi \notin \{t_1^2 : \sigma^r, t_2^2 : t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$.

Case 4. s is $s_1 \cdot s_2$ for some terms s_1 and s_2:

(1) Suppose $\phi \in *'(s_1 \cdot s_2)$. Since $s_1 \cdot s_2$ is not a constant, $s_1 \cdot s_2 : \phi \notin \mathcal{CS}^{\varnothing}$, thus this pair is added by the closure condition for $\cdot$, hence there is some τ s.t. $\tau \to \phi \in *'(s_1)$ and $\tau \in *'(s_2)$. Both s_1 and s_2 are subterms of t_1^2, by IH, $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \tau \to \phi$ and $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \tau$, so $\mathsf{LP}(\mathcal{CS}^{\varnothing}) \vdash \phi$, (i) holds. By IH, $t_1^2 \not\rhd \tau \to \phi$, which implies $t_1^2 \not\rhd \phi$, (ii) holds.

(2) Suppose $\phi \in *(s_1 \cdot s_2) \setminus *'(s_1 \cdot s_2)$. Subcase (a), via the initial set. Then $s_1 \cdot s_2 : \phi \in X$, which implies $\phi \in X^-$ with the help of (4). All of (iii,iv,v) are ensured by (6). Subcase (b), via the closure condition. Then there is some τ s.t. $\tau \to \phi \in *(s_1)$, $\tau \in *(s_2)$. Subcase (b.1), $\tau \to \phi \in *(s_1) \setminus *'(s_1)$ and $\tau \in *'(s_2)$. By IH, $t_1^2 \not\rhd \tau$. Since $\tau \to \phi$ is an implication, by IH, $\tau \to \phi = \delta$, i.e., $\tau = t_x^2 : \cdots : t_1^2 : \sigma^r$. Note that $t_1^2 \rhd \tau$; we have a contradiction. Subcase (b.2), $\tau \to \phi \in *(s_1) \setminus *'(s_1)$ and $\tau \in *(s_2) \setminus *'(s_2)$. By IH, we know that $\tau \notin \{t_1^2 : \sigma^r, t_2^2 : t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$. As in subcase (b.1), by applying IH on $\tau \to \phi$, we have $\tau = t_x^2 : \cdots : t_1^2 : \sigma^r \in \{t_1^2 : \sigma^r, t_2^2 : t_1^2 : \sigma^r, \ldots, t_x^2 : \cdots : t_1^2 : \sigma^r\}$, a contradiction. Subcase (b.3), $\tau \to \phi \in *'(s_1)$ and $\tau \in *(s_2) \setminus *'(s_2)$. By IH, $t_1^2 \not\rhd \tau \to \phi$. But also by IH, $t_1^2 \rhd \tau$, hence $t_1^2 \rhd \tau \to \phi$, a contradiction. Subcase (b.4), $\tau \to \phi \in *'(s_1)$ and $\tau \in *'(s_2)$. The closure condition for $\cdot$ in (2) of Definition 4 gives $\phi \in *'(s_1 \cdot s_2)$ and violates the assumption of (2). In summary, Subcase (b) is impossible.

Case 5. s is $s_1 + s_2$ for some terms s_1 and s_2:

(1) If $\phi \in *'(s_1 + s_2)$, then w.l.o.g., $\phi \in *'(s_1)$, hence both (i) and (ii) hold by IH.

(2) Suppose $\phi \in *(s_1 + s_2) \setminus *'(s_1 + s_2)$. Subcase (a), via initial set. Then $s_1 + s_2 : \phi \in X$, which implies $\phi \in X^-$. From (6), we have all of (iii,iv,v). Subcase (b), via the closure condition. That is, $\phi \notin *'(s_1)$, $\phi \notin *'(s_2)$, and w.l.o.g., $\phi \in *(s_1)$. Thus $\phi \in *(s_1) \setminus *'(s_1)$, which, by IH, gives all of (iii,iv,v). $\square$

Theorem 4 gives a natural class of self-referential $\mathsf{S4}$-theorems.[8] We now try to find corresponding classes of IPC-theorems.

[8] Instances of self-referential theorems in modal logics T, $\mathsf{K4}$, and $\mathsf{D4}$ (w.r.t. realizations in their explicit counterparts, JT, $\mathsf{J4}$, and $\mathsf{JD4}$, respectively) are given in [8], where T and $\mathsf{D4}$ share the same instance with $\mathsf{S4}$ (i.e., formula (1) in Theorem 2), and the instance in $\mathsf{K4}$ is $\neg\Box\bot \to \neg\Box\neg(p \to \Box p)$, which is also a T-theorem. Note that $\mathsf{S4} \nvdash \sigma$ implies $\mathsf{T} \nvdash \Diamond\Box\sigma$, and hence Theorem 4(3) gives a natural class of self-referential modal instances that are not T-theorems. For example, the $\mathsf{S4}$-theorem $\Diamond\Box(\Diamond\Box p \to \Box p)$, which is not provable in T, calls for direct self-referentiality in any possible realizations in LP.

There are several ways to faithfully embed IPC in $\mathsf{S4}$, with minor differences between them (cf. [7], [9], [4], and [12]). Three of them are presented in Table 1, where $(\cdot)^\circ$ and $(\cdot)^\Box$ are contained in [12], and $(\cdot)^\triangle$ is the well-known Gödel's "$\Box$ each subformula" embedding.[9]

Table 1. Three Faithful Embeddings of IPC in $\mathsf{S4}$

	$(\cdot)^\circ$	$(\cdot)^\Box$	$(\cdot)^\triangle$
$\bot$	$\bot$	$\bot$	$\Box\bot$
p	p	$\Box p$	$\Box p$
$\alpha \wedge \beta$	$\alpha^\circ \wedge \beta^\circ$	$\alpha^\Box \wedge \beta^\Box$	$\Box(\alpha^\triangle \wedge \beta^\triangle)$
$\alpha \vee \beta$	$\Box\alpha^\circ \vee \Box\beta^\circ$	$\alpha^\Box \vee \beta^\Box$	$\Box(\alpha^\triangle \vee \beta^\triangle)$
$\alpha \to \beta$	$\Box\alpha^\circ \to \beta^\circ$	$\Box(\alpha^\Box \to \beta^\Box)$	$\Box(\alpha^\triangle \to \beta^\triangle)$

Fact 1 ([12]). *Embeddings in Table 1 are all faithful, since* $\mathsf{IPC} \vdash \phi$ *iff* $\mathsf{S4} \vdash \phi^\times$ *for any embedding* $(\cdot)^\times \in \{(\cdot)^\circ, (\cdot)^\Box, (\cdot)^\triangle\}$. *Differences between these embeddings are minor, since by induction, we see that* $\mathsf{S4} \vdash \phi^\Box \leftrightarrow \phi^\triangle$ *and* $\mathsf{S4} \vdash \Box\phi^\times \leftrightarrow \Box\phi^+$ *for any propositional formula* ϕ*, any* $(\cdot)^\times, (\cdot)^+ \in \{(\cdot)^\circ, (\cdot)^\Box, (\cdot)^\triangle\}$.

In the following theorem, we show that $(\cdot)^\circ$ gives an easy description of IPC-theorems that fit the requirement of Theorem 4(3). By CPC, we mean the classical propositional calculus.

Theorem 5. *For any propositional formula* α *s.t.* $\mathsf{CPC} \vdash \alpha$ *and* $\mathsf{IPC} \nvdash \alpha$*, we have* $\mathsf{IPC} \vdash \neg\neg\alpha$ *and any realization of* $(\neg\neg\alpha)^\circ$ *calls for a directly self-referential constant specification.*

Proof. By Glivenko's Theorem (cf. [11]), $\mathsf{IPC} \vdash \neg\neg\alpha$ follows from $\mathsf{CPC} \vdash \alpha$. Since $\mathsf{IPC} \nvdash \alpha$ and $\mathsf{IPC} \vdash \neg\neg\alpha$, we have $\mathsf{S4} \nvdash \alpha^\circ$ and $\mathsf{S4} \vdash (\neg\neg\alpha)^\circ$. Now

$$\begin{aligned}(\neg\neg\alpha)^\circ &= ((\alpha \to \bot) \to \bot)^\circ = \Box(\alpha \to \bot)^\circ \to \bot^\circ \\ &= \neg\Box(\Box\alpha^\circ \to \bot^\circ) = \neg\Box\neg\Box\alpha^\circ = \Diamond\Box\alpha^\circ\end{aligned}$$

implies $\mathsf{S4} \vdash \Diamond\Box\alpha^\circ$.

By Theorem 4(3), any realization of $\Diamond\Box\alpha^\circ$ calls for a directly self-referential constant specification. We are done since $(\neg\neg\alpha)^\circ = \Diamond\Box\alpha^\circ$. $\Box$

There is a well-known formula that satisfies conditions of Theorem 5.

[9] In [7], Gödel presented a version of embedding along with some alternative options, say, whether or not to add $\Box$'s in the $\wedge$ case. The version Gödel presented has $\neg$, but not $\bot$, as a primitive connective, and hence looks slightly different. Except for this, if we take the alternative option that adds $\Box$'s in the $\wedge$ case, then the only difference between the resulting version and $(\cdot)^\triangle$ is the outermost $\Box$.

Example 1. *$\mathsf{CPC} \vdash \neg\neg p \to p$ and $\mathsf{IPC} \nvdash \neg\neg p \to p$.*

Though differences between embeddings $(\cdot)^\circ$, $(\cdot)^\Box$, and $(\cdot)^\triangle$ are minor, they can affect realizations in LP. In a realization, each $\Box$ is replaced by a term, hence the result of realization is quite sensitive to the exact syntactical form of the modal formula (cf. [2]). It is therefore reasonable to consider possible embeddings of IPC in $\mathsf{S4}$ other than $(\cdot)^\circ$ to see whether or not they are good at avoiding self-referentiality. In what follows, we consider the class of "basic" embeddings.

Definition 5 (Basic Embedding). *A potential embedding of IPC in $\mathsf{S4}$, denoted by $(\cdot)^\times$, is* basic *if it satisfies:*

$$\begin{cases} p^\times = \Box^h p, \\ \bot^\times = \Box^i \bot, \\ (\phi \oplus \psi)^\times = \Box^{j_\oplus}(\Box^{k_\oplus}\phi^\times \oplus \Box^{l_\oplus}\psi^\times) \text{ for } \oplus \in \{\wedge, \vee, \to\}. \end{cases}$$

We abbreviate $j_\to, k_\to, l_\to$ by j, k, l, respectively. By saying $(\cdot)^\times$ is a basic (potential) embedding, it is assumed that notations $h, i, j, k, l, j_\wedge, k_\wedge, l_\wedge, j_\vee, k_\vee, l_\vee$ are all reserved for the eleven parameters, in the way indicated above.

A potential embedding $(\cdot)^\times$ is an embedding *if it is actually a faithful embedding, i.e., if it satisfies: $\mathsf{IPC} \vdash \phi$ iff $\mathsf{S4} \vdash \phi^\times$.*

Fact 2. *The three embeddings in Table 1 are all basic embeddings. Actually,*

$$\begin{aligned} &\text{in } (\cdot)^\circ,\ k = k_\vee = l_\vee = 1; \\ &\text{in } (\cdot)^\Box,\ h = j = 1; \\ &\text{in } (\cdot)^\triangle,\ h = i = j = j_\wedge = j_\vee = 1; \end{aligned}$$

other parameters not mentioned here (for each of them, respectively) are all 0.

We present some macros that will be useful later.

Fact 3. *For any basic potential embedding $(\cdot)^\times$:*

$$\begin{aligned} (\neg\phi)^\times &= (\phi \to \bot)^\times = \Box^j(\Box^k\phi^\times \to \Box^l\bot^\times) = \Box^j(\Box^k\phi^\times \to \Box^{i+l}\bot), \\ (\neg\neg\phi)^\times &= \Box^j(\Box^k(\neg\phi)^\times \to \Box^{i+l}\bot) = \Box^j(\Box^k\Box^j(\Box^k\phi^\times \to \Box^{i+l}\bot) \to \Box^{i+l}\bot) \\ &= \Box^j(\Box^{j+k}(\Box^k\phi^\times \to \Box^{i+l}\bot) \to \Box^{i+l}\bot), \\ ((\phi \to q) \to q)^\times &= \Box^j(\Box^k(\phi \to q)^\times \to \Box^l q^\times) = \Box^j(\Box^k\Box^j(\Box^k\phi^\times \to \Box^l q^\times) \to \Box^{h+l}q) \\ &= \Box^j(\Box^{j+k}(\Box^k\phi^\times \to \Box^{h+l}q) \to \Box^{h+l}q). \end{aligned}$$

Some basic potential embeddings are not faithful embeddings. We have the following lemma, which provides a necessary condition on parameters:

Lemma 3. *If $(\cdot)^\times$ is a basic (faithful) embedding, then $j + k > 0$.*

Proof. Let $(\cdot)^\times$ be a potential embedding that satisfies $j = k = 0$. By Fact [3],

$$(\neg\neg\phi)^\times = \Box^0(\Box^{0+0}(\Box^0\phi^\times \to \Box^{i+l}\bot)\to\Box^{i+l}\bot) = (\phi^\times \to \Box^{i+l}\bot)\to\Box^{i+l}\bot.$$

Since $\mathsf{S4} \vdash \bot \leftrightarrow \Box^{i+l}\bot$, $\mathsf{S4} \vdash (\neg\neg\phi)^\times \leftrightarrow ((\phi^\times \to \bot)\to\bot)$, so $\mathsf{S4} \vdash (\neg\neg\phi)^\times \leftrightarrow \phi^\times$. Let ϕ be $\neg\neg p \to p$, we have $\mathsf{S4} \vdash (\neg\neg(\neg\neg p \to p))^\times \leftrightarrow (\neg\neg p \to p)^\times$.

Note that $\mathsf{IPC} \vdash \neg\neg(\neg\neg p \to p)$ and $\mathsf{IPC} \nvdash \neg\neg p \to p$. If $(\cdot)^\times$ is a faithful embedding, then $\mathsf{S4} \vdash (\neg\neg(\neg\neg p \to p))^\times$ and $\mathsf{S4} \nvdash (\neg\neg p \to p)^\times$. Thus we have a contradiction. □

We are now ready to show the following:

Theorem 6 (Direct Self-referentiality of IPC). *For any propositional formula α s.t. $\mathsf{CPC} \vdash \alpha$ and $\mathsf{IPC} \nvdash \alpha$:*

(1) $\mathsf{IPC} \vdash \neg\neg\alpha$.

(2) If α is an implication,[10] *then for any basic embedding $(\cdot)^\times$, any realization of $(\neg\neg\alpha)^\times$ calls for a directly self-referential constant specification.*

(3) There is a propositional formula β s.t. $\mathsf{IPC} \vdash \alpha \leftrightarrow \beta$ and for any basic embedding $(\cdot)^\times$, any realization of $(\neg\neg\beta)^\times$ calls for a directly self-referential constant specification.

(4) For any basic embedding $(\cdot)^\times$ that satisfies $k + j_\wedge > 0$ and $k + j_\vee > 0$, any realization of $(\neg\neg\alpha)^\times$ calls for a directly self-referential constant specification.

(5) Any realization of $(\neg\neg\alpha)^\triangle$ calls for a directly self-referential constant specification.

Proof. (1) $\mathsf{IPC} \vdash \neg\neg\alpha$ follows from $\mathsf{CPC} \vdash \alpha$ by Glivenko's Theorem (cf. [11]).

(2) Take any basic (faithful) embedding $(\cdot)^\times$. Since α is an implication, we assume $\alpha = \alpha_1 \to \alpha_2$, which implies $\alpha^\times = (\alpha_1 \to \alpha_2)^\times = \Box^j(\Box^k\alpha_1^\times \to \Box^l\alpha_2^\times)$.

Let $\sigma = \Box^k\alpha_1^\times \to \Box^l\alpha_2^\times$. We have

$$\alpha^\times = \Box^j\sigma. \tag{7}$$

By Fact [3], $(\neg\neg\alpha)^\times = \Box^j(\Box^{j+k}(\Box^k\alpha^\times \to \Box^{i+l}\bot)\to\Box^{i+l}\bot)$. Thus

$$(\neg\neg\alpha)^\times = \Box^j(\Box^{j+k}(\Box^{j+k}\sigma \to \Box^{i+l}\bot)\to\Box^{i+l}\bot). \tag{8}$$

Since $\mathsf{IPC} \nvdash \alpha$, $\mathsf{S4} \nvdash \alpha^\times$. By (7), $\mathsf{S4} \nvdash \Box^j\sigma$, which implies

$$\mathsf{S4} \nvdash \sigma. \tag{9}$$

Since $\mathsf{IPC} \vdash \neg\neg\alpha$, $\mathsf{S4} \vdash (\neg\neg\alpha)^\times$, then $\mathsf{S4} \vdash \Box^j(\Box^{j+k}(\Box^{j+k}\sigma \to \Box^{i+l}\bot)\to\Box^{i+l}\bot)$ follows from (8), which implies $\mathsf{S4} \vdash \Box^{j+k}(\Box^{j+k}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot$. Since $\mathsf{S4} \vdash \Box^{i+l}\bot \leftrightarrow \bot$, we have $\mathsf{S4} \vdash \Box^{j+k}(\Box^{j+k}\sigma \to \bot) \to \bot$. According to the assumption, $(\cdot)^\times$ is a faithful embedding. By Lemma [3],

$$j + k > 0, \tag{10}$$

[10] Note that the existence of such α is instanced by $\neg\neg p \to p$.

which implies $\mathsf{S4} \vdash \Box^{j+k}\phi \leftrightarrow \Box\phi$ for any ϕ. Therefore,

$$\mathsf{S4} \vdash \Box(\Box\sigma \to \bot) \to \bot = \Diamond\Box\sigma. \tag{11}$$

Having (9) and (11), from Theorem 4(2) we know that for any $w, x > 0$, any realization of $\Box^u(\Box^w(\Box^x\sigma \to \Box^y\bot) \to \Box^z\bot)$ calls for a directly self-referential constant specification.

Let $u = j$, $y = z = i+l$, $w = x = j+k$ (note that $w, x > 0$ by (10)), we know that

$$\Box^j(\Box^{j+k}(\Box^{j+k}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot)$$

calls for direct self-referentiality. Then by (8), any realization of $(\neg\neg\alpha)^\times$ calls for a directly self-referential constant specification.

(3) Let $\beta = \top \to \alpha$, which is an implication. $\mathsf{IPC} \nvdash \beta$ and $\mathsf{CPC} \vdash \beta$ follow from assumptions and the fact that $\mathsf{IPC} \vdash \alpha \leftrightarrow \beta$ and $\mathsf{CPC} \vdash \alpha \leftrightarrow \beta$. Now, β satisfies all requirements of (2).

(4) Take any basic (faithful) embedding $(\cdot)^\times$ that satisfies $k + j_\wedge > 0$ and $k + j_\vee > 0$. Since $\mathsf{CPC} \vdash \alpha$, α is not a prime formula. The case that α is an implication follows from (2). Thus it is sufficient to prove with the assumption that $\alpha = \alpha_1 \oplus \alpha_2$ for $\oplus \in \{\wedge, \vee\}$. Now $\alpha^\times = \Box^{j_\oplus}(\Box^{k_\oplus}\alpha_1^\times \oplus \Box^{l_\oplus}\alpha_2^\times)$. Let $\sigma = \Box^{k_\oplus}\alpha_1^\times \oplus \Box^{l_\oplus}\alpha_2^\times$, we have $\alpha^\times = \Box^{j_\oplus}\sigma$.

Since $\mathsf{IPC} \nvdash \alpha$, $\mathsf{S4} \nvdash \alpha^\times$, hence $\mathsf{S4} \nvdash \sigma$. Since $\mathsf{IPC} \vdash \neg\neg\alpha$, $\mathsf{S4} \vdash (\neg\neg\alpha)^\times$. By Fact 3 and our definition of σ,

$$\begin{aligned}(\neg\neg\alpha)^\times &= \Box^j(\Box^{j+k}(\Box^k\alpha^\times \to \Box^{i+l}\bot) \to \Box^{i+l}\bot)\\ &= \Box^j(\Box^{j+k}(\Box^{k+j_\oplus}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot),\end{aligned}$$

hence $\mathsf{S4} \vdash \Box^j(\Box^{j+k}(\Box^{k+j_\oplus}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot)$. Lemma 3 says $j + k > 0$ while $k + j_\oplus > 0$ is given, which implies $\mathsf{S4} \vdash \Box(\Box\sigma \to \bot) \to \bot = \Diamond\Box\sigma$.

In the sense that $u = j$, $w = j + k > 0$, $x = k + j_\oplus > 0$, $y = z = i + l$, Theorem 4(2) states that any realization of $\Box^j(\Box^{j+k}(\Box^{k+j_\oplus}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot)$ calls for a directly self-referential constant specification. This finishes our proof since $(\neg\neg\alpha)^\times = \Box^j(\Box^{j+k}(\Box^{k+j_\oplus}\sigma \to \Box^{i+l}\bot) \to \Box^{i+l}\bot)$.

(5) A consequence of (4), since in $(\cdot)^\triangle$, $j_\wedge = j_\vee = 1$.[11] $\Box$

By Theorem 6(2), we cannot embed-and-realize the IPC-theorem

$$\neg\neg(\neg\neg p \to p)$$

without direct self-referentiality. Since $\neg$ is defined on $\bot$ and $\to$, this IPC-theorem involves $\bot$. In Theorem 7, we will give a directly self-referential example in $\mathsf{IPC}_\to$, i.e., the purely implicational fragment of IPC.[12]

[11] Actually, $(\cdot)^\circ$ has parameter $k = 1$, and hence it also follows from (4) that any realization of $(\neg\neg\alpha)^\circ$ calls for a directly self-referential constant specification, although we have proved this in Theorem 5. Also note that $(\cdot)^\Box$ has $k = j_\wedge = j_\vee = 0$, and hence does not fit the requirements of (4).

[12] The question whether there is a self-referential example in $\mathsf{IPC}_\to$ was raised by Melvin Fitting.

Theorem 7 (Direct Self-referentiality of $\mathsf{IPC}_{\rightarrow}$). *Let γ be the* IPC*-theorem*

$$((((p\to q)\to p)\to p)\to q)\to q.$$

For any basic embedding $(\cdot)^{\times}$, any realization of $\gamma^{\times}$ calls for a directly self-referential constant specification.

Proof. Let $\chi = ((p\to q)\to p)\to p$, then $\gamma = ((((p\to q)\to p)\to p)\to q)\to q$ can be abbreviated as $(\chi\to q)\to q$.

Take any basic (faithful) embedding $(\cdot)^{\times}$. By Lemma 3, $j+k > 0$. By Fact 3, $((\chi\to q)\to q)^{\times} = \Box^{j}(\Box^{j+k}(\Box^{k}\chi^{\times}\to\Box^{h+l}q)\to\Box^{h+l}q)$.

Note that $\chi^{\times} = (((p\to q)\to p)\to p)^{\times} = \Box^{j}(\Box^{k}((p\to q)\to p)^{\times}\to\Box^{l}p^{\times})$. Let $\sigma = \Box^{k}((p\to q)\to p)^{\times}\to\Box^{l}p^{\times}$, we have $\chi^{\times} = \Box^{j}\sigma$, and hence

$$\gamma^{\times} = ((\chi\to q)\to q)^{\times} = \Box^{j}(\Box^{j+k}(\Box^{j+k}\sigma\to\Box^{h+l}q)\to\Box^{h+l}q). \tag{12}$$

Since $\mathsf{IPC}\nvdash\chi$, $\mathsf{S4}\nvdash\chi^{\times} = \Box^{j}\sigma$, and hence $\mathsf{S4}\nvdash\sigma$.

Note that q is a propositional formula, and $\mathsf{S4}\nvdash q$. Employing Theorem 4(1) in the sense that $u=j$, $w=x=j+k > 0$, $z=h+l$, $\theta = \Box^{h+l}q$, and $\zeta = q$, we know that any realization of $\Box^{j}(\Box^{j+k}(\Box^{j+k}\sigma\to\Box^{h+l}q)\to\Box^{h+l}q)$ has a counter $\mathcal{CS}^{\varnothing}$-model. By (12), any realization of $\gamma^{\times}$ has a counter $\mathcal{CS}^{\varnothing}$-model.

Suppose we were able to realize $\gamma^{\times}$ in LP without calling for a directly self-referential constant specification, then we must have an $\mathsf{LP}(\mathcal{CS}^{\varnothing})$-proof of that realization, say κ. By Theorem 3, $\mathcal{M}\vDash\kappa$ for any $\mathcal{CS}^{\varnothing}$-model $\mathcal{M}$, which contradicts the existence of the counter $\mathcal{CS}^{\varnothing}$-model given by Theorem 4(1). $\Box$

Theorem 7 indicates that the BHK semantics of intuitionistic implication involves direct self-referentiality.

3 Conclusions

In this work, we generalize Kuznets' method from [3] and find a natural class of $\mathsf{S4}$-theorems that call for direct self-referentiality in realization. Then we consider Gödel-style basic provability embeddings of IPC in $\mathsf{S4}$ and find IPC-theorems that call for direct self-referentiality under each of these embeddings. This suggests that the BHK semantics of intuitionistic logic (even its purely implicational fragment) is intrinsically self-referential. In particular, this could explain the well-known difficulties with formalizing the BHK semantics prior to the Logic of Proofs: any system of proof terms sufficient for realizing IPC is necessarily self-referential and hence requires some kind of fixed-point construction for its provability interpretation.

Acknowledgements. The author is grateful to Sergei Artemov for his supervision. This work is inspired by Roman Kuznets' earlier paper [3]. Many thanks to Melvin Fitting who raises helpful questions. An anonymous referee of LFCS 2013 offers comments in extraordinary detail. Also, the author is indebted to Karen Kletter for her help in editing this note.

References

1. Artemov, S.N.: Operational modal logic, Technical Report MSI 95-29. Cornell University (1995)
2. Artemov, S.N.: Explicit provability and constructive semantics. The Bulletin of Symbolic Logic 7(1), 1–36 (2001)
3. Brezhnev, V.N., Kuznets, R.: Making knowledge explicit: How hard it is. Theoretical Computer Science 357(1-3), 23–34 (2006)
4. Chagrov, A., Zakharyashchev, M.: Modal companions of intermediate propositional logics. Studia Logica 51, 49–82 (1992)
5. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
6. Fitting, M.: Realizations and LP. Annals of Pure and Applied Logic 161(3), 368–387 (2009)
7. Gödel, K.: Eine Interpretation des intuitionistischen Aussagenkalkuls. Ergebnisse Eines Mathematischen Kolloquiums 4, 39–40 (1933); English translation in: Feferman, S., et al (eds.): Kurt Gödel Collected Works, vol. 1, pp. 301–303. Oxford University Press, Clarendon Press, Oxford, New York (1986)
8. Kuznets, R.: Self-referential justifications in epistemic logic. Theory of Computing Systems 46(4), 636–661 (2010)
9. McKinsey, J.C.C., Tarski, A.: Some theorems about the sentential calculi of Lewis and Heyting. The Journal of Symbolic Logic 13, 1–15 (1948)
10. Mkrtychev, A.: Models for the Logic of Proofs. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)
11. Sørensen, M.H., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Elsevier, Amsterdam (2006)
12. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory, 2nd edn. Cambridge University Press, Cambridge (2000)

Author Index