

Alexandra Silva
Renata Wassermann
Ruy de Queiroz (Eds.)

LNCS 13038

Logic, Language, Information, and Computation

27th International Workshop, WoLLIC 2021
Virtual Event, October 5–8, 2021
Proceedings

 Springer



Founding Editors

Gerhard Goos, Germany
Juris Hartmanis, USA

Editorial Board Members

Elisa Bertino, USA
Wen Gao, China
Bernhard Steffen , Germany

Gerhard Woeginger , Germany
Moti Yung, USA

FoLLI Publications on Logic, Language and Information Subline of Lectures Notes in Computer Science

Subline Editors-in-Chief

Valentin Goranko, *Stockholm University, Sweden*
Michael Moortgat, *Utrecht University, The Netherlands*

Subline Area Editors

Nick Bezhanishvili, *University of Amsterdam, The Netherlands*
Anuj Dawar, *University of Cambridge, UK*
Philippe de Groot, *Inria Nancy, France*
Gerhard Jäger, *University of Tübingen, Germany*
Fenrong Liu, *Tsinghua University, Beijing, China*
Eric Pacuit, *University of Maryland, USA*
Ruy de Queiroz, *Universidade Federal de Pernambuco, Brazil*
Ram Ramanujam, *Institute of Mathematical Sciences, Chennai, India*

More information about this subseries at <http://www.springer.com/series/7407>

Alexandra Silva · Renata Wassermann ·
Ruy de Queiroz (Eds.)

Logic, Language, Information, and Computation

27th International Workshop, WoLLIC 2021
Virtual Event, October 5–8, 2021
Proceedings

Editors

Alexandra Silva
Cornell University
New York, NY, USA

Ruy de Queiroz
Centro de Informática
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil

Renata Wassermann
Instituto de Matemática e Estatística
Universidade de São Paulo
Sao Paulo, São Paulo, Brazil

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-88852-7 ISBN 978-3-030-88853-4 (eBook)
<https://doi.org/10.1007/978-3-030-88853-4>

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 27th Workshop on Logic, Language, Information and Computation (WoLLIC 2021) held virtually during October 5–8, 2021. The WoLLIC series of workshops started in 1994 with the aim of fostering interdisciplinary research in pure and applied logic. The idea is to have a forum which is large enough in the number of possible interactions between logic and the sciences related to information and computation, and yet is small enough to allow for concrete and useful interaction among participants.

There were 44 submissions. Each submission was reviewed by at least three program committee members. The committee decided to accept 25 papers. The program also includes six invited lectures by Catarina Dutilh Novaes (VU Amsterdam, The Netherlands), Santiago Figueira (Universidad de Buenos Aires, Argentina), Andreas Herzig (IRIT, France), and Cláudia Nalon (UnB, Brazil).

We would very much like to thank all Program Committee members and external reviewers for the work they put into reviewing the submissions. The help provided by the EasyChair system created by Andrei Vorokonkov is gratefully acknowledged. Finally, we would like to acknowledge the scientific sponsorship of the following organizations: Interest Group in Pure and Applied Logics (IGPL), The Association for Logic, Language and Information (FoLLI), Association for Symbolic Logic (ASL), European Association for Theoretical Computer Science (EATCS), European Association for Computer Science Logic (EACSL), Sociedade Brasileira de Computação (SBC), and Sociedade Brasileira de Lógica (SBL).

October 2021

Alexandra Silva
Renata Wassermann
Ruy de Queiroz

Organization

Program Committee

Carlos Areces	Universidad Nacional de Córdoba, Argentina
Arthur Amorim Azevedo	Boston University, USA
Paul Brunet	University College London, UK
Nina Gierasimczuk	Technical University of Denmark, Denmark
Helle Hvid Hansen	University of Groningen, The Netherlands
Justin Hsu	University of Pennsylvania, USA
Fairouz Kamareddine	Heriot-Watt University, UK
Sandra Kiefer	RWTH Aachen University, Germany
Clemens Kupke	University of Strathclyde, UK
Konstantinos Mamouras	Rice University, USA
Maria Vanina Martinez	Universidad de Buenos Aires, Argentina
Larry Moss	Indiana University, USA
Claudia Nalon	University of Brasilia, Brazil
Valeria de Paiva	Samsung Research, USA
Elaine Pimentel	Universidade Federal do Rio Grande do Norte, Brazil
Revantha Ramanayake	University of Groningen, The Netherlands
Jurriaan Rot	Radboud University, The Netherlands
Alexandra Silva	University College London
Christine Tasson	IRIF, France
Sebastiaan Terwijn	Radboud University, The Netherlands
Renata Wassermann	Univ São Paulo, Brazil

Program Committee Co-chairs

Alexandra Silva	University College London
Renata Wassermann	Univ São Paulo, Brazil

General Chair

Ruy de Queiroz	.
----------------	---

Contents

Formalized Soundness and Completeness of Epistemic Logic	1
<i>Asta Halkjær From</i>	
A Logical Characterization of Constant-Depth Circuits over the Reals	16
<i>Timon Barlag and Heribert Vollmer</i>	
Wanted Dead or Alive: Epistemic Logic for Impure Simplicial Complexes. . .	31
<i>Hans van Ditmarsch</i>	
Doubly Strongly First Order Dependencies.	47
<i>Pietro Galliani</i>	
Explicit Non-normal Modal Logic	64
<i>Atefeh Rohani and Thomas Studer</i>	
A General Relational Semantics of Propositional Logic: Axiomatization	82
<i>Shengyang Zhong</i>	
Meaning and Computing: Two Approaches to Computable Propositions	100
<i>Ivo Pezlar</i>	
Modal Logic via Global Consequence	117
<i>Xuefeng Wen</i>	
Games for Hybrid Logic: From Semantic Games to Analytic Calculi.	133
<i>Robert Freiman</i>	
Verifying the Conversion into CNF in Dafny	150
<i>Viorel Iordache and Ștefan Ciobâcă</i>	
Analysis in a Formal Predicative Set Theory	167
<i>Nissan Levi and Arnon Avron</i>	
Coherence via Focusing for Symmetric Skew Monoidal Categories.	184
<i>Niccolò Veltri</i>	
On the Subtle Nature of a Simple Logic of the Hide and Seek Game	201
<i>Dazhu Li, Sujata Ghosh, Fenrong Liu, and Yaxin Tu</i>	
Orthogonal Frames and Indexed Relations	219
<i>Philippe Balbiani and Saúl Fernández González</i>	
Computable Execution Traces.	235
<i>Declan Thompson</i>	

Axiomatic Reals and Certified Efficient Exact Real Computation	252
<i>Michal Konečný, Sewon Park, and Holger Thies</i>	
Lorenzen Won the Game, Lorenz Did Too: Dialogical Logic for Ellipsis and Anaphora Resolution	269
<i>Davide Catta and Symon Jory Stevens-Guille</i>	
Uniform Lyndon Interpolation for Basic Non-normal Modal Logics	287
<i>Amirhossein Akbar Tabatabai, Rosalie Iemhoff, and Raheleh Jalali</i>	
On the Expressive Power of TeamLTL and First-Order Team Logic over Hyperproperties	302
<i>Juha Kontinen and Max Sandström</i>	
Characterizations for XPath _R (↓)	319
<i>Nicolás González and Sergio Abriola</i>	
Uniform Interpolation via Nested Sequents	337
<i>Iris van der Giessen, Raheleh Jalali, and Roman Kuznets</i>	
Disjunction and Negation in Information Based Semantics	355
<i>Vít Punčochář and Andrew Tedder</i>	
Algorithmically Broad Languages for Polynomial Time and Space	372
<i>Daniel Leivant</i>	
A Pure View of Ecumenical Modalities	388
<i>Sonia Marin, Luiz Carlos Pereira, Elaine Pimentel, and Emerson Sales</i>	
Provability Games for Non-classical Logics: Mezhirov Game for MPC, KD!, and KD	408
<i>Alexandra Pavlova</i>	
Author Index	427



Formalized Soundness and Completeness of Epistemic Logic

Asta Halkjær From  

Technical University of Denmark, Kongens Lyngby, Denmark
ahfrom@dtu.dk

Abstract. We strengthen the foundations of epistemic logic by formalizing the family of normal modal logics in the proof assistant Isabelle/HOL. We define an abstract canonical model and prove a truth lemma for any logic in the family. We then instantiate it with logics based on various epistemic principles to obtain completeness results for systems from K to S5. Our work gives a disciplined treatment of completeness-via-canonicity arguments and demonstrates their compositionality.

Keywords: Epistemic logic · Isabelle/HOL · Completeness · Canonicity

1 Introduction

Epistemic logic provides a foundation for reasoning about the knowledge of agents, both factual (“I know the sky is blue”) and higher-order (“I know that you know that I know the sky is blue”). This has applications in computer science and artificial intelligence [17]. Basic epistemic logic extends propositional logic with a knowledge modality K_i that, for each agent i , expresses that agent’s knowledge. For example, the following formula states that: (i) agent 1 knows φ , (ii) agent 2 knows that agent 1 knows φ , but (iii) agent 1 does not know (ii):

$$K_1\varphi \wedge K_2K_1\varphi \wedge \neg K_1K_2K_1\varphi$$

Such formulas are understood on possible worlds models that represent the different situations that agents consider possible, including which situations are indistinguishable to them (due to a lack of observations or similar). Different things can be true at each possible world and the uncertainty of each agent is modeled by relating the worlds through so-called accessibility relations, one for each agent. The agent finds itself at some world and it considers those worlds possible that are accessible from this world. Thus, an agent *knows* something if it holds at every world that the agent considers possible, i.e. all accessible worlds.

To model different kinds of knowledge, we consider classes of possible worlds models. For instance, we may want *true knowledge*: if something is known then it is true. Then we consider models with reflexive accessibility relations, where the agents must always consider the current world. If we want *positive introspection*,

if the agent knows something then it knows that it knows it, we want transitive accessibility relations. There are a range of epistemic principles to consider.

With a deductive proof system, we can use just a few axioms and inference rules, with different principles resulting in different axioms, to reason about the consequences of such epistemic principles: what formulas classify different possible worlds models and what formulas are true on this class. To trust such reasoning we need to know that the system is sound and thus only allows us to deduce formulas that hold on our considered class. Moreover, we want the system to be complete: if we cannot prove a formula, then it is not due to a limitation of the proof system but because the formula is “incorrect”.

In this paper we formalize epistemic logic with countably many agents [9] in the proof assistant Isabelle/HOL [19]. We consider the so-called normal modal logics, from the smallest, system K, valid over all models to S5 where the accessibility relations must be reflexive, symmetric and transitive. We base our proofs on the textbooks *Reasoning About Knowledge* by Fagin, Halpern, Moses and Vardi [7], which mainly proves completeness for system K, and *Modal Logic* by Blackburn, de Rijke and Venema [3], which goes further.

Unfortunately, in textbooks it is easy to treat extensions to system K informally. For the completeness of system T on reflexive models, Fagin et al. [7] write: “A proof identical to that of Theorem 3.1.3 can now be used.” In a formalized setting we do not want to *copy/paste* our efforts but rather find a suitable abstraction of the theorem that works for both cases. We seek to achieve the compositionality expressed by Blackburn et al. [3] (emphasis ours):

The canonical frame of any normal logic containing T is reflexive, the canonical frame of any normal logic containing B is symmetric, and the canonical frame of any normal logic containing D is right unbounded. *This allows us to ‘add together’ our results.*

To this effect, we give a disciplined treatment of normal modal logics and completeness-via-canonicity [3] by formalizing an abstract account of the Henkin-style completeness method. This is made possible by parameterizing our proof system and the notion of a maximal consistent set to allow for an open-ended number of additional axioms. We obtain an abstract truth lemma that we reuse for each logic and fix the axioms pertaining to each system afterwards. Where Fagin et al. suggest using an identical proof, we reuse the exact same one, instantiated accordingly. Notably, our definitions and proofs are specified in the precise language of higher-order logic and every step of our reasoning is checked mechanically. While the results are not new, our approach, as well as this level of precision and guarantee, is. Our formalization [9] provides a recipe for extending the work with similar logics and can serve as starting point for related work.

A short extended abstract [12] describes an earlier version of the formalization, covering only system K and not the family of normal modal logics.

We discuss related work in Sect. 2 before diving into the precise syntax and semantics in Sect. 3. Section 4 introduces our formalization of normal modal logics as a family of proof systems and Sect. 5 presents the abstract completeness result. In Sect. 6 we consider concrete axiom systems (K, T, KB, K4, S4) and

how to prove their completeness in a composable way. Section 7 discusses two variants of system S5 and Sect. 8 concludes with future work.

Remarks on the Formalization. The formalization is available online [9] and consists of more than 1400 lines of Isabelle/HOL text. Half of the lines prove the abstract completeness result and the second half concerns concrete systems. We reproduce select definitions and results here, but none of the proofs. The Isabelle text is close enough to formal English, with notation from mathematical logic, that we trust the accompanying explanations make it understandable. We present it in this way to remind the reader of the formal guarantees behind each result: each proof was checked by the proof assistant.

Producing a formalization of this size is a significant undertaking: every step of reasoning in every proof must be written down explicitly and the gaps must be sufficiently small such that the proof assistant can fill them in. This is a craft. Some of this process is pure labor, but a significant part is getting the definitions right, not just to match our intended meaning but so that they are easy to work with. For instance, we can formalize substitution instances of propositional tautologies (cf. Sect. 4.1) or the worlds of the canonical model (cf. Sect. 5.3) in different ways. One choice can lead to a lot less work than another.

2 Related Work

Wu and Gore [20] formalize modal tableaux with histories for the modal logics K, KT and S4 in Lean, giving formally verified decision procedures for these logics. Bentzen [1] also works in Lean but formalizes a Hilbert-style system for single-agent S5 with a Henkin-style completeness proof similar to ours. Bentzen only considers S5 built from axioms T, B and 4 while we also consider the combination of T and 5, as well as a wider range of normal modal logics. Both Li [15] and Neeley [18] have recently formalized dynamic epistemic logic [6] in Lean including Henkin-style completeness proofs for multi-agent S5 and public announcement logic. We have recently formalized soundness and completeness of public announcement logic [11] in Isabelle/HOL by building on the work presented here. Hagemeyer [13] formalized intuitionistic epistemic logic in Coq including completeness of a natural deduction proof system. Slightly differently, Magessi and Brogi [16] have given a formal proof of modal completeness for provability logic in HOL Light. We note that most of this work is unpublished and all of it focuses on specific systems rather than a family of logics, as we do. The unverified tool SQEMA by Conradie et al. [5] proves canonicity of modal formulas, giving a purely algorithmic approach to proving canonical completeness in modal logic.

Xiong et al. [21] present a variant of epistemic logic that adds the notion of secret knowledge as a first-class citizen. They introduce a new modality of secrets instead of defining it in terms of the knowledge operator. The authors argue that the main principles can be studied this way, for instance when considering a language with an operator for secrets and without the usual knowledge operator. We think it would be interesting to formalize their work in a proof assistant.

Kądziołka [14] formalized a solution to a logic puzzle in Isabelle/HOL, using a logic tailored to the problem that is very similar to the possible worlds model of epistemic logic. Our formalization might be used for this reasoning instead.

Blanchette et al. [4] formalize an abstract completeness result for various flavors of first-order logic. Besides the different logic, they consider Gentzen and tableau systems, where we consider axiomatic proof systems. Their technique is analytic, based on inspecting infinite proof attempts, where ours is synthetic, in the Henkin style, using maximal consistent sets of formulas.

We have also used the synthetic technique to formalize completeness for a tableau system for hybrid logic [8, 10].

3 Syntax and Semantics

The well-formed formulas of our epistemic logic are given by the following grammar, where we denote propositional symbols by x and agent labels by i :

$$\phi, \psi ::= \perp \mid x \mid \phi \vee \psi \mid \phi \wedge \psi \mid \phi \rightarrow \psi \mid K_i \phi$$

The K_i operator, “agent i knows” is typically written \Box_i in non-epistemic multimodal logic. We write L_i for the dual modality, “agent i considers possible”, typically written \Diamond_i . We use x for propositional symbols, since we use p and q instead of ϕ and ψ in the Isabelle/HOL text. This is a common choice in formalizations of logic and their presentations [1, 2, 16], regardless of the chosen proof assistant. Similarly, we write $K i$ for K_i .

We interpret the language on Kripke models $\mathfrak{M} = (\mathcal{F}, V)$. The frame $\mathcal{F} = (W, R_1, R_2, \dots)$ consists of a non-empty set of worlds W and binary accessibility relations R_i between them, one for each agent. V is the valuation of propositional symbols. Formula satisfiability is defined as follows:

$$\begin{aligned} \mathfrak{M}, w &\not\models \perp \\ \mathfrak{M}, w &\models x && \text{iff } w \in V(x) \\ \mathfrak{M}, w &\models \phi \vee \psi && \text{iff } \mathfrak{M}, w \models \phi \text{ or } \mathfrak{M}, w \models \psi \\ \mathfrak{M}, w &\models \phi \wedge \psi && \text{iff } \mathfrak{M}, w \models \phi \text{ and } \mathfrak{M}, w \models \psi \\ \mathfrak{M}, w &\models \phi \rightarrow \psi && \text{iff } \mathfrak{M}, w \not\models \phi \text{ or } \mathfrak{M}, w \models \psi \\ \mathfrak{M}, w &\models K_i \phi && \text{iff } w R_i w' \text{ implies } \mathfrak{M}, w' \models \phi \text{ for all } w' \in W \end{aligned}$$

We deeply embed our language as a datatype in the higher-order logic, which the semantics interprets. Kripke models are formalized using a type variable $'w$ to represent the domain of worlds and $'i$ for the type of agent labels. We include the set of worlds explicitly as \mathcal{W} . In doing so, we can consider models over sets of worlds that are cumbersome to define as subtypes. We write π for the valuation and \mathcal{K} for the accessibility relations:

datatype ($'i$, $'w$) *kripke* =
Kripke (\mathcal{W} : $'w$ set) (π : $'w \Rightarrow id \Rightarrow bool$) (\mathcal{K} : $'i \Rightarrow 'w \Rightarrow 'w$ set)

We omit the so-called cartouches from the presentation of our Isabelle code. The type of the semantics is formalized as follows:

primrec semantics :: ('i, 'w) kripke \Rightarrow 'w \Rightarrow 'i fm \Rightarrow bool

In the case for K_i we only consider accessible worlds in the set \mathcal{W} (of M):

| $(M, w \models K_i p) = (\forall v \in \mathcal{W} M \cap \mathcal{K} M_i w. M, v \models p)$

4 Normal Modal Logic

We consider the family of normal modal logics, namely those that contain all propositional tautologies and formulas of the form $K_i \phi \wedge K_i (\phi \rightarrow \psi) \rightarrow K_i \psi$ and are closed under modus ponens and necessitation.

4.1 Proof System

In Isabelle, we formalize the family as the following inductive predicate \vdash , which is parameterized by a predicate A on the left-hand side. This is used to admit additional formulas as axioms. The implication \Longrightarrow allows us to conclude the right-hand side given the left-hand side:

inductive AK :: ('i fm \Rightarrow bool) \Rightarrow 'i fm \Rightarrow bool (- \vdash - [50, 50] 50)

for $A :: 'i fm \Rightarrow bool$ **where**

- $A1$: tautology $p \Longrightarrow A \vdash p$
- | $A2$: $A \vdash (K_i p \wedge K_i (p \longrightarrow q) \longrightarrow K_i q)$
- | Ax : $A p \Longrightarrow A \vdash p$
- | $R1$: $A \vdash p \Longrightarrow A \vdash (p \longrightarrow q) \Longrightarrow A \vdash q$
- | $R2$: $A \vdash p \Longrightarrow A \vdash K_i p$

Axiom $A1$ derives any tautology (under any axiom predicate) and $A2$ derives any instance of the distributivity of K_i over implication. The special axiom Ax derives any formula admitted by A . Finally, $R1$ is modus ponens and $R2$ is necessitation. Note that logics in this family are only closed under substitution if A is: it is harmless to permit an A that admits $K_i x \rightarrow x$ but not $K_i x' \rightarrow x'$ for distinct x and x' , but we will not consider any such concrete logics here.

We obtain the smallest normal modal logic K when A admits no extra axioms.

The formulas derivable by $A1$ are typically described as substitution instances of propositional tautologies. For instance, $K_i x \vee \neg K_i x$ is derivable because it is obtained by substituting $K_i x$ for p in the tautology $p \vee \neg p$. To avoid formalizing substitution, we classify tautologies semantically, giving a propositional semantics for our language. That is, we treat formulas of the form $K_i \phi$ as a different sort of propositional symbols whose truth value is given by another valuation. The tautologies are the formulas that are valid under this semantics:

abbreviation tautology $p \equiv \forall g h. eval\ g\ h\ p$

4.2 Soundness

We give a generalized soundness result for our family of logics. Fix a predicate P on models (e.g. admitting reflexive ones). If the axioms admitted by A are sound on P -models, then the normal modal logic based on A is sound on P -models:

theorem *soundness*:

fixes $M :: ('i, 'w) \text{ kripke}$

assumes $\bigwedge(M :: ('i, 'w) \text{ kripke}) w p. A p \implies P M \implies w \in \mathcal{W} M \implies M, w \models p$

shows $A \vdash p \implies P M \implies w \in \mathcal{W} M \implies M, w \models p$

The Isabelle symbol \bigwedge in the assumption quantifies universally over a model M , world w and formula p . This potentially quantifies over too many worlds, so we add the assumption that w is the model's set of worlds. The proof of the theorem relies on the simple fact that any *tautology* is valid in all models.

4.3 Derived Rules

The formalization contains a number of derived rules that ease the completeness proof. We note that the precise meta-language of the proof assistant allows us to state many such rules in a methodical way. Consider the following lemma, which allows us to lift any derivation of an implication into the K_i operator:

lemma *K-map*:

assumes $A \vdash (p \longrightarrow q)$

shows $A \vdash (K i p \longrightarrow K i q)$

In the derivation of another formula, we can readily reuse this result and the proof assistant makes sure we have applied it correctly: that p and q can be instantiated so the assumption is met and the conclusion matches the desired result. In this way, we can compose larger derivations out of smaller pieces in a disciplined way and the proof automation of Isabelle can even help find the right pieces and put them together for us.

5 Abstract Completeness

We follow the completeness proof for system K by Fagin et al. [7] but in our generalized setting. As such, we do not just talk about maximal consistent sets (MCSs) but MCSs with respect to a choice of additional axioms A (A -MCSs). Likewise, our canonical model is parameterized by such an A , which we will fix later to obtain completeness of various logics over various classes of frames.

5.1 Maximal Consistent Sets

A potentially infinite set of formulas S is A -consistent if there is no finite subset $S' \subseteq S$ from which, in the presence of A , we can derive a contradiction (\perp):

definition *consistent* :: (*i fm* \Rightarrow *bool*) \Rightarrow '*i fm set* \Rightarrow *bool* **where**
consistent *A S* $\equiv \nexists S'. \text{set } S' \subseteq S \wedge A \vdash \text{imply } S' \perp$

Here $S' = [\phi_1, \dots, \phi_n]$ is some (finite and potentially empty) list of formulas and the expression *imply* $S' \perp$ builds the formula $\phi_1 \rightarrow \dots \rightarrow \phi_n \rightarrow \perp$.

A set of formulas is *A*-maximal if any proper extension breaks *A*-consistency:

definition *maximal* :: (*i fm* \Rightarrow *bool*) \Rightarrow '*i fm set* \Rightarrow *bool* **where**
maximal *A S* $\equiv \forall p. p \notin S \longrightarrow \neg \text{consistent } A (\{p\} \cup S)$

It is straightforward to verify some classic properties about *A*-MCSs [3,7]:

theorem *mcs-properties*:

assumes *consistent A V* **and** *maximal A V*

shows $A \vdash p \Longrightarrow p \in V$

and $p \in V \longleftrightarrow (\neg p) \notin V$

and $p \in V \Longrightarrow (p \longrightarrow q) \in V \Longrightarrow q \in V$

Given an *A*-MCS *V* we have (i) any derivable formula (using *A*) is in *V* (and by *Ax* so is any formula admitted by *A*), (ii) exactly one of ϕ and $\neg\phi$ is in *V* and (iii) *V* is closed under modus ponens. See the formalization for the proofs.

5.2 Lindenbaum's Lemma

When the agent labels are countable, which we assume from now on, so is our language, allowing us to assume an enumeration (ϕ_n) of formulas. In Isabelle, the *countable-datatype* method realizes this enumeration automatically by providing a surjective function *from-nat* from natural numbers to formulas of our language.

We can then extend any *A*-consistent set of formulas S_0 into an *A*-MCS in the usual way by building an infinite sequence of consistent sets S_0, S_1, S_2, \dots and taking their union. Given S_n , construct S_{n+1} like so:

$$S_{n+1} = \begin{cases} S_n & \text{if } \{\phi_n\} \cup S_n \text{ is not } A\text{-consistent} \\ \{\phi_n\} \cup S_n & \text{otherwise} \end{cases}$$

In Isabelle, *extend A S f n* constructs element S_n of the sequence, taking $S_0 = S$ and using *f* to enumerate the formulas. *Extend A S f* gives the infinite union $\bigcup_n S_n$. The result is *A*-consistent when the starting point is *A*-consistent and maximal when the enumeration is surjective (cf. the formalization for proofs):

lemma *consistent-Extend*:

assumes *consistent A S*

shows *consistent A (Extend A S f)*

lemma *maximal-Extend*:

assumes *surj f*

shows *maximal A (Extend A S f)*

5.3 Model Existence

The worlds of the canonical model are A -MCSs, which we abbreviate:

abbreviation $mcss :: ('i\ fm \Rightarrow bool) \Rightarrow 'i\ fm\ set\ set$ **where**
 $mcss\ A \equiv \{W. consistent\ A\ W \wedge maximal\ A\ W\}$

The remaining parts are the valuation pi and the accessibility relation $reach$:

abbreviation $pi :: 'i\ fm\ set \Rightarrow id \Rightarrow bool$ **where**
 $pi\ V\ x \equiv Pro\ x \in V$

abbreviation $known :: 'i\ fm\ set \Rightarrow 'i \Rightarrow 'i\ fm\ set$ **where**
 $known\ V\ i \equiv \{p. K\ i\ p \in V\}$

abbreviation $reach :: ('i\ fm \Rightarrow bool) \Rightarrow 'i \Rightarrow 'i\ fm\ set \Rightarrow 'i\ fm\ set\ set$ **where**
 $reach\ A\ i\ V \equiv \{W. known\ V\ i \subseteq W\}$

The valuation pi states that proposition x holds in a world V iff $x \in V$. Where Fagin et al. [7] write V/K_i for the set of formulas known by agent i at V , i.e. $\{\phi \mid K_i\phi \in V\}$, we write $known\ V\ i$. The worlds *reachable* by i from V are those A -MCSs that contain all formulas *known* at V by i .

We formalize the usual truth lemma: the canonical model satisfies formula ϕ at A -MCS V iff $\phi \in V$ (we need the negated case too for the induction):

lemma *truth-lemma*:

fixes A **and** $p :: ('i :: countable)\ fm$

defines $M \equiv Kripke\ (mcss\ A)\ pi\ (reach\ A)$

assumes $consistent\ A\ V$ **and** $maximal\ A\ V$

shows $(p \in V \iff M, V \models p) \wedge ((\neg p) \in V \iff M, V \models \neg p)$

The proof is by structural induction on the formula ϕ . The only non-trivial case is for the K_i -operator where we need to show that when $K_i\psi$ is satisfied at V then $K_i\psi \in V$. We follow the proof by Fagin et al. [7] and note that $\neg\psi$ must be inconsistent with the formulas *known* at V by i . If they were consistent, they could be extended into an A -MCS satisfying both $\neg\psi$ and $known\ V\ i$ making it accessible from V , which would then satisfy $\neg K_i\psi$ as well as $K_i\psi$. A contradiction. Thus, we can derive ψ from some finite subset $L \subseteq known\ V\ i$ of the known formulas. By necessitation, agent i knows this and by distributivity of K_i over implication, if agent i knows all of L then it knows ψ . Since L is a subset of what i knows, the thesis follows immediately.

It is useful to gather Lindenbaum's lemma and the truth lemma in a lemma stating that any formula in an A -consistent set is satisfied at the constructed A -MCS in the canonical model:

lemma *canonical-model*:

assumes $consistent\ A\ S$ **and** $p \in S$

defines $V \equiv Extend\ A\ S\ from\ nat$ **and** $M \equiv Kripke\ (mcss\ A)\ pi\ (reach\ A)$

shows $M, V \models p$ **and** $consistent\ A\ V$ **and** $maximal\ A\ V$

Strong completeness follows directly. If ϕ is valid under assumptions G but has no derivation from a subset of G , then $\{\neg\phi\} \cup G$ is A -consistent and the canonical model satisfies both $\neg\phi$ and all of G . This, however, contradicts the validity of ϕ under G :

lemma *imply-completeness*:

assumes *valid*: $\forall (M :: ('i :: \text{countable}, 'i \text{ fm set}) \text{ kripke}). \forall w \in \mathcal{W} M.$

$(\forall q \in G. M, w \models q) \longrightarrow M, w \models p$

shows $\exists qs. \text{set } qs \subseteq G \wedge (A \vdash \text{imply } qs \ p)$

In the following section we consider choices of A that impose structure on the canonical model, giving us completeness over various classes of frames.

We have formalized strong completeness results for every system, but for brevity we display only the weak counterparts here.

Table 1. Epistemic axioms

Axiom	Formula	Frame condition	Principle
T	$K_i\varphi \rightarrow \varphi$	Reflexive	True knowledge
B	$\varphi \rightarrow K_i L_i\varphi$	Symmetric	Knowledge of consistency of truths ^a
4	$K_i\varphi \rightarrow K_i K_i\varphi$	Transitive	Positive introspection
5	$\neg K_i\varphi \rightarrow K_i \neg K_i\varphi$	Euclidean ^b	Negative introspection

^aName suggested to us by Rineke Verbrugge.

^bNot formalized. Blackburn et al. [3] give a proof.

6 Concrete Systems

We now consider logics based on the axioms in Table 1 and show how we can compose axioms to easily obtain completeness over the restricted class of frames.

6.1 System K

System K is the smallest normal modal logic so we fix A to always return false:

abbreviation *SystemK* :: *'i fm* \Rightarrow *bool* (\vdash_K - [50] 50) **where**

$\vdash_K p \equiv (\lambda\cdot. \text{False}) \vdash p$

Soundness follows immediately from the generalized result:

lemma *soundness_K*: $\vdash_K p \Longrightarrow w \in \mathcal{W} M \Longrightarrow M, w \models p$

We abbreviate validity over a class of frames as *valid_X* where X is a system from table 2, which identifies the corresponding class. For instance *valid_{K4}* abbreviates validity over transitive frames. For K there are no restrictions:

abbreviation $valid_K p \equiv \forall (M :: (nat, nat\ fm\ set)\ kripke). \forall w \in \mathcal{W} M. M, w \models p$

We obtain immediately that system K is sound and complete on all frames.

theorem $main_K: valid_K p \longleftrightarrow \vdash_K p$

We assume validity over one type of worlds only (as given by the type declaration on M in the definition of $valid_K$). This a stronger result than assuming validity in all universes. Composing the soundness and completeness results shows that validity in this universe implies validity in any other:

corollary

assumes $valid_K p$ **and** $w \in \mathcal{W} M$

shows $M, w \models p$

In the corollary above there is no restriction on the type of the model M .

Table 2. Soundness and completeness results

System	Axioms	Class
K		All frames
T	T	Reflexive frames
KB	B	Symmetric frames
K4	4	Transitive frames
S4	T, 4	Reflexive and transitive frames
S5	T, B 4 or T, 5	Frames with equivalence relations

6.2 System T

We consider system T as an example of our general recipe for proving completeness of a normal modal logic. The **inductive** command provides an easy way to define the axiom schema (recall that i and p can be instantiated at will):

inductive $AxT :: 'i\ fm \Rightarrow bool$ **where**

$AxT (K\ i\ p \longrightarrow p)$

Following Table 2 we abbreviate the resulting system \vdash_T . We will do similarly from now on without mentioning it explicitly:

abbreviation $SystemT :: 'i\ fm \Rightarrow bool$ ($\vdash_T - [50]$ 50) **where**

$\vdash_T p \equiv AxT \vdash p$

If A admits axiom T then every A -MCS can *reach* itself, by virtue of the canonical accessibility relation:

lemma *AxT-reflexive*:

assumes $\forall p. AxT\ p \longrightarrow A\ p$ **and** *consistent* $A\ V$ **and** *maximal* $A\ V$
shows $V \in reach\ A\ i\ V$

Therefore, when A admits T the canonical model is reflexive:

lemma *mcs_T-reflexive*:

assumes $\forall p. AxT\ p \longrightarrow A\ p$
shows *reflexive* (*Kripke* ($mcss\ A$) pi (*reach* A))

Completeness follows directly. Assume validity of ϕ over reflexive models (in the AxT -MCS universe) and that ϕ has no derivation. Then $\{\neg\phi\}$ is AxT -consistent and satisfied by the abstract canonical model. By the result above, this model is reflexive. But this contradicts the validity of ϕ over reflexive models, proving ϕ must be derivable. As a result:

theorem *main_T*: $valid_T\ p \longleftrightarrow \vdash_T\ p$

6.3 Systems KB, K4, S4

The previous section provides a recipe for formalizing completeness of other normal modal logics: define the axioms, prove that the canonical frame belongs to a certain class, and reuse the generic canonical model lemma to obtain completeness over that class.

We formalize axioms B and 4 from Table 1 as we did with T . These impose symmetry and transitivity on the canonical frame, respectively. The corresponding logics, KB and K4, are sound and complete over those classes of frames:

theorem *main_{KB}*: $valid_{KB}\ p \longleftrightarrow \vdash_{KB}\ p$

theorem *main_{K4}*: $valid_{K4}\ p \longleftrightarrow \vdash_{K4}\ p$

We introduce an abbreviation to combine axiom predicates such that $A \oplus A'$ admits ϕ as an axiom if either A or A' does:

abbreviation *Or* :: $('a \Rightarrow bool) \Rightarrow ('a \Rightarrow bool) \Rightarrow 'a \Rightarrow bool$ (**infixl** \oplus 65) **where**
 $A \oplus A' \equiv \lambda x. A\ x \vee A'\ x$

We can then define system S4 as the combination of axioms T and 4:

abbreviation *SystemS4* :: $'i\ fm \Rightarrow bool$ (\vdash_{S4} - [50] 50) **where**
 $\vdash_{S4}\ p \equiv AxT \oplus Ax4 \vdash p$

Each axiom imposes a condition on the canonical frame and this composes in our setup to give us completeness over reflexive and transitive frames:

theorem *main_{S4}*: $valid_{S4}\ p \longleftrightarrow \vdash_{S4}\ p$

7 System S5

We formalize two versions of System S5. The first version is obtained by combining the axioms T, B and 4. In formalizing the systems T, KB and K4, we have already proved that these axioms force the canonical model to be reflexive, symmetric and transitive, respectively. Thus, we can easily show that their composition guarantees equivalence relations. The second version uses the more traditional combination of axioms T and 5. We show completeness by deriving the axioms of the first system in the second and vice versa for soundness.

7.1 Compositional Version

We combine the axioms and follow the recipe from before, reusing the canonicity results, leading us to the expected result:

abbreviation $SystemS5 :: 'i fm \Rightarrow bool (\vdash_{S5} - [50] 50)$ **where**

$\vdash_{S5} p \equiv AxT \oplus AxB \oplus Ax4 \vdash p$

theorem $main_{S5}: valid_{S5} p \longleftrightarrow \vdash_{S5} p$

7.2 Alternative Version

We now define a different version of S5 from scratch and call it S5':

inductive $SystemS5' :: 'i fm \Rightarrow bool (\vdash_{S5''} - [50] 50)$ **where**

$A1'$: $tautology\ p \Longrightarrow \vdash_{S5'} p$

| $A2'$: $\vdash_{S5'} (K\ i\ (p \longrightarrow q) \longrightarrow K\ i\ p \longrightarrow K\ i\ q)$

| AT' : $\vdash_{S5'} (K\ i\ p \longrightarrow p)$

| $A5'$: $\vdash_{S5'} (\neg K\ i\ p \longrightarrow K\ i\ (\neg K\ i\ p))$

| $R1'$: $\vdash_{S5'} p \Longrightarrow \vdash_{S5'} (p \longrightarrow q) \Longrightarrow \vdash_{S5'} q$

| $R2'$: $\vdash_{S5'} p \Longrightarrow \vdash_{S5'} K\ i\ p$

We include again all propositional tautologies ($A1'$) and the distribution axiom ($A2'$), but formulated more traditionally for variety. We directly include the axioms T (AT') and 5 ($A5'$). The two rules, modus ponens ($R1'$) and necessitation ($R2'$), are the same as before.

To show that this formulation is as powerful as the former, we provide derivations of the two remaining axioms B and 4. First B:

lemma $S5'-B: \vdash_{S5'} (p \longrightarrow K\ i\ (L\ i\ p))$

We omit the proof, which instantiates axiom 5 to $L_i p \rightarrow K_i L_i p$ and uses axiom T to derive $p \rightarrow L_i p$. By transitivity, a simple tautology, we arrive at B.

Instantiating axiom 5 to $L_i(\neg p) \rightarrow K_i L_i(\neg p)$ allows us to derive axiom 4:

lemma $S5'-4: \vdash_{S5'} (K\ i\ p \longrightarrow K\ i\ (K\ i\ p))$

We can then easily show that any formula derivable from axioms T, B, 4 are derivable with T and 5 as well:

lemma $S5-S5'$: $\vdash_{S5} p \implies \vdash_{S5'} p$

The other direction ($S5'-S5$) follows directly from the completeness result for that system. In combination, we obtain formalized soundness and completeness for the more traditional choice of axioms over the class of S5 frames:

theorem $main_{S5'}$: $valid_{S5} p \iff \vdash_{S5'} p$

8 Conclusion and Future Work

We have formalized the syntax and semantics of epistemic logic with countably many agents in the proof assistant Isabelle/HOL, following the textbook by Fagin et al. [7] and providing a precise, mechanically-checked elaboration of their completeness proof. Instead of considering just one proof system for epistemic logic we have formalized the family of normal modal logics and given an abstract account of the Henkin-style completeness method, which can be instantiated with any logic in the family. We have formalized the canonicity of various epistemic principles and used this, alongside the abstract completeness result, to prove completeness of systems K, T, KB, K4, S4 and two variants of S5 over their respective classes of frames. In the composite systems we have reused our results about the constituent axioms, proving Blackburn et al.'s [3] statement that we can ‘add our results together’ in such completeness-via-canonicity arguments.

In the initial version of this paper we did not include the set of worlds in the model explicitly, as we do now, but only implicitly as the inhabitants of the type variable w . This, however, required us to explicitly define a subtype of each kind of A -MCS, to build the canonical model over. While doable this quickly proved tedious. We are grateful to the anonymous reviewer who pointed us back towards the current solution. While we find it slightly less intuitive to have both a domain of worlds, namely the type, and a concrete set of considered worlds, it turned out simpler in the end.

In the future we want to extend the formalization with a broader range of results about epistemic and modal logics, including a formalization of Salhquist’s correspondence theorem, which would subsume the present results. We also want to extend the techniques explored here to our formalization of public announcement logic [11].

Acknowledgements. We thank Alexander Birch Jensen and Jørgen Villadsen for discussions and comments on a draft, Valentin Goranko for suggesting both related and future work and Bruno Bentzen for informing us of relevant formalizations. We thank the anonymous reviewers for valuable comments.

References



1. Bentzen, B.: A Henkin-style completeness proof for the modal logic S5. CoRR abs/1910.01697 (2019). <https://arxiv.org/abs/1910.01697>

2. Berghofer, S.: First-order logic according to Fitting. *Archive of Formal Proofs* (2007). <https://isa-afp.org/entries/FOL-Fitting.html>. Formal proof development
3. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001). <https://doi.org/10.1017/CBO9781107050884>
4. Blanchette, J.C., Popescu, A., Traytel, D.: Soundness and completeness proofs by coinductive methods. *J. Autom. Reason.* **58**(1), 149–179 (2017). <https://doi.org/10.1007/s10817-016-9391-3>
5. Conradie, W., Goranko, V., Vakarelov, D.: Algorithmic correspondence and completeness in modal logic. i. The core algorithm SQEMA. *Log. Methods Comput. Sci.* **2**(1) (2006). [https://doi.org/10.2168/LMCS-2\(1:5\)2006](https://doi.org/10.2168/LMCS-2(1:5)2006)
6. van Ditmarsch, H., van der Hoek, W., Kooi, B.: *Dynamic Epistemic Logic*. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-1-4020-5839-4>
7. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: *Reasoning About Knowledge*. MIT Press, Cambridge (1995). <https://doi.org/10.7551/mitpress/5803.001.0001>
8. From, A.H., Blackburn, P., Villadsen, J.: Formalizing a Seligman-style tableau system for hybrid logic. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) *IJCAR 2020*. LNCS (LNAI), vol. 12166, pp. 474–481. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51074-9_27
9. From, A.H.: Epistemic logic: Completeness of modal logics. *Archive of Formal Proofs* (2018). https://devel.isa-afp.org/entries/Epistemic_Logic.html. Formal proof development
10. From, A.H.: Formalizing Henkin-style completeness of an axiomatic system for propositional logic. In: Pavlova, A. (ed.) *Proceedings of the ESSLLI & WeSSLLI Student Session 2020* (2020)
11. From, A.H.: Public announcement logic. *Archive of Formal Proofs*, June 2021. https://isa-afp.org/entries/Public_Announcement_Logic.html. Formal proof development
12. From, A.H., Jensen, A.B., Villadsen, J.: Formalized soundness and completeness of epistemic logic (2021). <https://lamassr.github.io/papers/Formalized-Soundness.pdf>. Extended abstract. International Workshop on Logical Aspects in Multi-Agent Systems and Strategic Reasoning (LAMAS & SR)
13. Hagemeyer, C.: Formalizing intuitionistic epistemic logic in Coq. BSc thesis (2021). <https://www.ps.uni-saarland.de/~hagemeyer/bachelor.php>
14. Kądziołka, J.: Solution to the XKCD blue eyes puzzle. *Archive of Formal Proofs* (2021). https://isa-afp.org/entries/Blue_Eyes.html. Formal proof development
15. Li, J.: Formalization of pal-s5 in proof assistant. *CoRR abs/2012.09388* (2020). <https://arxiv.org/abs/2012.09388>
16. Maggesi, M., Brogi, C.P.: A formal proof of modal completeness for provability logic. *CoRR abs/2102.05945* (2021). <https://arxiv.org/abs/2102.05945>
17. Meyer, J.J.C., van der Hoek, W.: *Epistemic Logic for AI and Computer Science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (1995). <https://doi.org/10.1017/CBO9780511569852>
18. Neeley, P.: Results in modal and dynamic epistemic logic: a formalization in Lean (2021). <https://leanprover-community.github.io/lt2021/slides/paula-LeanTogether2021.pdf>. Slides
19. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL–A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002). <https://doi.org/10.1007/3-540-45949-9>

20. Wu, M., Goré, R.: Verified decision procedures for modal logics. In: Harrison, J., O’Leary, J., Tolmach, A. (eds.) 10th International Conference on Interactive Theorem Proving (ITP 2019). Leibniz International Proceedings in Informatics (LIPIcs), vol. 141, pp. 31:1–31:19. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2019). <https://doi.org/10.4230/LIPIcs.ITP.2019.31>, <http://drops.dagstuhl.de/opus/volltexte/2019/11086>
21. Xiong, Z., Ågotnes, T., Zhang, Y.: The logic of secrets. In: LAMAS 2020–10th Workshop on Logical Aspects of Multi-Agent Systems (2020)



A Logical Characterization of Constant-Depth Circuits over the Reals

Timon Barlag^(✉)  and Heribert Vollmer 

Leibniz University, Hanover, Germany
{barlag,vollmer}@thi.uni-hannover.de
<https://www.thi.uni-hannover.de>

Abstract. In this paper we give an Immerman Theorem for real-valued computation, i.e., we define circuits of unbounded fan-in operating over real numbers and show that families of such circuits of polynomial size and constant depth decide exactly those sets of vectors of reals that can be defined in first-order logic on \mathbb{R} -structures in the sense of Cucker and Meer.

Our characterization holds both non-uniformly as well as for many natural uniformity conditions.

Keywords: Computation over the reals · Descriptive complexity · Constant-depth circuit families

1 Introduction

Computational complexity theory is a branch of theoretical computer science which focuses on the study and classification of problems with regard to their innate difficulty. This is done by dividing these problems into classes, according to the amount of resources necessary to solve them using particular models of computation. One of the most prominent such models is the Turing machine – a machine operating sequentially on a fixed, finite vocabulary.

If one wishes to study problems based on their parallel complexity or in the domain of the real numbers, one requires different models of computation. Theoretical models exist both for real-valued sequential and for real-valued parallel computation, going back to the seminal work by Blum, Shub and Smale [4], see also [3]. Their aim was to lay the foundation for a theory of scientific computation, an area going back to Newton, Euler and Gauss, with algorithms over the real numbers. Going even a step further, John von Neumann aimed for a formal logic amenable to mathematical analysis and the continuous concept of the real number.

Unlike Turing machines, machines over \mathbb{R} obtain not an unstructured sequence of bits as input but a vector of real numbers or an (encoding of an) \mathbb{R} -structure. The respective parallel model we are going to have a closer look at is a real analogue to the arithmetic circuit (see, e.g., [15]), which, as its name

Supported by DFG VO 630/8-1.

suggests, resembles electrical circuits in its functioning, however, contrary to these our model operates not on electrical signals, i.e., Boolean values, but real numbers.

Descriptive complexity is an area of computational complexity theory, which groups decision problems into classes not by bounds on the resources needed for their solution but by considering the syntactic complexity of a logical formalism able to express the problems. Best known is probably Fagin’s characterization of the class NP as those problems that can be described by existential second-order formulas of predicate logic [9]. Since then, many complexity classes have been characterized logically. Most important in our context is a characterization obtained by Neil Immerman, equating problems decidable by (families of) Boolean circuits of polynomial size and constant depth consisting of gates of unbounded fan-in, with those describable in first-order logic:

Theorem 1 ([14]). $AC^0 = FO$.

An important issue in circuit complexity is *uniformity*, i.e., the question if a finite description of an infinite family of circuits exists, and if yes, how complicated it is to obtain it. Immerman’s Theorem holds both non-uniformly, i.e., under no requirements on the constructability of the circuit family, as well as for many reasonable uniformity conditions [2]. In the non-uniform case, first-order logic is extended by allowing access to arbitrary numerical predicates, in symbols: non-uniform $AC^0 = FO[Arb]$.

The rationale behind the descriptive approach to complexity is the hope to make tools from logic on expressive power of languages available to resource-based complexity and use non-expressibility results to obtain lower bounds on resources such as time, circuit size or depth, etc.

Descriptive complexity seems very pertinent for real-valued computation, since formulas operate directly on structured inputs, which seems quite natural, while computation models generally work on encodings of the input structure, which is an additional abstraction.

In the area of descriptive complexity over the reals, one usually considers *metafinite structures*, that is finite first-order structures enriched with a set of functions into another possibly infinite structure, in our case the real numbers \mathbb{R} . This study was initiated by Grädel and Meer [11], presenting logical characterizations of $P_{\mathbb{R}}$ and $NP_{\mathbb{R}}$. Continuing this line of research, Cucker and Meer showed a few logical characterizations for bounded fan-in real arithmetic circuit classes [7], which is what the present paper builds on. Cucker and Meer first proved a characterization of $P_{\mathbb{R}}$ using fixed-point logic, and building on this characterized the classes of the NC-hierarchy (bounded fan-in circuits of polynomial size and polylogarithmic depth) restricting the number of updating in the definition of fixed points to a polylogarithmic number. They leave out the case of the very low circuit class $AC_{\mathbb{R}}^0$, a subclass of $NC_{\mathbb{R}}^1$. We now expand on their research by making the framework of logics over metafinite structures amenable for the description of *unbounded fan-in circuits*; we are particularly concerned with a real analogue to the class $AC_{\mathbb{R}}^0$ and show that it corresponds to first-order logic over metafinite structures:

Theorem 2. $\text{AC}_{\mathbb{R}}^0 = \text{FO}_{\mathbb{R}}$.

Cucker and Meer only note that “the expressive power of first-order logic is not too big” [7] since it can only describe properties in $\text{NC}_{\mathbb{R}}^1$. In a sense we close the missing detail in their picture by determining a circuit class corresponding to first-order logic.

The logical characterization of Theorem 2 holds for arbitrary uniformity conditions based on time-bounded construction of the circuit family, in particular $\text{P}_{\mathbb{R}}$ -uniformity and $\text{LT}_{\mathbb{R}}$ -uniformity. Extending the framework of Cucker and Meer, we also characterize non-uniform $\text{AC}_{\mathbb{R}}^0$ by first-order logic enhanced with arbitrary numerical predicates.

This paper is structured as follows: In the next section, we introduce the reader to machines and circuits over \mathbb{R} and the complexity classes they define. We also introduce logics over metafinite structures and prove a couple of auxiliary results concerning useful extensions of $\text{FO}_{\mathbb{R}}$. Section 3 contains our main results, first turning to non-uniform circuits and then generalizing our results to different uniformity conditions. We close by mentioning some questions for further work.

The proofs of some of our theorems are quite lengthy and due to space restrictions we only give sketches of basic ideas. For the full proofs, see [1].

2 Preliminaries

In this section, we give an introduction to the machine models and logic over \mathbb{R} used in this paper – which are mostly taken from Cucker and Meer [7] – and some extensions thereof which we will make use of later on.

2.1 Machines over \mathbb{R}

Machines over \mathbb{R} , which were first introduced by Blum, Shub and Smale [4], operate on an unbounded tape of registers containing real numbers. They can evaluate real polynomials and divisions of real polynomials in a single step and branch out by checking if the number contained in a cell is nonnegative. A function f is said to be (\mathbb{R} -)computable if and only if there exists an \mathbb{R} -machine M , whose input-output-function is exactly f . We say that such a machine works in polynomial (logarithmic) time, if the number of steps it takes before halting when given an input $x \in \mathbb{R}^{\infty}$ is bounded by a polynomial (logarithmic) function in $|x|$. Here, \mathbb{R}^{∞} denotes arbitrarily long \mathbb{R} -vectors (i.e., $\mathbb{R}^{\infty} = \bigcup_{k \in \mathbb{N}_0} \mathbb{R}^k$) and $|x|$ denotes the length of x , i.e., if $x \in \mathbb{R}^k$ then $|x| = k$.

A more formal definition of these machines can be found can also be found in the paper by Cucker and Meer [7] or the arXiv version of this paper [1].

2.2 Arithmetic Circuits over \mathbb{R}

Arithmetic circuits over \mathbb{R} were first introduced by Cucker [6] and are our main model of computation. We will define them in analogy to how they were defined

by Cucker and Meer [7], however in this paper we consider unbounded fan-in. Also, we disallow division and subtraction gates, since it is not clear, how these operations would be defined for unbounded fan-in. Since it can be shown that for decision problems, losing (the bounded version of) those gate types does not change computational power within polynomial size, disallowing them does not relativize our results.

Definition 3. An *arithmetic circuit* C over \mathbb{R} is a directed acyclic graph. Its nodes (also called gates) can be of the following types:

- Input nodes* have indegree 0 and contain the respective input values of the circuit.
- Constant nodes* have indegree 0 and are labelled with real numbers.
- Arithmetic nodes* can have an arbitrary indegree only bounded by the number of nodes in the circuit. They can be labelled with either $+$ or \times
- Sign nodes* have indegree 1.
- Output nodes* have outdegree 1 and contain the output values of the circuit after the computation.

Nodes cannot be predecessors of the same node more than once, which leads to the outdegree of nodes in these arithmetic circuits being bounded by the number of gates in the circuit.

For convenience, we also define arithmetic nodes with the labels $-$, $=$, $<$, $>$, \leq and \geq , though they do not grant us additional computational power in this context.

Arithmetic nodes compute the respective function they are labelled with (with $=$, $<$, $>$, \leq and \geq representing their respective binary characteristic functions) and sign gates compute the sign function. On any input x , a circuit C computes a function f_C by evaluating all gates according to their labels. The values of the output gates at the end of the computation are the result of the computation of C , i.e., $f_C(x)$.

In order to talk about complexity classes of arithmetic circuits, one considers the *depth* and the *size* of the circuit. The depth of a circuit is the longest path from an input gate to an output gate and the size of a circuit is the number of gates in a circuit.

We say that a directed acyclic graph C_{sub} is a *subcircuit* of a circuit C , if and only if C_{sub} is weakly connected (i.e. replacing all directed edges with undirected ones in C_{sub} would produce a connected graph), all nodes and edges in C_{sub} are also contained in C and it holds that if there is a path from an input gate to a gate g in C , then this path also exists in C_{sub} . For any node g in C , we denote by the subcircuit *induced* by g that subcircuit $C_{sub,g}$ of C , of which g is the top node. We then also say that g is the *root node* of $C_{sub,g}$.

A single circuit can only compute a function with a fixed number of arguments, which is why we call arithmetic circuits a *non-uniform* model of computation. In order to talk about arbitrary functions, we need to consider circuit families, i.e., sequences of circuits which contain one circuit for every input length

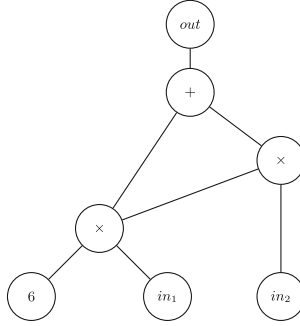


Fig. 1. This circuit has size 7, depth 4 and computes the binary function $f(x, y) = (6 \cdot x) + (6 \cdot x) \cdot y$.

$n \in \mathbb{N}$. The function computed by a circuit family $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$ is the function computed by the respective circuit, i.e.,

$$f_{\mathcal{C}}(x) = f_{C_{|x|}}(x). \tag{1}$$

A circuit family is said to decide a set if and only if it computes the characteristic function of the set. For a function $f: \mathbb{N} \rightarrow \mathbb{N}$, we say that a circuit family \mathcal{C} is of size f (depth f), if the size (depth) of C_n is bounded by $f(n)$.

Definition 4. The class $\text{AC}_{\mathbb{R}}^0$ is the class of sets decidable by arithmetic circuit families over \mathbb{R} of polynomial size and constant depth.

The circuit families we have just introduced do not have any restrictions on the difficulty of obtaining any individual circuit. For this reason, we also consider so-called *uniform* circuit families.

Definition 5. We say that a circuit family $\mathcal{C} = (C_n)_{n \in \mathbb{N}}$ is uniform if for each of its circuit the gates are numbered, the predecessors of each gate are ordered and for any given triple of numbers (n, v_{nr}, p_{idx}) , a corresponding triple (t, p_{nr}, c) can be computed by an \mathbb{R} -machine M , where

- i) t is the type of the v_{nr} -th gate v in C_n ,
- ii) p_{nr} is the number of the p_{idx} -th predecessor of v and
- iii) c is the value of v if v is a constant gate, the index i if v is the i -th input gate and 0 otherwise.

If v has less than p_{idx} predecessors, M returns $(t, 0, 0)$ and if v_{nr} does not encode a gate in C_n , M returns $(0, 0, 0)$.

If this computation only takes logarithmic time in n , we call \mathcal{C} $\text{LT}_{\mathbb{R}}$ -uniform. If it takes polynomial time in n , we call \mathcal{C} $\text{P}_{\mathbb{R}}$ -uniform.

For a circuit complexity class \mathfrak{C} , we will by $\text{U}_{\text{LT}_{\mathbb{R}}}\text{-}\mathfrak{C}$ denote the subclass of \mathfrak{C} , which only contains sets definable by $\text{LT}_{\mathbb{R}}$ -uniform circuit families. We will use $\text{U}_{\text{P}_{\mathbb{R}}}\text{-}\mathfrak{C}$ to analogously denote those sets in \mathfrak{C} definable by $\text{P}_{\mathbb{R}}$ -uniform families.

2.3 \mathbb{R} -Structures and First-Order Logic over \mathbb{R}

The logics we use to characterize real circuit complexity classes are based on first-order logic with arithmetics.

Definition 6 ([7, Definition 7]). Let L_s, L_f be finite vocabularies where L_s can contain function and predicate symbols and L_f only contains function symbols. An \mathbb{R} -structure of signature $\sigma = (L_s, L_f)$ is a pair $\mathcal{D} = (\mathcal{A}, \mathcal{F})$ where

1. \mathcal{A} is a finite structure of vocabulary L_s which we call the *skeleton* of \mathcal{D} whose universe A we will refer to as the *universe* of \mathcal{D} and whose cardinality we will refer to by $|A|$
2. and \mathcal{F} is a finite set which contains functions of the form $X: A^k \rightarrow \mathbb{R}$ for $k \in \mathbb{N}$ which interpret the function symbols in L_f .

We will use $Struct_{\mathbb{R}}(\sigma)$ to refer to the set of all \mathbb{R} -structures of signature σ and we will assume that for any fixed signature $\sigma = (L_s, L_f)$, we can fix an ordering on the symbols in L_s and L_f .

In order to use \mathbb{R} -structures as inputs for machines, we encode them in \mathbb{R}^∞ as follows: We start by choosing an arbitrary ranking r on A , i.e., a bijection $r: A \rightarrow \{0, \dots, |A| - 1\}$. We then replace all predicates in L_s by their respective characteristic functions and all functions $f \in L_s$ by $r \circ f$. Those functions are then considered to be elements of L_f . We represent each of these functions by concatenating their function values in lexicographical ordering on the respective function arguments according to r . To encode \mathcal{D} we only need to concatenate all representations of functions in L_f in the order fixed on the signature. We denote this encoding by $\text{enc}(\mathcal{D})$.

In order to be able to compute $|A|$ from $\text{enc}(\mathcal{D})$, we make an exception for functions and predicates of arity 0. We treat those as if they had arity 1, meaning that e.g. we encode a function $f_1() = 3$ as $|A|$ many 3s.

Since

$$|\text{enc}(\mathcal{D})| = \sum_{f \in L_f} |A|^{\max\{ar(f), 1\}}, \tag{2}$$

where $ar(f)$ is the arity of f , we can reconstruct $|A|$ from the arities of the functions in L_f and the length of $\text{enc}(\mathcal{D})$. We can do so by using for example binary search, since we know that $|A|$ is between 0 and $|\text{enc}(\mathcal{D})|$. We can therefore compute $|A|$ when given φ and $|\text{enc}(\mathcal{D})|$ in time logarithmic in $|\text{enc}(\mathcal{D})|$.

First-Order Logic over \mathbb{R}

Definition 7 (First-order logic). The language of first-order logic contains for each signature $\sigma = (L_s, L_f)$ a set of formulas and terms. The terms are divided into *index terms* which take values in universe of the skeleton and *number terms* which take values in \mathbb{R} . These terms are inductively defined as follows:

1. The set of index terms is defined as the closure of the set of variables V under applications of the function symbols of L_s .

2. Any real number is a number term.
3. For index terms h_1, \dots, h_k and a k -ary function symbol $X \in L_f$, $X(h_1, \dots, h_k)$ is a number term.
4. If t_1, t_2 are number terms, then so are $t_1 + t_2$, $t_1 \times t_2$ and $sign(t_1)$.

Atomic formulas are equalities of index terms $h_1 = h_2$ and number terms $t_1 = t_2$, inequalities of number terms $t_1 < t_2$ and expressions of the form $P(h_1, \dots, h_k)$, where $P \in L_s$ is a k -ary predicate symbol and h_1, \dots, h_k are index terms.

The set $\text{FO}_{\mathbb{R}}$ is the smallest set which contains the closure of atomic formulas under the Boolean connectives $\{\wedge, \vee, \neg, \implies, \iff\}$ and quantification $\exists v\psi$ and $\forall v\psi$ where v ranges over \mathcal{A} .

Equivalence of $\text{FO}_{\mathbb{R}}$ -formulas and sets defined by $\text{FO}_{\mathbb{R}}$ -formulas are done in the usual way, i.e., a formula φ defines a set S if and only if the elements of S are exactly the encodings of \mathbb{R} -structures under which φ holds and two such formulas are said to be equivalent if and only if they define the same set.

Extensions to $\text{FO}_{\mathbb{R}}$

In the following, we would like to extend $\text{FO}_{\mathbb{R}}$ by additional functions and relations that are not given in the input structure. To that end, we make a small addition to Definition 6 where we defined \mathbb{R} -structures. Whenever we talk about \mathbb{R} -structures over a signature (L_s, L_f) , we now also consider structures over signatures of the form (L_s, L_f, L_a) . The additional vocabulary L_a does not have any effect on the \mathbb{R} -structure, but it contains function and relation symbols, which can be used in a logical formula with this signature. This means that any \mathbb{R} -structure of signature (L_s, L_f) is also an \mathbb{R} -structure of signature (L_s, L_f, L_a) for any vocabulary L_a .

Definition 8. Let R be a set of finite relations and functions. We will write $\text{FO}_{\mathbb{R}}[R]$ to denote the class of sets that can be defined by $\text{FO}_{\mathbb{R}}$ -sentences which can make use of the functions and relations in R in addition to what they are given in their structure. Formally, this means that $\text{FO}_{\mathbb{R}}[R]$ describes exactly those sets $S \subseteq \mathbb{R}^{\infty}$ for which there exists an $\text{FO}_{\mathbb{R}}$ -sentence φ over a signature $\sigma = (L_s, L_f, L_a)$ such that for each length n , there is an interpretation I_n interpreting the symbols in L_a as elements of R such that for all \mathbb{R}^{∞} -tuples s of length n it holds that $s \in S$ if and only if s encodes an \mathbb{R} -structure over (L_s, L_f, L_a) which models φ when using I_n .

With the goal in mind to create a logic which can define sets decided by circuits with unbounded fan-in, we introduce new rules for building number terms: the *sum* and the *product rule*. We will also give another rule, which we call the *maximization rule*, but will later show that we can define this rule in $\text{FO}_{\mathbb{R}}$ and thus do not gain expressive power by using it. We will use this rule to show that we can represent characteristic functions in $\text{FO}_{\mathbb{R}}$.

Definition 9 (sum, product and maximization rule). Let t be a number term in which the variable i occurs freely with other variables $\bar{w} = w_1, \dots, w_j$ and let A denote the universe of the given input structure. Then

$$sum_i(t(i, \bar{w})) \tag{3}$$

is also a number term which is interpreted as $\sum_{i \in A} t(i, \bar{w})$. The number terms $prod_i(t(i, \bar{w}))$ and $max_i(t(i, \bar{w}))$ are defined analogously.

We also write $sum_i^q(t(i_1, \dots, i_q, \bar{w}))$ to denote $sum_{i_1}(\dots sum_{i_q}(t(i_1, \dots, i_q, \bar{w})))$ for convenience and we will use $prod_i^q$ analogously.

For a logic \mathcal{L} , we will by $\mathcal{L} + \text{SUM}_{\mathbb{R}}$, $\mathcal{L} + \text{PROD}_{\mathbb{R}}$ and $\mathcal{L} + \text{MAX}_{\mathbb{R}}$ denote \mathcal{L} extended by the sum rule, the product rule or the maximization rule respectively.

We will now evaluate which logics can already natively use some of the aforementioned rules. As it turns out, the maximization rule can be used in $\text{FO}_{\mathbb{R}}$ without any extensions and the sum and product rule extend neither $\text{FO}_{\mathbb{R}}[\text{Arb}]$ nor a polynomial extension of $\text{FO}_{\mathbb{R}}$ which we will see later.

Lemma 10. $\text{FO}_{\mathbb{R}} = \text{FO}_{\mathbb{R}} + \text{MAX}_{\mathbb{R}}$.

Proof. For each $\text{FO}_{\mathbb{R}} + \text{MAX}_{\mathbb{R}}$ formula, we can construct an equivalent $\text{FO}_{\mathbb{R}}$ -formula. For each such term containing $max_i(F(i))$, the basic idea is to add a quantifier prefix which makes sure that there exists an element $x \in A$ such that for all elements $y \in A$, $F(x) \geq F(y)$.

Definition 11. Let Arb denote the set of all finitary relations over \mathbb{R}^{∞} and all functions $f: \mathbb{R}^k \rightarrow \mathbb{R}$ for $k \in \mathbb{N}$.

Lemma 12. $\text{FO}_{\mathbb{R}}[\text{Arb}] = \text{FO}_{\mathbb{R}}[\text{Arb}] + \text{SUM}_{\mathbb{R}} = \text{FO}_{\mathbb{R}}[\text{Arb}] + \text{PROD}_{\mathbb{R}}$.

Proof. Let φ be an $\text{FO}_{\mathbb{R}}[\text{Arb}]$ -sentence which contains sum_i -constructions. For each sum_i -occurrence $sum_i(t(i, \bar{w}))$, starting at the deepest level of nesting, create a new function symbol $sum_{(t,i)}$ of arity $|\bar{w}|$ which for any input structure $\mathcal{D} = (\mathcal{A}, \mathcal{F})$ is interpreted as

$$sum_{(t,i)}(\bar{w}) = \sum_{i \in A} t(i, \bar{w}). \tag{4}$$

Now replace each sum_i -occurrence in φ by the respective function symbol. The result is an $\text{FO}_{\mathbb{R}}[\text{Arb}]$ -formula which is equivalent to φ but does not contain any sum_i -constructions. $\text{FO}_{\mathbb{R}}[\text{Arb}] = \text{FO}_{\mathbb{R}}[\text{Arb}] + \text{PROD}_{\mathbb{R}}$ can be shown analogously.

Remark 13. For the sake of simplicity we only consider *functional* \mathbb{R} -structures in the following, i.e., \mathbb{R} -structures whose signatures do not contain any predicate symbols. This does not restrict what we can express, since any relation $P \in A^k$ can be replaced by its characteristic function $\chi_P: A^k \rightarrow \{0, 1\}$.

As mentioned before, the reason why we need the maximization rule is that we would like to write characteristic functions as number terms. This will become

useful when we characterize our circuit models logically. For a first-order formula $\varphi(v_1, \dots, v_r)$ we define its characteristic function $\chi[\varphi]$ on a structure \mathcal{D} by

$$\chi[\varphi](a_1, \dots, a_r) = \begin{cases} 1 & \text{if } \mathcal{D} \models \varphi(a_1, \dots, a_r) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The following result is a slight modification of a result presented by Cucker and Meer [7].

Proposition 14. *Let R be a set of functions and predicates. For every $\text{FO}_{\mathbb{R}}[R]$ -formula φ , there is an $\text{FO}_{\mathbb{R}}[R]$ number term $t_{\chi[\varphi]}$ such that for all structures \mathcal{D} it holds that $t_{\chi[\varphi]}$ evaluates to 1 under \mathcal{D} if $\mathcal{D} \models \varphi$ and to 0, otherwise.*

We will write $\chi[\varphi]$ to denote the use of $t_{\chi[\varphi]}$ when writing number terms.

3 Characterizing $\text{AC}_{\mathbb{R}}^0$

In this section, we give descriptive complexity results for the non-uniform set $\text{AC}_{\mathbb{R}}^0$ and some of its uniform subsets. In order to achieve this, we use the previously defined first-order logic over the real numbers and the extensions we defined.

3.1 A Characterization for Non-uniform $\text{AC}_{\mathbb{R}}^0$

First of all we show an equality which is close to a classical result shown by Immermann [13]. We show that extending our first-order logic over the reals with arbitrary functions lets us exactly describe the non-uniform set $\text{AC}_{\mathbb{R}}^0$.

In the proof for the upcoming theorem, we make use of a convenient property of circuits deciding $\text{AC}_{\mathbb{R}}^0$ -sets, namely that for each of those circuits, there exist *tree-like* circuits deciding the same set. We call a circuit tree-like, if it is a directed tree with the exception of the input nodes. Those nodes, which would represent the leaves, can have multiple successor nodes. That means that tree-like circuits are trees up until the penultimate level and would be actual trees, if one would copy every input gate for each outgoing edge, rather than letting them have multiple successors.

Lemma 15. For every $\text{AC}_{\mathbb{R}}^0$ -circuit family $(C_n)_{n \in \mathbb{N}}$, there exists a tree-like $\text{AC}_{\mathbb{R}}^0$ -circuit family $(C'_n)_{n \in \mathbb{N}}$ computing the same function, such that for all $n \in \mathbb{N}$ and every gate v in C'_n , every path from an input gate to v has the same length.

Proof. In order to prove this we show that any $\text{AC}_{\mathbb{R}}^0$ -family can be transformed into an $\text{AC}_{\mathbb{R}}^0$ -family which exhibits the specified property. For any given circuit of a $\text{AC}_{\mathbb{R}}^0$ -family, we first make sure that all non-input gates have outdegree 1. In order to achieve this, for each gate g with outdegree $k > 1$ we copy the subcircuit $C_{sub,g}$ induced by g $k - 1$ times, so that we now have k copies of $C_{sub,g}$. For each of the previously outgoing edges $g \rightarrow v$ of g , the root of one of the copies of $C_{sub,g}$ then has v as its (sole) successor.

We do this iteratively, in each step only modifying gates with outdegree ≥ 2 that are closest to input gates. Afterwards, we pad all paths from input gates to the output gate with addition gates to ensure that they have the same length. This can be done with only a polynomial overhead in size and a constant overhead in depth without changing the computed function. Figure 2 show an example of this construction. \square

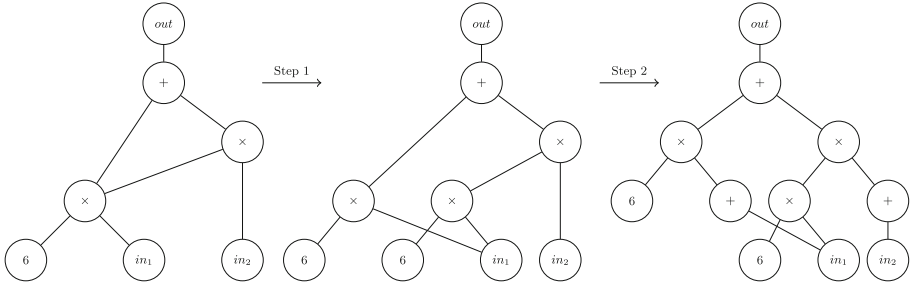


Fig. 2. An example of turning the circuit from Fig. 1 into a tree-like circuit as described in Lemma 15

For the upcoming proof we need some additional notation: For every $\text{FO}_{\mathbb{R}}$ formula φ and every variable x let $\varphi[a/x]$ denote φ where each occurrence of x is replaced by a . We write $\varphi[a_1/x_1, \dots, a_n/x_n]$ to denote several such replacements.

Theorem 16. $\text{FO}_{\mathbb{R}}[\text{Arb}] = \text{AC}_{\mathbb{R}}^0$.

Proof. The proof for this equality follows a similar pattern as the proof for the respective discrete result as presented in [15].

The main idea is to show that for any given $\text{FO}_{\mathbb{R}}$ sentence φ , a circuit family can be constructed which accepts its input if and only if the input encodes an \mathbb{R} -structure that satisfies ϕ . This is achieved by using addition and multiplication gates to mimic the functionality of existential and universal quantifiers and Boolean connectives and using the available gate types to represent the different kinds of number and index terms that can appear in $\text{FO}_{\mathbb{R}}$ formulae. This is a similar basic idea as in the proof in [15], however, the technical execution of that idea is quite different thanks to the fact that we are dealing with arithmetic circuits and a logic which deals with Boolean and arithmetic terms of a dyadic structure.

Let φ be an $\text{FO}_{\mathbb{R}}[\text{Arb}]$ sentence of signature σ . For any subformula ψ of φ with exactly k free variables x_1, \dots, x_k , and any vector $(m_1, \dots, m_k) \in A^k$ we can construct an arithmetic circuit $C_n^{\psi(m_1, \dots, m_k)}$ with the following property: For any input structure \mathcal{D} such that $|\text{enc}(\mathcal{D})| = n$ it holds that $\mathcal{D} \models \psi[m_1/x_1, \dots, m_k/x_k]$ if and only if $\text{enc}(\mathcal{D})$ is accepted by $C_n^{\psi(m_1, \dots, m_k)}$.

If for example $\varphi = \exists \psi$, then C_n^{φ} has a sign gate at the top, followed by a multiplication gate, which in turn has as its predecessors the circuits $C_n^{\psi(i)}$ for

$1 \leq i \leq n$. The final circuit for the sentence φ is defined by structural induction on φ . The construction is relatively technical and can be found in [1].

The idea for the converse inclusion is to construct a formula for a given circuit family \mathcal{C} that is satisfied by exactly those structures whose encodings are evaluated to 1 by the circuits of \mathcal{C} . This is accomplished by defining number terms which encode the structure of the given circuit.

This idea is again very similar to the proof in [15], nevertheless, again the differences lie in the technical details. While the structures used in [15] are word structures, the functional structures used here require interpreting the circuit inputs as an encoded \mathbb{R} -structure which contains a single unary function that maps an index i to the value of the i th input gate of the circuit. These real values then need to be accumulated and “carried” through the circuit by defining a number term for each level of the circuit, which maps each gate on that level to its value during the computation.

We want to show that for any given set $S \subseteq \mathbb{R}^\infty$ defined by an $\text{AC}_{\mathbb{R}}^0$ family \mathcal{C} , we can construct an $\text{FO}_{\mathbb{R}}[\text{Arb}]$ sentence φ defining S . Since we have access to arbitrary functions, we can essentially just encode the structure of any given circuit into functions and have the interpretation of the function symbols we use be dependent on the length of the input n . However, the function symbols themselves, and thus the formula, do not depend on n . The functions we use will give us information about the type of any gate, its value if it is a constant gate, its index if it is an input gate and about its predecessor gates. Given that we are dealing with $\text{AC}_{\mathbb{R}}^0$ circuits, we can by Lemma 15 assume that they are tree-like and that each path from an input to the output gate has the same length. Therefore we can construct a sentence which essentially describes the gates of each level of the circuit.

We thus define φ inductively by defining a number term val_x for every $x \leq \text{depth}(\mathcal{C})$, which for any given gate on level x holds the value of that gate in the computation of C_n on the given input structure. φ then states that if the given gate g has the type 6 – meaning it is the output gate – then $val_{\text{depth}(\mathcal{C})}(g) = 1$. Again, for details see [1]. \square

3.2 Characterizations for uniform $\text{AC}_{\mathbb{R}}^0$

Having now developed a description for non-uniform $\text{AC}_{\mathbb{R}}^0$, in the upcoming part of this paper we derive descriptions for two of its uniform variations and a generalization. We start by giving a description for the logarithmic time uniform $\text{U}_{\text{LT}_{\mathbb{R}}}\text{-AC}_{\mathbb{R}}^0$.

For this reason, we introduce another notation here:

Definition 17. By $\text{FTIME}_{\mathbb{R}}(f(n))$ we will denote all functions that for a finite set S and $k \in \mathbb{N}$ map from S^k to \mathbb{R} or to S and that are computable by an \mathbb{R} -machine in time bounded by $\mathcal{O}(f(|S|))$.

Theorem 18. $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(\log n)] + \text{SUM}_{\mathbb{R}} + \text{PROD}_{\mathbb{R}} = \text{U}_{\text{LT}_{\mathbb{R}}}\text{-AC}_{\mathbb{R}}^0$.

Proof. Showing the equality of $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(\log n)]$ and $\text{U}_{\text{LT}_{\mathbb{R}}}\text{-AC}_{\mathbb{R}}^0$ works very similarly to the proof of Theorem 16. The respective circuit family \mathcal{C} for showing $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(\log n)] \subseteq \text{U}_{\text{LT}_{\mathbb{R}}}\text{-AC}_{\mathbb{R}}^0$ is defined in the same way. To show logtime uniformity, we need to additionally number the gates of the circuits of \mathcal{C} , and provide an \mathbb{R} -machine running in logarithmic time which produces the gates of C_n for all $n \in \mathbb{N}$. To achieve this, we number the gates of C_n gates in a post-order fashion. If we now want to produce the information for any given gate g , we can simply construct every gate on the path from the output gate to g , since our numbering tells us for each gate on our way, at which predecessor to continue. We can assume that this path is unique, since by Lemma 15, we can assume that C_n is tree-like. As C_n has constant depth, we only need to construct a constant number of gates in this way and each gate can be constructed in at most logarithmic time. In fact, we only need logarithmic time, if the gate is the evaluation of a function of $\text{FTIME}_{\mathbb{R}}(\log n)$. In all other cases, gate information can be procured in constant time. An example for this construction can be seen in Fig. 3.

The reverse containment again works very similarly as it did in the proof of Theorem 16. We again define val_x in the same way, and all auxiliary number and index terms we need can be defined in $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(\log n)] + \text{SUM}_{\mathbb{R}} + \text{PROD}_{\mathbb{R}}$, since \mathcal{C} is $\text{LT}_{\mathbb{R}}$ -uniform. \square

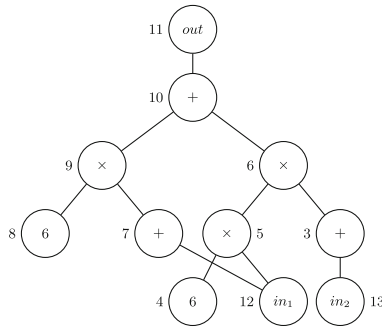


Fig. 3. The circuit from Fig. 1 has been transformed as shown in Fig. 2 and been numbered as in (a simplified version of) the numbering for Theorem 18. If we for example wanted to construct the addition gate numbered 7, we would start at 11, construct the gate 10 and we would then know to keep going at gate 9, since 7 is less than 9 but larger than 6. Note, that the input gates are exceptions in this numbering, since they do not behave tree-like. Their numbering starts just above the size of the circuit, so if a machine producing the gates of the circuit gets a number $11 < k \leq 13$ as an input, it can immediately return $(1, 0, k)$ (as per Definition 5).

Our second uniformity result is a classification for $\text{P}_{\mathbb{R}}$ -uniform $\text{AC}_{\mathbb{R}}^0$. This result basically follows in the same way as the previous one did. It is of note, however, that the sum and product rule do not actually extend

$\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(n^{\mathcal{O}(1)})]$, since the respective sums and products can already be computed in polynomial time. In analogy to Lemma 12 we get the following.

Lemma 19. $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(n^{\mathcal{O}(1)})] = \text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(n^{\mathcal{O}(1)})] + \text{SUM}_{\mathbb{R}} + \text{PROD}_{\mathbb{R}}$.

Corollary 20. $\text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(n^{\mathcal{O}(1)})] = \text{U}_{\text{P}_{\mathbb{R}}} - \text{AC}_{\mathbb{R}}^0$.

With the construction shown in Theorem 18 we can now generalize that, whenever we have a variant of $\text{AC}_{\mathbb{R}}^0$ given by a time complexity uniformity criterion that is at least logarithmic, we can describe it using first-order logic extended with functions of that class' time complexity and the sum and product rule. This result is formalized as follows:

Corollary 21. *For any function $f: \mathbb{N} \rightarrow \mathbb{N}$ with $f(n) \geq \log n$ for all n , it holds that*

$$\text{U}_f - \text{AC}_{\mathbb{R}}^0 = \text{FO}_{\mathbb{R}}[\text{FTIME}_{\mathbb{R}}(f(n))] + \text{SUM}_{\mathbb{R}} + \text{PROD}_{\mathbb{R}}, \quad (6)$$

where $\text{U}_f - \text{AC}_{\mathbb{R}}^0$ is the class of sets decidable by circuit families, which can be constructed as described in Definition 5 in time bounded by $\mathcal{O}(f(n))$.

Remark 22. The logarithmic bound for f in Corollary 21 stems from the time it takes to decode an encoded \mathbb{R} -structure. For details, see [1].

Remark 23. Even though we have only considered functional \mathbb{R} -structures in this paper, our findings can be generalized for \mathbb{R} -structures which use relations as well, since any relation can be expressed via its characteristic function.

4 Conclusion

We showed that the computational power of circuits of polynomial size and constant depth over the reals can be characterized in a logical way by first-order logic on metafinite structures. This result is in analogy to corresponding characterizations for Boolean circuits [14] and arithmetic circuits [12]. The results presented in this paper mostly do not make use of any special properties of the real numbers and can be generalized for other fields with suitably adapted logic and circuit definitions.

In the Boolean and arithmetic context, it is known [2] that the numerical predicates of addition and multiplication play a special role: If we enhance first-order logic by these, we obtain a logic as powerful as dlogtime-uniform AC^0 -circuits, i.e., $\text{U}_{\text{D}} - \text{AC}^0 = \text{FO}[+, \times]$ (see also [15]). This does not seem to hold in our case of computation over the real numbers: $\text{U}_{\text{LT}_{\mathbb{R}}} - \text{AC}_{\mathbb{R}}^0$ looks more powerful than $\text{FO}_{\mathbb{R}}[+, \times]$, since real numbers can be manipulated more generally by \mathbb{R} -machines operating in logarithmic time than in first-order formulas, because it seems that a logarithmic number of operations on reals cannot be simulated in first-order logic. Maybe an equivalence can be obtained with a more powerful logic for real numbers, but this is a question for further research. However, an analogue to the Boolean equality seems to hold if we consider uniformity defined

itself in a logical way: The identity $U_{\text{FO-AC}}^0 = \text{FO}[+, \times]$, well known in the Boolean world, seems to hold in the real-valued setting as well. We will turn to this issue in the final version of this paper.

While investigating uniform circuit classes over the reals, we found that uniformity behaves somewhat differently in the real setting than it does in the Boolean one. In the classical setting, the question of uniformity arises quite naturally, since small classes like non-uniform AC^0 already contain undecidable problems with respect to Turing machines. In the case of $\text{AC}_{\mathbb{R}}^0$ and \mathbb{R} -machines, the same is at least not quite obvious and worth looking into further.

We consider it worthwhile to study logical characterizations of analogues of further circuit classes of unbounded or semi-unbounded fan-in, most prominently $\text{SAC}_{\mathbb{R}}^1$ and $\text{AC}_{\mathbb{R}}^1$. In the theory of arithmetic complexity, i.e., computation over arbitrary semi-rings, first an analogue of Immerman's Theorem was shown in [12], and this was later used to obtain logical characterizations of the larger arithmetic classes $\#\text{NC}^1$, $\#\text{SAC}^1$ and $\#\text{AC}^1$ [8]. Remarkably these characterizations did not build on logics with repeated quantifier blocks (like in [14]) or restricted fixed-point logic (like in [7]). Instead, new logical characterizations of the Boolean classes NC^1 , SAC^1 and AC^1 were given, somewhat similar to earlier ideas from Compton and Laflamme [5], and these were then shifted to the arithmetic setting. Maybe this can also be useful in our context to develop characterizations for $\text{SAC}_{\mathbb{R}}^1$ and $\text{AC}_{\mathbb{R}}^1$ (and maybe obtain a new characterization of $\text{AC}_{\mathbb{R}}^1$).

In the theory of computation over the reals, separations among classes are known which are widely open in the discrete world; we only mention the separation of $\text{NC}_{\mathbb{R}}$ and $\text{P}_{\mathbb{R}}$ [6]. In the circuit world, the most prominent open question is if $\text{TC}^0 = \text{NC}^1$ (see the discussion in [15]). In our context, it is intriguing to study the landscape between $\text{AC}_{\mathbb{R}}^0$ and $\text{NC}_{\mathbb{R}}^1$. Is there any meaningful way to add computational power to $\text{AC}_{\mathbb{R}}^0$ without already arriving at the full power of $\text{NC}_{\mathbb{R}}^1$? Observe that up to date, no reasonable real analogue of the class TC^0 is known. In Boolean complexity, TC^0 is obtained by enriching AC^0 -circuits with majority gates. Here, the class $\text{AC}_{\mathbb{R}}^0$ is closed under all reasonable forms of majority and threshold operations. A first step forward will be to separate $\text{AC}_{\mathbb{R}}^0$ and $\text{NC}_{\mathbb{R}}^1$, a real world analogue of a classical circuit separation from the eighties [10].

References

1. Barlag, T., Vollmer, H.: A logical characterization of constant-depth circuits over the reals. CoRR (2020). <https://arxiv.org/abs/2005.04916>
2. Barrington, D.A.M., Immerman, N., Straubing, H.: On uniformity within NC^1 . J. Comput. Syst. Sci. **41**(3), 274–306 (1990). [https://doi.org/10.1016/0022-0000\(90\)90022-D](https://doi.org/10.1016/0022-0000(90)90022-D)
3. Blum, L.: Complexity and Real Computation. Springer, Heidelberg (1998). <https://www.worldcat.org/oclc/37004484>
4. Blum, L., Shub, M., Smale, S.: On a theory of computation over the real numbers; NP completeness, recursive functions and universal machines (extended abstract). In: 29th Annual Symposium on Foundations of Computer Science, pp. 387–397. IEEE Computer Society (1988). <https://doi.org/10.1109/SFCS.1988.21955>

5. Compton, K.J., Laflamme, C.: An algebra and a logic for NC^1 . *Inf. Comput.* **87**(1/2), 240–262 (1990). [https://doi.org/10.1016/0890-5401\(90\)90063-N](https://doi.org/10.1016/0890-5401(90)90063-N)
6. Cucker, F.: $P_{\mathbb{R}} \neq NC_{\mathbb{R}}$. *J. Complexity* **8**(3), 230–238 (1992). [https://doi.org/10.1016/0885-064X\(92\)90024-6](https://doi.org/10.1016/0885-064X(92)90024-6)
7. Cucker, F., Meer, K.: Logics which capture complexity classes over the reals. *J. Symb. Log.* **64**(1), 363–390 (1999). <https://doi.org/10.2307/2586770>
8. Durand, A., Haak, A., Vollmer, H.: Model-theoretic characterization of Boolean and arithmetic circuit classes of small depth. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 354–363. ACM (2018). <https://doi.org/10.1145/3209108.3209179>
9. Fagin, R.: Generalized first-order spectra and polynomial time recognizable sets. In: Karp, R. (ed.) *Complexity of Computations*, SIAM-AMS Proceedings, vol. 7, pp. 43–73. American Mathematical Society, Providence (1974)
10. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory* **17**(1), 13–27 (1984). <https://doi.org/10.1007/BF01744431>
11. Grädel, E., Meer, K.: Descriptive complexity theory over the real numbers. In: *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pp. 315–324 (1995). <https://doi.org/10.1145/225058.225151>
12. Haak, A., Vollmer, H.: A model-theoretic characterization of constant-depth arithmetic circuits. *Ann. Pure Appl. Log.* **170**(9), 1008–1029 (2019). <https://doi.org/10.1016/j.apal.2019.04.006>
13. Immerman, N.: Languages that capture complexity classes. *SIAM J. Comput.* **16**, 760–778 (1987). <https://doi.org/10.1137/0216051>
14. Immerman, N.: Expressibility and parallel complexity. *SIAM J. Comput.* **18**(3), 625–638 (1989). <https://doi.org/10.1137/0218043>
15. Vollmer, H.: *Introduction to Circuit Complexity - A Uniform Approach*. Springer, Heidelberg (1999). <https://doi.org/10.1007/978-3-662-03927-4>



Wanted Dead or Alive: Epistemic Logic for Impure Simplicial Complexes

Hans van Ditmarsch^(✉)

Open University of the Netherlands, Heerlen, The Netherlands
hans.vanditmarsch@ou.nl

Abstract. We propose a logic of knowledge for impure simplicial complexes. Impure simplicial complexes represent distributed systems under uncertainty over which processes are still active (are alive) and which processes have failed or crashed (are dead). Our work generalizes the logic of knowledge for pure simplicial complexes, where all processes are alive, by Goubault et al. Our logical semantics has a satisfaction relation defined simultaneously with a definability relation. The latter restricts which formulas are allowed to have a truth value: dead processes cannot know or be uncertain about any proposition, and live processes cannot know or be uncertain about propositions involving processes they know to be dead. The logic satisfies some but not all axioms and rules of the modal logic **S5**. Impure simplicial complexes correspond to Kripke models where each agent's accessibility relation is an equivalence relation on a subset of the domain only. We also propose a notion of bisimulation for impure simplicial complexes and show bisimulation correspondence.

Keywords: Epistemic logic · Simplicial complex · Combinatorial topology

1 Introduction

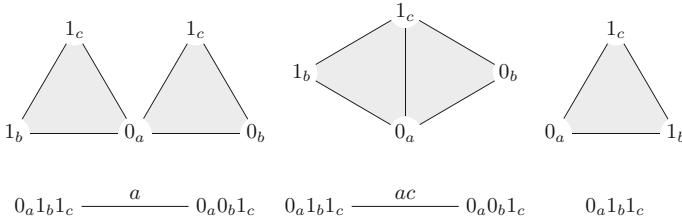
Epistemic logic investigates knowledge and belief, and change of knowledge and belief, in multi-agent systems. A foundational study is [24]. Knowledge change was extensively modelled in temporal epistemic logics [2, 19, 31] and more recently in dynamic epistemic logics [3, 11] including semantics based on histories of epistemic actions, both synchronously [4] and asynchronously [6].

Combinatorial topology has been used in distributed computing to model concurrency and asynchrony since [14, 27], with higher-dimensional topological properties entering the scene in [22, 23]. The basic structure in combinatorial topology is the *simplicial complex*, a collection of subsets called *simplices* of a set of *vertices*, closed under containment. Geometric manipulations such as subdivision have natural combinatorial counterparts.

An epistemic logic interpreted on simplicial complexes has been proposed in [17, 18, 26], including exact correspondence between certain multi-agent Kripke models where all relations are equivalence relations, and simplicial complexes.

Also, in those works and in e.g. [35] the action models of [3] are used to model distributed computing tasks and algorithms, where asynchronous histories are treated somewhat as in [6]. And we even find action models in their combinatorial topological incarnations as simplicial complexes [18].

To illustrate the subject matter to a reader familiar with encoding uncertainty in Kripke models, below some examples of simplicial complexes and (under those) corresponding Kripke models. These simplicial complexes are for three agents. As simplices are required to be labelled with different agents, a maximum size simplex consists of three vertices. This is called dimension 2. These are the triangles in the figure. (For 2 agents we get lines/edges, for 4 agents we get tetrahedra, etc.) Such a triangle corresponds to a state in a Kripke model. The variables $0_a, 1_b, 1_c$ (local value of a is 0, local value of b is 1, ...) that together determine the valuation of the single state Kripke model are distributed over the agents in the simplicial complex. The single triangle corresponds to the singleton $S5$ model on its right. (We assume reflexivity and symmetry of accessibility relations, and we label a state with the valuation of variables.) With two triangles, if they only intersect in a it means that agent a cannot distinguish these states (so that a is uncertain about the value of b), whereas if they intersect in a and c both a and c are uncertain about the value of b .



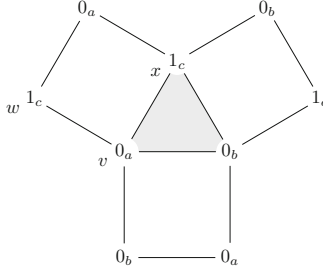
The current state of the distributed system is represented by a distinguished maximal simplex (called facet) of the simplicial complex, just as we need a distinguished or actual state in a Kripke model in order to evaluate propositions. For example, in the leftmost triangle, as well as in the leftmost state/world, a is uncertain whether the value of b is 0 or 1, whereas b knows that its value is 1, and all three agents know that the value of c is 1.

Let us proceed to the novel results of this contribution. The so-called *impure simplicial complexes* were beyond the scope of the epistemic logic of [18]. Impure simplicial complexes encode uncertainty over which processes are still active. They model information states in synchronous message passing with crash failures (processes ‘dying’). The impure complex below is found in [22, Section 13.5.2], here decorated with local values in order to illustrate epistemic features.

This simplicial complex represents the result of possibly failed message passing between a, b, c . In the (2-dimensional) triangle in the middle the messages have all been received, whereas in the (1-dimensional) edges on the side a process has crashed: on the left, b is dead, on the right, a is dead, and below, c is dead.

We propose to interpret this in terms of knowledge and uncertainty: if the point of evaluation is vertex v for agent a , then a knows that the value of c is 1

and that the value of b is 0, but a is also uncertain whether c knows the value of b . This is because the vertex v is contained in the edge $\{w, v\}$ and in vertex w agent c knows that b is dead (that is, w is not contained in a simplex with a vertex for b), and v is also contained in the edge $\{v, x\}$ and in vertex x agent c knows that the value of b is 0. Similarly, in v agent a is uncertain whether b knows the value of c . And so on.



A semantics of knowledge involving impure complexes was considered in [34] and also in an unpublished manuscript by Goubault. An issue for such semantics is what an agent knows when she is dead. This quickly leads to counterintuitive scenarios. In [34, Section 3.3, Sect. 3.4], projections are proposed from impure complexes to pure subcomplexes or Kripke models for the subset of agents that are alive. Inspired by the epistemic logic for pure complexes of [18], the knowledge semantics in [34], and synchronous message passing in [22], we propose an epistemic logic for impure complexes.

Why impure complexes? Crashed processes can alternatively be modelled as non-responding processes in asynchronous message passing, on (larger) pure simplicial complexes. Dually, non-responding processes can be modelled as dead processes in a synchronous setting, for example after a time-out, on (smaller) impure complexes. The impure simplicial complex therefore seems a useful abstraction. Maybe this also could result in lower computational complexities for decision problems such as model checking. And why complexes at all? Our epistemic logic can alternatively be interpreted on certain Kripke models, as we will see. Such models however contain more, unused, information, as explicit in our results on bisimulation invariance.

Section 2 gives an introduction to simplicial complexes. Section 3 presents the epistemic semantics for impure complexes. Section 4 transforms impure complexes into a certain kind of Kripke models and vice versa. Section 5 defines bisimulation for impure complexes. Section 6 compares our results to the literature on knowledge and awareness, on knowledge and belief, and on multi-valued logics; we discuss: (i) how uncertainty over which agents are alive and dead relates to epistemic logics of awareness (of other agents) [1, 7, 13, 20], (ii) to what extent our epistemic notion that comes ‘just short of knowledge’ differs from various notions of belief [21, 24, 29, 33], (iii) how our semantics with the three values true, false and undefined relates to other multi-valued epistemic logics [15, 28, 30, 32], and (iv) the advantages of simplicial complexes versus Kripke models.

Only some results have proofs. All omitted proofs are found in an extended version on <https://arxiv.org/abs/2103.03032>.

2 Technical Preliminaries: Simplicial Complexes

Given are a countable set A of *agents* a_0, a_1, \dots (or a, b, \dots) and a countable set $P = \bigcup_{a \in A} P_a$ of *variables*. The elements p_a, q_a, \dots of P_a for some $a \in A$ are the *local variables for agent a* . We often assume that A is finite and $|A| = n + 1$.

Definition 1 (Language). *The language $\mathcal{L}_K(A, P)$ is defined as: $\varphi ::= p_a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \widehat{K}_a\varphi$.*

We will write $\mathcal{L}_K(P)$ if A is clear from the context, and \mathcal{L}_K if A and P are clear from the context. Other connectives $\top, \rightarrow, \leftrightarrow, \vee$ and K_a are defined by abbreviation as usual, where for the record $K_a\varphi := \neg\widehat{K}_a\neg\varphi$. In the semantics and inductive proofs of our work, $\widehat{K}_a\varphi$ is more elegant and allows more succinct proofs than the more common primitive $K_a\varphi$; however, this is mere syntactic and semantic sugar as they are interdefinable as above (for the semantics of K_a , see Lemma 4). For $B \subseteq A$, we write $\mathcal{L}_K|B$ for $\mathcal{L}_K(B, \bigcup_{a \in B} P_a)$, and where $\mathcal{L}_K|b$ means $\mathcal{L}|b$. For $\neg p$ we may write \bar{p} . Expression $K_a\varphi$ stands for ‘agent a knows (that) φ ’ and $\widehat{K}_a\varphi$ stands for ‘agent a considers it possible that φ ’ which we will abbreviate as ‘agent a considers (that) φ .’

Given a set of *vertices* V (representing *local states*; singular form *vertex*), a (*simplicial*) *complex* C is a set of non-empty finite subsets of V , called *simplices*, that is closed under subsets (for all $X \in C, Y \subseteq X$ implies $Y \in C$), and that contains all singleton subsets of V .

If $Y \subseteq X$ we say that Y is a *face* of X . A maximal simplex in C is a *facet*. The facets of a complex C are denoted as $\mathcal{F}(C)$, and the vertices of a complex C are denoted as $\mathcal{V}(C)$. The *star* of X , denoted $\text{star}(X)$, is defined as $\{Y \in C \mid X \subseteq Y\}$, where for $\text{star}(\{v\})$ we write $\text{star}(v)$. The *dimension* of a simplex X is $|X| - 1$, e.g., vertices are of dimension 0, while edges are of dimension 1. The dimension of a complex is the maximal dimension of its facets. A simplicial complex is *pure* if all facets have the same dimension. Otherwise it is *impure*.

Complex D is a *subcomplex* of complex C if $D \subseteq C$. The *m -skeleton* C of a n -dimensional complex C is the maximal subcomplex D of C of dimension $m < n$. We will use this term for pure and impure complexes, where we typically consider a pure m -skeleton of an impure n -dimensional complex.

We decorate the vertices of simplicial complexes with agent’s names, that we often refer to as *colours*. A *chromatic map* $\chi : \mathcal{V}(C) \rightarrow A$ assigns colours to vertices such that different vertices of the same simplex are assigned different colours. Thus, $\chi(v) = a$ denotes that the local state or vertex v belongs to agent a . Dually, the vertex of a simplex X coloured with a is denoted X_a . A pair (C, χ) consisting of a simplicial complex C and a colouring map χ is a *chromatic simplicial complex*. From now on, all simplicial complexes will be chromatic simplicial complexes.

We extend the usage of the term ‘skeleton’ as follows to chromatic simplicial complexes. Given processes $B \subseteq A$, the B -skeleton of a chromatic complex (C, χ) , denoted $(C, \chi)|B$, is defined as $\{X \in C, \chi(X) \subseteq B\}$ (it is required to be non-empty).

We decorate the vertices of simplicial complexes with local variables $p_a \in P_a$ for $a \in A$, where we recall that $\bigcup_{a \in A} P_a = P$. *Valuations* (valuation functions) assign sets of local variables for agents a to vertices coloured a are denoted ℓ, ℓ', \dots . For any $X \in C$, $\ell(X)$ stands for $\bigcup_{v \in X} \ell(v)$.

A *simplicial model* \mathcal{C} is a triple (C, χ, ℓ) where (C, χ) is a chromatic simplicial complex and ℓ a valuation function. A *pointed simplicial model* is a pair (\mathcal{C}, X) with $X \in C$; X is called the *designated face* or the *point*.

3 Epistemic Logic on Impure Simplicial Models

Given agents A and variables $P = \bigcup_{a \in A} P_a$, let $\mathcal{C} = (C, \chi, \ell)$ be a simplicial model, $X \in C$, and $\varphi \in \mathcal{L}_K(A, P)$. Informally, we now wish to define a satisfaction relation \models between **some** but not all pairs (\mathcal{C}, X) and formulas φ . Not all, because if agent a does not occur in X (if $a \notin \chi(X)$), we do not wish to interpret certain formulas involving a , such as p_a and formulas of shape $\widehat{K}_a \varphi$. This is, because, if X were a facet (a maximal simplex), the absence of a would mean that the process is absent/dead, and dead processes do not have local values or know anything. The relation should therefore be partial. However, this relation is fairly complex, because we may wish to interpret formulas $K_b \varphi$ in (\mathcal{C}, X) , with $b \in \chi(X)$, where after all agent a ‘occurs’ in φ , for example expressing that a ‘live’ process b is uncertain whether process a is ‘dead’. Formally, we therefore proceed slightly differently.

We first define an auxiliary relation \bowtie between (all) pairs (\mathcal{C}, X) and formulas φ , where $\mathcal{C}, X \bowtie \varphi$ informally means that (the interpretation of) φ is *defined in* (\mathcal{C}, X) . For “not $\mathcal{C}, X \bowtie \varphi$ ” we write $\mathcal{C}, X \not\bowtie \varphi$, for “ φ is undefined.” Subsequently we then formally also define relation \models between (after all) all pairs (\mathcal{C}, X) and formulas φ , where as usual $\mathcal{C}, X \models \varphi$ means that φ is *true in* (\mathcal{C}, X) . In expressions $\mathcal{C}, X \bowtie \varphi$ and $\mathcal{C}, X \models \varphi$ we omit the parentheses in (\mathcal{C}, X) , as above. For “not $(\mathcal{C}, X \models \varphi)$ ” we write $\mathcal{C}, X \not\models \varphi$. Unusually, $\mathcal{C}, X \not\models \varphi$ does not mean that φ is false in (\mathcal{C}, X) but only means that φ is not true in (\mathcal{C}, X) .

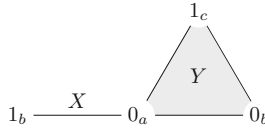
Definition 2 (Definability and satisfaction relation). *We define the definability relation \bowtie and the satisfaction relation \models by induction on φ .*

$$\begin{aligned}
 \mathcal{C}, X \bowtie p_a & \quad \text{iff } a \in \chi(X) \\
 \mathcal{C}, X \bowtie \varphi \wedge \psi & \quad \text{iff } \mathcal{C}, X \bowtie \varphi \text{ and } \mathcal{C}, X \bowtie \psi \\
 \mathcal{C}, X \bowtie \neg \varphi & \quad \text{iff } \mathcal{C}, X \not\bowtie \varphi \\
 \mathcal{C}, X \bowtie \widehat{K}_a \varphi & \quad \text{iff } \mathcal{C}, Y \bowtie \varphi \text{ for some } Y \in C \text{ with } a \in \chi(X \cap Y)
 \end{aligned}$$

$$\begin{aligned}
\mathcal{C}, X \models p_a & \text{ iff } a \in \chi(X) \text{ and } p_a \in \ell(X) \\
\mathcal{C}, X \models \varphi \wedge \psi & \text{ iff } \mathcal{C}, X \models \varphi \text{ and } \mathcal{C}, X \models \psi \\
\mathcal{C}, X \models \neg\varphi & \text{ iff } \mathcal{C}, X \not\models \varphi \text{ and } \mathcal{C}, X \not\models \varphi \\
\mathcal{C}, X \models \widehat{K}_a\varphi & \text{ iff } \mathcal{C}, Y \models \varphi \text{ for some } Y \in \mathcal{C} \text{ with } a \in \chi(X \cap Y)
\end{aligned}$$

Given $\varphi, \psi \in \mathcal{L}_K$, φ is equivalent to ψ iff for all (\mathcal{C}, X) : $(\mathcal{C}, X \not\models \varphi \text{ iff } \mathcal{C}, X \not\models \psi)$ implies $(\mathcal{C}, X \models \varphi \text{ iff } \mathcal{C}, X \models \psi)$; and φ is valid iff for all (\mathcal{C}, X) : $\mathcal{C}, X \not\models \varphi$ implies $\mathcal{C}, X \models \varphi$.

Example 1. Consider the following impure simplicial model \mathcal{C} for three agents a, b, c with local variables respectively p_a, p_b, p_c . A vertex v is labelled 0_a if $\chi(v) = a$ and $p_a \notin \ell(v)$, 1_b if $\chi(v) = b$ and $p_b \in \ell(v)$, etc. Some simplices have been named.



As expected, $\mathcal{C}, X \models p_b \wedge \neg p_a$, where the conjunct $\mathcal{C}, X \models \neg p_a$ is justified by $\mathcal{C}, X \not\models p_a$ and $\mathcal{C}, X \not\models p_a$. We also have $\mathcal{C}, X \models \widehat{K}_a p_b$, because $a \in \chi(X)$ and $\mathcal{C}, X \models p_b$.

Illustrating the novel aspects of the semantics, $\mathcal{C}, X \not\models p_c$, because $c \notin \chi(X)$ so that $\mathcal{C}, X \not\models p_c$. Similarly $\mathcal{C}, X \not\models \neg p_c$. Also, $\mathcal{C}, X \not\models \widehat{K}_c \neg p_a$ and similarly $\mathcal{C}, X \not\models \neg \widehat{K}_c \neg p_a$, again because $c \notin \chi(X)$. Still, $\neg p_a$ is true in both facets: $\mathcal{C}, X \models \neg p_a$ and $\mathcal{C}, Y \models \neg p_a$.

Although $\mathcal{C}, X \not\models p_c$, after all $\mathcal{C}, X \models \widehat{K}_a p_c$, because $a \in \chi(X \cap Y)$ and $\mathcal{C}, Y \models p_c$. Statement $\widehat{K}_a p_c$ says that agent a considers it possible that atom p_c is true. For this to be true agent c does not have to be alive in facet X . It is sufficient that agent a considers it possible that agent c is alive.

We also have $\mathcal{C}, Y \models K_b p_c$. This is easier to see after we have introduced the (derived) semantics for knowledge directly. We then explain why even $\mathcal{C}, X \models K_a p_c$ (not a typo).

Because $\mathcal{C}, X \not\models \varphi$ is not equivalent to $\mathcal{C}, X \models \neg\varphi$, we need to prove many intuitively expected results anew. The main results are as follows. We recall that omitted proofs are found in the extended version <https://arxiv.org/abs/2103.03032>.

Lemma 1. *If $\mathcal{C}, X \models \varphi$ then $\mathcal{C}, X \not\models \varphi$.*

Also, $\mathcal{C}, X \not\models \varphi$, iff $\mathcal{C}, X \models \varphi$ or $\mathcal{C}, X \models \neg\varphi$. In the next lemma, let $\xi[p/\varphi]$ be uniform substitution in ξ of p by φ .

Lemma 2. *Let (\mathcal{C}, X) be given. If φ is equivalent to ψ , then $\mathcal{C}, X \not\models \xi[p/\varphi]$ iff $\mathcal{C}, X \not\models \xi[p/\psi]$.*

Lemma 3. *If φ is equivalent to ψ , then $\xi[p/\varphi]$ is equivalent to $\xi[p/\psi]$.*

Lemma 4 presents the direct semantics for non-primitive logical connectives, that were defined by abbreviation. It is proved using those definitions.

Lemma 4. *Let $\mathcal{C} = (C, \chi, \ell)$, $X \in C$, and $\varphi, \psi \in \mathcal{L}_K$ be given. Then:*

$$\begin{aligned} \mathcal{C}, X \models \varphi \vee \psi & \text{ iff } \mathcal{C}, X \bowtie \varphi, \mathcal{C}, X \bowtie \psi, \text{ and } \mathcal{C}, X \models \varphi \text{ or } \mathcal{C}, X \models \psi \\ \mathcal{C}, X \models \varphi \rightarrow \psi & \text{ iff } \mathcal{C}, X \bowtie \varphi, \mathcal{C}, X \bowtie \psi, \text{ and } \mathcal{C}, X \models \varphi \text{ implies } \mathcal{C}, X \models \psi \\ \mathcal{C}, X \models \varphi \leftrightarrow \psi & \text{ iff } \mathcal{C}, X \bowtie \varphi, \mathcal{C}, X \bowtie \psi, \text{ and } \mathcal{C}, X \models \varphi \text{ iff } \mathcal{C}, X \models \psi \\ \mathcal{C}, X \models K_a \varphi & \text{ iff } \mathcal{C}, X \bowtie K_a \varphi, \text{ and} \\ & \mathcal{C}, Y \bowtie \varphi \text{ implies } \mathcal{C}, Y \models \varphi \text{ for all } Y \in C \text{ with } a \in \chi(X \cap Y) \end{aligned}$$

The final three lemmas relate (definability and) truth between simplices containing each other. Their inductive proofs are representative of most proofs of results in the contribution.

Lemma 5. *If $\mathcal{C}, X \bowtie \varphi$ and $Y \in C$ such that $X \subseteq Y$, then $\mathcal{C}, Y \bowtie \varphi$.*

Proof. By induction on formula structure. Let $Y \in C$ with $X \subseteq Y$.

- $\mathcal{C}, X \bowtie p_a$, iff $a \in \ell(X)$, which implies $a \in \ell(Y)$, iff $\mathcal{C}, Y \bowtie p_a$.
- $\mathcal{C}, X \bowtie \neg\varphi$, iff $\mathcal{C}, X \bowtie \varphi$, which implies (by induction) $\mathcal{C}, Y \bowtie \varphi$, iff $\mathcal{C}, Y \bowtie \neg\varphi$.
- $\mathcal{C}, X \bowtie \varphi \wedge \psi$, iff $\mathcal{C}, X \bowtie \varphi$ and $\mathcal{C}, X \bowtie \psi$, which implies (by induction) $\mathcal{C}, Y \bowtie \varphi$ and $\mathcal{C}, Y \bowtie \psi$, iff $\mathcal{C}, Y \bowtie \varphi \wedge \psi$.
- $\mathcal{C}, X \bowtie \tilde{K}_a \varphi$, iff $\mathcal{C}, Z \bowtie \varphi$ for some $Z \in C$ with $a \in \chi(X \cap Z)$, iff (as $X \subseteq Y$) $\mathcal{C}, Z \bowtie \varphi$ for some $Z \in C$ with $a \in \chi(Y \cap Z)$, iff $\mathcal{C}, Y \bowtie \tilde{K}_a \varphi$.

Lemma 6. *If $\mathcal{C}, X \models \varphi$ and $Y \in C$ such that $X \subseteq Y$, then $\mathcal{C}, Y \models \varphi$.*

Proof. The proof is by induction on the structure of formulas φ in negation normal form, where we note that inductive proof on the usual formula structure fails for the case $\neg\varphi$. In the arXiv version it is shown over several lemmas that every formula is equivalent to one in negation normal form. Let (\mathcal{C}, X) with $\mathcal{C} = (C, \chi, \ell)$ and $X \subseteq Y$ be given.

- $\mathcal{C}, X \models p_a$ iff $a \in \chi(X)$ and $p_a \in \ell(X)$. As $X \subseteq Y$, also $a \in \chi(Y)$. We recall that $p_a \in \ell(X)$ means that there is $v \in X$ with $\chi(v) = a$ and $p_a \in \ell(v)$. Therefore, as $X \subseteq Y$, also $p_a \in \ell(Y)$. By definition, $a \in \chi(Y)$ and $p_a \in \ell(Y)$ means that $\mathcal{C}, Y \models p_a$.
- $\mathcal{C}, X \models \neg p_a$, iff $\mathcal{C}, X \bowtie p_a$ and $\mathcal{C}, X \not\models p_a$, iff $a \in \chi(X)$ and $\mathcal{C}, X \not\models p_a$. Again, we obtain that $a \in \chi(Y)$. Therefore also $\mathcal{C}, Y \bowtie p_a$. Towards a contradiction, assume that $\mathcal{C}, Y \models p_a$. Then $p_a \in \ell(Y)$, and as $v \in X \subseteq Y$, also $p_a \in \ell(X)$ and thus $\mathcal{C}, X \models p_a$, contradicting $\mathcal{C}, X \not\models p_a$. Therefore $\mathcal{C}, Y \not\models p_a$. From $\mathcal{C}, Y \bowtie p_a$ and $\mathcal{C}, Y \not\models p_a$ now follows by definition that $\mathcal{C}, Y \models \neg p_a$.
- $\mathcal{C}, X \models \varphi \wedge \psi$, iff $\mathcal{C}, X \models \varphi$ and $\mathcal{C}, X \models \psi$, which implies (by induction) $\mathcal{C}, Y \models \varphi$ and $\mathcal{C}, Y \models \psi$, iff $\mathcal{C}, Y \models \varphi \wedge \psi$.

- $\mathcal{C}, X \models \varphi \vee \psi$, iff (Lemma 4) $\mathcal{C}, X \bowtie \varphi$, $\mathcal{C}, X \bowtie \psi$, and $\mathcal{C}, X \models \varphi$ or $\mathcal{C}, X \models \psi$. This implies by Lemma 5 and induction that: $\mathcal{C}, Y \bowtie \varphi$, $\mathcal{C}, Y \bowtie \psi$, and $\mathcal{C}, Y \models \varphi$ or $\mathcal{C}, Y \models \psi$, which is by definition $\mathcal{C}, Y \models \varphi \vee \psi$.
- $\mathcal{C}, X \models K_a \varphi$, iff $\mathcal{C}, X \bowtie K_a \varphi$, and $\mathcal{C}, Z \bowtie \varphi$ implies $\mathcal{C}, Z \models \varphi$ for all $Z \in C$ with $a \in \chi(X \cap Z)$. From $\mathcal{C}, X \bowtie K_a \varphi$, $X \subseteq Y$ and Lemma 5 we obtain $\mathcal{C}, Y \bowtie K_a \varphi$. Further, $a \in \chi(X)$ and $X \subseteq Y$ implies that $a \in \chi(X \cap Z)$ iff $a \in \chi(Y \cap Z)$. From all this we obtain that $\mathcal{C}, Y \bowtie K_a \varphi$, and $\mathcal{C}, Z \models \varphi$ implies $\mathcal{C}, Z \models \varphi$ for all $Z \in C$ with $a \in \chi(Y \cap Z)$, which is by definition equivalent to $\mathcal{C}, Y \models K_a \varphi$.
- $\mathcal{C}, X \models \widehat{K}_a \varphi$, iff $\mathcal{C}, Z \models \varphi$ for some $Z \in C$ with $a \in \chi(X \cap Z)$, which implies (as $X \subseteq Y$) that $\mathcal{C}, Z \models \varphi$ for some $Z \in C$ with $a \in \chi(Y \cap Z)$, iff $\mathcal{C}, Y \models \widehat{K}_a \varphi$.

Lemma 7. *If $\mathcal{C}, X \models \varphi$ and $Y \in C$ s.t. $Y \subseteq X$ and $\mathcal{C}, Y \bowtie \varphi$, then $\mathcal{C}, Y \models \varphi$.*

Proof. Let now $Y \in C$ such that $Y \subseteq X$. In all inductive cases we assume that the formula is defined in Y .

- $\mathcal{C}, X \models p_a$, iff $a \in \chi(X)$ and $p_a \in \ell(X)$, that is, $p_a \in \ell(X_a)$. As $\mathcal{C}, Y \bowtie p_a$, also $a \in \chi(Y)$, so that $a \in \chi(X \cap Y)$. Therefore $X_a \subseteq X \cap Y$, so that $p_a \in \ell(X_a) \subseteq \ell(Y)$.
- $\mathcal{C}, X \models \neg \varphi$, iff $\mathcal{C}, X \bowtie \varphi$ and $\mathcal{C}, X \not\models \varphi$. Using the contrapositive of Lemma 6, $\mathcal{C}, X \not\models \varphi$ implies $\mathcal{C}, Y \bowtie \varphi$. From that, together with the assumption $\mathcal{C}, Y \bowtie \varphi$, we obtain by definition $\mathcal{C}, Y \models \neg \varphi$.
- $\mathcal{C}, X \models \varphi \wedge \psi$, iff $\mathcal{C}, X \models \varphi$ and $\mathcal{C}, X \models \psi$, which implies (by induction) $\mathcal{C}, Y \models \varphi$ and $\mathcal{C}, Y \models \psi$, iff $\mathcal{C}, Y \models \varphi \wedge \psi$.
- $\mathcal{C}, X \models \widehat{K}_a \varphi$, iff $\mathcal{C}, Z \models \varphi$ for some $Z \in C$ with $a \in \chi(X \cap Z)$. Assumption $\mathcal{C}, Y \bowtie \widehat{K}_a \varphi$ implies $a \in \chi(Y)$, so that it follows from $a \in \chi(X \cap Z)$ and $Y \subseteq X$ that $a \in \chi(Y \cap Z)$. Therefore $\mathcal{C}, Z \models \varphi$ for some $Z \in C$ with $a \in \chi(Y \cap Z)$, which is by definition $\mathcal{C}, Y \models \widehat{K}_a \varphi$.

3.1 Validities and Differences with the Logic S5

name	validity	restriction	open question
T	$K_a \varphi \rightarrow \varphi$	—	
4	$K_a \varphi \rightarrow K_a K_a \varphi$	—	
5	$\widehat{K}_a \varphi \rightarrow K_a \widehat{K}_a \varphi$	—	
K	$K_a(\varphi \rightarrow \psi) \rightarrow K_a \varphi \rightarrow K_a \psi$	$\varphi, \psi \in \mathcal{L}_K b$	$\varphi, \psi \in \mathcal{L}_K \{a, b\}$?
N	from φ infer $K_a \varphi$	—	
MP	from $\varphi \rightarrow \psi$ and φ infer ψ	$\varphi, \psi \in \mathcal{L}_K a$	$\varphi, \psi \in \mathcal{L}_K$?
L	$K_a p_a \vee K_a \neg p_a$	—	

The table is an overview of our results in relation to the logic **S5**. Given the linguistic restrictions on the **K** axiom and on, possibly, **MP**, it seems unclear what the complete axiomatization for the epistemic logic of impure simplices is. Without the language restriction in **K** to a single agent, there are

counterexamples. In Example 1 we can observe that $\mathcal{C}, X \models K_a(p_c \rightarrow \neg p_b)$, and also $\mathcal{C}, X \models K_a p_c$, but $\mathcal{C}, X \not\models K_a \neg p_b$. Given that **K** does not hold for three or more agents, it is easy to see that the often interchangeable principle $K_a(\varphi \wedge \psi) \leftrightarrow (K_a \varphi \wedge K_a \psi)$ is also invalid.

A consequence of the knowledge semantics is that the agent may know a proposition even if that proposition is not defined in all possible simplices. In particular the proposition may not be true in the actual simplex (although it cannot be false there): $K_a \varphi$ rather means “as far as I know, φ ” than “I know φ .” We recall Example 1 wherein it is easy to see that $\mathcal{C}, X \models K_a p_c$ but $\mathcal{C}, X \not\models p_c$ (because $\mathcal{C}, X \not\models p_c$). So, in this case it does **not** hold that $\mathcal{C}, X \models K_a p_c$ implies $\mathcal{C}, X \models p_c$, and also $\mathcal{C}, X \not\models K_a p_c \rightarrow p_c$ (because $\mathcal{C}, X \not\models K_a p_c \rightarrow p_c$). But we still have that $\models K_a p_c \rightarrow p_c$, as for that we only need to consider (\mathcal{C}', X') such that both $K_a p_c$ and p_c are defined in X' . The validity of $K_a p_c \rightarrow p_c$ means that there is no (\mathcal{C}', X') such that $\mathcal{C}', X' \models \neg p_c \wedge K_a p_c$: knowledge cannot be verifiably incorrect.

A good way to understand knowledge on impure complexes is dynamically: even if $K_a \varphi$ is true, an update (such as a subdivision) may be possible after which $K_a \varphi$ is no longer true, namely when φ is no longer defined: agent/process a had some remaining uncertainty whether the processes involved in φ were alive or dead and the update confirmed that they were dead. However, no update is possible after which φ is false.

When the simplicial complexes are pure, the logical semantics should become that of [18, 26] and the logic should be **S5**. This is indeed the case, and the ‘sanity check’ that was expected. Proofs are in the arXiv version. So we now have unrestricted validity of **K** and **MP**. We still need the \bowtie relation, as unlike [18] our semantics is ‘local’: we interpret formulas on any face, not merely on facets. So, still $\mathcal{C}, X \not\models p_a$ and $\mathcal{C}, X \not\models \neg p_a$, iff $a \notin \chi(X)$, etc.

4 Correspondence to Kripke Models

A one-to-one correspondence between simplicial models and multi-agent Kripke models satisfying the following three conditions is given in [18, 26]: (i) all accessibility relations are equivalences, (ii) all propositional variables are local, that is, there is an agent who knows the value of that variable, and (iii) the intersection of all relations is the identity. In [34] it is observed that transforming impure simplicial models results in Kripke models where dead agents (crashed processes) do not have equivalence accessibility relations, and projections to subsets of live agents are proposed. Combining [18] and [34], we propose Kripke models where agents may be dead **or** alive. We call them *local epistemic models*.

As before, given are a countable set A of *agents* and a countable set $P = \bigcup_{a \in A} P_a$ of (*local*) *variables*, where both are typically assumed finite. Given an abstract domain of objects called *states* and an agent a , a *local equivalence relation* ($\sim(a)$ or) \sim_a is a binary relation between elements of S that is an equivalence relation on a subset of S denoted S_a and otherwise empty. So, \sim_a induces a partition on S_a , whereas $\sim_a = \emptyset$ on the complement $\bar{S}_a := S \setminus S_a$ of

S_a . For $(s, t) \in \sim_a$ we write $s \sim_a t$, and for $\{t \mid s \sim_a t\}$ we write $[s]_a$: this is an equivalence class of the relation \sim_a on S_a . Given $s \in S$, let $A_s := \{a \in A \mid s \in S_a\}$. Set A_s contains the agents that are alive in state s .

Definition 3 (Local epistemic model). Local epistemic frames are pairs $\mathcal{M} = (S, \sim)$ where S is the domain of (global) states, and \sim maps each agent a to a local equivalence relation \sim_a . Local epistemic models are triples $\mathcal{M} = (S, \sim, L)$, where (S, \sim) is a local epistemic frame, and where valuation L is a function from S to $\mathcal{P}(P)$ satisfying that for all $a \in A$, $p_a \in P_a$ and $s, t \in S_a$, if $s \sim_a t$ then $p_a \in L(s)$ iff $p_a \in L(t)$. We say that all variables p_a are local for agent a . A pair (\mathcal{M}, s) where $s \in S$ is a pointed local epistemic model.

There is no requirement for the valuation of variables p_a on the complement \bar{S}_a !

The interpretation of a formula $\varphi \in \mathcal{L}_K$ in a global state of a given pointed local epistemic model (\mathcal{M}, s) is by induction on the structure of φ . As before, we need relations \boxtimes of to determine whether the interpretation is defined, and \models to determine its truth value when defined.

Definition 4. Given $\mathcal{M} = (S, \sim, L)$, define \boxtimes and \models by induction on $\varphi \in \mathcal{L}_K$.

$$\begin{array}{l|l} \mathcal{M}, s \boxtimes p_a & \text{iff } s \in S_a & \mathcal{M}, s \models p_a & \text{iff } s \in S_a \& p_a \in L(s) \\ \mathcal{M}, s \boxtimes \neg\varphi & \text{iff } \mathcal{M}, s \not\boxtimes \varphi & \mathcal{M}, s \models \neg\varphi & \text{iff } \mathcal{M}, s \boxtimes \varphi \& \mathcal{M}, s \not\models \varphi \\ \mathcal{M}, s \boxtimes \varphi \wedge \psi & \text{iff } \mathcal{M}, s \boxtimes \varphi \& \mathcal{M}, s \boxtimes \psi & \mathcal{M}, s \models \varphi \wedge \psi \text{ iff } \mathcal{M}, s \models \varphi \& \mathcal{M}, s \models \psi \\ \mathcal{M}, s \boxtimes \widehat{K}_a\varphi & \text{iff } \mathcal{M}, t \not\boxtimes \varphi \text{ for } a \ t \sim_a \ s & \mathcal{M}, s \models \widehat{K}_a\varphi & \text{iff } \mathcal{M}, t \models \varphi \text{ for } a \ t \sim_a \ s \end{array}$$

Formula φ is valid iff for all (\mathcal{M}, s) , $\mathcal{M}, s \boxtimes \varphi$ implies $\mathcal{M}, s \models \varphi$. We let $\llbracket \varphi \rrbracket_{\mathcal{M}}$ stand for $\{s \in S \mid \mathcal{M}, s \models \varphi\}$. This set is called the denotation of φ in \mathcal{M} .

Analogous results as for the semantics on simplicial complexes can be obtained for the semantics on local epistemic models, demonstrating the tricky interaction between \boxtimes and \models .

Generalizing [18], we define maps $\sigma : \mathcal{K} \rightarrow \mathcal{S}$ (σ for Simplicial) and $\kappa : \mathcal{S} \rightarrow \mathcal{K}$ (κ for Kripke), such that σ maps each local epistemic model \mathcal{M} to a simplicial model $\sigma(\mathcal{M})$, and κ maps each simplicial model \mathcal{C} to a local epistemic model $\kappa(\mathcal{C})$.¹ As σ maps a state s in \mathcal{M} to a facet $X = \sigma(s)$ in $\sigma(\mathcal{M})$, and κ maps each facet X in \mathcal{C} to a state $s = \kappa(X)$ in $\kappa(\mathcal{C})$, these maps are also between pointed structures (\mathcal{M}, s) respectively (\mathcal{C}, X) . Subsequently we then show that for all $\varphi \in \mathcal{L}_K$, $\mathcal{M}, s \models \varphi$ iff $\sigma(\mathcal{M}, s) \models \varphi$, and that (for facets X) $\mathcal{C}, X \models \varphi$ iff $\kappa(\mathcal{C}, X) \models \varphi$.

Definition 5. Given local epistemic $\mathcal{M} = (S, \sim, L)$, we define $\sigma(\mathcal{M}) = (C, \chi, \ell)$:

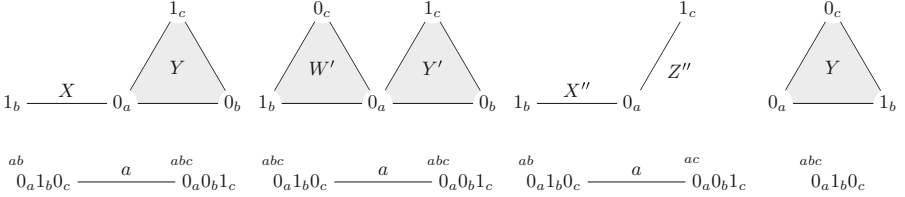
$$\begin{array}{l} X \in C & \text{iff } X = \{[s]_a \mid a \in B, B \subseteq A_s, B \neq \emptyset\} \text{ for some } s \in S \text{ with } A_s \neq \emptyset \\ \chi([s]_a) & = a \\ p_a \in \ell([s]_a) & \text{iff } p_a \in L(s) \end{array}$$

¹ Another construction to make simplicial models and epistemic models correspond is found in an unpublished manuscript by Goubault, that he was kind enough to share.

Given $\mathcal{C} = (C, \chi, \ell)$, we define $\kappa(\mathcal{C}) = (S, \sim, L)$ (where $X, Y \in \mathcal{F}(C)$):

$$\begin{aligned} S &= \mathcal{F}(C) \\ X \sim_a Y &\text{ iff } a \in \chi(X \cap Y) \\ p_a \in L(X) &\text{ iff } p_a \in \ell(X) \end{aligned}$$

Below, various examples of the transformation via κ of simplicial models into local epistemic models. Labels of states in epistemic models list the agents that are alive. Without the agents labels of states, the visualization would be ambiguous, as all two-state models below would then be identical.



Proposition 1. For all formulas $\varphi \in \mathcal{L}_K$, for all pointed local epistemic models (\mathcal{M}, s) : $\mathcal{M}, s \vDash \varphi$ iff $\sigma(\mathcal{M}, s) \vDash \varphi$, and $\mathcal{M}, s \models \varphi$ iff $\sigma(\mathcal{M}, s) \models \varphi$.

Proposition 2. For all formulas $\varphi \in \mathcal{L}_K$, for all pointed simplicial models (\mathcal{C}, X) with X a facet: $\mathcal{C}, X \vDash \varphi$ iff $\kappa(\mathcal{C}, X) \vDash \varphi$, and $\mathcal{C}, X \models \varphi$ iff $\kappa(\mathcal{C}, X) \models \varphi$.

5 Bisimulation for Impure Simplicial Complexes

We propose a notion of bisimulation between (possibly impure) simplicial models. It generalizes the notion for pure simplicial models proposed in [9, 17, 26]. We then show that bisimilarity implies modal equivalence and vice versa, on certain finitary simplicial models. For the sake of completeness and logical hygiene, we also ‘propose’ a notion called *local bisimulation* between local epistemic models for which the same results hold. As our logical semantics is somewhat non-standard, because it is based on \vDash and \models , this is weaker than the standard notion of bisimulation [5]. For the standard notion we still have that bisimilarity implies modal equivalence, but not, on image-finite models, in the other direction. The standard notion of bisimulation is too strong for that.

Definition 6 (Bisimulation between simplicial models). Let simplicial models $\mathcal{C} = (C, \chi, \ell)$ and $\mathcal{C}' = (C', \chi', \ell')$ be given. A relation R between $\mathcal{F}(C)$ and $\mathcal{F}(C')$ is rigid iff for all $X \in C$ and $X' \in C'$ with RXX' , $\chi(X) = \chi'(X')$. A non-empty rigid relation R between $\mathcal{F}(C)$ and $\mathcal{F}(C')$ is a (simplicial) bisimulation between \mathcal{C} and \mathcal{C}' , notation $R : \mathcal{C} \rightleftarrows \mathcal{C}'$, iff for all $Y \in \mathcal{F}(C)$ and $Y' \in \mathcal{F}(C')$ with $RY Y'$ the following three conditions are satisfied:

- **atoms:** for all $a \in \chi(Y)$ and $p_a \in P_a$, $p_a \in \ell(Y)$ iff $p_a \in \ell(Y')$.
- **forth:** for all $a \in \chi(Y)$, if $Z \in \mathcal{F}(C)$ and $a \in \chi(Y \cap Z)$ there is a $Z' \in \mathcal{F}(C')$ with $a \in \chi(Y' \cap Z')$ such that RZZ' .

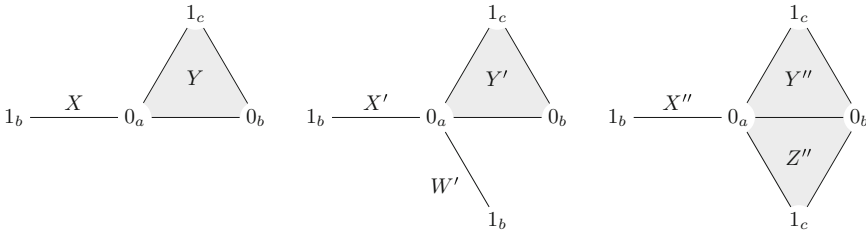
- **back:** for all $a \in \chi(Y')$, if $Z' \in \mathcal{F}(C')$ and $a \in \chi(Y' \cap Z')$ there is a $Z \in \mathcal{F}(C)$ with $a \in \chi(Y \cap Z)$ such that RZZ' .

A total simplicial bisimulation R is a simplicial bisimulation such that for all $X \in \mathcal{F}(C)$ there is a $X' \in \mathcal{F}(C')$ with RXX' and for all $X' \in \mathcal{F}(C')$ there is a $X \in \mathcal{F}(C)$ with RXX' . If there is a bisimulation between C and C' we write $C \Leftrightarrow C'$. A bisimulation between pointed simplicial models (C, X) and (C', X') , where $X \in \mathcal{F}(C)$ and $X' \in \mathcal{F}(C')$, is a bisimulation R between C and C' such that RXX' , notation $R : (C, X) \Leftrightarrow (C', X')$, and if there is such a bisimulation we write $(C, X) \Leftrightarrow (C', X')$.

Given $B \subseteq A$, a B -restricted bisimulation or B -bisimulation between C and C' is a bisimulation between the B -skeletons of C and C' .

A relation $R : \mathcal{F}(C) \rightarrow \mathcal{F}(C')$ between facets induces a similarly denoted chromatic relation $R : \mathcal{V}(C) \rightarrow \mathcal{V}(C')$ between vertices by way of: if RXX' then for all $a \in A$, $RX_aX'_a$, where a relation R between vertices is *chromatic* if for all v and v' , if Rvv' then $\chi(v) = \chi(v')$. Dually, given a chromatic relation $R : \mathcal{V}(C) \rightarrow \mathcal{V}(C')$ between vertices, then for any simplices X, X' with $\chi(X) = \chi(X')$, RXX' denotes that for all $a \in \chi(X)$, $RX_aX'_a$. (Induced relations cannot be primitive to define bisimulations.)

Example 2. The three simplicial models below are all (totally) bisimilar. It will be obvious that we need to require RXX' as well as RXW' between the left and the middle, and RYY'' as well as RYZ'' between the left and the right.



Let (C, X) and (C', X') be given, with $C = (C, \chi, \ell)$ and $C' = (C', \chi', \ell')$. Model (C, X) is *modally equivalent* to (C', X') , notation $(C, X) \equiv (C', X')$, iff for all $\varphi \in \mathcal{L}_K$: $C, X \models \varphi$ iff $C', X' \models \varphi$. A simplicial complex C is *star-finite* if for all $v \in \mathcal{V}(C)$, $star(v)$ is finite. This is the obvious analogue of what is called *image-finite* for Kripke models.

Proposition 3. Given (C, X) , (C', X') , where $C = (C, \chi, \ell)$, $C' = (C', \chi', \ell')$.

- $(C, X) \Leftrightarrow (C', X')$ implies $(C, X) \equiv (C', X')$.
- $(C, X) \equiv (C', X')$ implies $(C, X) \Leftrightarrow (C', X')$, if C and C' are star-finite.

We close the section with a similar notion of bisimulation between Kripke models. A *local bisimulation* between local epistemic models $\mathcal{M} = (S, \sim, L)$ and $\mathcal{M}' = (S', \sim', L')$, notation $R : \mathcal{M} \Leftrightarrow^{loc} \mathcal{M}'$, is a non-empty relation $R \subseteq S \times S'$ such that for all $s \in S$, $s' \in S'$ with Rss' the following three conditions are satisfied:

- **atoms**: for all $a \in A_s$ and $p_a \in P_a$, $p_a \in L(s)$ iff $p_a \in L'(s')$.
- **forth**: for all $a \in A_s$, for all $t \sim_a s$, there is a $t' \sim'_a s'$ such that Rtt' .
- **back**: for all $a \in A_{s'}$, for all $t' \sim'_a s'$, there is a $t \sim_a s$ such that Rtt' .

The terminology and notation around bisimulations (as in Definition 6) also applies to local bisimulation. To get a *standard bisimulation* [5], notation $\leftrightarrow^{\text{st}}$, we replace the requirements: $a \in A_s$ (twice) and $a \in A_{s'}$ in the above definition by $a \in A$. It is therefore clear that standardly bisimilar implies locally bisimilar. It will also be obvious that on the class of multi-agent $\mathcal{S5}$ models, $\leftrightarrow^{\text{loc}} = \leftrightarrow^{\text{st}}$.

We define $(\mathcal{M}, s) \equiv^{\text{loc}} (\mathcal{M}', s')$ as: for all $\varphi \in \mathcal{L}_K$, $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}', s' \models \varphi$. Informally, this means that these pointed models contain the same information. (On the class of multi-agent $\mathcal{S5}$ models, \equiv^{loc} is the usual modal equivalence of pointed structures.) Local epistemic model $\mathcal{M} = (S, \sim, V)$ is image-finite if for all $a \in A$ and $s \in S_a$, $[s]_a$ is finite.

Proposition 4. *Let (\mathcal{M}, s) and (\mathcal{M}', s') be given. If $(\mathcal{M}, s) \leftrightarrow^{\text{loc}} (\mathcal{M}', s')$, then $(\mathcal{M}, s) \equiv^{\text{loc}} (\mathcal{M}', s')$. Let now $\mathcal{M}, \mathcal{M}'$ be image-finite. If $(\mathcal{M}, s) \equiv^{\text{loc}} (\mathcal{M}', s')$, then $(\mathcal{M}, s) \leftrightarrow^{\text{loc}} (\mathcal{M}', s')$.*

Example 3. It is not the case that $(\mathcal{M}, s) \equiv^{\text{loc}} (\mathcal{M}', s')$ implies $(\mathcal{M}, s) \leftrightarrow^{\text{st}} (\mathcal{M}', s')$. Consider once more the simplicial model \mathcal{C} of Example 1 and its counterpart $\kappa(\mathcal{C})$, and a different model \mathcal{M}' . They make the same formulas in \mathcal{L}_K true, and they are locally bisimilar, but they are not standardly bisimilar. (Variable p_c is false on the left but true on the right! See arXiv.)

$$\kappa(\mathcal{C}) : \begin{array}{ccc} & ab & \\ & 0_a 1_b 0_c & \xrightarrow{a} \\ & & abc \\ & & 0_a 0_b 1_c \end{array} \quad \mathcal{M}' : \begin{array}{ccc} & ab & \\ & 0_a 1_b 1_c & \xrightarrow{a} \\ & & abc \\ & & 0_a 0_b 1_c \end{array}$$

6 Comparison to the Literature and Further Research

Awareness of Agents. Knowing that an agent is alive is like saying that you are aware of that agent. Various (propositional) modal logics combine knowledge with (un)awareness [1, 7, 8, 13, 20]. However, in all those except [7] this is awareness of sets of *formulas*, not awareness of *agents*. One could consider ‘tricking’ awareness of agents in a given state as awareness of all formulas involving those agents. But this would be very different from our truly modal setting: whether a formula is defined in a given state is not a function of which agents are alive in that state, but a function of what agents are alive in indistinguishable states.

Belief and Knowledge. The logic of knowledge on impure complexes is ‘almost’ **S5**. Is it yet another notion of belief that is ‘almost’ like knowledge? It is not Hintikka’s favourite **S4** [24] nor **S4.4** [21, 33], as it satisfies **5**. It is not **KD45** [10], as it satisfies **T**. It is none of the proposals in [29] either, for similar reasons. We recall $B^\alpha \varphi$ of [29]: the agent believes φ on *assumption* α . Could this assumption not be that ‘ φ is defined in state s ’ (i.e., $M, s \vDash \varphi$)? Not really, as their results are for assumptions that are *formulas*, and in a binary semantics. But our definability assumption is not a linguistic feature, and results in a three-valued semantics.

Multi-valued Logic. Our semantics is a three-valued modal logic with a propositional basis known as Kleene’s weak three-valued logic: the value of any binary connective is unknown if one of its arguments is unknown [16]. Modal logical (including epistemic) extensions of multi-valued logics are found in [15, 28, 30, 32]. There is no relation to embeddings of three-valued propositional logics into modal logics [25]. In multi-valued modal logics the modal extension tends to be independent from the multi-valued propositional base. But not in our case: it is multi-modal, which is not so bad, but the value of a proposition does not merely depend on the variables occurring in it but also on the agent names occurring in the modalities. Worse, our logic does not satisfy **K** (nor $K_a(\varphi \wedge \psi) \leftrightarrow (K_a\varphi \wedge K_a\psi)$) nor **N**, so it is not a normal modal logic.

Simplicial Models Versus Kripke Models. Why simplicial models? We see three independent reasons. (i) Rediscovering epistemic logic in its entirety (including group epistemics, and any kind of dynamics) but interpreted on completely dual structures increases our insight in the foundations of knowledge. (ii) Local epistemic models contain too much information. See Proposition 4 and Example 3: the value of p_c in the left state of $\kappa(\mathcal{C})$ and \mathcal{M}' is irrelevant. Such superfluous information is absent in the corresponding complex \mathcal{C} . (iii) We can evaluate formulas in any simplex, not facets only. In Kripke models this involves *sets* of states, not states. Some $\mathcal{C}, v \models \varphi$ with $\chi(v) = a$ compares to some $\mathcal{M}, [s]_a \models \varphi$ ($[s]_a = \{t \mid t \sim_a s\}$). Some edge X for agents a, b compares to some set $[s]_a \cap [s]_b$. And so on.

Further Research. (i) We cannot formalize ‘ a knows that b is dead’ (or alive). This was on purpose: we targeted the simplest epistemic logic. But it is quite possible: add variables a_\downarrow for ‘ a is alive’, where $a_\uparrow := \neg a_\downarrow$ means ‘ a is dead’. Define $\mathcal{C}, X \bowtie a_\uparrow$ iff $a \notin \chi(X)$ and $\mathcal{C}, X \bowtie a_\downarrow$ iff $a \in \chi(X)$, for facets X only, and with the obvious truth definition. Valid is then $K_a a_\downarrow$. (ii) It is straightforward to add dynamics as *action models* [3], generalizing [17, 18, 26, 34, 35]. We envisage (possibly impure) *simplicial action models* representing incompletely specified tasks and algorithms, for example Byzantine agreement [12] on dynamically evolving (with agents dying) impure complexes. It should be noted however that our framework does not enjoy the property that positive formulas (intuitively, those without negations before K_a operators; corresponding to the universal fragment in first-order logic) are preserved after update. It might therefore be challenging to generalize results for pure complexes employing such ‘positive knowledge gain’ after update [17, 18] to truly failure-prone distributed systems modelled with impure complexes. (iii) We intend to resolve the bounds on **MP** and **K**, and subsequently prove the completeness of the axiomatization.

Acknowledgements. I thank the reviewers for their comments and support. I gratefully acknowledge comments on the extended arXiv version from: Armando Castañeda, Éric Goubault, Jérémy Ledent, Yoram Moses, Diego Velázquez and David Rosenblueth.

References

1. Ågotnes, T., Alechina, N.: A logic for reasoning about knowledge of unawareness. *J. Logic Lang. Inform.* **23**(2), 197–217 (2014)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. *J. ACM* **49**, 672–713 (2002)
3. Baltag, A., Moss, L., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: *Proceedings of 7th TARK*, pp. 43–56. Morgan Kaufmann (1998)
4. van Benthem, J., Gerbrandy, J., Hoshi, T., Pacuit, E.: Merging frameworks for interaction. *J. Philos. Log.* **38**, 491–526 (2009)
5. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press, Cambridge (2001)
6. Degremont, C., Löwe, B., Witzel, A.: The synchronicity of dynamic epistemic logic. In: *Proceedings of 13th TARK*, pp. 145–152. ACM (2011)
7. van Ditmarsch, H., French, T.: Semantics for knowledge and change of awareness. *J. Logic Lang. Inform.* **23**(2), 169–195 (2014)
8. van Ditmarsch, H., French, T., Velázquez-Quesada, F., Wáng, Y.: Implicit, explicit and speculative knowledge. *Artif. Intell.* **256**, 35–67 (2018). <https://doi.org/10.1016/j.artint.2017.11.004>
9. van Ditmarsch, H., Goubault, E., Ledent, J., Rajsbaum, S.: Knowledge and simplicial complexes (2021, to appear). <https://arxiv.org/abs/2002.08863>
10. van Ditmarsch, H., Halpern, J., van der Hoek, W., Kooi, B.: An introduction to logics of knowledge and belief. In: van Ditmarsch, H., Halpern, J., van der Hoek, W., Kooi, B. (eds.) *Handbook of Epistemic Logic*, pp. 1–51. College Publications (2015)
11. van Ditmarsch, H., van der Hoek, W., Kooi, B.: *Dynamic Epistemic Logic*. Synthese Library, vol. 337. Springer, Cham (2008). <https://doi.org/10.1007/978-1-4020-5839-4>
12. Dwork, C., Moses, Y.: Knowledge and common knowledge in a byzantine environment: crash failures. *Inf. Comput.* **88**(2), 156–186 (1990). [https://doi.org/10.1016/0890-5401\(90\)90014-9](https://doi.org/10.1016/0890-5401(90)90014-9)
13. Fagin, R., Halpern, J.: Belief, awareness, and limited reasoning. *Artif. Intell.* **34**(1), 39–76 (1988)
14. Fischer, M., Lynch, N., Paterson, M.: Impossibility of distributed consensus with one faulty process. *J. ACM* **32**(2), 374–382 (1985). <https://doi.org/10.1145/3149.214121>
15. Fitting, M.C.: Many-valued modal logics. In: *Fundamenta Informaticae*, pp. 365–448. Kluwer Academic Publishers (1992)
16. Gottwald, S.: Many-valued logic. In: Zalta, E. (ed.) *The Stanford Encyclopedia of Philosophy* (2020). <https://plato.stanford.edu/archives/sum2020/entries/logic-manyvalued/>
17. Goubault, É., Lazić, M., Ledent, J., Rajsbaum, S.: A dynamic epistemic logic analysis of the equality negation task. In: Soares Barbosa, L., Baltag, A. (eds.) *DALI 2019. LNCS*, vol. 12005, pp. 53–70. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38808-9_4
18. Goubault, E., Ledent, J., Rajsbaum, S.: A simplicial complex model for dynamic epistemic logic to study distributed task computability. In: *Proceedings of 9th GandALF. EPTCS*, vol. 277, pp. 73–87 (2018). <https://doi.org/10.4204/EPTCS.277.6>

19. Halpern, J., Moses, Y.: Knowledge and common knowledge in a distributed environment. *J. ACM* **37**(3), 549–587 (1990)
20. Halpern, J., Rêgo, L.: Reasoning about knowledge of unawareness revisited. *Math. Soc. Sci.* **65**(2), 73–84 (2013)
21. Halpern, J., Samet, D., Segev, E.: Defining knowledge in terms of belief: the modal logic perspective. *Rew. Symb. Logic* **2**(3), 469–487 (2009)
22. Herlihy, M., Kozlov, D., Rajsbaum, S.: *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, Burlington (2013)
23. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. *J. ACM* **46**(6), 858–923 (1999). <https://doi.org/10.1145/331524.331529>
24. Hintikka, J.: *Knowledge and Belief*. Cornell University Press, Ithaca (1962)
25. Kooi, B., Tamminga, A.: Three-valued logics in modal logic. *Stud. Logica.* **101**(5), 1061–1072 (2013). <https://doi.org/10.1007/s11225-012-9420-0>
26. Ledent, J.: *Geometric semantics for asynchronous computability*. Ph.D. thesis, École Polytechnique, Palaiseau, France (2019)
27. Loui, M., Abu-Amara, H.: Memory requirements for agreement among unreliable asynchronous processes. *Adv. Comput. Res.* **4**, 163–183 (1987)
28. Morikawa, O.: Some modal logics based on a three-valued logic. *Notre Dame J. Formal Log.* **30**(1), 130–137 (1989). <https://doi.org/10.1305/ndjfl/1093635000>
29. Moses, Y., Shoham, Y.: Belief as defeasible knowledge. *Artif. Intell.* **64**(2), 299–321 (1993). [https://doi.org/10.1016/0004-3702\(93\)90107-M](https://doi.org/10.1016/0004-3702(93)90107-M)
30. Odintsov, S.P., Wansing, H.: Modal logics with Belnapian truth values. *J. Appl. Non-Classical Logics* **20**(3), 279–301 (2010)
31. Pnueli, A.: The temporal logic of programs. In: *Proceedings of 18th FOCS*, pp. 46–57. IEEE Computer Society (1977). <https://doi.org/10.1109/SFCS.1977.32>
32. Rivieccio, U., Jung, A., Jansana, R.: Four-valued modal logic: Kripke semantics and duality. *J. Log. Comput.* **27**(1), 155–199 (2017). <https://doi.org/10.1093/logcom/exv038>
33. Stalnaker, R.: On logics of knowledge and belief. *Philos. Stud.* **128**(1), 169–199 (2005)
34. Velázquez, D.: *Una relación entre las lógicas modales y el enfoque topológico del cómputo distribuido*. Master’s thesis, UNAM, Mexico (2019)
35. Velázquez, D., Castañeda, A., Rosenblueth, D.: Communication pattern models: an extension of action models for dynamic-network distributed systems. In: *Proceedings of TARK XVIII* (2021)



Doubly Strongly First Order Dependencies

Pietro Galliani^(✉)

Free University of Bozen-Bolzano, Universitätsplatz 1 - Piazza Università 1,
39100 Bozen-Bolzano, Italy
Pietro.Galliani@unibz.it

Abstract. Team Semantics is a generalization of Tarskian Semantics that can be used to add to First Order Logic atoms and connectives expressing dependencies between the possible values of variables. Some of the resulting logics are more expressive than First Order Logic, while others are not. I characterize the (relativizable) atoms and families of atoms that do not increase the expressive power of First Order Logic when they and their complements are added to it, separately or jointly.

Keywords: Team semantics · Dependence logic · Second order logic

1 Introduction

Team Semantics generalizes Tarskian Semantics for First Order Logic by defining satisfaction with respect to sets of assignments (called *Teams*) rather than with respect to single assignments. This semantics arises naturally from the analysis of the game theoretic semantics of First Order Logic and its extensions: in brief, a team represents a set of possible game states (= variable assignments) that can be played at some subformula, and a team satisfies a subformula if the existential player has a¹ strategy that is winning for the corresponding subgame for every starting assignment in the team. This approach works equally well for extensions of First Order Logic whose game-theoretic semantics yield *imperfect information* games, as is the case for *Independence-Friendly Logic* [14, 15, 22] which was the reason for the original development of Team Semantics (then called “Trump Semantics”) in [16].

Jouko Väänänen [25] observed that a logic roughly equivalent to Independence-Friendly Logic, but with more convenient formal properties (for example *locality*, in the sense that the interpretation of a formula in a team depends only on the restriction of the team to the free variables of the formula), can be obtained by adding to First Order Logic, in place of the so-called *slashed quantifiers* (“there exists a y , chosen independently from x , such that ...”) of Independence-Friendly Logic, *functional dependence atoms* $=(x; y)$ that state

¹ Non-deterministic, for the commonly-used “lax” (see [3]) version of Team Semantics that we consider in this work.

that the values of y are *determined* by those of x . The resulting logic, called *Dependence Logic*, has been the subject of a considerable amount of research that cannot be summarized here (for an up-to-date introduction, we refer the reader to [9]). It was soon noticed that Team Semantics could also be used to extend First Order Logic via other atoms (see e.g. [3, 12]) or connectives (like the “contradictory negation” of [26], the “intuitionistic implication” of [1], or the generalized quantifiers of [2]) that have no direct analogues in Tarskian Semantics as their definitions likewise involve possible interactions between different assignments. Other Team Semantics-based logics use families of connectives different from the ones arising directly from the Game-Theoretic Semantics of First Order Logic: of particular interest in this context is the **FOT** logic of [19], that relates to ordinary First Order Logic not through Game Theoretic Semantics but on the level of *team definability*, in the sense that a family of teams is defined by a **FOT** formula if and only if the corresponding family of relations is first order definable and every sentence of **FOT** is equivalent to some first order sentence.²

Team Semantics – aside from its applications and connections with other areas, which we will not discuss here – can thus be seen as a generalization of Tarskian Semantics that allows for the construction of new kinds of extensions and fragments of First Order Logic; and while some of these extensions have been studied in some depth by now (see e.g. [18, 21] for the contradictory negation, [27] for the intuitionistic implications, or [3, 10, 11, 13, 23] for database-theoretic atoms), not much is yet known regarding the *general properties* of the extensions of First Order Logic obtainable through Team Semantics.

This work is a partial answer to the following question: which extensions of First Order Logic based on Team Semantics are genuinely more expressive than First Order Logic itself, and which ones instead can only specify properties that were already first order definable? This question is the obvious starting point for a classification of Team Semantics-based extensions of First Order Logic; and yet, at the moment only some very limited answers (see [6, 7]) are known.

The main result of this work is a full characterization – aside from the technical condition of *relativizability*, that most dependencies of interest satisfy – of the dependencies that are *doubly strongly first order* in the sense that both they and their complements can be added (jointly or separately) to First Order Logic with Team Semantics without increasing their expressive power. This may be regarded as another step towards the classification of the expressive capabilities – and computational costs – of logics based on Team Semantics, a topic of both practical (especially given the applications of Team Semantics to knowledge representation and database theory, for which we refer the reader to [9]) and theoretical interest.

² Thus, all families \mathcal{D} of first order dependencies are “strongly first order” for **FOT** in the sense analogous to Definition 6. Additionally, whenever $\mathbf{FO}(\mathcal{D}) \equiv \mathbf{FO}$ all formulas (not just all sentences) of $\mathbf{FO}(\mathcal{D})$ are equivalent to formulas of **FOT**.

2 Preliminaries

2.1 Team Semantics

Definition 1 (Team). Let \mathfrak{M} be a first order model with domain $\mathbf{Dom}(\mathfrak{M}) = M$ and let V be a finite set of variables. Then a team X over \mathfrak{M} with domain $\mathbf{Dom}(X) = V$ is a set of variable assignments $s : V \rightarrow M$. Given such a team and some tuple of variables $\vec{v} = v_1 \dots v_k \in V^k$, we will write $X(\vec{v})$ for the $|\vec{v}|$ -ary relation $\{s(\vec{v}) : s \in X\} \subseteq M^{|\vec{v}|}$, where $s(\vec{v})$ is the tuple $s(v_1) \dots s(v_k)$.

Definition 2 (Team Semantics for First Order Logic). Let \mathfrak{M} be a first order model, let ϕ be a first order formula in Negation Normal Form³ over the signature of \mathfrak{M} , and let X be a team over \mathfrak{M} whose domain contains the free variables of ϕ . Then we say that X satisfies ϕ in \mathfrak{M} , and we write $\mathfrak{M} \models_X \phi$, if this follows from the following rules:

- TS-lit:** If ϕ is a first order literal, $\mathfrak{M} \models_X \phi$ if and only if, for all assignments $s \in X$, $\mathfrak{M} \models_s \phi$ in the usual sense of Tarskian Semantics;
- TS- \vee :** $\mathfrak{M} \models_X \phi_1 \vee \phi_2$ if and only if $X = Y \cup Z$ for two $Y, Z \subseteq X$ such that $\mathfrak{M} \models_Y \phi_1$ and $\mathfrak{M} \models_Z \phi_2$;
- TS- \wedge :** $\mathfrak{M} \models_X \phi_1 \wedge \phi_2$ if and only if $\mathfrak{M} \models_X \phi_1$ and $\mathfrak{M} \models_X \phi_2$;
- TS- \exists :** $\mathfrak{M} \models_X \exists v \psi$ if and only if there exists some function⁴ $H : X \rightarrow \mathcal{P}(M) \setminus \{\emptyset\}$ such that $\mathfrak{M} \models_{X[H/v]} \psi$, where $X[H/v] = \{s[m/v] : s \in X, m \in H(s)\}$;
- TS- \forall :** $\mathfrak{M} \models_X \forall v \psi$ if and only if $\mathfrak{M} \models_{X[M/v]} \psi$, where $X[M/v] = \{s[m/v] : s \in X, m \in M\}$.

A sentence ϕ is true in a model \mathfrak{M} if and only if $\mathfrak{M} \models_{\{\emptyset\}} \phi$, where $\{\emptyset\}$ is the team containing only the empty assignment. In that case, we write that $\mathfrak{M} \models \phi$.

Over First Order Logic, Team Semantics reduces to Tarskian Semantics:

Proposition 1. ([25], Corollary 3.32) Let \mathfrak{M} be a first order model, let ϕ be a first order formula in Negation Normal Form over the signature of \mathfrak{M} , and let X be a team over \mathfrak{M} whose domain contains the free variables of ϕ . Then $\mathfrak{M} \models_X \phi$ if and only if, for all $s \in X$, $\mathfrak{M} \models_s \phi$ in the sense of Tarskian Semantics.

In particular, if ϕ is a sentence, $\mathfrak{M} \models \phi$ in the sense of Team Semantics if and only if $\mathfrak{M} \models \phi$ in the sense of Tarskian Semantics.

Nonetheless, Team Semantics makes it possible to extend First Order Logic via new types of atoms specifying *collective* properties of (that is to say, *dependencies* between) the assignments in a team. The earliest and arguably most important atoms studied in this context are the *functional dependence atoms* [25]

TS-dep: If \vec{x} and \vec{y} are tuples of variables, $\mathfrak{M} \models_{X=(\vec{x};\vec{y})}$ if and only if, for all $s, s' \in X$, if $s(\vec{x}) = s'(\vec{x})$ then $s(\vec{y}) = s'(\vec{y})$.

³ In this work we will assume that all expressions are in Negation Normal Form.

⁴ Here $\mathcal{P}(M)$ represents the powerset $\{X : X \subseteq M\}$ of M .

This rule corresponds precisely to the database-theoretic notion of *functional dependence*; and it was soon recognized that other dependency notions may also be studied in the same context, such as *independence atoms* [12]

TS-ind: $\mathfrak{M} \models_X \vec{x} \perp_{\vec{y}} \vec{z}$ if and only if for any two $s, s' \in X$ with $s(\vec{y}) = s'(\vec{y})$ there exists some $s'' \in X$ with $s''(\vec{x}\vec{y}) = s(\vec{x}\vec{y})$ and $s''(\vec{y}\vec{z}) = s'(\vec{y}\vec{z})$

which as per [2] have a close connection with database-theoretic *embedded multivalued dependencies*, *inclusion dependencies* [3, 11, 13]

TS-inc: $\mathfrak{M} \models_X \vec{x} \subseteq \vec{y}$ if and only if $X(\vec{x}) \subseteq X(\vec{y})$

and *exclusion dependencies* [3, 24]

TS-exc: $\mathfrak{M} \models_X \vec{x} \bar{\parallel} \vec{y}$ if and only if $X(\vec{x}) \cap X(\vec{y}) = \emptyset$.

But what is, in general, a dependency? The following definition is from [20]:

Definition 3 (Generalized Dependency). A k -ary generalized dependency is a class \mathbf{D} , closed under isomorphisms, of models $\mathfrak{M} = (M, R)$ over the signature $\{R\}$, where R is a k -ary relation symbol. \mathbf{D} corresponds to the rule

TS-D: $\mathfrak{M} \models_X \mathbf{D}\vec{x}$ if and only if $(\mathbf{Dom}(\mathfrak{M}), R := X(\vec{x})) \in \mathbf{D}$

where \vec{x} is any tuple of k variables and $(\mathbf{Dom}(\mathfrak{M}), R := X(\vec{x}))$ is the model with the same domain $\mathbf{Dom}(\mathfrak{M})$ of \mathfrak{M} and with signature $\{R\}$, where R is interpreted as $R^{\mathfrak{M}} = X(\vec{x}) = \{s(\vec{x}) : s \in X\}$.

Definition 4 (FO(\mathcal{D})). Let $\mathcal{D} = \{\mathbf{D}_1, \mathbf{D}_2, \dots\}$ be a set of generalized dependencies. Then **FO(\mathcal{D})** is the logic obtained by taking First Order Logic (with Team Semantics) **FO** and adding to it all the generalized dependency atoms $\mathbf{D} \in \mathcal{D}$.

An important class of generalized dependencies is that of the *first order ones*:

Definition 5 (First Order Generalized Dependencies). A k -ary generalized dependency \mathbf{D} is first order if there exists some first order sentence $\mathbf{D}(R)$, over the signature $\{R\}$ where R is a k -ary relation symbol, such that $\mathbf{D} = \{(M, R) : (M, R) \models \mathbf{D}(R)\}$.

If \mathbf{D} is a first order generalized dependency, the rule **TS-D** is equivalent to

TS-D-FO: $\mathfrak{M} \models_X \mathbf{D}\vec{x}$ if and only if $(\mathbf{Dom}(\mathfrak{M}), R := X(\vec{x})) \models \mathbf{D}(R)$.

Functional dependence atoms, independence atoms, inclusion atoms, and exclusion atoms are all first order; but the logics obtained by adding them to First Order Logic are more expressive than First Order Logic itself. This differs from the case of generalized quantifiers in Tarskian Semantics, in which if a quantifier \mathbf{Q} is first order definable then adding it to First Order Logic yields nothing new. The intriguing phenomenon of first order dependencies increasing the expressiveness of First Order Logic is a consequence of the higher order character of

Team Semantics, and in particular to the existential second order quantification implicit in the rules **TS- \vee** and **TS- \exists** ,⁵ and it raises immediately a natural (and still open) problem: can we identify the dependencies (or the families of dependencies, or even more in general the families of connectives and dependencies) that do *not* increase the expressive power of First Order Logic?

2.2 Strongly First Order Dependencies

In this section, we will recall some partial results related to the problem of characterizing the dependencies that are “safe” to add to First Order Logic:

Definition 6 (Strongly First Order Dependencies). *A generalized dependency \mathbf{D} is strongly first order if and only if $\mathbf{FO}(\mathbf{D}) \equiv \mathbf{FO}$,⁶ i.e., if and only if every sentence of $\mathbf{FO}(\mathbf{D})$ is equivalent to some first order sentence. Likewise, a family of dependencies \mathcal{D} is strongly first order if and only if $\mathbf{FO}(\mathcal{D}) \equiv \mathbf{FO}$.*

If \mathbf{D} is strongly first order then it is first order in the sense of Definition 5: indeed, it is definable via the first order sentence equivalent to the $\mathbf{FO}(\mathbf{D})$ sentence $\forall \vec{w}(\neg R\vec{w} \vee (R\vec{w} \wedge \mathbf{D}\vec{w}))$. However, as mentioned at the end of the previous section, not all first order dependencies are strongly first order.

A notion closely related to strong first-orderness is that of *safety*:

Definition 7 (Safe Dependencies). *Let \mathbf{D} be a generalized dependency and let \mathcal{E} be a family of dependencies. Then \mathbf{D} is safe for \mathcal{E} if $\mathbf{FO}(\mathbf{D}, \mathcal{E}) \equiv \mathbf{FO}(\mathcal{E})$, that is, if every sentence of $\mathbf{FO}(\mathbf{D}, \mathcal{E})$ is equivalent to some sentence of $\mathbf{FO}(\mathcal{E})$.*

If \mathcal{D} and \mathcal{E} are families of dependencies, \mathcal{D} is safe for \mathcal{E} if $\mathbf{FO}(\mathcal{D}, \mathcal{E}) \equiv \mathbf{FO}(\mathcal{E})$.

Clearly, a dependency \mathbf{D} or a family of dependencies \mathcal{D} is strongly first order if and only if it is safe for \emptyset : in this sense, safety is a generalization of strong first-orderness. However, as shown in ([8], Theorem 53), strongly first order dependencies are not necessarily safe for all families of dependencies.

An example of strongly first order dependencies is given by the *constancy atoms* ([3], Corollary 3.13)

TS-Const: $\mathfrak{M} \models_X =(\vec{x})$ if and only if for all $s, s' \in X$ it holds that $s(\vec{x}) = s'(\vec{x})$.

A more general strongly first order family of generalized dependencies is given by first order *upwards closed dependencies*, that are strongly first order even taken together with each other and with the constancy atom:

Definition 8 (Upwards and Downwards Closed Dependencies). *A k -ary dependency \mathbf{D} is upwards closed if and only if $(M, R) \in \mathbf{D}$ implies $(M, R') \in \mathbf{D}$*

⁵ Ultimately, the existential second order character of Team Semantics derives from the existential second order character of Game-Theoretic Semantics, that defines truth in terms of the existence of winning strategies – that is to say, functions from positions to moves or sets of moves – in certain semantic games.

⁶ Strictly speaking we should write $\mathbf{FO}(\{\mathbf{D}\})$ instead of $\mathbf{FO}(\mathbf{D})$, $\mathbf{FO}(\{\mathbf{D}\} \cup \mathcal{E})$ instead of $\mathbf{FO}(\mathbf{D}, \mathcal{E})$ and so on, but this would clutter our notation too much.

for all k -ary relations $R' \supseteq R$ over M . We write \mathcal{UC} for the family of upwards closed, first order dependencies. Likewise, we write \mathcal{DC} for the family of all downwards closed first order dependencies, defined analogously.

\mathcal{DC} is not strongly first order: for example, functional dependencies $=(\vec{x}; \vec{y})$ are first order and downwards closed, but $\mathbf{FO}(=(\cdot; \cdot))$ is as expressive as Existential Second Order Logic ([25], Corollary 6.3). On the other hand,

Theorem 1 ([4], **Theorem 21**). $\mathcal{UC} \cup \{=(\cdot)\}$ is strongly first order.

An upwards closed, first order generalized dependency atom that will be of some use is the *totality atom* $\mathbf{All}(\vec{x})$, that says that \vec{x} takes all possible values:

TS-All: $\mathfrak{M} \models_X \mathbf{All}(\vec{x})$ if and only if $X(\vec{x}) = \mathbf{Dom}(\mathfrak{M})^{|\vec{x}|}$.

As shown in [8], totality is safe for any collection of dependencies:

Proposition 2 ([8], **Theorem 38**). $\mathbf{FO}(\mathbf{All}, \mathcal{D}) \equiv \mathbf{FO}(\mathcal{D})$ for any collection of dependencies \mathcal{D} .

Also of interest are the families \mathcal{D}_0 and \mathcal{D}_1 of 0-ary and unary first order dependencies. A 0-ary first order dependency atom $[\psi] = \{M : M \models \psi\}$ is nothing but a family of models over the empty signature characterized by some first order sentence ψ , and $\mathfrak{M} \models_X [\psi]$ if and only if $M \models \psi$ (that is to say, 0-ary dependencies do not “look” at the team X but only at the domain $\mathbf{Dom}(\mathfrak{M}) = M$). A unary dependency atom is instead a family of models (M, P) over the signature $\{P\}$, where P is a unary predicate, characterized by some first order sentence over the signature $\{P\}$. As shown in ([5], Proposition 9 and Theorems 9 and 10), both of these families are strongly first order.

The main result of [6] characterizes the strongly first order dependencies \mathbf{D} that are downwards closed, have the *empty team property* and are *relativizable*:

Definition 9 (Empty Team Property). A dependency \mathbf{D} has the empty team property if $(M, \emptyset) \in \mathbf{D}$ for all domains M .

Definition 10 (Relativization of a dependency; Relativizable Dependencies). Let P be a unary predicate and let \mathbf{D} be a k -ary generalized dependency. Then the relativization of \mathbf{D} to P is the k -ary atom $\mathbf{D}^{(P)}$, whose semantics - for models \mathfrak{M} whose signature contains the predicate P interpreted as $P^{\mathfrak{M}}$ - is

TS- $\mathbf{D}^{(P)}$: $\mathfrak{M} \models_X \mathbf{D}^{(P)}\vec{x}$ if and only if $(P^{\mathfrak{M}}, X(\vec{x})) \in \mathbf{D}$.

A dependency \mathbf{D} is *relativizable* if every sentence of $\mathbf{FO}(\mathbf{D}^{(P)})$, i.e. of First Order Logic with Team Semantics augmented with the above rule, is equivalent to some sentence of $\mathbf{FO}(\mathbf{D})$. Likewise, a family of dependencies \mathcal{D} is *relativizable* if $\mathbf{FO}(\mathcal{D}^{(P)}) \equiv \mathbf{FO}(\mathcal{D})$, where $\mathcal{D}^{(P)} = \{\mathbf{D}^{(P)} : \mathbf{D} \in \mathcal{D}\}$.

As discussed in [6], non-relativizable generalized dependencies exist.⁷ However, the vast majority of the dependencies considered in the context of Team Semantics so far (and all the ones whose corresponding logics have been studied in some

⁷ The existence of non-relativizable dependencies was first observed in (Barbero, personal communication).

depth) are relativizable, and most are even *universe-independent* in the sense of [17] (in brief, \mathbf{D} is universe-independent if the satisfaction of $\mathbf{D}\vec{x}$ in a team X and in a model \mathfrak{M} does not depend on the existence in \mathfrak{M} of elements that are not in $X(\vec{x})$). \mathbf{All} is not universe-independent, but it is still relativizable;⁸ and no strongly first order non-relativizable dependencies are currently known.

Theorem 2 ([6], **Theorem 4.5**). *Let \mathbf{D} be a downwards closed, relativizable generalized dependency with the empty team property. Then \mathbf{D} is strongly first order if and only if it is definable in $\mathbf{FO}(=(\cdot))$.*

The same notions of strong first-orderness and safety can also be applied to *connectives*. Three connectives of particular interest in Team Semantics are the *Global* (or *Boolean*) *Disjunction* $\phi_1 \sqcup \phi_2$, the *Possibility Operator* $\diamond\phi$ and the *Contradictory Negation* $\sim\phi$, corresponding to the rules

TS- \sqcup : $\mathfrak{M} \models_X \phi_1 \sqcup \phi_2$ if and only if $\mathfrak{M} \models_X \phi_1$ or $\mathfrak{M} \models_X \phi_2$;
TS- \diamond : $\mathfrak{M} \models_X \diamond\phi$ if and only if $\mathfrak{M} \models_Y \phi$ for some $Y \subseteq X$, $Y \neq \emptyset$;
TS- \sim : $\mathfrak{M} \models_X \sim\phi$ if and only if $\mathfrak{M} \not\models_X \phi$.

As shown in [7], global disjunction is not safe for arbitrary dependencies, but it is safe for strongly first order dependencies.

Proposition 3 ([7], **Proposition 14**). *If \mathcal{D} is a strongly first order family of dependencies then every sentence of $\mathbf{FO}(\sqcup, \mathcal{D})$ (i.e. of First Order Logic with Team Semantics, plus global disjunctions and the atoms in \mathcal{D}) is equivalent to some sentence of \mathbf{FO} .*

Instead \diamond is safe for any collection of dependencies, in the sense that

Proposition 4 ([8], **Corollary 42**). *Let \mathcal{D} be any family of generalized dependencies, not necessarily strongly first order. Then every sentence of $\mathbf{FO}(\diamond, \mathcal{D})$ (i.e. of First Order Logic with Team Semantics, plus the possibility operator \diamond and the atoms in \mathcal{D}) is equivalent to some sentence of $\mathbf{FO}(\mathcal{D})$.*

Differently from global disjunction and from the possibility operator, contradictory negation is extremely unsafe. Augmenting Dependence Logic $\mathbf{FO}(=(\cdot; \cdot))$ with contradictory negation yields *Team Logic* $\mathbf{FO}(\sim, =(\cdot; \cdot))$ [26], which is as expressive as Second Order Logic; and as observed in ([5], Corollary 2), even $\mathbf{FO}(\sim, =(\cdot))$ is already equivalent to full Second Order Logic. On the other hand, \sim is still “strongly first order”, in the sense that $\mathbf{FO}(\sim) \equiv \mathbf{FO}$: this is mentioned in [5] as a consequence of ([5], Theorem 4), but it may be verified more simply by observing that if ϕ is first order then – as a consequence of Proposition 1 – $\sim\phi$ is logically equivalent to $\diamond(\neg\phi)$, and then applying Proposition 4.

We end this section with two simple results that will be of some use:

Proposition 5. *Let \mathcal{D} be a strongly first order, relativizable collection of dependencies. Then every sentence of $\mathbf{FO}(\mathcal{D}, =(\cdot))$ is equivalent to some sentence of \mathbf{FO} , and so is every sentence of $\mathbf{FO}(\mathcal{D}^{(P)}, =(\cdot))$.*

⁸ Indeed, $\mathbf{All}^{(P)}(\vec{x})$ is equivalent to $(\bigwedge_{x \in \vec{x}} Px) \wedge \exists \vec{v} ((\bigvee_{v \in \vec{v}} \neg Pv \vee \vec{v} = \vec{x}) \wedge \mathbf{All}(\vec{v}))$.

Proof (Sketch). Given a sentence $\phi \in \mathbf{FO}(\mathcal{D}, =(\cdot))$ [respectively $\mathbf{FO}(\mathcal{D}^{(P)}, =(\cdot))$], replace every constancy atom $=(\vec{x})$ with $\vec{x} = \vec{d}_{\vec{x}}$, where $\vec{d}_{\vec{x}}$ is a tuple of constant symbols (each variable corresponding to a unique distinct constant). The result will be in $\mathbf{FO}(\mathcal{D})$ [respectively $\mathbf{FO}(\mathcal{D}^{(P)})$], and so it will be equivalent to some $\phi'(\vec{d}) \in \mathbf{FO}$, where \vec{d} contains all new constants; and then ϕ will be equivalent to $\exists \vec{w} \phi'(\vec{w})$ for some new tuple of variables \vec{w} .

Proposition 6. $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All}) \equiv \mathbf{FO}$.

Proof. By Propositions 3 and 2, it is enough to show that $\mathbf{FO}(\mathcal{D}_0, =(\cdot)) \equiv \mathbf{FO}$. This follows from Proposition 5, since 0-ary dependencies are strongly first order.

2.3 Relations Definable over the Empty Signature

Finally, in this work we will need a couple of simple results – whose proofs we omit – about the relations that are definable via First Order Logic formulas:

Definition 11. Let \mathfrak{M} be a first order model with domain M and let $\theta(\vec{x}, \vec{y})$ be a first order formula. Then a $|\vec{x}|$ -ary relation R over \mathfrak{M} is defined by θ if there is a tuple of elements $\vec{a} \in M^{|\vec{y}|}$ such that $R = \{\vec{m} \in M^{|\vec{x}|} : \mathfrak{M} \models \theta(\vec{m}, \vec{a})\}$.

Definition 12. A first order formula $\theta(\vec{x}, \vec{y})$ is said to fix the identity type of \vec{y} if, for every two variables $y_i, y_j \in \vec{y}$, $\models \forall \vec{x} \vec{y} (\theta(\vec{x}, \vec{y}) \rightarrow y_i = y_j)$ or $\models \forall \vec{x} \vec{y} (\theta(\vec{x}, \vec{y}) \rightarrow y_i \neq y_j)$.

Proposition 7. If a relation R over \mathfrak{M} is defined by a formula $\theta(\vec{x}, \vec{y})$, it is defined by some $\theta'(\vec{x}, \vec{y})$ over the same signature that fixes the identity type of \vec{y} .

Proposition 8. If two nonempty relations R and S over the same model \mathfrak{M} with domain $\mathbf{Dom}(\mathfrak{M}) = M$ are defined by the same formula $\theta(\vec{x}, \vec{y})$ over the empty signature and θ fixes the identity type of \vec{y} then there is a bijection $\mathfrak{h} : M \rightarrow M$ such that $\mathfrak{h}[R] = S$.

3 Doubly Strongly First Order Dependencies

We will now characterize the relativizable dependencies \mathbf{D} such that $\{\mathbf{D}, \sim \mathbf{D}\}$ is strongly first order, where $\sim \mathbf{D}$ is the *complement* of \mathbf{D} :

Definition 13 (Complement of a Dependency). Let \mathbf{D} be any generalized dependency. Then $\sim \mathbf{D}$ is the generalized dependency $\{(M, R) : (M, R) \notin \mathbf{D}\}$. If \mathcal{D} is a family of dependencies, we write $\sim \mathcal{D}$ for the family $\{\sim \mathbf{D} : \mathbf{D} \in \mathcal{D}\}$.

Definition 14 (Doubly Strongly First Order Dependencies). Let \mathbf{D} be a generalized dependency. Then \mathbf{D} is doubly strongly first order if and only if $\{\mathbf{D}, \sim \mathbf{D}\}$ is strongly first order. Likewise, a family \mathcal{D} of dependencies is doubly strongly first order if and only if $\mathcal{D} \cup \sim \mathcal{D}$ is strongly first order.

Clearly $\mathfrak{M} \models_X (\sim \mathbf{D})\vec{v}$ if and only if $\mathfrak{M} \not\models_X \mathbf{D}\vec{v}$ if and only if $\mathfrak{M} \models_X \sim(\mathbf{D}\vec{v})$. Also, if α is a first order literal, $\sim\alpha$ is equivalent to $\diamond(\neg\alpha)$:⁹ therefore, via Proposition 4, it can be shown that \mathcal{D} is doubly strongly first order if and only if $\mathbf{FO}(\sim_0, \mathcal{D}) \equiv \mathbf{FO}$, where $\mathbf{FO}(\sim_0, \mathcal{D})$ is the fragment of $\mathbf{FO}(\sim, \mathcal{D})$ in which the contradictory negation \sim only occurs in front of literals or dependency atoms.

Our characterization will be based on the following result about strongly first order, relativizable dependencies from [7]:

Definition 15 (\mathbf{D}_{\max}). *Let \mathbf{D} be any dependency. Then $\mathbf{D}_{\max} = \{(M, R) \in \mathbf{D} : \forall R' \supseteq R, (M, R') \notin \mathbf{D}\}$ is the dependency containing the maximal $(M, R) \in \mathbf{D}$.*

Theorem 3 ([7], **Theorem 23 and Proposition 22**). *Let \mathbf{D} be a strongly first order, relativizable dependency. Then there exists some first order sentence*

$$\mathbf{D}^m(R) = \bigvee_{i=1}^n \exists \vec{y} \forall \vec{x} (R\vec{x} \leftrightarrow \theta_i(\vec{x}, \vec{y})), \quad (1)$$

where each θ_i is a first order formula over the empty signature, such that, for all (M, R) , if $(M, R) \in \mathbf{D}_{\max}$ then $(M, R) \models \mathbf{D}^m(R)$. Also, for all $(M, R) \in \mathbf{D}$, R is contained in some R' such that $(M, R') \in \mathbf{D}_{\max}$.

We will now see that if \mathbf{D} and $\sim\mathbf{D}$ are both strongly first order, there can be no infinite ascending “stair” of relations satisfying alternatively \mathbf{D} and $\sim\mathbf{D}$:

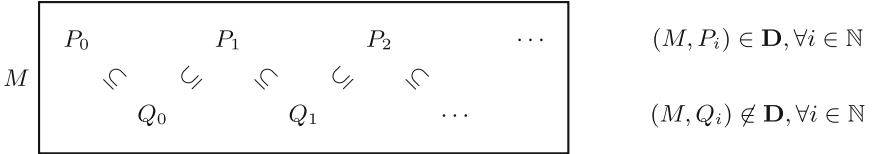


Fig. 1. If both \mathbf{D} and $\sim\mathbf{D}$ are (separately) strongly first order, this configuration of k -ary relations P_i and Q_i over some domain M is forbidden by Lemma 1.

Lemma 1. *Let \mathbf{D} be a k -ary dependency, and let $(P_i)_{i \in \mathbb{N}}$ and $(Q_i)_{i \in \mathbb{N}}$ be k -ary relations over the same domain M such that*

1. For all $i \in \mathbb{N}$, $(M, P_i) \in \mathbf{D}$;
2. For all $i \in \mathbb{N}$, $(M, Q_i) \in \sim\mathbf{D}$;
3. For all $i \in \mathbb{N}$, $P_i \subseteq Q_i \subseteq P_{i+1}$.

Then at least one between \mathbf{D} and $\sim\mathbf{D}$ is not strongly first order.

⁹ Indeed, $\mathfrak{M} \models_X \sim\alpha$ iff $\mathfrak{M} \not\models_X \alpha$ iff $\exists s \in X$ s.t. $\mathfrak{M} \models_{\{s\}} \neg\alpha$ iff $\mathfrak{M} \models_X \diamond\neg\alpha$.

Proof. Let \mathbf{D} and $\sim\mathbf{D}$ be strongly first order and suppose that relations P_i, Q_i as per our hypothesis exist over some domain M .

M is clearly infinite, since $P_0 \subsetneq P_1 \subsetneq P_2 \subsetneq \dots$; so by the Löwenheim-Skolem Theorem we can assume that M is countable and we can identify it with \mathbb{N} . Now let P and Q be the $(k+1)$ -ary relations over \mathbb{N} whose interpretations are $\{(\vec{m}, i) \in \mathbb{N}^{k+1} : \vec{m} \in P_i\}$ and $\{(\vec{m}, j) \in \mathbb{N}^{k+1} : \vec{m} \in Q_j\}$ respectively. Now consider the model $\mathfrak{M} = (\mathbb{N}, <, P, Q)$ where $<$ is the usual ordering over \mathbb{N} . I state that, if $\mathbf{FO}(\mathbf{D}) \equiv \mathbf{FO}(\sim\mathbf{D}) \equiv \mathbf{FO}$, \mathfrak{M} has no uncountable elementary extensions; but this is impossible due to the Löwenheim-Skolem Theorem.

In order to show that \mathfrak{M} has no uncountable (in fact, no non-standard) elementary extensions, consider the $\mathbf{FO}(\sim\mathbf{D})$ sentence

$$\exists i(i < d \wedge \forall \vec{w}(\neg P\vec{w}i \vee (P\vec{w}i \wedge (\sim\mathbf{D})\vec{w}))) \quad (2)$$

in the signature of \mathfrak{M} augmented by some new constant symbol d . Since $\sim\mathbf{D}$ is strongly first order, this sentence is equivalent to some first order sentence $\phi(d)$. I state that $\phi(d)$ is true if and only if there exists a nonempty set of indexes $I \subseteq M$ such that $i < d$ for all $i \in I$ and such that $(M, \bigcup_{i \in I} P_i) \notin \mathbf{D}$, where P_i is the relation $\{\vec{m} : (\vec{m}, i) \in P\}$. Indeed, if such a family of indexes exists, we can satisfy (2) by choosing the values of I as the values of the variable i ,¹⁰ then taking all possible values of \vec{w} for all chosen i , and then splitting the team by putting in the right disjunct all the assignments s for which $P\vec{w}i$ (that is, for which $s(\vec{w}) \in P_{s(i)}$); and conversely, if (2) can be satisfied, the values that the variable i can take will form a set I of indexes $< d$ such that $\bigcup_{i \in I} P_i$ does not satisfy \mathbf{D} .

Now, for the model \mathfrak{M} with domain \mathbb{N} described above no such family I may be found no matter the choice of d . Indeed, there will be only finitely many indexes less than d , and so if all elements of I are less than d then $\bigcup_{i \in I} P_i = P_{\max(I)}$, which satisfies \mathbf{D} . Hence, $\mathfrak{M} \models \neg \exists n \phi(n)$.

Similarly, the $\mathbf{FO}(\mathbf{D})$ sentence $\exists j(j < d \wedge \forall \vec{w}(\neg Q\vec{w}j \vee (Q\vec{w}j \wedge \mathbf{D}\vec{w})))$ is true if and only there exists a nonempty set J of indexes $< d$ such that $\bigcup_{j \in J} Q_j$ satisfies \mathbf{D} ; and as above, this sentence must be equivalent to some first order $\psi(d)$, because \mathbf{D} is strongly first order, and $\mathfrak{M} \models \neg \exists n \psi(n)$.

Now let \mathfrak{M}' be any elementary extension of \mathfrak{M} , and let d be any nonstandard element of it (that is, any element greater than all $n \in \mathbb{N}$). Then at least one between $\phi(d)$ and $\psi(d)$ will hold in \mathfrak{M}' . Indeed, in \mathfrak{M}' – like in \mathfrak{M} – we will have that $P_i \subseteq Q_i \subseteq P_{i+1}$ for all indexes $i \in \mathbb{N}$; and therefore, $\bigcup_{i \in \mathbb{N}} P_i = \bigcup_{i \in \mathbb{N}} Q_i$ and all indexes in \mathbb{N} are less than our element d . If this union satisfies \mathbf{D} , $\psi(d)$ holds; and if instead it does not satisfy \mathbf{D} , $\phi(d)$ holds. So $\mathfrak{M}' \models (\exists n \phi(n)) \vee (\exists n \psi(n))$ and \mathfrak{M}' is not an elementary extension of \mathfrak{M} , contradicting our premise. Thus, \mathfrak{M} cannot have elementary extensions with non-standard elements (and in particular it cannot have uncountable elementary extensions).

The next lemma can be verified by applying the rules of Team Semantics:

¹⁰ Note that Rule **TS- \exists** of Team Semantics allows for the selection of multiple values for the variable i , e.g. via the function $H : \{\emptyset\} \rightarrow \mathcal{P}(M) \setminus \{\emptyset\}$, $H(\emptyset) = I$.

Lemma 2. For all models \mathfrak{M} with domain M , teams X over M , and formulas $\theta(\vec{v}, \vec{y})$ over the empty signature with \vec{v} contained in the variables of X ,¹¹

$$\begin{aligned} (M, R := X(\vec{v})) \models \exists \vec{y} \forall \vec{x} (R\vec{x} \rightarrow \theta(\vec{x}, \vec{y})) &\Leftrightarrow \mathfrak{M} \models_X \exists \vec{y} (= (\vec{y}) \wedge \theta(\vec{v}, \vec{y})); \\ (M, R := X(\vec{v})) \not\models \exists \vec{y} \forall \vec{x} (R\vec{x} \rightarrow \theta(\vec{x}, \vec{y})) &\Leftrightarrow \mathfrak{M} \models_X \top \vee \exists \vec{y} (\mathbf{A}u(\vec{y}) \wedge \neg \theta(\vec{v}, \vec{y})). \end{aligned}$$

Lemma 3. Let \mathbf{D} be a k -ary strongly first order, relativizable dependency and let $\theta(\vec{x}, \vec{y})$ be a first order formula over the empty signature, where \vec{x} is a tuple of k distinct variables. Then $\mathbf{D}_\theta = \{(M', R') : (M', R') \in \mathbf{D}, (M', R') \models \exists \vec{y} \forall \vec{x} (R'\vec{x} \rightarrow \theta(\vec{x}, \vec{y}))\}$ is also strongly first order and relativizable.

Proof. Observe that by Lemma 2, $\mathbf{D}_\theta \vec{x}$ is logically equivalent to the $\mathbf{FO}(\mathbf{D}, =(\cdot))$ formula $\mathbf{D}\vec{x} \wedge \exists \vec{a} (= (\vec{a}) \wedge \theta(\vec{x}, \vec{a}))$. Therefore, every $\mathbf{FO}(\mathbf{D}_\theta)$ sentence is equivalent to some $\mathbf{FO}(\mathbf{D}, =(\cdot))$ sentence and hence – by Proposition 5 – to some \mathbf{FO} sentence. Therefore, \mathbf{D}_θ is strongly first order. To show that \mathbf{D}_θ is also relativizable, observe that its relativization to some unary predicate P can be defined in terms of constancy atoms and of the relativization of \mathbf{D} to P , since $\mathbf{D}_\theta^{(P)} \vec{x} \equiv \mathbf{D}^{(P)} \vec{x} \wedge \exists \vec{a} (= (\vec{a}) \wedge \bigwedge_{a \in \vec{a}} Pa \wedge \theta^{(P)}(\vec{x}, \vec{a}))$, where $\theta^{(P)}$ is the relativization (in the usual First Order Logic sense) of θ to the unary predicate P . Therefore, every $\mathbf{FO}(\mathbf{D}_\theta^{(P)})$ sentence is equivalent to some $\mathbf{FO}(\mathbf{D}^P, =(\cdot))$ sentence, and hence – again by Proposition 5 – to some first order sentence.

Proposition 9. Let \mathbf{D} be a first order, relativizable dependency such that both \mathbf{D} and $\sim \mathbf{D}$ are strongly first order and let $\mathfrak{M} = (M, R) \in \mathbf{D}$ for M countable. Then there exists a first order sentence $\eta_{\mathfrak{M}}$ of the form

$$\eta_{\mathfrak{M}} = \psi \wedge \bigwedge_{i=1}^n \exists \vec{y}_i (\forall \vec{x} (R\vec{x} \rightarrow \theta_i(\vec{x}, \vec{y}_i))) \wedge \bigwedge_{j=1}^{n'} \neg \exists \vec{z}_j (\forall \vec{x} (R\vec{x} \rightarrow \xi_j(\vec{x}, \vec{z}_j))) \quad (3)$$

where ψ is a first order sentence over the empty signature and all the θ_i and the ξ_j are first order formulas over the empty signature, such that $\mathfrak{M} \models \eta_{\mathfrak{M}}$ and $\eta_{\mathfrak{M}} \models \mathbf{D}(R)$.¹²

Proof. Let T be the theory

$$\begin{aligned} T = \{ \psi : M \models \psi \} \cup \{ \exists \vec{y} \forall \vec{x} (R\vec{x} \rightarrow \theta(\vec{x}, \vec{y})) : (M, R) \models \exists \vec{y} \forall \vec{x} (R\vec{x} \rightarrow \theta(\vec{x}, \vec{y})) \} \cup \\ \{ \neg \exists \vec{z} \forall \vec{x} (R\vec{x} \rightarrow \xi(\vec{x}, \vec{z})) : (M, R) \not\models \exists \vec{z} \forall \vec{x} (R\vec{x} \rightarrow \xi(\vec{x}, \vec{z})) \} \end{aligned}$$

where \vec{x} is a tuple of distinct variables such that $|\vec{x}|$ is the arity of \mathbf{D} , ψ ranges over all first order sentences over the empty signature, \vec{y} and \vec{z} range over tuples

¹¹ Here \top is the always-true first order literal and $\neg \theta(\vec{v}, \vec{y})$ stands for the corresponding first order formula in Negation Normal Form.

¹² Here $\mathbf{D}(R)$ is the first order sentence characterizing \mathbf{D} as per Definition 5: thus, every model of η_M with signature $\{R\}$ is in \mathbf{D} .

of distinct variables disjoint from \vec{x} of all finite lengths (including the empty tuple of variables, in which case their existential quantification is vacuous), and $\theta(\vec{x}, \vec{y})$ and $\xi(\vec{x}, \vec{z})$ range over first order formulas with free variables in $\vec{x}\vec{y}$ (respectively $\vec{x}\vec{z}$) over the empty signature. If we can show that $T \models \mathbf{D}(R)$, the conclusion follows: indeed, by compactness we can then find a finite theory $T_f \subseteq T$ such that $T_f \models \mathbf{D}(R)$, and then $\bigwedge T_f$ has the required form.

Suppose that this is not true, and let $\mathfrak{A} = (A, S)$ be some model such that $\mathfrak{A} \models T$ but $\mathfrak{A} \notin \mathbf{D}$. Since \mathfrak{A} and \mathfrak{M} satisfy the same sentences over the empty signature, we can assume that \mathfrak{A} and \mathfrak{M} have the same cardinality (finite or – via Löwenheim-Skolem – countably infinite) and therefore that, up to isomorphism, the domain A of \mathfrak{A} is the same as that M of \mathfrak{M} , i.e., $\mathfrak{A} = (M, S)$. Also, $R \neq \emptyset$: indeed, if R were empty then $\forall \vec{x}(R\vec{x} \rightarrow \perp)$ would be in T , and so since $(M, S) \models T$ the relation S would also be \emptyset , which is impossible since $(M, R) \in \mathbf{D}$ but $(M, S) \notin \mathbf{D}$. Therefore $\neg \forall \vec{x}(R\vec{x} \rightarrow \perp)$ is in T and $S \neq \emptyset$ too.

I aim to prove, by induction on n , that for every $n \in \mathbb{N}$ there exists in M a descending chain of relations $S_0^n \supseteq R_0^n \supseteq S_1^n \supseteq R_1^n \dots S_n^n \supseteq R_n^n \supseteq R$ such that

1. $(M, S_i^n) \in \sim \mathbf{D}$ and $(M, R_i^n) \in \mathbf{D}$ for all $i = 1 \dots n$;
2. Every R_i^n , for $0 \leq i \leq n$, is defined by some formula $\theta_i(\vec{x}, \vec{y})$ over the empty signature that fixes the identity type of \vec{y} and by some tuple of elements \vec{a}_i^n , in the sense that $R_i^n = \{\vec{m} : M \models \theta_i(\vec{m}, \vec{a}_i^n)\}$;
3. Every S_i^n , for $0 \leq i \leq n$, is defined by some formula $\xi_i(\vec{x}, \vec{z})$ over the empty signature that fixes the identity type of \vec{z} and by some tuple of elements \vec{b}_i^n , in the sense that $S_i^n = \{\vec{m} : M \models \xi_i(\vec{m}, \vec{b}_i^n)\}$.

Base Case: See Fig. 2. Since $(M, S) \notin \mathbf{D}$, $(M, S) \in \sim \mathbf{D}$; and therefore, by Theorem 3, there exists some $S' \supseteq S$ such that $(M, S') \in (\sim \mathbf{D})_{\max}$. Also by Theorem 3, S' is first order definable over the empty signature: $(M, S') \models \exists \vec{z} \forall \vec{x}(S'\vec{x} \leftrightarrow \xi_0(\vec{x}, \vec{z}))$, where by Proposition 7 we can assume that ξ_0 fixes the identity type of \vec{z} . Since $S \subseteq S'$, $(M, S) \models \exists \vec{z} \forall \vec{x}(S\vec{x} \rightarrow \xi_0(\vec{x}, \vec{z}))$; and since $(M, S) \models T$, $(M, R) \models \exists \vec{z} \forall \vec{x}(R\vec{x} \rightarrow \xi_0(\vec{x}, \vec{z}))$ too.

Now consider the dependency $\mathbf{E}_0 = \{(M', R') \in \mathbf{D} : (M', R') \models \exists \vec{z} \forall \vec{x}(R'\vec{x} \rightarrow \xi_0(\vec{x}, \vec{z}))\}$. By Lemma 3, \mathbf{E}_0 is strongly first order and relativizable, and $(M, R) \in \mathbf{E}_0$; therefore, by Theorem 3, there exists some $R_0^0 \supseteq R$ such that $(M, R_0^0) \in (\mathbf{E}_0)_{\max}$, and this R_0^0 is definable by some $\theta_0(\vec{x}, \vec{y})$ that fixes the identity type of \vec{y} and by some tuple $\vec{a}_0^0 \in M^{|\mathcal{Y}|}$, in the sense that $R_0^0 = \{\vec{m} : M \models \theta_0(\vec{m}, \vec{a}_0^0)\}$.

Since $(M, R_0^0) \in (\mathbf{E}_0)_{\max}$, $(M, R_0^0) \in \mathbf{E}_0$. Therefore, $(M, R_0^0) \in \mathbf{D}$, and there exists some tuple \vec{b}_0^0 in M such that $R_0^0 \subseteq S_0^0 = \{\vec{m} : M \models \xi_0(\vec{m}, \vec{b}_0^0)\}$. Now S_0^0 is nonempty, as it contains R_0^0 and hence R , and S' is nonempty, as it contains S , and they are both defined by the same formula $\xi_0(\vec{x}, \vec{z})$ that fixes the identity type of \vec{z} . Therefore, by Proposition 8, there exists a bijection $\mathfrak{h} : M \rightarrow M$ that maps S' into S_0^0 . This implies that (M, S_0^0) is isomorphic to (M, S') , and thus that $(M, S_0^0) \in \sim \mathbf{D}$ as required.

Induction Case: See Fig. 3. Suppose that a chain $S_0^n \supseteq R_0^n \dots S_n^n \supseteq R_n^n \supseteq R$ exists as per our hypothesis. Then, in particular, R_n^n is defined by some

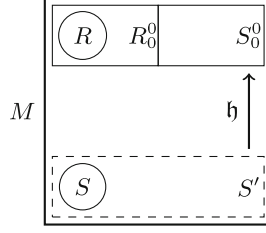


Fig. 2. The base case.

$\theta_n(\vec{x}, \vec{y})$ that fixes the identity type of \vec{y} ; and since $R \subseteq R_n^n$, $(M, R) \models \exists \vec{y} \forall \vec{x} (R\vec{x} \rightarrow \theta_n(\vec{x}, \vec{y}))$. But then, since $(M, S) \models T$, $(M, S) \models \exists \vec{y} \forall \vec{x} (S\vec{x} \rightarrow \theta_n(\vec{x}, \vec{y}))$ too. Now consider the dependency $\mathbf{F}_n = \{(M', R') \in \sim \mathbf{D} : (M', R') \models \exists \vec{y} \forall \vec{x} (R'\vec{x} \rightarrow \theta_n(\vec{x}, \vec{y}))\}$. By Lemma 3, \mathbf{F}_n is strongly first order and relativizable, and $(M, S) \in \mathbf{F}_n$; therefore, there exists some $S' \supseteq S$ such that $(M, S') \in (\mathbf{F}_n)_{\max}$, and this S' is defined by some $\xi_{n+1}(\vec{x}, \vec{z})$ over the empty signature that (by Proposition 7) fixes the identity type of \vec{z} . Therefore, since $S \subseteq S'$, $(M, S) \models \exists \vec{z} \forall \vec{x} (S\vec{x} \rightarrow \xi_{n+1}(\vec{x}, \vec{z}))$; and since $(M, S) \models T$, this implies that $(M, R) \models \exists \vec{z} \forall \vec{x} (R\vec{x} \rightarrow \xi_{n+1}(\vec{x}, \vec{z}))$ as well.

Consider the dependency $\mathbf{E}_{n+1} = \{(M', R') \in \mathbf{D} : (M', R') \models \exists \vec{z} \forall \vec{x} (R'\vec{x} \rightarrow \xi_{n+1}(\vec{x}, \vec{z}))\}$. Again, by Lemma 3, \mathbf{E}_{n+1} is strongly first order and relativizable, and $(M, R) \in \mathbf{E}_{n+1}$; therefore, there exists some $R_{n+1}^{n+1} \supseteq R$ such that $(M, R_{n+1}^{n+1}) \in (\mathbf{E}_{n+1})_{\max}$. This R_{n+1}^{n+1} will, again, be first order definable over the empty signature by some $\theta_{n+1}(\vec{x}, \vec{y})$ that fixes the identity type of \vec{y} and by some tuple \vec{a}_{n+1}^{n+1} . Furthermore, since $(M, R_{n+1}^{n+1}) \in \mathbf{E}_{n+1}$, it will be the case that $(M, R_{n+1}^{n+1}) \in \mathbf{D}$ and that there exists some \vec{b}_{n+1}^{n+1} such that $R_{n+1}^{n+1} \subseteq S_{n+1}^{n+1} = \{\vec{m} : M \models \xi_{n+1}(\vec{m}, \vec{b}_{n+1}^{n+1})\}$. Now since S_{n+1}^{n+1} and S' are defined by the same $\xi_{n+1}(\vec{x}, \vec{z})$ and are both nonempty, by Proposition 8 there exists some bijection $\mathfrak{g} : M \rightarrow M$ such that $\mathfrak{g}[S'] = S_{n+1}^{n+1}$. Therefore, $(M, S_{n+1}^{n+1}) \in \mathbf{F}_n$: thus, $(M, S_{n+1}^{n+1}) \in \sim \mathbf{D}$, and there exists some \vec{a}_n^{n+1} for which $S_{n+1}^{n+1} \subseteq R_n^{n+1} = \{\vec{m} : \theta_n(\vec{m}, \vec{a}_n^{n+1})\}$.

R_n^{n+1} and R_n^n are defined by the same formula $\theta_n(\vec{x}, \vec{y})$, which fixes the identity type of \vec{y} , and they are both nonempty since they both contain R . Thus by Proposition 8 there exists some bijection $\mathfrak{h} : M \rightarrow M$ that maps R_n^n into R_n^{n+1} . Then, for all $i = 0 \dots n$, let $R_i^{n+1} = \mathfrak{h}[R_i^n]$ and $S_i^{n+1} = \mathfrak{h}[S_i^n]$.

Then $S_0^{n+1} \supseteq R_0^{n+1} \supseteq S_1^{n+1} \supseteq \dots \supseteq S_n^{n+1} \supseteq R_n^{n+1} \supseteq S_{n+1}^{n+1} \supseteq R_{n+1}^{n+1} \supseteq R$, because \mathfrak{h} preserves inclusions. Additionally, $(M, S_i^{n+1}) \in \sim \mathbf{D}$ and $(M, R_i^{n+1}) \in \mathbf{D}$ for all $i = 0 \dots n + 1$, as required, since \mathbf{D} and $\sim \mathbf{D}$ are closed under isomorphisms, and for all $i \in 0 \dots n$ the S_i^{n+1} and R_i^{n+1} are still defined respectively by ξ_i and θ_i and by $\vec{a}_i^{n+1} = \mathfrak{h}(\vec{a}_i^n)$, $\vec{b}_i^{n+1} = \mathfrak{h}(\vec{b}_i^n)$, since $S_i^{n+1} = \mathfrak{h}[S_i^n] = \{\mathfrak{h}(\vec{m}) : M \models \xi_i(\vec{m}, \vec{b}_i^n)\} = \{\vec{m}' : M \models \xi_i(\vec{m}', \mathfrak{h}(\vec{b}_i^n))\}$ and $R_i^{n+1} = \mathfrak{h}[R_i^n] = \{\mathfrak{h}(\vec{m}) : M \models \theta_i(\vec{m}, \vec{a}_i^n)\} = \{\vec{m}' : M \models \theta_i(\vec{m}', \mathfrak{h}(\vec{a}_i^n))\}$.

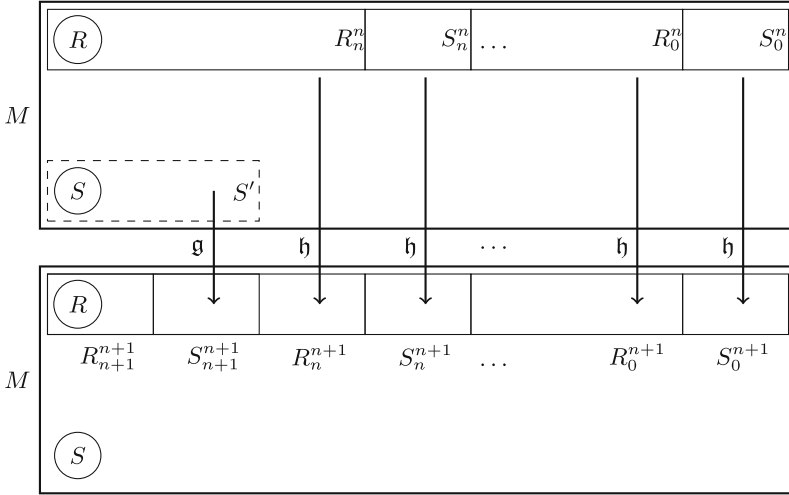


Fig. 3. The induction case. M is showed twice to avoid cluttering the figure too much.

Finally, consider the theory

$$U = \{\forall \vec{x}(P_i \vec{x} \rightarrow Q_i \vec{x}) \wedge (Q_i \vec{x} \rightarrow P_{i+1} \vec{x}) : i \in \mathbb{N}\} \cup \{\mathbf{D}(P_i), \neg \mathbf{D}(Q_i)\} : i \in \mathbb{N}\}$$

that states that there is an infinite *ascending* chain $P_0 \subseteq Q_0 \subseteq P_1 \subseteq Q_1 \subseteq \dots$ of relations satisfying alternatively \mathbf{D} and $\neg \mathbf{D}$ as per Fig. 1. U is finitely satisfiable: indeed, for any finite subset U_f of U , if n is the highest index for which P_n or Q_n appear in U_f , the model with domain M in which $P_0 \dots P_n$ are interpreted as $R_n^n \dots R_0^n$ (note the inverse order) and $Q_0 \dots Q_n$ are interpreted as $S_n^n \dots S_0^n$ (likewise in inverse order) satisfies U_f , since $P_i = R_{n-i}^n \subseteq S_{n-i}^n = Q_i$ and $Q_i = S_{n-i} \subseteq R_{n-i-1} = P_{i+1}$. Therefore, by compactness, U is satisfiable; and by Lemma 1, at least one between \mathbf{D} and $\sim \mathbf{D}$ is not strongly first order.

We can now prove the main result of this work:

Theorem 4. *Let \mathbf{D} be a relativizable first order dependency. Then the following are equivalent:*

i) $\mathbf{D}(R)$ is equivalent to some sentence of the form

$$\bigvee_{k=1}^l \left(\psi_k \wedge \bigwedge_{i=1}^{n_k} \exists \vec{y}_i^k (\forall \vec{x} (R \vec{x} \rightarrow \theta_i^k(\vec{x}, \vec{y}_i^k))) \wedge \bigwedge_{j=1}^{n'_k} \neg \exists \vec{z}_j^k (\forall \vec{x} (R \vec{x} \rightarrow \xi_j^k(\vec{x}, \vec{z}_j^k))) \right) \tag{4}$$

where all the ψ_k are first order sentences over the empty vocabulary and all the θ_i^k and the ξ_j^k are first order formulas over the empty vocabulary;

- ii) Both \mathbf{D} and $\sim \mathbf{D}$ are definable in $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$;
- iii) \mathbf{D} is doubly strongly first order.
- iv) Both \mathbf{D} and $\sim \mathbf{D}$ are (separately) strongly first order.

Proof.

i) \Rightarrow ii): Suppose that $\mathbf{D}(R)$ is in the form of Eq. (4) and, for each first order sentence ψ over the empty signature ψ , let $[\psi] \in \mathcal{D}_0$ be the 0-ary first order dependency defined as $[\psi] = \{M : M \models \psi\}$. Then $\mathbf{D}\vec{v}$ is equivalent to the $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$ formula

$$\bigsqcup_{k=1}^l \left([\psi_k] \wedge \bigwedge_{i=1}^{n_k} \exists \vec{y}_i^k (= (\vec{y}_i^k) \wedge \theta_i^k(\vec{v}, \vec{y}_i^k)) \wedge \bigwedge_{j=1}^{n'_k} (\top \vee \exists \vec{z}_j^k (\mathbf{All}(\vec{z}_j^k) \wedge \neg \xi_j^k(\vec{v}, \vec{z}_j^k))) \right)$$

where, up to renaming, we can assume that \vec{v} is disjoint from all the \vec{y}_i^k and \vec{z}_j^k and where each $\neg \xi_j^k$ stands for the corresponding expression in Negation Normal Form. Indeed, by Lemma 2 – as well as the rules **TS- \wedge** and **TS- \sqcup** for conjunction and global disjunction – the above expression is satisfied by a team X in a model \mathfrak{M} if and only if there exists some $k \in 1 \dots l$ such that

1. $M \models \psi_k$;
2. For all $i \in 1 \dots n_k$, $(M, X(\vec{v})) \models \exists \vec{y}_i^k \forall \vec{x} (R\vec{x} \rightarrow \theta_i^k(\vec{x}, \vec{y}_i^k))$;
3. For all $j \in 1 \dots n'_k$, $(M, X(\vec{v})) \not\models \exists \vec{z}_j^k \forall \vec{x} (R\vec{x} \rightarrow \xi_j^k(\vec{x}, \vec{z}_j^k))$.

These are precisely the conditions for Eq. (4) to be true in $(M, X(\vec{v}))$, that is, for it to be the case that $\mathfrak{M} \models_X \mathbf{D}\vec{v}$. Likewise, $\sim \mathbf{D}\vec{v}$ is equivalent to

$$\bigwedge_{k=1}^l \left([\neg \psi_k] \sqcup \bigsqcup_{i=1}^{n_k} (\top \vee \exists \vec{y}_i^k (\mathbf{All}(\vec{y}_i^k) \wedge \neg \theta_i^k(\vec{v}, \vec{y}_i^k))) \sqcup \bigsqcup_{j=1}^{n'_k} \exists \vec{z}_j^k (= (\vec{z}_j^k) \wedge \xi_j^k(\vec{v}, \vec{z}_j^k)) \right).$$

Therefore, both \mathbf{D} and $\sim \mathbf{D}$ are indeed definable in $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$.

ii) \Rightarrow iii) Since both \mathbf{D} and $\sim \mathbf{D}$ are definable in $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$, every sentence of $\mathbf{FO}(\mathbf{D}, \sim \mathbf{D})$ is equivalent to some sentence of $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$ and therefore – by Proposition 6 – to some sentence of \mathbf{FO} . Therefore, \mathbf{D} is doubly strongly first order.

iii) \Rightarrow iv) Obvious, because $\mathbf{FO}(\mathbf{D}), \mathbf{FO}(\sim \mathbf{D}) \subseteq \mathbf{FO}(\mathbf{D}, \sim \mathbf{D}) \equiv \mathbf{FO}$.

iv) \Rightarrow i) Suppose that both \mathbf{D} and $\sim \mathbf{D}$ are strongly first order. Then, by Proposition 9, for every countable $\mathfrak{M} = (M, R) \in \mathbf{D}$ there exists some first order sentence $\eta_{\mathfrak{M}}$ of the form of Eq. (3) such that $\mathfrak{M} \models \eta_{\mathfrak{M}}$ and that $\eta_{\mathfrak{M}} \models \mathbf{D}(R)$. Now consider the first order theory

$$T = \{\neg \eta_{\mathfrak{M}} : \mathfrak{M} \in \mathbf{D}, \mathfrak{M} \text{ is countable}\} \cup \{\mathbf{D}(R)\}$$

T is unsatisfiable: indeed, if it had a model then by the Löwenheim-Skolem Theorem it would have a countable model $\mathfrak{M} = (M, R)$, but this is impossible because we would have that $\mathfrak{M} \in \mathbf{D}$ (since $\mathbf{D}(R) \in T$) and thus $\mathfrak{M} \models \eta_{\mathfrak{M}}$, despite the fact that $\neg \eta_{\mathfrak{M}} \in T$. Therefore, T is finitely unsatisfiable and $\mathbf{D}(R) \models \bigvee_{k=1}^l \eta_{\mathfrak{M}_k}$ for some finite set $\mathfrak{M}_1 \dots \mathfrak{M}_l$ of countable models of \mathbf{D} . But each such $\eta_{\mathfrak{M}_k}$ entails $\mathbf{D}(R)$, and therefore $\mathbf{D}(R)$ is equivalent to $\bigvee_{k=1}^l \eta_{\mathfrak{M}_k}$ which is in the form of Eq. (4).

Corollary 1. *Let \mathcal{D} be a family of relativizable dependencies. Then \mathcal{D} is doubly strongly first order if and only if every $\mathbf{D} \in \mathcal{D}$ is doubly strongly first order.*

Proof. If \mathcal{D} is doubly strongly first order and $\mathbf{D} \in \mathcal{D}$, every sentence of $\mathbf{FO}(\mathbf{D}, \sim \mathbf{D})$ is a sentence of $\mathbf{FO}(\mathcal{D}, \sim \mathcal{D}) \equiv \mathbf{FO}$, and so \mathbf{D} is doubly strongly first order.

Conversely, suppose that every $\mathbf{D} \in \mathcal{D}$ is doubly strongly first order and relativizable. Then by Theorem 4, for every $\mathbf{D} \in \mathcal{D}$ both \mathbf{D} and $\sim \mathbf{D}$ are definable in $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$. Therefore, every sentence of $\mathbf{FO}(\mathcal{D}, \sim \mathcal{D})$ is equivalent to some sentence of $\mathbf{FO}(\mathcal{D}_0, \sqcup, =(\cdot), \mathbf{All})$, and thus – because of Proposition 6 – to some first order sentence. Therefore, \mathcal{D} is doubly strongly first order.

Acknowledgements. I thank the reviewers for their helpful comments and suggestions.

References

1. Abramsky, S., Väänänen, J.: From IF to BI. *Synthese* **167**(2), 207–230 (2009). <https://doi.org/10.1007/s11229-008-9415-6>
2. Engström, F.: Generalized quantifiers in dependence logic. *J. Logic Lang. Inform.* **21**(3), 299–324 (2012). <https://doi.org/10.1007/s10849-012-9162-4>
3. Galliani, P.: Inclusion and exclusion dependencies in team semantics: On some logics of imperfect information. *Ann. Pure Appl. Logic* **163**(1), 68–84 (2012). <https://doi.org/10.1016/j.apal.2011.08.005>
4. Galliani, P.: Upwards closed dependencies in team semantics. *Inf. Comput.* **245**, 124–135 (2015). <https://doi.org/10.1016/j.ic.2015.06.008>
5. Galliani, P.: On strongly first-order dependencies. In: Abramsky, S., Kontinen, J., Väänänen, J., Vollmer, H. (eds.) *Dependence Logic*, pp. 53–71. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31803-5_4
6. Galliani, P.: Characterizing downwards closed, strongly first-order, relativizable dependencies. *J. Symb. Log.* **84**(3), 1136–1167 (2019). <https://doi.org/10.1017/jsl.2019.12>
7. Galliani, P.: Characterizing strongly first order dependencies: the non-jumping relativizable case. In: *Electronic Proceedings in Theoretical Computer Science*, vol. 305, pp. 66–82 (2019)
8. Galliani, P.: Safe dependency atoms and possibility operators in team semantics. *Inf. Comput.* 104593 (2020)
9. Galliani, P.: Dependence logic. In: Zalta, E.N. (ed.) *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2021 edn. (2021)
10. Galliani, P., Hannula, M., Kontinen, J.: Hierarchies in independence logic. In: Rocca, S.R.D. (ed.) *Computer Science Logic 2013 (CSL 2013)*. Leibniz International Proceedings in Informatics (LIPIcs), vol. 23, pp. 263–280. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013). <https://doi.org/10.4230/LIPIcs.CSL.2013.263>
11. Galliani, P., Hella, L.: Inclusion logic and fixed point logic. In: Rocca, S.R.D. (ed.) *Computer Science Logic 2013 (CSL 2013)*. Leibniz International Proceedings in Informatics (LIPIcs), vol. 23, pp. 281–295. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2013). <https://doi.org/10.4230/LIPIcs.CSL.2013.281>
12. Grädel, E., Väänänen, J.: Dependence and independence. *Stud. Logica*. **101**(2), 399–410 (2013). <https://doi.org/10.1007/s11225-013-9479-2>

13. Hannula, M.: Hierarchies in inclusion logic with lax semantics. In: Banerjee, M., Krishna, S.N. (eds.) ICLA 2015. LNCS, vol. 8923, pp. 100–118. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-45824-2_7
14. Hintikka, J.: *The Principles of Mathematics Revisited*. Cambridge University Press, Cambridge (1996). <https://doi.org/10.1017/cbo9780511624919>
15. Hintikka, J., Sandu, G.: Informational independence as a semantic phenomenon. In: Fenstad, J., Frolov, I., Hilpinen, R. (eds.) *Logic, Methodology and Philosophy of Science*, pp. 571–589. Elsevier (1989). [https://doi.org/10.1016/S0049-237X\(08\)70066-1](https://doi.org/10.1016/S0049-237X(08)70066-1)
16. Hodges, W.: Compositional semantics for a language of imperfect information. *J. Interest Group Pure Appl. Logics* **5**(4), 539–563 (1997). <https://doi.org/10.1093/jigpal/5.4.539>
17. Kontinen, J., Kuusisto, A., Virtema, J.: Decidability of predicate logics with team semantics. In: 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016). *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 58, pp. 60:1–60:14 (2016). <https://doi.org/10.4230/LIPIcs.MFCS.2016.60>
18. Kontinen, J., Nurmi, V.: Team logic and second-order logic. *Fund. Inform.* **106**(2–4), 259–272 (2011)
19. Kontinen, J., Yang, F.: Logics for first-order team properties. In: Iemhoff, R., Moortgat, M., de Queiroz, R. (eds.) *WoLLIC 2019*. LNCS, vol. 11541, pp. 392–414. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-59533-6_24
20. Kuusisto, A.: A double team semantics for generalized quantifiers. *J. Logic Lang. Inform.* **24**(2), 149–191 (2015)
21. Lück, M.: On the complexity of team logic and its two-variable fragment. In: Potapov, I., Spirakis, P., Worrell, J. (eds.) 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018). *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 117, pp. 27:1–27:22. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). <https://doi.org/10.4230/LIPIcs.MFCS.2018.27>
22. Mann, A.L., Sandu, G., Sevenster, M.: *Independence-Friendly Logic: A Game-Theoretic Approach*. Cambridge University Press, Cambridge (2011). <https://doi.org/10.1017/CBO9780511981418>
23. Rönholm, R.: Capturing k-ary existential second order logic with k-ary inclusion-exclusion logic. *Ann. Pure Appl. Logic* **169**(3), 177–215 (2018). <https://doi.org/10.1016/j.apal.2017.10.005>
24. Rönholm, R.: The expressive power of k-ary exclusion logic. *Ann. Pure Appl. Logic* **170**(9), 1070–1099 (2019)
25. Väänänen, J.: *Dependence Logic*. Cambridge University Press, Cambridge (2007). <https://doi.org/10.1017/CBO9780511611193>
26. Väänänen, J.: Team logic. In: van Benthem, J., Gabbay, D., Löwe, B. (eds.) *Interactive Logic. Selected Papers from the 7th Augustus de Morgan Workshop*, pp. 281–302. Amsterdam University Press (2007)
27. Yang, F.: Expressing second-order sentences in intuitionistic dependence logic. In: Kontinen, J., Väänänen, J. (eds.) *Proceedings of Dependence and Independence in Logic, ESSLLI 2010*, pp. 118–132 (2010)



Explicit Non-normal Modal Logic

Atefeh Rohani and Thomas Studer^(✉)

Institute of Computer Science, University of Bern, Bern, Switzerland
{atefeh.rohani, thomas.studer}@inf.unibe.ch

Abstract. Faroldi argues that deontic modals are hyperintensional and thus traditional modal logic cannot provide an appropriate formalization of deontic situations. To overcome this issue, we introduce novel justification logics as hyperintensional analogues to non-normal modal logics. We establish soundness and completeness with respect to various models and we study the problem of realization.

Keywords: Justification logic · Non-normal modal logic · Deontic modals

1 Introduction

Justification logic [4, 16] is a variant of modal logic that replaces the implicit \Box -operator with explicit justifications. Instead of formulas $\Box A$, meaning, e.g., *A is known* or *A is obligatory*, the language of justification logic features formulas of the form $t : A$ that stand for *t justifies the agent's knowledge of A* or *A is obligatory for reason t*, where t is a so-called justification term.

The first justification logic, the Logic of Proofs [1], has been developed by Artemov in order to provide a classical provability semantics for the modal logic S4 (and thus also for intuitionistic logic) [1, 15]. Starting with the work of Fitting [10], several interpretations of justification logic have been presented that combine justifications with traditional possible world models [3, 14, 18]. This opened the door for numerous applications of justification logic, e.g., in epistemic and deontic contexts [2, 5, 13, 21, 23].

One of the features of a normal modal logic is that it is closed under the rule of necessitation, that is if F is valid, then so is $\Box F$. Hence we can easily derive the rule of monotonicity: Suppose $A \rightarrow B$ is valid. By necessitation, we get $\Box(A \rightarrow B)$. By axiom K and modus ponens we conclude $\Box A \rightarrow \Box B$.

Pacuit [20] mentions several interpretations of \Box for which the validities and rules of inference of normal modal logic can be questioned. A well-known example is the paradox of gentle murder [11], where \Box is read as *ought to*. Consider the statements:

This work was supported by the Swiss National Science Foundation grant 200020_184625.

Jones murders Smith. (1)

Jones ought not to murder Smith. (2)

If Jones murders Smith, then Jones ought to murder Smith gently. (3)

These sentences seem to be consistent. However, from (1) and (3) we infer

Jones ought to murder Smith gently. (4)

Moreover, we have the following implication

If Jones murders Smith gently, then Jones murders Smith. (5)

By the rule of monotonicity, (5) implies

If Jones ought to murder Smith gently,
then Jones ought to murder Smith. (6)

Now (4) and (6) together yield

Jones ought to murder Smith. (7)

This contradicts (2). This argument suggests that deontic modal logic should not validate the rules of normal modal logic and thus a semantics different from Kripke semantics is needed. The traditional approach for models of non-normal modal logics is to use neighborhood semantics. There, a so-called neighborhood function N assigns to each world w a set of sets of worlds $N(w)$ and a formula $\Box F$ is true at w if the truth set of F is an element of $N(w)$.

Justification logics are parametrized by a constant specification, which is a set

$$\text{CS} \subseteq \{(c, A) \mid c \text{ is a constant justification term and } A \text{ is an axiom of justification logic}\}.$$

Instead of the rule of necessitation, justification logics include a rule called *axiom necessitation* saying that one is allowed to infer $c : A$ if $(c, A) \in \text{CS}$. Hence, in epistemic settings, we can calibrate the reasoning power of the agents by adapting the constant specification.

Faroldi and Protopopescu [8, 9] suggest to use this mechanism also in deontic settings in order to avoid the usual paradoxes. For instance, they discuss Ross' paradox [22], which is:

You ought to mail the letter. (8)

implies

You ought to mail the letter or burn it. (9)

The reason is as before. It is a classical validity that

$$\textit{you mail the letter implies you mail the letter or burn it.} \quad (10)$$

By the monotonicity rule we find that (8) implies (9).

Fardoli and Protopopescu avoid this paradox by restricting the constant specification such that although (10) is a logical validity, there will no justification term for it. Thus the rule of monotonicity cannot be derived and there is no paradox.

One of the reasons why Faroldi prefers justification logic over using neighborhood models is that he claims that deontic modalities are *hyperintensional* [7], i.e. they can distinguish between logically equivalent formulas. This property is one of the distinguishing features of justification logics: they are hyperintensional by design. Even if A and B are logically equivalent, we may have that a term t justifying A does not justify B . Think of the Logic of Proofs, where the terms represent proofs in a formal system (like Peano arithmetic). Let A and B be logically equivalent formulas. In general, a proof of A will not also be a proof of B . In order to obtain a proof of B we have to extend the proof of A with a proof of $A \rightarrow B$ and an application of modus ponens. Thus in justification logic, terms do distinguish between equivalent formulas, which, according to Faroldi, makes it a suitable framework for deontic reasoning.

There is a problem with restricting the constant specification. Namely, the resulting constant specification will not be *axiomatically appropriate*, i.e. there will be axioms that are not justified by any term. This implies, however, that the Internalization property (saying that a justification logic internalizes its own notion of proof) does not hold, which is a problem for several reasons.

First, Internalization is needed to obtain completeness with respect to fully explanatory models. That is models where each formula that is obligatory (or believed) in the sense of the modal \Box operator has a justification.

Further, Internalization is often required to obtain completeness when a form of the D axiom is present [14, 18, 19]. In deontic settings, this is often the case since obligations are supposed not to contradict each other. Hence restricting the constant specification leads to deductive systems that are not complete. Conflicting obligations in justification logic have been studied in [6]. Recently, it turned out that this approach can also be used to analyze an epistemic paradox of quantum physics [24].

Moreover, Internalization is essential to obtain realization results. A justification logic realizes a modal logic if, given any theorem F of the modal logic, each occurrence of \Box in F can be replaced with some justification term such that the resulting formula is a theorem of the justification logic. Realization is an important property connecting implicit and explicit modalities.

In the present paper, we introduce two novel justification logics JE_{CS} and JEM_{CS} that are the explicit counterparts of the non-normal modal logics E and EM , respectively. As usual for justification logics, JE_{CS} and JEM_{CS} are hyperintensional and can therefore serve as appropriate formalization of deontic modals. On a technical level, the main novelty of our paper is the use of two types of

terms for JE_{CS} and JEM_{CS} . This makes it possible to formalize the characteristic principle of JE_{CS} and JEM_{CS} as an axiom (and not as a rule) and, therefore, our logics have the Internalization property. We show soundness and completeness of JE_{CS} and JEM_{CS} . For JE_{CS} we also prove completeness with respect to fully explanatory models whereas for JEM_{CS} this will only be a conjecture. Moreover, we show that the justification logic JEM_{CS} realizes the modal logic EM .

2 Justification Logic

To define the language of our novel justification logic JE_{CS} , we extend the usual language of justification logic by introducing two types of terms.

We consider two types of terms, *proof terms* and *justification terms*, where each type is built-up from countably many constants and variables. So if we denote proof constants by α_i and proof variables by ξ_i , the set of proof terms is defined inductively as follows:

$$\lambda ::= \alpha_i \mid \xi_i \mid (\lambda \cdot \lambda) \mid (\lambda + \lambda) \mid !\lambda .$$

Justification terms are built inductively as follows:

$$t ::= c_i \mid x_i \mid (t \cdot t) \mid (t + t) \mid \mathbf{e}(\lambda, \lambda, t) ,$$

where justification constants are shown by c_i , justification variables by x_i , and λ is a proof term. We denote the set of proof terms by PTm and the set of justification terms by JTm . Therefore, the set of all terms is $\text{Tm} := \text{PTm} \cup \text{JTm}$. We use λ, κ, γ for elements of PTm and r, s, t for elements of JTm .

Let Prop be a countable set of atomic propositions. Then formulas are inductively defined as follows:

$$F ::= P_i \mid \perp \mid (F \rightarrow F) \mid \lambda : F \mid [t]F ,$$

where $P_i \in \text{Prop}$, $\lambda \in \text{PTm}$, and $t \in \text{JTm}$. We use Fm for the set of formulas. The axioms of JE are:

$$\begin{array}{ll} \mathbf{j} & \lambda : (F \rightarrow G) \rightarrow (\kappa : F \rightarrow \lambda \cdot \kappa : G) \\ \mathbf{j}+_1 & (\lambda : F \vee \kappa : F) \rightarrow (\lambda + \kappa) : F \\ \mathbf{jt} & \lambda : F \rightarrow F \\ \mathbf{j4} & \lambda : F \rightarrow !\lambda : \lambda : F \\ \mathbf{j}+_2 & ([t]F \vee [s]F) \rightarrow [t + s]F \\ \mathbf{je} & (\lambda_1 : (F \rightarrow G) \wedge \lambda_2 : (G \rightarrow F)) \rightarrow ([t]F \rightarrow [\mathbf{e}(\lambda_1, \lambda_2, t)]G) \end{array}$$

In order to define the deductive system for our logic, we first need the notion of a constant specification.

Definition 1 (Constant Specification). *A constant specification CS is any subset:*

$$\text{CS} \subseteq \{ \alpha : A \mid \alpha \text{ is a proof constant and } A \text{ is an axiom of } \text{JE} \} .$$

A constant specification CS is called axiomatically appropriate if for each axiom A of JE there is a constant α with $(\alpha, A) \in \text{CS}$.

Definition 2 (Logic JE_{CS}). For a constant specification CS , the logic JE_{CS} is defined by a Hilbert-style system with the axioms **JE** and the inference rules *modus ponens* (**MP**) and *axiom necessitation* (**AN_{CS}**), given by:

$$\frac{}{\alpha : A} \text{ where } (\alpha, A) \in \text{CS} .$$

We write $\text{JE}_{\text{CS}} \vdash A$ to express that a formula A is provable in JE_{CS} . If the deductive system is clear from the context and we only want to stress the constant specification, we simply use $\vdash_{\text{CS}} A$. When the constant specification does not matter or is clear from the context, we drop the subscript CS and write $\vdash A$.

Note that the axioms **j**, **j₊₁**, **jt**, and **j4** are exactly the axioms of the Logic of Proofs. The fragment of JE_{CS} with only proof terms (but no justifications terms) actually is the Logic of Proofs.

It is a standard result that justification logics with an axiomatically appropriate constant specification internalize their own notion of proof [1, 4, 16].

Lemma 1 (Internalization). Let CS be an axiomatically appropriate constant specification. For any formula A with $\vdash A$, there exists a proof term λ such that $\vdash \lambda : A$.

Moreover, justification logics enjoy a deduction theorem [1, 4, 16].

Lemma 2 (Deduction). Let CS be an arbitrary constant specification. For any set Δ of formulas and for any formulas A and B ,

$$\Delta, A \vdash B \quad \text{iff} \quad \Delta \vdash A \rightarrow B .$$

Let us now turn to semantics. In order to present basic evaluations for JE_{CS} we need some operations on sets of formulas.

Definition 3. Let X, Y, Z be sets of formulas and λ be a proof term. We define the following operations:

$$\begin{aligned} \lambda : X &:= \{\lambda : F \mid F \in X\}; \\ X \cdot Y &:= \{F \mid G \rightarrow F \in X \text{ for some } G \in Y\}; \\ \odot_Y^X Z &:= \{G \mid F \rightarrow G \in X \text{ and } G \rightarrow F \in Y \text{ for some } F \in Z\} . \end{aligned}$$

Definition 4 (Basic evaluation). Let CS be an arbitrary constant specifications. A basic evaluation for JE_{CS} is a function ε that maps atomic propositions to 0 or 1

$$\varepsilon(P_i) \in \{0, 1\} \text{ for } P_i \in \text{Prop}$$

and maps terms to a set of formulas:

$$\varepsilon : \text{PTm} \cup \text{JTm} \rightarrow \mathcal{P}(\text{Fm}) ,$$

such that for arbitrary $\lambda, \kappa \in \text{PTm}$, and $s, t \in \text{JTm}$:

1. $\varepsilon(\lambda) \cdot \varepsilon(\kappa) \subseteq \varepsilon(\lambda \cdot \kappa)$;
2. $\varepsilon(\lambda) \cup \varepsilon(\kappa) \subseteq \varepsilon(\lambda + \kappa)$;

3. $F \in \varepsilon(\lambda)$ if $(\lambda, F) \in \text{CS}$;
4. $\lambda : \varepsilon(\lambda) \subseteq \varepsilon(!\lambda)$;
5. $\varepsilon(t) \cup \varepsilon(s) \subseteq \varepsilon(t + s)$;
6. $\bigodot_{\varepsilon(\lambda_2)}^{\varepsilon(\lambda_1)} \varepsilon(t) \subseteq \varepsilon(\mathbf{e}(\lambda_1, \lambda_2, t))$.

Definition 5 (Truth under a basic evaluation). We define truth of a formula F under a basic evaluation ε inductively as follows:

1. $\varepsilon \not\models \perp$;
2. $\varepsilon \models P$ iff $\varepsilon(P) = 1$ for $P \in \text{Prop}$;
3. $\varepsilon \models F \rightarrow G$ iff $\varepsilon \not\models F$ or $\varepsilon \models G$;
4. $\varepsilon \models \lambda : F$ iff $F \in \varepsilon(\lambda)$;
5. $\varepsilon \models [t]F$ iff $F \in \varepsilon(t)$.

Definition 6 (Factive basic evaluation). A basic evaluation ε is called *factive* if for any formula $\lambda : F$ we have $\varepsilon \models \lambda : F$ implies $\varepsilon \models F$.

Definition 7 (Basic model). Given an arbitrary CS, a basic model for JE_{CS} is a basic evaluation that is *factive*.

As expected, we have soundness and completeness with respect to basic models. The following theorem is established in Appendix A.

Theorem 1 (Soundness and completeness w.r.t. basic models). Let CS be an arbitrary constant specification. The logic JE_{CS} is sound and complete with respect to basic models. For any formula F ,

$$\text{JE}_{\text{CS}} \vdash F \quad \text{iff} \quad \varepsilon \models F \text{ for all basic models } \varepsilon \text{ for } \text{JE}_{\text{CS}} .$$

3 Neighborhood Semantics and Modular Models

The main purpose of modular models is to connect justification logic to traditional modal logic. To define modular models for JE_{CS} , we start with a neighborhood model (like for the modal logic E) and assign to each possible world a basic evaluation. This, however, is not enough since these basic evaluations may have nothing to do with the neighborhood structure of the model. Hence we introduce the following principle:

having a specific justification for F must yield

$$\Box F \text{ in the sense of the neighborhood structure.}$$

This principle was first introduced in epistemic contexts and is, therefore, called *justification yields belief* (JYB).

Definition 8 (Neighborhood function). For a non-empty set of worlds W , a neighborhood function is any $N : W \rightarrow \mathcal{P}(\mathcal{P}(W))$.

Definition 9 (Quasi-model). A quasi-model for JE_{CS} is a triple

$$\mathcal{M} = \langle W, N, \varepsilon \rangle$$

where W is a non-empty set of worlds, N is a neighborhood function and ε is an evaluation function that maps each world to a basic evaluation ε_w .

Definition 10 (Truth in quasi-model). Let $\mathcal{M} = \langle W, N, \varepsilon \rangle$ be a quasi-model. Truth of a formula at a world w in a quasi-model is defined inductively as follows:

1. $\mathcal{M}, w \not\models \perp$;
2. $\mathcal{M}, w \models P$ iff $\varepsilon_w(P) = 1$, for $P \in \text{Prop}$;
3. $\mathcal{M}, w \models F \rightarrow G$ iff $\mathcal{M}, w \not\models F$ or $\mathcal{M}, w \models G$;
4. $\mathcal{M}, w \models \lambda : F$ iff $F \in \varepsilon_w(\lambda)$;
5. $\mathcal{M}, w \models [t]F$ iff $F \in \varepsilon_w(t)$.

We will write $\mathcal{M} \models F$ if $\mathcal{M}, w \models F$ for all $w \in W$.

Remark 1. The neighborhood function plays no rule in the definition of truth in quasi-models. Hence truth in quasi-models is local. Let $\mathcal{M} = \langle W, N, \varepsilon \rangle$ be a quasi-model. For any $w \in W$ and any formula F ,

$$\mathcal{M}, w \models F \quad \text{iff} \quad \varepsilon_w \models F . \quad (11)$$

Definition 11 (Factive quasi-model). A quasi-model $\mathcal{M} = \langle W, N, \varepsilon \rangle$ is factive if for each world w , we have that for any formula $\lambda : F$,

$$\mathcal{M}, w \models \lambda : F \quad \text{implies} \quad \mathcal{M}, w \models F .$$

Definition 12 (Truth set). Let $\mathcal{M} = \langle W, N, \varepsilon \rangle$ be a quasi-model. The truth set of a formula F , denoted by $|F|^{\mathcal{M}}$, is the set of all worlds in which F is true, i.e.,

$$|F|^{\mathcal{M}} := \{ w \in W \mid \mathcal{M}, w \models F \} .$$

Further, we define

$$\square_w := \{ F \mid |F|^{\mathcal{M}} \in N(w) \} .$$

Looking back at neighborhood models for E , it is easy to see that $F \in \square_w$ means (modulo the different language that we are using) that $\square F$ holds at world w . As a result, we can formulate principle of justification yields belief as follows:

$$\text{for any } t \in \text{JTm} \text{ and } w \in W, \text{ we have that } \varepsilon_w(t) \subseteq \square_w . \quad (\text{JYB})$$

Definition 13 (Modular model). A JE_{CS} modular model is a quasi-model for JE_{CS} that is factive and satisfies (JYB).

JE_{CS} is sound and complete with respect to modular models. A proof of the following theorem is given in Appendix B.

Theorem 2 (Soundness and completeness w.r.t. modular models). *Let CS be an arbitrary constant specification. For each formula F we have*

$$\text{JE}_{\text{CS}} \vdash F \quad \text{iff} \quad \mathcal{M} \Vdash F \text{ for all } \text{JE}_{\text{CS}} \text{ modular models } \mathcal{M}.$$

It is natural to ask whether every obligatory formula in a modular model is justified by a justification term.

Definition 14 (Fully explanatory modular model). *A JE_{CS} modular model $\mathcal{M} = \langle W, N, \varepsilon \rangle$ is fully explanatory if for any $w \in W$ and any formula F ,*

$$|F|^{\mathcal{M}} \in N(w) \quad \text{implies} \quad F \in \varepsilon_w(t) \text{ for some } t \in \text{JTm} .$$

The fully explanatory property can be seen as the converse of justification yields belief. In fully explanatory models we have that for each world w ,

$$\bigcup_{t \in \text{JTm}} \varepsilon_w(t) = \Box_w .$$

For axiomatically appropriate constant specifications, we can show that JE_{CS} is sound and complete with respect to fully explanatory JE_{CS} modular models. The proof is presented in Appendix C.

Theorem 3 (Soundness and completeness for fully explanatory modular models). *Let CS be an axiomatically appropriate constant specification. JE_{CS} is sound and complete with respect to fully explanatory JE_{CS} modular models.*

4 Monotonic Justification Logic

There are several applications for which the modal logic E is too weak and one considers the extension of E with the axiom $\Box(A \wedge B) \rightarrow (\Box A \wedge \Box B)$ or, equivalently, with the rule

$$\frac{A \rightarrow B}{\Box A \rightarrow \Box B} .$$

The resulting logic is called EM. In this section we introduce an explicit counterpart JEM of the modal logic EM.

First, we extend the language as follows. If λ is a proof term and t is a justification term, then $m(\lambda, t)$ is a justification term, too. Formulas are then built using this extended set of justification terms. It will always be clear from the context whether we work with the basic language for JE or with the extended language for JEM.

The axioms of JEM are the axioms of JE extended by

$$\mathbf{jm} \quad \lambda : (F \rightarrow G) \rightarrow ([t]F \rightarrow [m(\lambda, t)]G).$$

For a constant specification CS , we now consider axioms of JEM ; and the system JEM_{CS} consists of the axioms of JEM plus the rules of modus ponens and axiom necessitation. Note that Internalization and the Deduction theorem hold for JEM_{CS} , too.

A *basic evaluation* for JEM_{CS} is defined like a basic evaluation for JE_{CS} with the additional requirement that for arbitrary $\lambda \in \text{PTm}$ and $t \in \text{Jm}$,

$$\varepsilon(\lambda) \cdot \varepsilon(t) \subseteq \varepsilon(\mathbf{m}(\lambda, t)) .$$

Further we define a *monotonic basic model* (for JEM_{CS}) as a basic evaluation for JEM_{CS} that is factive.

Similar to JE_{CS} , we can show that JEM_{CS} is sound and complete with respect to monotonic basic models.

Theorem 4. *Let CS be an arbitrary constant specification. The logic JEM_{CS} is sound and complete with respect to monotonic basic models. For any formula F ,*

$$\text{JEM}_{\text{CS}} \vdash F \quad \text{iff} \quad \varepsilon \Vdash F \text{ for all monotonic basic models } \varepsilon \text{ for } \text{JEM}_{\text{CS}} .$$

Now we are going to adapt modular models to JEM_{CS} . A neighborhood function N for a non-empty set of worlds W is called *monotonic* provided that for each $w \in W$ and for each $X \subseteq W$,

$$\text{if } X \in N(w) \text{ and } X \subseteq Y \subseteq W \text{ then } Y \in N(w).$$

A *monotonic quasi-model* for JEM_{CS} is defined like a quasi-model for JE_{CS} but we use a monotonic neighborhood function and each world is mapped to a basic evaluation for JEM_{CS} . A *monotonic modular model* is then defined like a modular model but the underlying quasi-model is required to be monotonic. As for JE_{CS} we get completeness of JEM_{CS} with respect to monotonic modular models.

Theorem 5. *Let CS be an arbitrary constant specification. For each formula F we have*

$$\text{JEM}_{\text{CS}} \vdash F \quad \text{iff} \quad \mathcal{M} \Vdash F \text{ for all } \text{JEM}_{\text{CS}} \text{ monotonic modular models } \mathcal{M}.$$

We do not yet have completeness with respect to fully explanatory monotonic modular models. To achieve this, one needs some additional construction to guarantee that the neighborhood function constructed in the canonical model is monotonic. We conjecture that a construction like in the completeness proof for EM [20] should provide this.

Conjecture 1. Let CS be an axiomatically appropriate constant specification. JEM_{CS} is sound and complete with respect to fully explanatory JEM_{CS} monotonic modular models.

5 Realisation

This section is concerned with the exact relationship between some non-normal modal logic \mathbf{M} and its explicit counterpart \mathbf{J} . Let $\mathbf{Fm}^{\mathbf{M}}$ denote the set of formulas from modal logic and $\mathbf{Fm}^{\mathbf{J}}$ the set of all \mathbf{Fm} -formulas that do not contain proof terms. There is the so-called forgetful translation $^\circ$ from $\mathbf{Fm}^{\mathbf{J}}$ to $\mathbf{Fm}^{\mathbf{M}}$ given by

$$\perp^\circ := \perp \quad P^\circ := P \quad (A \rightarrow B)^\circ := A^\circ \rightarrow B^\circ \quad ([t]A)^\circ := \Box A^\circ .$$

However, we are mainly interested in the converse direction. A *realization* is a mapping from $\mathbf{Fm}^{\mathbf{M}}$ to $\mathbf{Fm}^{\mathbf{J}}$ such that for all $A \in \mathbf{Fm}^{\mathbf{M}}$, we have $(r(A))^\circ = A$.

Now the question is whether a realization theorem holds, i.e. given a modal logic \mathbf{M} and a justification logic \mathbf{J} , does there exist a realization r such that for all $A \in \mathbf{Fm}^{\mathbf{M}}$, we have $\mathbf{M} \vdash A$ implies $\mathbf{J} \vdash r(A)$.

In order to establish such a realization theorem, we need the notion of a schematic constant specification.

Definition 15. *A constant specification \mathbf{CS} is called schematic if it satisfies the following property: for each constant c , the set of axioms $\{A \mid (c, A) \in \mathbf{CS}\}$ consists of all instances one or several (possibly zero) axioms schemes of the justification logic.*

Schematic constant specifications are important in the context of substitutions, where a substitution replaces atomic propositions with formulas, proof variables with proof terms, and justification variables with justification terms. The following lemma is standard [16].

Lemma 3. *Let \mathbf{CS} be a schematic constant specification. We have for any set of formulas Δ , any formula A , and any substitution σ*

$$\Delta \vdash A \text{ implies } \Delta\sigma \vdash A\sigma .$$

In order to show a realization result, we further need a cut-free sequent calculus for the given modal logic. The system \mathbf{GE} is given by the following propositional axioms and rules, the structural rules, and the rule **(RE)**. If we replace **(RE)** with **(RM)**, we obtain the system \mathbf{GM} . In these systems, a *sequent* is an expression of the form $\Gamma \supset \Delta$ where Γ and Δ are finite multisets of formulas.

Propositional axioms and rules:

$$\begin{array}{c} P \supset P \\ \hline \Gamma \supset \Delta, A \quad B, \Gamma \supset \Delta \\ \hline A \rightarrow B, \Gamma \supset \Delta \quad (\rightarrow\supset) \end{array} \qquad \begin{array}{c} \perp \supset \\ \hline A, \Gamma \supset \Delta, B \\ \hline \Gamma \supset \Delta, A \rightarrow B \quad (\supset\rightarrow) \end{array}$$

Structural rules:

$$\frac{\Gamma \supset \Delta}{A, \Gamma \supset \Delta} (w \supset) \qquad \frac{\Gamma \supset \Delta}{\Gamma \supset \Delta, A} (\supset w)$$

$$\frac{A, A, \Gamma \supset \Delta}{A, \Gamma \supset \Delta} (c \supset) \qquad \frac{\Gamma \supset \Delta, A, A}{\Gamma \supset \Delta, A} (\supset c)$$

Modal rules:

$$\frac{A \supset B \quad B \supset A}{\Box A \supset \Box B} (\mathbf{RE}) \qquad \frac{A \supset B}{\Box A \supset \Box B} (\mathbf{RM})$$

The systems **GE** and **GM** are sound and complete [12, 17].

Theorem 6. *For each modal logic formula A , we have*

1. $\mathbf{GE} \vdash \supset A$ iff $\mathbf{E} \vdash A$;
2. $\mathbf{GM} \vdash \supset A$ iff $\mathbf{EM} \vdash A$.

We need some technical notions about occurrence of \Box operators. We assign a *positive or negative polarity* to each sub-formula occurrence within a fixed formula A as follows:

1. To the only occurrence of A in A we assign the positive polarity.
2. If a polarity is assigned to a sub-formula of the form $B \rightarrow C$ in A , then the same polarity is assigned to C and opposite polarity is assigned to B .
3. If a polarity is already assigned to a sub-formula of the form $\Box B$ in A , then the same polarity is assigned to B .

Let $\Box B$ be a sub-formula of A . If $A \in \Delta$ in a sequent $\Gamma \supset \Delta$, then the \Box -operator of $\Box B$ has the same *polarity* as the sub-formula occurrence of $\Box B$ in A . If $A \in \Gamma$ in a sequent $\Gamma \supset \Delta$, then the \Box -operator of $\Box B$ has the opposite *polarity* as the sub-formula occurrence of $\Box B$ in A .

Remark 2. All rules of **GM** respect the polarities of \Box -operators. The rule **(RM)** introduces negative \Box -occurrence to the left side, and positive \Box -occurrence to the right side of the conclusion.

In the following we only consider the system **GM**. Let \mathcal{D} be a derivation in **GM**. We say that occurrences of \Box in \mathcal{D} are *related* if they occur in the same position in related formulas of premises and conclusions of a rule instance in \mathcal{D} . We close this relationship of related occurrences under transitivity.

All occurrences of \Box in \mathcal{D} naturally split into disjoint *families* of related \Box -occurrences. We call such a family *essential* if at least one of its members is a positive \Box -occurrence introduced by an instance of **(RM)**.

Now we are ready to formulate and prove the realization theorem.

Definition 16 (Normal realization). *A realization is called normal if all negative occurrences of \Box are realized by distinct justification variables.*

Theorem 7 (Constructive realization). *For any axiomatically appropriate and schematic constant specifications CS, there exist a normal realization r such that for each formula $A \in \text{Fm}^M$, we have*

$$\text{GM} \vdash \supset A \quad \text{implies} \quad \text{JEM}_{\text{CS}} \vdash r(A) .$$

For space limitations, we cannot present the full proof of the realization theorem. The essence is the same as in the proof of the constructive realization theorem for the Logic of Proofs [1, 16].

Let \mathcal{D} be the GM-proof of $\supset A$. The realization r is constructed by the following algorithm. We reserve a large enough set of justification variables as *provisional variables*.

1. For each non-essential family of \Box -occurrences, replace all occurrences of \Box by $[x]$ such that each family has a distinct justification variable.
2. For an essential family of \Box -occurrences, enumerate all occurrences of (RM) rules that introduce a \Box -operator to this family. Let n be the number of such occurrences. Replace each \Box -occurrence of this family with $[v_1 + \dots + v_n]$ where each v_i is a fresh provisional variable. Applying this step for all essential families yields a derivation tree \mathcal{D}' labeled by Fm^J -formulas.
3. Replace all provisional justification variables in \mathcal{D}' from the leaves toward the root. By induction on the depth of a node in \mathcal{D}' , we show that after each replacement, the resulting sequent of this step is derivable in JEM_{CS} where for finite multisets Γ and Δ of Fm^J -formulas, derivability of $\Gamma \supset \Delta$ means $\Gamma \vdash_{\text{CS}} \bigvee \Delta$.

Let us show the case of an instance of (RM) with number i in an essential family. The corresponding node in \mathcal{D}' is labelled by

$$\frac{A \supset B}{[x]A \supset [v_1 + \dots + v_i + \dots + v_n]B} \quad (\text{RM})$$

where the v 's are justification terms and v_i is a justification variable. By I.H. we get $A \vdash_{\text{CS}} B$. By the Deduction Theorem we get $\vdash_{\text{CS}} A \rightarrow B$ and Internalization yields a proof term λ with $\vdash_{\text{CS}} \lambda : (A \rightarrow B)$. By **jm** we get $\vdash_{\text{CS}} [x]A \rightarrow [m(\lambda, x)]B$. Hence, again by the Deduction Theorem, we find $[x]A \vdash_{\text{CS}} [m(\lambda, x)]B$ and thus $[x]A \vdash_{\text{CS}} [v_1 + \dots + m(\lambda, x) + \dots + v_n]B$ by axiom **j**₊₂. Substitute $m(\lambda, x)$ for v_i everywhere in \mathcal{D}' . By Lemma 3 this does not affect the already established derivability results since CS is schematic.

6 Conclusion

We have presented two new justification logics JE_{CS} and JEM_{CS} as explicit counterparts of the non-normal modal logics E and EM, respectively. Having a justification analogue of the modal logic E is particularly important in deontic contexts since, according to Faroldi [7], deontic modalities are hyperintensional. On a technical level, the main novelty in our work is the introduction of two types of terms. This facilitates the formulation of axiom **je**, which corresponds

to the rule of equivalence. Having this principle as an axiom (and not as a rule) in justification logic is important to obtain Internalization (Lemma 1).

We have established soundness and completeness of JE_{CS} and JEM_{CS} with respect to basic models and with respect to modular models. For JE_{CS} we have also proved completeness with respect to fully explanatory modular models whereas for JEM_{CS} this is only a conjecture.

Moreover, we have shown that for an axiomatically appropriate and schematic constant specification, the justification logic JEM_{CS} realizes the modal logic EM.

It is an open question whether JE_{CS} realizes E. The proof idea of Theorem 7 cannot be applied to E. The reason is that this proof relies on the fact that the rules of GM respect the polarities of \Box occurrences, see Remark 2. The rule (RE), however, does not satisfy this property. In the right premise, the formulas A and B have the opposite polarity of A and B , respectively, in the conclusion. We conjecture that additional axioms are needed in JE_{CS} to make a realization result possible.

A Soundness and Completeness with Respect to Basic Models

Theorem 8 (Soundness w.r.t. basic models). *The Logic JE_{CS} is sound with respect to basic models. For an arbitrary constant specifications CS and any formula F ,*

$$\text{JE}_{\text{CS}} \vdash F \implies \varepsilon \Vdash F \text{ for any basic model } \varepsilon .$$

Proof. As usual, the proof is by induction on the length of JE_{CS} derivations and a case distinction on the last rule. The only interesting case is when F is an instance of **je**. Suppose

$$\varepsilon \Vdash \lambda_1 : (A \rightarrow B) \quad \text{and} \quad \varepsilon \Vdash \lambda_2 : (B \rightarrow A) \quad \text{and} \quad \varepsilon \Vdash [t]A .$$

Thus we have

$$(A \rightarrow B) \in \varepsilon(\lambda_1) \quad \text{and} \quad (B \rightarrow A) \in \varepsilon(\lambda_2) \quad \text{and} \quad A \in \varepsilon(t) .$$

By Definition 3 we find $B \in \bigodot_{\varepsilon(\lambda_2)}^{\varepsilon(\lambda_1)} \varepsilon(t)$. Hence, by the definition of basic model we get $B \in \varepsilon(\mathbf{e}(\lambda_1, \lambda_2, t))$, which is $\varepsilon \Vdash [\mathbf{e}(\lambda_1, \lambda_2, t)]B$.

To prove the completeness theorem, we need to know that JE_{CS} is consistent.

Lemma 4. *For any constant specification CS, JE_{CS} is consistent.*

Proof. As usual, one can show that JE_{CS} is a conservative extension of classical propositional logic. This immediately yields consistency of JE_{CS} .

Definition 17. *A set of formulas Γ is called JE_{CS} -consistent if for each finite subset $\Sigma \subseteq \Gamma$, we have $\not\vdash_{\text{CS}} \bigwedge \Sigma \rightarrow \perp$. The set Γ is maximal JE_{CS} -consistent if Γ is consistent and none of its proper supersets is.*

As usual, any consistent set can be extended to a maximal consistent set.

Lemma 5 (Lindenbaum). *For each JE_{CS} -consistent set Δ , there exists a maximal JE_{CS} -consistent set $\Gamma \supseteq \Delta$.*

Lemma 6. *For any constant specification CS and any maximal JE_{CS} -consistent set Γ , there is a canonical basic model ε^c induced by Γ that is defined as follows:*

$$\begin{aligned} \varepsilon^c(P) &:= 1, \text{ if } P \in \Gamma \text{ and } \varepsilon^c(P) := 0, \text{ if } P \notin \Gamma; \\ \varepsilon^c(\lambda) &:= \{F \mid \lambda : F \in \Gamma\}; \\ \varepsilon^c(t) &:= \{F \mid [t]F \in \Gamma\}. \end{aligned}$$

Proof. First we have to establish that ε^c is a basic evaluation. We only show the condition

$$\bigcirc_{\varepsilon^c(\lambda_2)}^{\varepsilon^c(\lambda_1)} \varepsilon^c(t) \subseteq \varepsilon^c(\mathfrak{e}(\lambda_1, \lambda_2, t)). \quad (12)$$

Suppose $B \in \bigcirc_{\varepsilon^c(\lambda_2)}^{\varepsilon^c(\lambda_1)} \varepsilon^c(t)$, which means there is a formula $A \in \varepsilon^c(t)$ with $(A \rightarrow B) \in \varepsilon^c(\lambda_1)$ and $(B \rightarrow A) \in \varepsilon^c(\lambda_2)$. By the definition of ε^c , we have

$$\lambda_1 : (A \rightarrow B) \in \Gamma \quad \text{and} \quad \lambda_2 : (B \rightarrow A) \in \Gamma \quad \text{and} \quad [t]B \in \Gamma.$$

Since Γ is a maximal consistent set and

$$(\lambda_1 : (A \rightarrow B) \wedge \lambda_2 : (B \rightarrow A)) \rightarrow ([t]A \rightarrow [e(\lambda_1, \lambda_2, t)]B)$$

is an instance of **je**, we obtain $[e(\lambda_1, \lambda_2, t)]B \in \Gamma$. This yields $B \in \varepsilon^c(e(\lambda_1, \lambda_2, t))$ and (12) is established.

Next, a truth lemma can be established as usual by induction on formula complexity. For all formulas F ,

$$F \in \Gamma \quad \text{iff} \quad \varepsilon^c \Vdash F. \quad (13)$$

Finally, we show that our basic evaluation ε^c is factive and hence a basic model. Suppose $\varepsilon^c \Vdash \lambda : F$. Hence $\lambda : F \in \Gamma$. Since Γ is maximal consistent, we get by axiom **jt** that $F \in \Gamma$. By (13) we conclude $\varepsilon^c \Vdash F$.

Using the Lindenbaum lemma, the canonical basic model and the established truth lemma (13), we immediately get the following completeness result.

Theorem 9 (Completeness w.r.t. basic models). *Let CS be an arbitrary constant specification. The logic JE_{CS} is complete with respect to basic models. For any formula F ,*

$$\text{JE}_{\text{CS}} \vdash F \quad \text{iff} \quad \varepsilon \Vdash F \text{ for all basic models } \varepsilon \text{ for } \text{JE}_{\text{CS}}.$$

B Soundness and Completeness with Respect to Modular Models

Theorem 10 (Soundness and completeness w.r.t. modular models). *Let CS be an arbitrary constant specification. For each formula F we have*

$$\text{JE}_{\text{CS}} \vdash F \quad \text{iff} \quad \mathcal{M} \Vdash F \quad \text{for all } \text{JE}_{\text{CS}} \text{ modular models } \mathcal{M}.$$

Proof. To prove soundness, suppose $\mathcal{M} = \langle W, N, \varepsilon \rangle$ is a JE_{CS} modular model, and $\text{JE}_{\text{CS}} \vdash A$. We need to show that A is true in every world $w \in W$. Assume that ε_w is a basic model. Then by soundness with respect to basic models we get $\varepsilon_w \Vdash A$ and by (11) we conclude $\mathcal{M}, w \Vdash A$. It remains to show that ε_w indeed is a basic model, i.e. that it is factive. Suppose $\varepsilon_w \Vdash \lambda : F$. By (11) we get $\mathcal{M}, w \Vdash \lambda : F$. By factivity of modular models we get $\mathcal{M}, w \Vdash F$ and by (11) again we conclude $\varepsilon_w \Vdash F$.

For completeness, suppose that $\text{JE}_{\text{CS}} \not\vdash F$. Since JE_{CS} is complete with respect to basic models, there is a JE_{CS} -basic model ε with $\varepsilon \not\vdash F$. Now we construct a quasi-model $\mathcal{M} := \langle \{w\}, N, \varepsilon' \rangle$ with $\varepsilon'_w := \varepsilon$ and

$$N(w) = \{|G|^{\mathcal{M}} \mid G \in \varepsilon'_w(t), \text{ for any } t \in \text{JTm}\}.$$

By (11) we find $\mathcal{M}, w \not\vdash F$. It only remains to show that \mathcal{M} is a modular model: Factivity follows immediately from (11) and the fact that ε is factive. To show (JYB), we suppose $F \in \varepsilon'_w(t)$. By the definition of N we get $|F|^{\mathcal{M}} \in N(w)$, which means $F \in \Box_w$.

C Soundness and Completeness with Respect to Fully Explanatory Modular Models

The next step is to prove that JE_{CS} is sound and complete with respect to fully explanatory JE_{CS} modular models. Before starting to prove the theorem, we need an auxiliary notion:

Definition 18 (Proof set). *Let M_{JE} be the set of all maximal JE_{CS} -consistent sets of formulas. We set*

$$\text{M}_{\text{JE}} := \{\Gamma \mid \Gamma \text{ is a maximal } \text{JE}_{\text{CS}}\text{-consistent set}\}.$$

For any formula F we define $\|F\| := \{\Gamma \in \text{M}_{\text{JE}} \text{ and } F \in \Gamma\}$, called the proof set of F .

Proof sets share a number of properties, which are given in the following lemma.

Lemma 7. *For formulas F, G following properties hold:*

1. $\|F \wedge G\| = \|F\| \cap \|G\|;$

2. $\|\neg F\| = M_{JE} \setminus \|F\|$;
3. $\|F \vee G\| = \|F\| \cup \|G\|$;
4. $\|F\| \subseteq \|G\|$ iff $\vdash F \rightarrow G$;
5. $\vdash (F \leftrightarrow G)$ iff $\|F\| = \|G\|$;
6. if $\|\lambda : G\| \subseteq \|G\|$ for any proof term λ .

Proof. Let only show claim 4. The claim from right to left immediately follows from closure of maximal consistent sets under modus ponens. For the other direction, suppose $\|F\| \subseteq \|G\|$, but not $\vdash F \rightarrow G$. Then $\neg(F \rightarrow G)$ is consistent and by Lindenbaum's Lemma there is a maximal consistent set $\Gamma \ni \neg(F \rightarrow G)$. This means $F, \neg G \in \Gamma$. Since $F \in \Gamma$ and $\|F\| \subseteq \|G\|$, we get $G \in \Gamma$, which contradicts $\neg G \in \Gamma$.

Theorem 11 (Soundness and completeness for fully explanatory modular models). *Let CS be an axiomatically appropriate constant specification. JE_{CS} is sound and complete with respect to fully explanatory JE_{CS} modular models.*

Proof. Soundness is a direct consequence of soundness for the class of JE_{CS} modular models.

To prove completeness, we define a canonical model $\mathcal{M}^c := \langle W^c, N^c, \varepsilon^c \rangle$ by

- $W^c := M_{JE}$;
- $N^c : W^c \rightarrow \mathcal{P}(\mathcal{P}(W^c))$, such that for each maximal consistent set $\Gamma \in W^c$,

$$\|F\| \in N^c(\Gamma) \text{ iff } [t]F \in \Gamma, \text{ for some } t \in \mathbf{JTm} ;$$
- $\varepsilon^c(t) := \{F \mid [t]F \in \Gamma\}$.

We define $\Gamma/\square := \{F \mid [t]F \in \Gamma, \text{ for some } t \in \mathbf{JTm}\}$. Now the canonical neighborhood function can be reformulated as

$$N^c(\Gamma) = \{\|F\| \mid F \in \Gamma/\square\} .$$

Before establishing that this canonical model is a fully explanatory modular model, we show that the neighborhood function is well-defined. The issue is that different formulas may have the same proof set. Thus we need to show the following lemma.

Lemma 8. *Let CS be axiomatically appropriate. The neighborhood mapping N^c is well-defined, i.e. for any $\Gamma \in M_{JE}$ and any formulas F, G , if $\|F\| \in N(\Gamma)$ and $\|F\| = \|G\|$ then $[t]G \in \Gamma$ for some $t \in \mathbf{JTm}$.*

Proof. Let F, G be two formulas such that $\|F\| = \|G\|$. For some $\Gamma \in M_{JE}$, suppose $\|F\| \in N^c(\Gamma)$. Thus $[s]F \in \Gamma$ for some $s \in \mathbf{JTm}$ by the definition of the canonical model. By Lemma 7, we have $\vdash F \leftrightarrow G$ and thus $\vdash F \rightarrow G$ as well as $\vdash G \rightarrow F$. Since CS is axiomatically appropriate, there are proof terms λ_1, λ_2 such that $\vdash \lambda_1 : (F \rightarrow G)$ and $\vdash \lambda_2 : (G \rightarrow F)$. By the **je** axiom, we conclude $[e(\lambda_1, \lambda_2, s)]G \in \Gamma$.

Next we can establish the truth lemma.

Lemma 9 (Truth lemma). *For each formula F , we have $|F|^{\mathcal{M}^c} = \|F\|$.*

Proof. As usual the proof is by induction on the structure of F . We only show the case when F is $[t]G$. We have the following equivalences: $\Gamma \in \|[t]G\|^{\mathcal{M}^c}$ iff $\mathcal{M}^c, \Gamma \Vdash [t]G$ iff $F \in \varepsilon_{\Gamma}^c(t)$ iff $[t]F \in \Gamma$ iff $\Gamma \in \|[t]F\|$.

Now we show that the canonical model is a modular model. First, we show that $W^c \neq \emptyset$. Recall that by Lindenbaum's Lemma, for every consistent set of formulas Γ , there exist a maximally consistent set of formulas that contains Γ . Since the empty set is consistent, by Lindenbaum's Lemma, there is a maximal consistent set that contains the empty set and is an element of W^c .

Next we show factivity. Suppose $\mathcal{M}^c, \Gamma \Vdash \lambda : G$. By the truth lemma we get $\lambda : G \in \Gamma$. Since Γ is maximally consistent, we obtain by axiom **jt** that $G \in \Gamma$. Again by the truth lemma we conclude $\mathcal{M}^c, \Gamma \Vdash G$.

Now we show that the canonical model satisfies justification yields belief (**JYB**). Suppose $F \in \varepsilon_{\Gamma}^c(t)$ for some term t , some formula F , and some $\Gamma \in W^c$. Since $\varepsilon_{\Gamma}^c(t) = \{G \mid [t]G \in \Gamma\}$, we have $[t]F \in \Gamma$. By the definition of N^c we obtain $\|F\| \in N^c(\Gamma)$. Thus, using the truth lemma, we get $|F|^{\mathcal{M}^c} \in N^c(\Gamma)$. Thus (**JYB**) is established.

It remains to show that the canonical model is fully explanatory. Suppose $|F|^{\mathcal{M}^c} \in N^c(\Gamma)$ for some formula F and some $\Gamma \in W^c$. By the truth lemma we find $\|F\| \in N^c(\Gamma)$. By the definition of N^c , this implies $[t]F \in \Gamma$ for some term t . By the definition of ε_{Γ}^c we finally conclude $F \in \varepsilon_{\Gamma}^c(t)$.


References

1. Artemov, S.: Explicit provability and constructive semantics. *BSL* **7**(1), 1–36 (2001)
2. Artemov, S.: Justified common knowledge. *TCS* **357**(1–3), 4–22 (2006)
3. Artemov, S.: The ontology of justifications in the logical setting. *Stud. Log.* **100**(1–2), 17–30 (2012)
4. Artemov, S., Fitting, M.: *Justification Logic: Reasoning with Reasons*. Cambridge University Press, Cambridge (2019)
5. Bucheli, S., Kuznets, R., Studer, T.: Justifications for common knowledge. *Appl. Non-Class. Log.* **21**(1), 35–60 (2011)
6. Faroldi, F.L., Ghari, M., Lehmann, E., Studer, T.: Impossible and conflicting obligations in justification logic. In: Marra, A., Liu, F., Portner, P., Van De Putte, F. (eds.) *Proceedings of the DEON 2020* (2020)
7. Faroldi, F.L.G.: Deontic modals and hyperintensionality. *Log. J. IGPL* **27**, 387–410 (2019)
8. Faroldi, F.L.G.: Normative properties and higher-order supervenience. In: *Hyperintensionality and Normativity*, pp. 181–199. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-03487-0_8
9. Faroldi, F.L.G., Protopopescu, T.: A hyperintensional logical framework for deontic reasons. *Log. J. IGPL* **27**, 411–433 (2019)
10. Fitting, M.: The logic of proofs, semantically. *APAL* **132**(1), 1–25 (2005)

11. Forrester, J.W.: Gentle murder, or the adverbial samaritan. *J. Philos.* **81**(4), 193–197 (1984)
12. Indrzejczak, A.: Admissibility of cut in congruent modal logics. *Log. Log. Philos.* **20**(3), 189–203 (2011)
13. Kokkinis, I., Maksimović, P., Ognjanović, Z., Studer, T.: First steps towards probabilistic justification logic. *Log. J. IGPL* **23**(4), 662–687 (2015)
14. 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)
15. Kuznets, R., Studer, T.: Weak arithmetical interpretations for the logic of proofs. *Log. J. IGPL* **24**(3), 424–440 (2016)
16. Kuznets, R., Studer, T.: *Logics of Proofs and Justifications*. College Publications, Norcross (2019)
17. Lavendhomme, R., Lucas, T.: Sequent calculi and decision procedures for weak modal systems. *Stud. Log.* **66**(1), 121–145 (2000)
18. Lehmann, E., Studer, T.: Subset models for justification logic. In: Iemhoff, R., Moortgat, M., de Queiroz, R. (eds.) *WoLLIC 2019*. LNCS, vol. 11541, pp. 433–449. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-59533-6_26
19. Pacuit, E.: A note on some explicit modal logics. In: *Proceedings of the 5th Panhellenic Logic Symposium*, pp. 117–125. University of Athens, Athens, Greece, 25–28 July 2005
20. Pacuit, E.: *Neighborhood Semantics for Modal Logic*. Springer, Heidelberg (2017)
21. Renne, B.: *Dynamic Epistemic Logic with Justification*. PhD thesis, City University of New York, May 2008
22. Ross, A.: Imperatives and logic. *Theoria* **7** (1941)
23. Studer, T.: Decidability for some justification logics with negative introspection. *JSL* **78**(2), 388–402 (2013)
24. Studer, T.: A conflict tolerant logic of explicit evidence. *Log. Investig.* **27**(1), 124–144 (2021)



A General Relational Semantics of Propositional Logic: Axiomatization

Shengyang Zhong^(✉) 

Institute of Foreign Philosophy, Department of Philosophy and Religious Studies,
Peking University, Beijing, China

Abstract. In the chapter on quantum logic in Volume 6 of *Handbook of Philosophical Logic*, Dalla Chiara and Giuntini make an interesting observation that there is a unified relational semantics underlying both the $\{\neg, \wedge\}$ -fragment of intuitionistic logic and ortho-logic. In this paper, we contribute to a systematic investigation of this relational semantics by providing an axiomatization of its logic.

Keywords: Relational semantics · Intuitionistic logic · Quantum logic

1 Introduction

Intuitionistic logic [7] and quantum logic [2] are two important kinds of non-classical logic. They were inspired by the observation that some laws in classical propositional logic are untenable under some considerations from philosophy of mathematics or quantum physics. At the beginning, they were studied as formal proof calculi in which some laws in classical logic are absent. To be precise, we do not have in propositional intuitionistic logic the law of double negation and a part of De Morgan Laws, while in quantum logic the distributive laws between conjunction and disjunction are missing.

Around 1970, after the genesis of relational semantics in modal logic, relational semantics of intuitionistic logic [1, 8] and that of quantum logic [6] were also proposed. These formal semantics prove to be fruitful in the study of these two logics and in addition reveal some interesting technical similarities of the logics. (Please refer to the textbook [4] for intuitionistic logic and the handbook chapter [5] for quantum logic.) The main idea is to use a Kripke frame, i.e. a non-empty set W equipped with a binary relation \rightarrow , to interpret these two propositional logics. The intuition behind W is in intuitionistic logic the set of stages of mental construction of mathematical objects and in quantum logic the set of pure states of a quantum system, respectively. The intuition behind \rightarrow is in intuitionistic logic the progress from one stage of mental construction to another and in quantum logic the non-orthogonality relation between pure states, respectively.

Supported by NSSF (No. 20CZX048).

© Springer Nature Switzerland AG 2021

A. Silva et al. (Eds.): WoLLIC 2021, LNCS 13038, pp. 82–99, 2021.

https://doi.org/10.1007/978-3-030-88853-4_6

There are two key features of this interpretation. One of them is that the fundamental level of interpretation, i.e. the notion of truth, is local and bivalent. To be precise, the formal notion of truth is among a Kripke frame (W, \rightarrow) , an element $s \in W$ and a formula φ , instead of being between a Kripke frame and a formula. Moreover, either φ is true at s in (W, \rightarrow) or φ is false at s in (W, \rightarrow) (Page 26 in [4] for intuitionistic logic and Definition 2.2 in [6] for quantum logic). Under such an interpretation the *proposition* expressed by a formula φ is the set of elements of W at which φ is true. The other feature of this interpretation is that not every subset of W is a proposition. A proposition satisfies in intuitionistic logic a property called ‘persistent’¹ and in quantum logic a property called ‘bi-orthogonally closed’². If every subset of W is a proposition, we get classical logic; by restricting our attention to special subsets of W , from an algebraic point of view, we get sublattices of the power set algebra on W which are not Boolean.

This interpretation provides a unified framework for both intuitionistic logic and quantum logic which reflects some intuitions behind these two logics and thus is not as purely technical as algebraic semantics. However, the logics still seem very different. In intuitionistic logic we study the persistent subsets of W in a Kripke frame (W, \rightarrow) where \rightarrow satisfies at least Reflexivity and Transitivity (Page 25 in [4]), while in quantum logic we study the bi-orthogonally closed subsets of W in a Kripke frame (W, \rightarrow) where \rightarrow satisfies at least Reflexivity and Symmetry (Definition 2.1 in [6]³).

In [5] (pp. 139–140) Dalla Chiara and Giuntini make an interesting observation that, despite the difference on the involved classes of Kripke frames, intuitionistic logic and quantum logic care about the same kind of special subsets of W . In other words, there is a unified property of subsets of W such that, when \rightarrow satisfies Reflexivity and Transitivity, it is equivalent to being persistent and, when \rightarrow satisfies Reflexivity and Symmetry, it is equivalent to being bi-orthogonally closed. Technically this points to a unified notion of propositions in a relational semantics for both intuitionistic logic and quantum logic. We acknowledge that the motivations behind intuitionistic logic and quantum logic are completely different, and that at present there is no meaningful interpretation of this unified notion of propositions, as far as we know. However, we still hope that a study of the technical aspect of this unified notion of propositions in relational semantics of propositional logic may eventually lead to something common and interesting about the notion of propositions in intensional logics.

In this paper we contribute to a systematic investigation of this formal semantics by providing an axiomatization of its logic. The formal language we use only

¹ Please refer to Page 25 and Proposition 2.1 in [4], where ‘persistent’ in this paper (Item (i) in Lemma 9 below) is called ‘upward closed’ there.

² Please refer to Page 24 and Lemma 3.1 in [6], where ‘bi-orthogonally closed’ in this paper (Item (i) in Lemma 10 below) is called ‘ \perp -closed’ there.

³ Please note that in [6] the orthogonality relation, instead of the non-orthogonality relation, is primitive, so there the binary relations are required to be irreflexive and symmetric.

has two connectives \neg and \wedge . We leave out disjunction and implication, because both of them are defined differently in intuitionistic logic and quantum logic. (Please refer to Page 26 in [4] for intuitionistic logic and Page 137 and Sect. 3 in [5] for quantum logic). The axiomatization is essentially in the style of natural deduction, but we adopt an abstract, rigorous and concise treatment and define it as the smallest relation between a set of formulas and a formula that satisfies some syntactic properties. Finally, since the logic is weak, we write the proofs in greater details than normal so that it is convenient for the readers to check what is used in which step and whether we take some unwarranted things for granted.

The remaining part of this paper is organized as follows: In Sect. 2 we present the logic, including the formal language and the definitions of semantic consequence and syntactic consequence. We also prove the soundness and completeness theorem. In Sect. 3 we briefly discuss how the $\{\neg, \wedge\}$ -fragment of intuitionistic logic and quantum logic can be considered as extensions of our logic, which is the observation in [5]. We add nothing new here but just flesh out this observation by proving some claims without proofs in [5]. Our techniques are useful in proving soundness and completeness of the $\{\neg, \wedge\}$ -fragment of intuitionistic logic and ortho-logic, the smallest logic in quantum logic; and this will be presented in the appendices. In Sect. 4 we discuss some interesting topics for future work.

2 The Logic PL

2.1 Formal Language

Definition 1. *PV is a countably infinite set of propositional variables.*

Formulas are defined as follows:

$$\varphi ::= p \mid (\neg\varphi) \mid (\varphi \wedge \varphi), \text{ where } p \in \text{PV}$$

Denote by Form the set of formulas. Moreover, we may omit the parentheses with the assumption that \neg has priority over \wedge .

2.2 Semantic Consequence

Definition 2. *A frame is an ordered pair $\mathfrak{F} = (W, \rightarrow)$ such that W is a non-empty set and $\rightarrow \subseteq W \times W$ satisfies Reflexivity, i.e. $s \rightarrow s$ for each $s \in W$.*

A proposition⁴ in a frame $\mathfrak{F} = (W, \rightarrow)$ is a set $X \subseteq W$ such that, for each $s \in W$, the following are equivalent:

- (i) $s \in X$;
- (ii) for each $t \in W$, $s \rightarrow t$ implies that there is a $u \in X$ satisfying $u \rightarrow t$.

Denote by $\mathcal{P}(\mathfrak{F})$ the set of propositions of \mathfrak{F} .

⁴ This terminology and its definition are Definition 7 on page 139 in [5].

Remark 1. Note that, for each $X \subseteq W$ in a frame $\mathfrak{F} = (W, \rightarrow)$, for each $s \in W$, the direction from (i) to (ii) is trivial and only the direction from (ii) to (i) is substantial in the definition of propositions. Hence in the following we only show the proof of the latter direction when proving that a set is a proposition.

Definition 3. A model is an ordered pair $\mathfrak{M} = (\mathfrak{F}, V)$, where $\mathfrak{F} = (W, \rightarrow)$ is a frame and V is a function from PV to $\mathcal{P}(\mathfrak{F})$.

Remark 2. Note that V maps each $p \in PV$ to an element of $\mathcal{P}(\mathfrak{F})$, instead of an element of $\wp(W)$.⁵

Definition 4. Define a satisfaction relation \models between a model $\mathfrak{M} = (W, \rightarrow, V)$, $s \in W$ and $\varphi \in \text{Form}$ recursively as follows:

- $\mathfrak{M}, s \models p$, if $s \in V(p)$;
- $\mathfrak{M}, s \models (\varphi \wedge \psi)$, if $\mathfrak{M}, s \models \varphi$ and $\mathfrak{M}, s \models \psi$;
- $\mathfrak{M}, s \models (\neg\varphi)$, if, for each $t \in W$, $s \rightarrow t$ implies that $\mathfrak{M}, t \not\models \varphi$.

For each $\Gamma \subseteq \text{Form}$, $\mathfrak{M}, s \models \Gamma$ means that, for each $\gamma \in \Gamma$, $\mathfrak{M}, s \models \gamma$. We write $\|\varphi\|$ for $\{s \in W \mid \mathfrak{M}, s \models \varphi\}$.

Remark 3. Note that in a model $\mathfrak{M} = (W, \rightarrow, V)$ by definition, for any $p \in PV$ and $\varphi, \psi \in \text{Form}$,

1. $\|p\| = V(p)$;
2. $\|(\varphi \wedge \psi)\| = \|\varphi\| \cap \|\psi\|$;
3. $\|(\neg\varphi)\| = -\|\varphi\|$, where, for an $X \subseteq W$, $-X$, called the *strong complement* of X , is defined to be $\{s \in W \mid \text{for each } t \in W, \text{ if } s \rightarrow t, \text{ then } t \notin X\}$.

Lemma 1. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model. For each $\varphi \in \text{Form}$, $\|\varphi\|$ is a proposition.

Proof. Use induction on the structure of formula.

In the base step, φ is a $p \in PV$. By definition $\|p\| = V(p)$ is a proposition.

In the induction step, consider two cases.

In the first case, φ is $\neg\psi$. Assume that, for each t , if $s \rightarrow t$, then there is a u such that $u \in \|\neg\psi\|$ and $u \rightarrow t$. For each t satisfying $s \rightarrow t$, by the assumption there is a u such that $u \in -\|\psi\|$ and $u \rightarrow t$, so $t \notin \|\psi\|$. Thus $s \in -\|\psi\| = \|\neg\psi\|$.

In the second case, φ is $\psi \wedge \theta$. Assume that, for each t , if $s \rightarrow t$, then there is a u such that $u \in \|\psi \wedge \theta\|$ and $u \rightarrow t$. For each t satisfying $s \rightarrow t$, by the assumption there is a u such that $u \in \|\psi\| \cap \|\theta\|$ and $u \rightarrow t$, so this u is such that $u \in \|\psi\|$ and $u \rightarrow t$. By IH $\|\psi\|$ is a proposition, so $s \in \|\psi\|$. Similarly we can show that $s \in \|\theta\|$. Therefore, $s \in \|\psi\| \cap \|\theta\| = \|\psi \wedge \theta\|$. \square

Definition 5. For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, φ is a semantic consequence of Γ , denoted by $\Gamma \models \varphi$, if, for any model $\mathfrak{M} = (W, \rightarrow, V)$ and $s \in W$, $\mathfrak{M}, s \models \Gamma$ implies that $\mathfrak{M}, s \models \varphi$.

⁵ For a set A , $\wp(A)$ denotes the power set of A .

2.3 Syntactic Consequence

Definition 6. Let $\vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form}$.

1. For any $\Gamma \in \wp(\text{Form})$ and $\varphi \in \text{Form}$, φ is an \mathbf{L} -syntactic consequence of Γ , if $(\Gamma, \varphi) \in \vdash_{\mathbf{L}}$.
In this case, for convenience, we write $\Gamma \vdash_{\mathbf{L}} \varphi$. Moreover, we write $\psi \vdash_{\mathbf{L}} \varphi$ for $\{\psi\} \vdash_{\mathbf{L}} \varphi$ and $\vdash_{\mathbf{L}} \varphi$ for $\emptyset \vdash_{\mathbf{L}} \varphi$.
2. $\Gamma \in \wp(\text{Form})$ is \mathbf{L} -consistent, if there is no $\varphi \in \text{Form}$ such that $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \neg\varphi$.
3. $\Gamma \in \wp(\text{Form})$ is \mathbf{L} -closed, if, for each $\varphi \in \text{Form}$, $\Gamma \vdash_{\mathbf{L}} \varphi$ implies that $\varphi \in \Gamma$.

The following are some syntactic properties of a relation $\vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form}$ (to make them easier to read, we omit the universal quantifiers for $\Gamma, \Delta \in \wp(\text{Form})$ and $\varphi, \psi \in \text{Form}$ at the beginning of each of the properties):

- (A) $\Gamma \cup \{\varphi\} \vdash_{\mathbf{L}} \varphi$;
- (\wedge I) if $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \psi$, then $\Gamma \vdash_{\mathbf{L}} \varphi \wedge \psi$;
- (\wedge E) if $\Gamma \vdash_{\mathbf{L}} \varphi \wedge \psi$, then $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \psi$;
- (Exp) if $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \neg\varphi$, then $\Gamma \vdash_{\mathbf{L}} \psi$;
- (Mon) if $\Gamma \subseteq \Delta$ and $\Gamma \vdash_{\mathbf{L}} \varphi$, then $\Delta \vdash_{\mathbf{L}} \varphi$;
- (Cut) if $\Gamma \cup \{\psi\} \vdash_{\mathbf{L}} \varphi$ and $\Delta \vdash_{\mathbf{L}} \psi$, then $\Gamma \cup \Delta \vdash_{\mathbf{L}} \varphi$;
- (\neg Iw) if $\varphi \vdash_{\mathbf{L}} \psi$ and $\varphi \vdash_{\mathbf{L}} \neg\psi$, then $\vdash_{\mathbf{L}} \neg\varphi$;
- (Ctr) if $\varphi \vdash_{\mathbf{L}} \psi$, then $\neg\psi \vdash_{\mathbf{L}} \neg\varphi$;
- (\neg I) if $\Gamma \cup \{\varphi\} \vdash_{\mathbf{L}} \psi$ and $\Gamma \cup \{\varphi\} \vdash_{\mathbf{L}} \neg\psi$, then $\Gamma \vdash_{\mathbf{L}} \neg\varphi$;
- (\neg^2 I) $\Gamma \cup \{\varphi\} \vdash_{\mathbf{L}} \neg\neg\varphi$;
- (\neg^2 E) $\Gamma \cup \{\neg\neg\varphi\} \vdash_{\mathbf{L}} \varphi$;
- (Com) if $\Gamma \vdash_{\mathbf{L}} \varphi$, there is a finite set $\Gamma' \subseteq \Gamma$ such that $\Gamma' \vdash_{\mathbf{L}} \varphi$.

Definition 7. We define three special syntactic consequence relations and mark them by special subscripts \mathbf{PL} , \mathbf{IL} and \mathbf{OL} , respectively. \mathbf{IL} stands for intuitionistic logic and \mathbf{OL} stands for ortho-logic.

$$\begin{aligned} \vdash_{\mathbf{PL}} &= \bigcap \left\{ \vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form} \mid \right. \\ &\quad \left. \vdash_{\mathbf{L}} \text{ satisfies } (A), (\wedge I), (\wedge E), (\text{Exp}), (\text{Mon}), (\text{Cut}), (\neg Iw), (\text{Ctr}) \right\} \\ \vdash_{\mathbf{IL}} &= \bigcap \left\{ \vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form} \mid \right. \\ &\quad \left. \vdash_{\mathbf{L}} \text{ satisfies } (A), (\wedge I), (\wedge E), (\text{Exp}), (\text{Mon}), (\text{Cut}), (\neg I) \right\} \\ \vdash_{\mathbf{OL}} &= \bigcap \left\{ \vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form} \mid \right. \\ &\quad \left. \vdash_{\mathbf{L}} \text{ satisfies } (A), (\wedge I), (\wedge E), (\text{Exp}), (\text{Mon}), (\text{Cut}), (\neg Iw), (\text{Ctr}), (\neg^2 I), (\neg^2 E) \right\} \end{aligned}$$

Lemma 2

1. $\vdash_{\mathbf{PL}}$ satisfies (A) , $(\wedge I)$, $(\wedge E)$, (Exp) , (Mon) , (Cut) , $(\neg Iw)$ and (Ctr) .
2. $\vdash_{\mathbf{IL}}$ satisfies (A) , $(\wedge I)$, $(\wedge E)$, (Exp) , (Mon) , (Cut) and $(\neg I)$.
3. $\vdash_{\mathbf{OL}}$ satisfies (A) , $(\wedge I)$, $(\wedge E)$, (Exp) , (Mon) , (Cut) , $(\neg Iw)$, (Ctr) , $(\neg^2 I)$ and $(\neg^2 E)$.

Proof. The proof is an easy verification, because each syntactic property under consideration is expressed by a universal statement. \square

Next we prove three important lemmas of syntactic consequence relations.

Lemma 3 (Extension Lemma). *Let $\vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form}$ satisfies (A), (Exp), (Mon), (Cut) and (Com). For any $\Gamma \subseteq \text{Form}$ and $\theta \in \text{Form}$, if $\Gamma \not\vdash_{\mathbf{L}} \theta$, there is a $\Delta \subseteq \text{Form}$ such that Δ is \mathbf{L} -closed and \mathbf{L} -consistent, $\Gamma \subseteq \Delta$ and $\theta \notin \Delta$.*

Proof. Enumerate Form without repetition as $(\varphi_i : i \in \mathbb{N})$.

Define a sequence of sets of formulas $(\Gamma_i : i \in \mathbb{N})$ recursively as follows:

$$\begin{aligned} \Gamma_0 &= \Gamma \\ \Gamma_{i+1} &= \begin{cases} \Gamma_i \cup \{\varphi_i\}, & \text{if } \Gamma_i \cup \{\varphi_i\} \not\vdash_{\mathbf{L}} \theta \\ \Gamma_i, & \text{otherwise} \end{cases} \end{aligned}$$

By induction it can be shown that (a) $\Gamma_i \subseteq \Gamma_j$, for any $i, j \in \mathbb{N}$ satisfying $i \leq j$ and (b) $\Gamma_i \not\vdash_{\mathbf{L}} \theta$, for each $i \in \mathbb{N}$.

Let $\Delta = \bigcup_{i \in \mathbb{N}} \Gamma_i$. Obviously $\Gamma = \Gamma_0 \subseteq \bigcup_{i \in \mathbb{N}} \Gamma_i = \Delta$.

Note that $\Delta \not\vdash_{\mathbf{L}} \theta$: Otherwise, by (Com) there is a finite subset $\Delta' \subseteq \Delta$ such that $\Delta' \vdash_{\mathbf{L}} \theta$; since Δ' is finite, there is an $n \in \mathbb{N}$ such that $\Delta' \subseteq \Gamma_n$, so by (Mon) $\Gamma_n \vdash_{\mathbf{L}} \theta$, contradicting that $\Gamma_n \not\vdash_{\mathbf{L}} \theta$.

Then it follows from (A) that $\theta \notin \Delta$ and by (Exp) Δ is \mathbf{L} -consistent.

Note that Δ is \mathbf{L} -closed: Assume that $\Delta \not\vdash_{\mathbf{L}} \varphi$. Then there is an $n \in \mathbb{N}$ such that $\varphi_n = \varphi$. Hence $\Gamma_n \cup \{\varphi_n\} \not\vdash_{\mathbf{L}} \theta$; otherwise, by the assumption and (Cut) $\Gamma_n \cup \Delta \vdash_{\mathbf{L}} \theta$, i.e. $\Delta \vdash_{\mathbf{L}} \theta$, contradicting what has just been proved. Therefore, $\varphi \in \Gamma_n \cup \{\varphi_n\} = \Gamma_{n+1} \subseteq \bigcup_{i \in \mathbb{N}} \Gamma_i = \Delta$. \square

Lemma 4 (Conjunction Lemma). *Let $\vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form}$ satisfies (A), $(\wedge I)$ and $(\wedge E)$. For any \mathbf{L} -closed and \mathbf{L} -consistent $\Gamma \subseteq \text{Form}$ and $\varphi, \psi \in \text{Form}$, $\varphi \wedge \psi \in \Gamma$, if and only if $\varphi \in \Gamma$ and $\psi \in \Gamma$.*

Proof. For the ‘only if’ part, by (A) $\Gamma \vdash_{\mathbf{L}} \varphi \wedge \psi$. By $(\wedge E)$ $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \psi$. Since Γ is \mathbf{L} -closed, $\varphi \in \Gamma$ and $\psi \in \Gamma$.

For the ‘if’ part, by (A) $\Gamma \vdash_{\mathbf{L}} \varphi$ and $\Gamma \vdash_{\mathbf{L}} \psi$. By $(\wedge I)$ $\Gamma \vdash_{\mathbf{L}} \varphi \wedge \psi$. Since Γ is \mathbf{L} -closed, $\varphi \wedge \psi \in \Gamma$. \square

Lemma 5 (Negation Lemma). *Let $\vdash_{\mathbf{L}} \subseteq \wp(\text{Form}) \times \text{Form}$ satisfies (A), $(\neg Iw)$, (Ctr), (Mon), (Cut) and (Com). For any \mathbf{L} -closed and \mathbf{L} -consistent $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, the following are equivalent:*

- (i) $\neg\varphi \notin \Gamma$;
- (ii) there is a $\Delta \subseteq \text{Form}$ such that Δ is \mathbf{L} -closed and \mathbf{L} -consistent, $\varphi \in \Delta$ and there is no formula θ such that $\neg\theta \in \Gamma$ and $\theta \in \Delta$.

Proof. For the direction from (ii) to (i), since $\varphi \in \Delta$, $\neg\varphi \notin \Gamma$.

For the direction from (i) to (ii), following the idea in the proof of Theorem 1.4 in [6], we let $\Delta = \{\psi \in \text{Form} \mid \varphi \vdash_{\mathbf{L}} \psi\}$.

1. Show that $\varphi \in \Delta$.
By (A) $\varphi \vdash_{\mathbf{L}} \varphi$, so $\varphi \in \Delta$.
2. Show that there is no formula θ such that $\neg\theta \in \Gamma$ and $\theta \in \Delta$.
Suppose (towards a contradiction) that there is a formula θ such that $\neg\theta \in \Gamma$ and $\theta \in \Delta$. Then $\varphi \vdash_{\mathbf{L}} \theta$. By (Ctr) $\neg\theta \vdash_{\mathbf{L}} \neg\varphi$. Since $\neg\theta \in \Gamma$, by (Mon) $\Gamma \vdash_{\mathbf{L}} \neg\varphi$. Since Γ is \mathbf{L} -closed, $\neg\varphi \in \Gamma$, contradicting (i).
3. Show that Δ is \mathbf{L} -closed.
Assume that $\Delta \vdash_{\mathbf{L}} \psi$. By (Com) there is a finite set $\Delta' \subseteq \Delta$ such that $\Delta' \vdash_{\mathbf{L}} \psi$. By the definition of Δ , for each $\delta \in \Delta'$, $\varphi \vdash_{\mathbf{L}} \delta$. By (Cut) $\varphi \vdash_{\mathbf{L}} \psi$, so $\psi \in \Delta$.
4. Show that Δ is \mathbf{L} -consistent.
Suppose (towards a contradiction) that there is a formula ψ such that $\Delta \vdash_{\mathbf{L}} \psi$ and $\Delta \vdash_{\mathbf{L}} \neg\psi$. By Item 3 $\psi \in \Delta$ and $\neg\psi \in \Delta$. By definition $\varphi \vdash_{\mathbf{L}} \psi$ and $\varphi \vdash_{\mathbf{L}} \neg\psi$. By (\neg Iw) $\vdash_{\mathbf{L}} \neg\varphi$. By (Mon) $\Gamma \vdash_{\mathbf{L}} \neg\varphi$. Since Γ is \mathbf{L} -closed, $\neg\varphi \in \Gamma$, contradicting (i). \square

We end this subsection by proving the compactness of \mathbf{PL} .

Theorem 1 (Compactness Theorem of \mathbf{PL}). $\vdash_{\mathbf{PL}}$ satisfies (Com).

Proof. Let

$$\vdash = \{(\Gamma, \varphi) \in \wp(\mathbf{Form}) \times \mathbf{Form} \mid \text{there is a finite } \Gamma' \subseteq \Gamma \text{ such that } \Gamma' \vdash_{\mathbf{PL}} \varphi\}$$

It suffices to show that \vdash satisfies (A), (\wedge I), (\wedge E), (Exp), (Mon), (Cut), (\neg Iw) and (Ctr). After this is done, by definition $\vdash_{\mathbf{PL}} \subseteq \vdash$, so $\vdash_{\mathbf{PL}}$ satisfies (Com).

- (A) Since $\{\varphi\}$ is a finite subset of $\Gamma \cup \{\varphi\}$ and $\varphi \vdash_{\mathbf{PL}} \varphi$ by (A), $\Gamma \cup \{\varphi\} \vdash \varphi$.
- (\wedge I) Assume that $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$. By definition there is a finite set $\Gamma' \subseteq \Gamma$ and a finite set $\Gamma'' \subseteq \Gamma$ such that $\Gamma' \vdash_{\mathbf{PL}} \varphi$ and $\Gamma'' \vdash_{\mathbf{PL}} \psi$. By (Mon) $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \varphi$ and $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \psi$. By (\wedge I) $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \varphi \wedge \psi$. Since $\Gamma' \cup \Gamma''$ is a finite subset of Γ , $\Gamma \vdash \varphi \wedge \psi$.
- (\wedge E) Assume that $\Gamma \vdash \varphi \wedge \psi$. By definition there is a finite set $\Gamma' \subseteq \Gamma$ such that $\Gamma' \vdash_{\mathbf{PL}} \varphi \wedge \psi$. By (\wedge E) $\Gamma' \vdash_{\mathbf{PL}} \varphi$ and $\Gamma' \vdash_{\mathbf{PL}} \psi$. Hence $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$.
- (Exp) Assume that $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$. By definition there is a finite set $\Gamma' \subseteq \Gamma$ and a finite set $\Gamma'' \subseteq \Gamma$ such that $\Gamma' \vdash_{\mathbf{PL}} \varphi$ and $\Gamma'' \vdash_{\mathbf{PL}} \neg\varphi$. By (Mon) $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \varphi$ and $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \neg\varphi$. By (Exp) $\Gamma' \cup \Gamma'' \vdash_{\mathbf{PL}} \psi$. Since $\Gamma' \cup \Gamma''$ is a finite subset of Γ , $\Gamma \vdash \psi$.
- (Mon) Assume that $\Gamma \subseteq \Delta$ and $\Gamma \vdash \varphi$. By definition there is a finite $\Gamma' \subseteq \Gamma$ such that $\Gamma' \vdash_{\mathbf{PL}} \varphi$. By the assumption Γ' is a finite subset of Δ . Hence $\Delta \vdash \varphi$.
- (Cut) Assume that $\Gamma \cup \{\psi\} \vdash \varphi$ and $\Delta \vdash \psi$. By (Mon) and definition there are two finite sets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that $\Gamma' \cup \{\psi\} \vdash_{\mathbf{PL}} \varphi$ and $\Delta' \vdash_{\mathbf{PL}} \psi$. By (Cut) $\Gamma' \cup \Delta' \vdash_{\mathbf{PL}} \varphi$. Since $\Gamma' \cup \Delta'$ is a finite subset of $\Gamma \cup \Delta$, $\Gamma \cup \Delta \vdash \varphi$.
- (\neg Iw) Assume that $\varphi \vdash \psi$ and $\varphi \vdash \neg\psi$. By (Mon) $\varphi \vdash_{\mathbf{PL}} \psi$ and $\varphi \vdash_{\mathbf{PL}} \neg\psi$. By (\neg Iw) $\vdash_{\mathbf{PL}} \neg\varphi$. Since \emptyset is finite, $\vdash \neg\varphi$.

- (Ctr) Assume that $\varphi \vdash \psi$. By definition either $\varphi \vdash_{\mathbf{PL}} \psi$ or $\vdash_{\mathbf{PL}} \psi$. In both cases by (Mon) $\varphi \vdash_{\mathbf{PL}} \psi$. By (Ctr) $\neg\psi \vdash_{\mathbf{PL}} \neg\varphi$. Since $\{\neg\psi\}$ is finite, $\neg\psi \vdash \neg\varphi$. \square

Remark 4. By Theorem 1 the conclusions of Lemmas 3, 4 and 5 apply to $\vdash_{\mathbf{PL}}$.

2.4 Soundness and Completeness

We start from proving the soundness theorem.

Theorem 2 (Soundness Theorem of PL). *For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \vdash_{\mathbf{PL}} \varphi$ implies that $\Gamma \models \varphi$.*

Proof. Similar to the proof of Theorem 1, it suffices to show that \models satisfies (A), (\wedge I), (\wedge E), (Exp), (Mon), (Cut), (\neg Iw) and (Ctr), from which $\vdash_{\mathbf{PL}} \subseteq \models$ follows.

- (A) Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Assume that $\mathfrak{M}, s \models \Gamma \cup \{\varphi\}$. Then $\mathfrak{M}, s \models \varphi$.
- (\wedge I) Assume that $\Gamma \models \varphi$ and $\Gamma \models \psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose that $\mathfrak{M}, s \models \Gamma$. By the assumption $\mathfrak{M}, s \models \varphi$ and $\mathfrak{M}, s \models \psi$. Hence $\mathfrak{M}, s \models \varphi \wedge \psi$.
- (\wedge E) Assume that $\Gamma \models \varphi \wedge \psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose that $\mathfrak{M}, s \models \Gamma$. By the assumption $\mathfrak{M}, s \models \varphi \wedge \psi$. Then $\mathfrak{M}, s \models \varphi$ and $\mathfrak{M}, s \models \psi$.
- (Exp) Assume that $\Gamma \models \varphi$ and $\Gamma \models \neg\varphi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Note that $\mathfrak{M}, s \not\models \Gamma$: Suppose (towards a contradiction) that $\mathfrak{M}, s \models \Gamma$. By the assumption $\mathfrak{M}, s \models \varphi$ and $\mathfrak{M}, s \models \neg\varphi$. Since $\mathfrak{M}, s \models \neg\varphi$ and $s \rightarrow s$ by Reflexivity, $\mathfrak{M}, s \not\models \varphi$, contradicting that $\mathfrak{M}, s \models \varphi$. Therefore, $\Gamma \models \psi$.
- (Mon) Assume that $\Gamma \models \varphi$ and $\Gamma \subseteq \Delta$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose that $\mathfrak{M}, s \models \Delta$. By the assumption $\mathfrak{M}, s \models \Gamma$ and thus $\mathfrak{M}, s \models \varphi$.
- (Cut) Assume that $\Gamma \cup \{\psi\} \models \varphi$ and $\Delta \models \psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose that $\mathfrak{M}, s \models \Gamma \cup \Delta$. By the assumption $\mathfrak{M}, s \models \Gamma \cup \{\psi\}$, so $\mathfrak{M}, s \models \varphi$.
- (\neg Iw) Assume that $\varphi \models \psi$ and $\varphi \models \neg\psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose (towards a contradiction) that $\mathfrak{M}, s \not\models \neg\varphi$. Then there is a $t \in W$ such that $s \rightarrow t$ and $\mathfrak{M}, t \models \varphi$. By the assumption $\mathfrak{M}, t \models \psi$ and $\mathfrak{M}, t \models \neg\psi$. Since $\mathfrak{M}, t \models \neg\psi$ and $t \rightarrow t$ by Reflexivity, $\mathfrak{M}, t \not\models \psi$, contradicting that $\mathfrak{M}, t \models \psi$. Hence $\mathfrak{M}, s \models \neg\varphi$.
- (Ctr) Assume that $\varphi \models \psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be a model and $s \in W$. Suppose that $\mathfrak{M}, s \models \neg\psi$. Then, for each $t \in W$, if $s \rightarrow t$, then $\mathfrak{M}, t \not\models \psi$, so $\mathfrak{M}, t \not\models \varphi$ by the assumption. Hence $\mathfrak{M}, s \models \neg\varphi$. \square

For the completeness theorem, we need the notion of the canonical model. Our definition follows Definition 3.3 in [6] and the definition on Page 160 in [5].

Definition 8. $\mathfrak{F}^{\mathbf{PL}} = (W^{\mathbf{PL}}, \rightarrow^{\mathbf{PL}})$ is called the canonical frame of \mathbf{PL} , where

- $W^{\mathbf{PL}} = \{\Gamma \subseteq \text{Form} \mid \Gamma \text{ is } \mathbf{PL}\text{-consistent and } \mathbf{PL}\text{-closed}\}$;
- $\Gamma \rightarrow^{\mathbf{PL}} \Delta$, if there is no $\theta \in \text{Form}$ such that $\neg\theta \in \Gamma$ and $\theta \in \Delta$.

The canonical model of \mathbf{PL} is an ordered pair $\mathfrak{M}^{\mathbf{PL}} = (\mathfrak{F}^{\mathbf{PL}}, V^{\mathbf{PL}})$ such that $\mathfrak{F}^{\mathbf{PL}}$ is the canonical frame of \mathbf{PL} and $V^{\mathbf{PL}} : \text{PV} \rightarrow \wp(W^{\mathbf{PL}})$ is a function satisfying, for each $p \in \text{PV}$, $V^{\mathbf{PL}}(p) = \{\Gamma \in W^{\mathbf{PL}} \mid p \in \Gamma\}$.

Lemma 6 (Canonical Model Lemma).

1. $\rightarrow^{\mathbf{PL}}$ satisfies Reflexivity.
2. For any $\Gamma \in W^{\mathbf{PL}}$ and $\varphi \in \text{Form}$, $\mathfrak{M}^{\mathbf{PL}}, \Gamma \models \varphi$ if and only if $\varphi \in \Gamma$.

Proof. For Item 1, suppose (towards a contradiction) that $\Gamma \not\rightarrow^{\mathbf{PL}} \Gamma$ for some $\Gamma \in W^{\mathbf{PL}}$. By definition there is a $\theta \in \text{Form}$ such that $\neg\theta \in \Gamma$ and $\theta \in \Gamma$. By (A) $\Gamma \vdash_{\mathbf{PL}} \neg\theta$ and $\Gamma \vdash_{\mathbf{PL}} \theta$, contradicting that Γ is \mathbf{PL} -consistent.

For Item 2, use induction on the structure of formula.

In the base step, φ is a $p \in \text{PV}$. By definition $\mathfrak{M}^{\mathbf{PL}}, \Gamma \models p \Leftrightarrow p \in \Gamma$.

In the induction step, consider two cases. In the first case, φ is $\psi \wedge \theta$.

$$\begin{aligned} & \mathfrak{M}^{\mathbf{PL}}, \Gamma \models \psi \wedge \theta \\ \Leftrightarrow & \mathfrak{M}^{\mathbf{PL}}, \Gamma \models \psi \text{ and } \mathfrak{M}^{\mathbf{PL}}, \Gamma \models \theta \\ \Leftrightarrow & \psi \in \Gamma \text{ and } \theta \in \Gamma && \text{(by IH)} \\ \Leftrightarrow & \psi \wedge \theta \in \Gamma && \text{(by Lemma 4)} \end{aligned}$$

In the second case, φ is $\neg\psi$.

$$\begin{aligned} & \mathfrak{M}^{\mathbf{PL}}, \Gamma \models \neg\psi \\ \Leftrightarrow & \text{for each } \Delta \in W^{\mathbf{PL}}, \text{ if } \Gamma \rightarrow^{\mathbf{PL}} \Delta, \text{ then } \mathfrak{M}^{\mathbf{PL}}, \Delta \not\models \psi \\ \Leftrightarrow & \text{for each } \Delta \in W^{\mathbf{PL}}, \text{ if } \Gamma \rightarrow^{\mathbf{PL}} \Delta, \text{ then } \psi \notin \Delta && \text{(by IH)} \\ \Leftrightarrow & \neg\psi \in \Gamma && \text{(by Lemma 5)} \end{aligned}$$

□

It is not obvious that the canonical model is a model in the sense of Definition 3. We can find a $\Gamma \in W^{\mathbf{PL}}$ and a $p \in \text{PV}$ such that $p \notin \Gamma$ and thus $\Gamma \notin V^{\mathbf{PL}}(p)$. For $\mathfrak{M}^{\mathbf{PL}}$ to be a model, there must be a $\Delta \in W^{\mathbf{PL}}$ such that $\Gamma \rightarrow^{\mathbf{PL}} \Delta$ and, for each $\Theta \in W^{\mathbf{PL}}$, $\Theta \rightarrow^{\mathbf{PL}} \Delta$ implies that $\Theta \notin V^{\mathbf{PL}}(p)$. There is little cue on what is this Δ . Our way of solving this problem is to add a ‘twin sister’ Γ' of Γ such that (a) only Γ and Γ' access to Γ' via the binary relation and (b) the set of formulas true at Γ' is the same as that at Γ . In this case, there are only two elements accessing to Γ' , namely Γ and Γ' , both of which are not in $V^{\mathbf{PL}}(p)$; and thus Γ' has the required property. The tricky point is that the set of formulas true at Γ' must be the same as that at Γ . The idea is to let Γ' ‘see the same panorama’ as Γ . To achieve this in a way that takes care of all points in a model, a well-known technique in modal logic called ‘unravelling’ is at our

disposal (Proposition 2.15 in [3]). Finally, the model obtained by unravelling in [3] does not satisfy Reflexivity. This can be solved by taking the reflexive closure of the binary relation, and it does not affect truth in the model because \rightarrow^{PL} satisfies Reflexivity. The above consideration leads to the following definition:

Definition 9. Let $\Gamma \in W^{\text{PL}}$. The Γ -model is $\mathfrak{M}^\Gamma = (W^\Gamma, \rightarrow^\Gamma, V^\Gamma)$, where:

1. $W^\Gamma = \{(\Theta_0, \dots, \Theta_n) \mid n \in \mathbb{N}, \Theta_0, \dots, \Theta_n \in W^{\text{PL}} \text{ satisfy that } \Theta_0 = \Gamma \text{ and } \Theta_i \rightarrow^{\text{PL}} \Theta_{i+1} \text{ for each } i = 0, \dots, n-1\}$.
2. $(\Theta_0, \dots, \Theta_m) \rightarrow^\Gamma (\Theta'_0, \dots, \Theta'_n)$, if and only if one of the following is true:
 - (a) $(\Theta_0, \dots, \Theta_m) = (\Theta'_0, \dots, \Theta'_n)$;
 - (b) $(\Theta_0, \dots, \Theta_m) = (\Theta'_0, \dots, \Theta'_{n-1})$ (in this case $m+1 = n$).
3. For each $p \in \text{PV}$, $V^\Gamma(p) = \{(\Theta_0, \dots, \Theta_n) \in W^\Gamma \mid \mathfrak{M}^{\text{PL}}, \Theta_n \models p\}$.

Lemma 7 (Γ -Model Lemma) Let $\Gamma \in W^{\text{PL}}$.

1. \rightarrow^Γ satisfies Reflexivity;
2. for each $p \in \text{PV}$, $V^\Gamma(p)$ is closed.
3. \mathfrak{M}^Γ is a model.

Proof. Item 1 follows from Condition (a) in the definition of \rightarrow^Γ .

For Item 2, assume that $(\Theta_0, \dots, \Theta_{n-1}, \Theta_n) \notin V^\Gamma(p)$. By definition $\mathfrak{M}^{\text{PL}}, \Theta_n \not\models p$. Consider $(\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n)$.

1. We show that $(\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n) \in W^\Gamma$.
By the Canonical Model Lemma $\Theta_n \rightarrow^{\text{PL}} \Theta_n$. Since $(\Theta_0, \dots, \Theta_{n-1}, \Theta_n) \in W^\Gamma$, $(\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n) \in W^\Gamma$.
2. We show that, for each $(\Theta'_0, \dots, \Theta'_m) \in W^\Gamma$, $(\Theta'_0, \dots, \Theta'_m) \notin V^\Gamma(p)$, if $(\Theta'_0, \dots, \Theta'_m) \rightarrow^\Gamma (\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n)$.
Assume that $(\Theta'_0, \dots, \Theta'_m) \rightarrow^\Gamma (\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n)$.
By definition consider 2 cases.
In the first case, $(\Theta'_0, \dots, \Theta'_m) = (\Theta_0, \dots, \Theta_{n-1}, \Theta_n, \Theta_n)$. Then $\Theta'_m = \Theta_n$. Since $\mathfrak{M}^{\text{PL}}, \Theta_n \not\models p$, $\mathfrak{M}^{\text{PL}}, \Theta'_m \not\models p$, so by definition $(\Theta'_0, \dots, \Theta'_m) \notin V^\Gamma(p)$.
In the second case, $(\Theta'_0, \dots, \Theta'_m) = (\Theta_0, \dots, \Theta_{n-1}, \Theta_n)$. By the assumption at the very beginning $(\Theta'_0, \dots, \Theta'_m) = (\Theta_0, \dots, \Theta_{n-1}, \Theta_n) \notin V^\Gamma(p)$.

Therefore, $V^\Gamma(p)$ is closed.

Item 3 follows from Items 1 and 2 directly. □

Lemma 8 (Γ -Model Truth Lemma). Let $\Gamma \in W^{\text{PL}}$. For any $\varphi \in \text{Form}$ and $(\Theta_0, \dots, \Theta_n) \in W^\Gamma$, $\mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \models \varphi$ if and only if $\mathfrak{M}^{\text{PL}}, \Theta_n \models \varphi$.

Proof. Use induction on the structure of formula.

In the base step, φ is a $p \in \text{PV}$. $\mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \models p \Leftrightarrow (\Theta_0, \dots, \Theta_n) \in V^\Gamma(p) \Leftrightarrow \mathfrak{M}^{\text{PL}}, \Theta_n \models p$.

In the induction step, consider two cases.

In the first case, φ is $\psi \wedge \theta$.

$$\begin{aligned}
& \mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \models \psi \wedge \theta \\
& \Leftrightarrow \mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \models \psi \text{ and } \mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \models \theta \\
& \Leftrightarrow \mathfrak{M}^{\text{PL}}, \Theta_n \models \psi \text{ and } \mathfrak{M}^{\text{PL}}, \Theta_n \models \theta && \text{(by IH)} \\
& \Leftrightarrow \mathfrak{M}^{\text{PL}}, \Theta_n \models \psi \wedge \theta
\end{aligned}$$

In the second case, φ is $\neg\psi$.

First assume that $\mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \not\models \neg\psi$. There is a $(\Theta'_0, \dots, \Theta'_m) \in W^\Gamma$ such that $(\Theta_0, \dots, \Theta_n) \rightarrow^\Gamma (\Theta'_0, \dots, \Theta'_m)$ and $\mathfrak{M}^\Gamma, (\Theta'_0, \dots, \Theta'_m) \models \psi$. By definition in both cases $\Theta_n \rightarrow^{\text{PL}} \Theta'_m$. By IH $\mathfrak{M}^{\text{PL}}, \Theta'_m \models \psi$. Hence $\mathfrak{M}^{\text{PL}}, \Theta_n \not\models \neg\psi$.

Second assume that $\mathfrak{M}^{\text{PL}}, \Theta_n \not\models \neg\psi$. There is a $\Delta \in W^{\text{PL}}$ such that $\Theta_n \rightarrow^{\text{PL}} \Delta$ and $\mathfrak{M}^{\text{PL}}, \Delta \models \psi$. Consider $(\Theta_0, \dots, \Theta_n, \Delta)$. Since $(\Theta_0, \dots, \Theta_n) \in W^\Gamma$ and $\Theta_n \rightarrow^{\text{PL}} \Delta$, $(\Theta_0, \dots, \Theta_n, \Delta) \in W^\Gamma$ and $(\Theta_0, \dots, \Theta_n) \rightarrow^\Gamma (\Theta_0, \dots, \Theta_n, \Delta)$. Since $\mathfrak{M}^{\text{PL}}, \Delta \models \psi$, by IH $\mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n, \Delta) \models \psi$. Hence $\mathfrak{M}^\Gamma, (\Theta_0, \dots, \Theta_n) \not\models \neg\psi$. □

Theorem 3 (Completeness Theorem of PL). *For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \models \varphi$ implies that $\Gamma \vdash_{\text{PL}} \varphi$.*

Proof. Assume that $\Gamma \not\vdash_{\text{PL}} \varphi$. By Lemma 3 there is a $\Delta \in W^{\text{PL}}$ such that $\Gamma \subseteq \Delta$ and $\varphi \notin \Delta$. By Lemma 6 $\mathfrak{M}^{\text{PL}}, \Delta \models \Gamma$ and $\mathfrak{M}^{\text{PL}}, \Delta \not\models \varphi$. By Lemma 8 $\mathfrak{M}^\Delta, (\Delta) \models \Gamma$ and $\mathfrak{M}^\Delta, (\Delta) \not\models \varphi$, where (Δ) is a sequence of length 1. Since \mathfrak{M}^Δ is a model, $\Gamma \not\models \varphi$. □

3 Extensions of PL

In this section we briefly discuss how the $\{\neg, \wedge\}$ -fragment of intuitionistic logic and ortho-logic can be considered as extensions of our logic, which is the observation in [5]. Here we add nothing new but just flesh out this observation by proving some claims without proofs in [5].

3.1 Intuitionistic Logic

Definition 10. *An I-frame is a frame $\mathfrak{F} = (W, \rightarrow)$ such that \rightarrow satisfies Transitivity, i.e., for any $s, t, u \in W$, $s \rightarrow t$ and $t \rightarrow u$ imply that $s \rightarrow u$.⁶*

Lemma 9. *In an I-frame $\mathfrak{F} = (W, \rightarrow)$, for each $X \in \wp(W)$, the following are equivalent:*

- (i) X is persistent, i.e., for any $s, t \in W$, if $s \in X$ and $s \rightarrow t$, then $t \in X$;
- (ii) $X \in \mathcal{P}(\mathfrak{F})$.

⁶ Note that, due to the definition of a frame (Definition 2), the relation in an I-frame satisfies both Reflexivity and Transitivity.

Proof. For the direction from (i) to (ii), let s be arbitrary. Assume that, for each $t \in W$, if $s \rightarrow t$, then there is a $u \in X$ such that $u \rightarrow t$. By Reflexivity $s \rightarrow s$. Hence there is a u such that $u \in X$ and $u \rightarrow s$. By (i) $s \in X$.

For the direction from (ii) to (i), let s and t be arbitrary such that $s \in X$ and $s \rightarrow t$. By (ii) there is a u such that $u \in X$ and $u \rightarrow t$.

Let v be arbitrary such that $t \rightarrow v$. Since $u \rightarrow t$ and $t \rightarrow v$, by Transitivity $u \rightarrow v$. So $u \in X$ is such that $u \rightarrow v$.

By the arbitrariness of v and (ii) $t \in X$. □

Definition 11. An I-model is an ordered pair $\mathfrak{M} = (\mathfrak{F}, V)$, where \mathfrak{F} is an I-frame and V is a function from PV to $\mathcal{P}(\mathfrak{F})$.

For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, φ is an I-semantic consequence of Γ , denoted by $\Gamma \models_I \varphi$, if, for any I-model $\mathfrak{M} = (W, \rightarrow, V)$ and $s \in W$, $\mathfrak{M}, s \models \Gamma$ implies that $\mathfrak{M}, s \models \varphi$.

Remark 5. According to Lemma 9, the definition of an I-model is the same as the usual definition of a model in the relational semantics of propositional intuitionistic logic. (Please refer to Page 25 in [4]. There the binary relation is required to be anti-symmetric in addition, but in fact this does not affect the logic.) Lemma 9 is claimed without proof on pages 139–140 in [5].

Since every I-frame is a frame by definition, $\models \subseteq \models_I$, and thus the $\{\neg, \wedge\}$ -fragment of intuitionistic logic is an extension of our logic.

3.2 Ortho-Logic

Definition 12. An O-frame is a frame $\mathfrak{F} = (W, \rightarrow)$ such that \rightarrow satisfies Symmetry, i.e., for any $s, t \in W$, $s \rightarrow t$ implies $t \rightarrow s$.⁷

Lemma 10. In an O-frame $\mathfrak{F} = (W, \rightarrow)$, for each $X \in \wp(W)$, the following are equivalent:

- (i) X is bi-orthogonally closed, i.e. $-(-X) = X$;
- (ii) $X \in \mathcal{P}(\mathfrak{F})$.

Proof.

$X \in \mathcal{P}(\mathfrak{F})$

\Leftrightarrow for each $s \in W$, $s \in X$, if and only if, for each t , $s \rightarrow t$ implies that
 there is a u such that $u \in X$ and $u \rightarrow t$

\Leftrightarrow for each $s \in W$, $s \in X$, if and only if, for each t , $s \rightarrow t$ implies that
 there is a u such that $u \in X$ and $t \rightarrow u$ (by Symmetry)

\Leftrightarrow for each $s \in W$, $s \in X$, if and only if, for each t , $s \rightarrow t$ implies that $t \notin -X$

\Leftrightarrow for each $s \in W$, $s \in X$, if and only if $s \in -(-X)$

$\Leftrightarrow -(-X) = X$

□

⁷ Note that, due to the definition of a frame (Definition 2), the relation in an O-frame satisfies both Reflexivity and Symmetry.

Definition 13. An O-model is an ordered pair $\mathfrak{M} = (\mathfrak{F}, V)$, where \mathfrak{F} is an O-frame and V is a function from PV to $\mathcal{P}(\mathfrak{F})$.

For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, φ is an O-semantic consequence of Γ , denoted by $\Gamma \models_O \varphi$, if, for any O-model $\mathfrak{M} = (W, \rightarrow, V)$ and $s \in W$, $\mathfrak{M}, s \models \Gamma$ implies that $\mathfrak{M}, s \models \varphi$.

Remark 6. According to Lemma 10, the definition of an O-model is the same as the usual definition of a model in the relational semantics of ortho-logic [6], despite the fact that in the literature usually $\perp \stackrel{\text{def}}{=} (W \times W) \setminus \rightarrow$ is the primitive binary relation. Lemma 10 is claimed without proof on page 140 in [5].

Since every O-frame is a frame by definition, $\models \subseteq \models_O$, and thus ortho-logic is an extension of our logic.

4 Future Work

The axiomatization result in this paper is only a part of a systematic study of this general relational semantics of propositional logic, and much more could and should be done.

First, it is interesting to pinpoint the expressive power of this relational semantics in describing Kripke frames by a van Benthem Characterization Theorem. We defer this study to an extension of this paper.

Second, we see that the present theory of this relational semantics is not as modular as that of the relational semantics of modal logic. Remember that the logics **IL** and **OL** are related to the modal logics **S4** and **KTB** via the Tarski-Mckinsey translation and Goldblatt's translation, respectively, and there is a unified theory of relational semantics for **S4** and **KTB** and many other modal logics. However, it is not the present situation for **IL**, **OL** and **PL**. For an example, according to Lemma 11 in the appendix, in defining **IL**, $(\neg Iw)$ and (Ctr) are no longer needed when $(\neg I)$ is present. For another example, in the completeness proofs in the appendices we see that the model for **IL** and that for **OL** are different and both are different from that for **PL**, although they are related. It is interesting to see whether there is a logical theory of this relational semantics, which is as modular as the theory of relational semantics of modal logic. Conceptually, this will lead to interesting interplay between syntax and semantics. Technically, this may involve works on proof theory to find the appropriate formal system and improvement on the current completeness proofs.

Acknowledgements. I'm very grateful to Prof. Roberto Giuntini for a helpful discussion about the observation in [5]. Sections 2.1 to 2.3 were presented and discussed at Workshop on Modal Logic 2018 held in Hangzhou, and I thank the participants very much for their suggestions. The last but not least, I'm very grateful to the three reviewers of this paper for their helpful comments.

A Intuitionistic Logic

In this appendix, we apply our techniques to prove the soundness and completeness theorem for **IL**, which is the $\{\neg, \wedge\}$ -fragment of intuitionistic logic.

We start with a remark about the semantics.

Remark 7. Since every I-model is a model, by Lemmas 1 and 9 in an I-model $\|\varphi\|$ is persistent for each $\varphi \in \text{Form}$.

Second, we prove some results about the syntactic consequence relation $\vdash_{\mathbf{IL}}$.

Lemma 11. $\vdash_{\mathbf{IL}}$ satisfies $(\neg Iw)$ and (Ctr) , and thus $\vdash_{\mathbf{PL}} \subseteq \vdash_{\mathbf{IL}}$

Proof. $(\neg Iw)$ is a special case of $(\neg I)$ when $\Gamma = \emptyset$.

For (Ctr) , assume that $\varphi \vdash_{\mathbf{IL}} \psi$. By (Mon) $\{\varphi, \neg\psi\} \vdash_{\mathbf{IL}} \psi$. By (A) $\{\varphi, \neg\psi\} \vdash_{\mathbf{IL}} \neg\psi$. By $(\neg I)$ $\neg\psi \vdash_{\mathbf{IL}} \neg\varphi$. \square

Theorem 4 (Compactness Theorem of IL). $\vdash_{\mathbf{IL}}$ satisfies (Com) .

Proof. Let

$$\vdash = \{(\Gamma, \varphi) \in \wp(\text{Form}) \times \text{Form} \mid \text{there is a finite } \Gamma' \subseteq \Gamma \text{ such that } \Gamma' \vdash_{\mathbf{IL}} \varphi\}$$

Similar to the proof of Theorem 1, it suffices to show that \vdash satisfies (A) , $(\wedge I)$, $(\wedge E)$, (Exp) , (Mon) , (Cut) and $(\neg I)$. The proofs for the first 6 properties are the same as that in Theorem 1. Here we only need to deal with $(\neg I)$.

Assume that $\Gamma \cup \{\varphi\} \vdash \psi$ and $\Gamma \cup \{\varphi\} \vdash \neg\psi$. By definition and (Mon) there are two finite subsets Γ' and Γ'' of Γ such that $\Gamma' \cup \{\varphi\} \vdash_{\mathbf{IL}} \psi$ and $\Gamma'' \cup \{\varphi\} \vdash_{\mathbf{IL}} \neg\psi$. By (Mon) $\Gamma' \cup \Gamma'' \cup \{\varphi\} \vdash_{\mathbf{IL}} \psi$ and $\Gamma' \cup \Gamma'' \cup \{\varphi\} \vdash_{\mathbf{IL}} \neg\psi$. By $(\neg I)$ $\Gamma' \cup \Gamma'' \vdash_{\mathbf{IL}} \neg\varphi$. Since $\Gamma' \cup \Gamma''$ is a finite subset of Γ , $\Gamma \vdash \neg\varphi$. \square

Remark 8. By Lemma 11 and Theorem 4 the conclusions of Lemmas 3 and 4 apply to $\vdash_{\mathbf{IL}}$.

Lemma 12 (Negation Lemma of IL). Let $\Gamma \subseteq \text{Form}$ be **IL**-closed and **IL**-consistent and $\varphi \in \text{Form}$. The following are equivalent:

- (i) $\neg\varphi \notin \Gamma$;
- (ii) there is an **IL**-closed and **IL**-consistent $\Delta \subseteq \text{Form}$ such that $\Gamma \cup \{\varphi\} \subseteq \Delta$.

Proof. For the direction from (ii) to (i), suppose (towards a contradiction) that $\neg\varphi \in \Gamma$. Since $\Gamma \subseteq \Delta$, $\neg\varphi \in \Delta$. By (A) $\Delta \vdash_{\mathbf{IL}} \varphi$ and $\Delta \vdash_{\mathbf{IL}} \neg\varphi$, contradicting that Δ is **IL**-consistent.

For the direction from (i) to (ii), note that $\Gamma \cup \{\varphi\} \not\vdash_{\mathbf{IL}} \neg\varphi$: Suppose (towards a contradiction) that $\Gamma \cup \{\varphi\} \vdash_{\mathbf{IL}} \neg\varphi$. By (A) $\Gamma \cup \{\varphi\} \vdash_{\mathbf{IL}} \varphi$. By $(\neg I)$ $\Gamma \vdash_{\mathbf{IL}} \neg\varphi$. Since Γ is **IL**-closed, $\neg\varphi \in \Gamma$, contradicting (i).

By Lemma 3 there is a $\Delta \subseteq \text{Form}$ such that Δ is **IL**-closed and **IL**-consistent and $\Gamma \cup \{\varphi\} \subseteq \Delta$. \square

Third, we prove the soundness theorem.

Theorem 5 (Soundness Theorem of \mathbf{IL}). *For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \vdash_{\mathbf{IL}} \varphi$ implies that $\Gamma \models_I \varphi$.*

Proof. It suffices to show that \models satisfies (A), $(\wedge I)$, $(\wedge E)$, (Exp), (Mon), (Cut) and $(\neg I)$. Since every I-model is a model, the proofs for the first 6 properties are the same as in Theorem 2. Here we only need to show $(\neg I)$.

Assume that $\Gamma \cup \{\varphi\} \models \psi$ and $\Gamma \cup \{\varphi\} \models \neg\psi$. Let $\mathfrak{M} = (W, \rightarrow, V)$ be an I-model and $s \in W$ such that $\mathfrak{M}, s \models \Gamma$. Suppose (towards a contradiction) that $\mathfrak{M}, s \not\models \neg\varphi$. Then there is a $t \in W$ such that $s \rightarrow t$ and $\mathfrak{M}, t \models \varphi$. Since $s \rightarrow t$ and $\mathfrak{M}, s \models \Gamma$, by Remark 7 $\mathfrak{M}, t \models \Gamma$, so $\mathfrak{M}, t \models \Gamma \cup \{\varphi\}$. By the assumption $\mathfrak{M}, t \models \psi$ and $\mathfrak{M}, t \models \neg\psi$. By Reflexivity $t \rightarrow t$, so $\mathfrak{M}, t \not\models \psi$, contradicting $\mathfrak{M}, t \models \psi$. Therefore, $\mathfrak{M}, s \models \neg\varphi$. \square

Finally, we define the canonical frame of \mathbf{IL} , which is standard in the literature (Pages 132–133 in [4]), and prove the completeness theorem.

Definition 14. $\mathfrak{F}^{\mathbf{IL}} = (W^{\mathbf{IL}}, \rightarrow^{\mathbf{IL}})$ is the canonical frame of \mathbf{IL} , where:

- $W^{\mathbf{IL}} = \{\Gamma \subseteq \text{Form} \mid \Gamma \text{ is } \mathbf{IL}\text{-consistent and } \mathbf{IL}\text{-closed}\}$;
- $\rightarrow^{\mathbf{IL}} = \{(\Gamma, \Delta) \in W^{\mathbf{IL}} \times W^{\mathbf{IL}} \mid \Gamma \subseteq \Delta\}$.

The canonical model of \mathbf{IL} is an ordered pair $\mathfrak{M}^{\mathbf{IL}} = (\mathfrak{F}^{\mathbf{IL}}, V^{\mathbf{IL}})$ such that $\mathfrak{F}^{\mathbf{IL}}$ is the canonical frame of \mathbf{IL} and $V^{\mathbf{IL}} : \text{PV} \rightarrow \wp(W^{\mathbf{IL}})$ is a function such that, for each $p \in \text{PV}$, $V^{\mathbf{IL}}(p) = \{\Gamma \in W^{\mathbf{IL}} \mid p \in \Gamma\}$.

Lemma 13 (Canonical Model Lemma of \mathbf{IL}).

1. $\rightarrow^{\mathbf{IL}}$ satisfies Reflexivity and Transitivity.
2. $\mathfrak{M}^{\mathbf{IL}}$ is an I-model.
3. For any $\Gamma \in W^{\mathbf{IL}}$ and $\varphi \in \text{Form}$, $\mathfrak{M}^{\mathbf{IL}}, \Gamma \models \varphi$ if and only if $\varphi \in \Gamma$.

Proof. For Item 1, $\rightarrow^{\mathbf{IL}} = \subseteq$ satisfies Reflexivity and Transitivity.

For Item 2, for any $p \in \text{PV}$ and $\Gamma, \Delta \in W^{\mathbf{IL}}$ such that $\Gamma \subseteq \Delta$ and $\Gamma \in V^{\mathbf{IL}}(p)$, $p \in \Gamma$, so $p \in \Delta$, i.e. $\Delta \in V^{\mathbf{IL}}(p)$.

For Item 3, the proof is the same as that of Item 2 in Lemma 6, besides that here we use Lemma 12 instead of Lemma 5. \square

Theorem 6 (Completeness Theorem of \mathbf{IL}). *For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \models_I \varphi$ implies that $\Gamma \vdash_{\mathbf{IL}} \varphi$.*

Proof. Assume that $\Gamma \not\vdash_{\mathbf{IL}} \varphi$. By Lemma 3 there is a $\Delta \in W^{\mathbf{IL}}$ such that $\Gamma \subseteq \Delta$ and $\varphi \notin \Delta$. By Lemma 13 $\mathfrak{M}^{\mathbf{IL}}, \Delta \models \Gamma$ and $\mathfrak{M}^{\mathbf{IL}}, \Delta \not\models \varphi$. Since $\mathfrak{M}^{\mathbf{IL}}$ is an I-model, $\Gamma \not\models_I \varphi$. \square

B Ortho-logic

In this appendix, we apply the techniques developed before to prove the soundness and completeness theorem for ortho-logic **OL**.

Our axiomatization of ortho-logic is essentially the same as that on Pages 158–159 in [5], and the results in this appendix are all in [5] (with or without proofs). Here we give detailed proofs using the results in this paper.

We start with a remark about the semantics.

Remark 9. Since every O-model is a model, by Lemmas 1 and 10 in an O-model $\|\varphi\|$ is bi-orthogonally closed for each $\varphi \in \text{Form}$.

Second, we prove some results about the syntactic consequence relation \vdash_{OL} .

Remark 10. By definition $\vdash_{\text{PL}} \subseteq \vdash_{\text{OL}}$.

Theorem 7 (Compactness Theorem of OL). \vdash_{OL} satisfies (Com).

Proof. Let

$$\vdash = \{(\Gamma, \varphi) \in \wp(\text{Form}) \times \text{Form} \mid \text{there is a finite } \Gamma' \subseteq \Gamma \text{ such that } \Gamma' \vdash_{\text{OL}} \varphi\}$$

Similar to the proof of Theorem 1, it suffices to show that \vdash satisfies (A), $(\wedge\text{I})$, $(\wedge\text{E})$, (Exp), (Mon), (Cut), $(\neg\text{Iw})$, (Ctr), $(\neg^2\text{I})$ and $(\neg^2\text{E})$. The proofs for the first 8 properties are the same as that in Theorem 1. Here we only need to deal with $(\neg^2\text{I})$ and $(\neg^2\text{E})$.

By $(\neg^2\text{I})$ $\{\varphi\} \vdash_{\text{OL}} \neg\neg\varphi$. Since $\{\varphi\}$ is finite, $\Gamma \cup \{\varphi\} \vdash \neg\neg\varphi$. Similarly we can show that $\Gamma \cup \{\neg\neg\varphi\} \vdash \varphi$. \square

Remark 11. By Theorem 7 the conclusions of Lemmas 3, 4 and 5 apply to \vdash_{OL} .

Third, we prove the soundness theorem.

Theorem 8 (Soundness Theorem of OL). For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \vdash_{\text{OL}} \varphi$ implies that $\Gamma \models_{\text{O}} \varphi$.

Proof. Similar to the proof of Theorem 2, it suffices to show that \models satisfies (A), $(\wedge\text{I})$, $(\wedge\text{E})$, (Exp), (Mon), (Cut), $(\neg\text{Iw})$, (Ctr), $(\neg^2\text{I})$ and $(\neg^2\text{E})$. Since every O-model is a model, the proofs for the first 8 properties are the same as in Theorem 2. Here we only need to show $(\neg^2\text{I})$ and $(\neg^2\text{E})$.

Let $\mathfrak{M} = (W, \rightarrow, V)$ be an O-model and $s \in W$ such that $\mathfrak{M}, s \models \Gamma \cup \{\varphi\}$. Then $s \in \|\varphi\|$. By Remark 9 $s \in -(\|\varphi\|)$. By Remark 3 $-(\|\varphi\|) = \|\neg\neg\varphi\|$. Hence $s \in \|\neg\neg\varphi\|$, i.e. $\mathfrak{M}, s \models \neg\neg\varphi$. Therefore, $\Gamma \cup \{\varphi\} \models_{\text{O}} \neg\neg\varphi$.

Similarly we can show that $\Gamma \cup \{\neg\neg\varphi\} \models_{\text{O}} \varphi$. \square

Next, we define the canonical model of **OL** in exactly the same way as that of **PL** and thus as that in the literature [5,6]. We will see that the canonical model of **OL** is an O-model.

Definition 15. $\mathfrak{F}^{\mathbf{OL}} = (W^{\mathbf{OL}}, \rightarrow^{\mathbf{OL}})$ is the canonical frame of \mathbf{OL} , where:

- $W^{\mathbf{OL}} = \{\Gamma \subseteq \text{Form} \mid \Gamma \text{ is } \mathbf{OL}\text{-consistent and } \mathbf{OL}\text{-closed}\}$;
- $\rightarrow^{\mathbf{OL}} = \{(\Gamma, \Delta) \in W^{\mathbf{OL}} \times W^{\mathbf{OL}} \mid \text{there is no } \theta \in \text{Form} \text{ such that } \neg\theta \in \Gamma \text{ and } \theta \in \Delta\}$.

The canonical model of \mathbf{OL} is an ordered pair $\mathfrak{M}^{\mathbf{OL}} = (\mathfrak{F}^{\mathbf{OL}}, V^{\mathbf{OL}})$ such that $\mathfrak{F}^{\mathbf{OL}}$ is the canonical frame of \mathbf{OL} and $V^{\mathbf{OL}} : \text{PV} \rightarrow \wp(W^{\mathbf{OL}})$ is a function such that, for each $p \in \text{PV}$, $V^{\mathbf{OL}}(p) = \{\Gamma \in W^{\mathbf{OL}} \mid p \in \Gamma\}$.

Lemma 14 (Canonical Model Lemma of OL)

1. $\rightarrow^{\mathbf{OL}}$ satisfies Reflexivity and Symmetry.
2. $\mathfrak{M}^{\mathbf{OL}}$ is an O-model.
3. For any $\Gamma \in W^{\mathbf{OL}}$ and $\varphi \in \text{Form}$, $\mathfrak{M}^{\mathbf{OL}}, \Gamma \models \varphi$ if and only if $\varphi \in \Gamma$.

Proof. For Item 1, Reflexivity can be proved in exactly the same way as that of $\rightarrow^{\mathbf{PL}}$. For Symmetry, assume that $\Gamma \not\rightarrow^{\mathbf{OL}} \Delta$. Then there is a $\theta \in \text{Form}$ such that $\neg\theta \in \Gamma$ and $\theta \in \Delta$. By $(\neg^2\text{I})$ $\Delta \cup \{\theta\} \vdash_{\mathbf{OL}} \neg\neg\theta$, i.e. $\Delta \vdash_{\mathbf{OL}} \neg\neg\theta$. Since Δ is \mathbf{OL} -closed, $\neg\neg\theta \in \Delta$. Then $\Delta \not\rightarrow^{\mathbf{OL}} \Gamma$, for $\neg\neg\theta \in \Delta$ and $\neg\theta \in \Gamma$.

For Item 2, first assume that $\Gamma \in V^{\mathbf{OL}}(p)$. Then, for each $\Delta \in W^{\mathbf{OL}}$ satisfying $\Gamma \rightarrow^{\mathbf{OL}} \Delta$, $\Delta \rightarrow^{\mathbf{OL}} \Gamma$ by Symmetry and $\Gamma \in V^{\mathbf{OL}}(p)$. Hence $\Gamma \in -(-V^{\mathbf{OL}}(p))$.

Second, assume that $\Gamma \notin V^{\mathbf{OL}}(p)$. By definition $p \notin \Gamma$. Since Γ is \mathbf{OL} -closed, $\Gamma \not\vdash_{\mathbf{OL}} p$. Then $\Gamma \not\vdash_{\mathbf{OL}} \neg\neg p$; otherwise, $\neg\neg p \in \Gamma$ and, since by $(\neg^2\text{E})$ $\Gamma \cup \{\neg\neg p\} \vdash_{\mathbf{OL}} p$, $\Gamma \vdash_{\mathbf{OL}} p$, contradicting that $\Gamma \not\vdash_{\mathbf{OL}} p$. By (A) $\neg\neg p \notin \Gamma$. By Lemma 5 there is a $\Delta \in W^{\mathbf{OL}}$ such that $\Gamma \rightarrow^{\mathbf{OL}} \Delta$ and $\neg p \in \Delta$. Since $\neg p \in \Delta$, by definition for each $\Theta \in W^{\mathbf{OL}}$, $\Delta \rightarrow^{\mathbf{OL}} \Theta$ implies that $p \notin \Theta$, i.e. $\Theta \notin V^{\mathbf{OL}}(p)$. Hence $\Delta \in -V^{\mathbf{OL}}(p)$. Since $\Gamma \rightarrow^{\mathbf{OL}} \Delta$, $\Gamma \notin -(-V^{\mathbf{OL}}(p))$.

The proof of Item 3 is the same as that of Item 2 in Lemma 6. □

Finally, we prove the completeness theorem.

Theorem 9 (Completeness Theorem of OL). For any $\Gamma \subseteq \text{Form}$ and $\varphi \in \text{Form}$, $\Gamma \models_{\mathbf{O}} \varphi$ implies that $\Gamma \vdash_{\mathbf{OL}} \varphi$.

Proof. Assume that $\Gamma \not\vdash_{\mathbf{OL}} \varphi$. By Lemma 3 there is a $\Delta \in W^{\mathbf{OL}}$ such that $\Gamma \subseteq \Delta$ and $\varphi \notin \Delta$. By Lemma 14 $\mathfrak{M}^{\mathbf{OL}}, \Delta \models \Gamma$ and $\mathfrak{M}^{\mathbf{OL}}, \Delta \not\models \varphi$. Since $\mathfrak{M}^{\mathbf{OL}}$ is an O-model, $\Gamma \not\models_{\mathbf{O}} \varphi$. □


References

1. Beth, E.: Semantic construction of intuitionistic logic. Koninklijke Nederlandse Akad. von Wetenschappen **19**(11), 357–388 (1956)
2. Birkhoff, G., von Neumann, J.: The logic of quantum mechanics. Ann. Math. **37**, 823–843 (1936)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
4. Chagrov, A., Zakharyashev, W.: Modal Logic. Clarendon Press, Oxford (1997)

5. Dalla Chiara, M., Giuntini, R.: Quantum logics. In: Gabbay, D., Guentner, F. (eds.) *Handbook of Philosophical Logic*, vol. 6, pp. 129–228. Kluwer Academic Publishers (2002)
6. Goldblatt, R.: Semantic analysis of orthologic. *J. Philos. Log.* **3**, 19–35 (1974)
7. Heyting, A.: *Intuitionism: An Introduction*. North-Holland Publishing (1956)
8. Kripke, S.: Semantical analysis of intuitionistic logic. In: Crossley, J., Dummett, M. (eds.) *Formal Systems and Recursive Functions*, pp. 92–130 (1965)



Meaning and Computing: Two Approaches to Computable Propositions

Ivo Pezlar^(✉) 

Czech Academy of Sciences, Institute of Philosophy, Jilska 1, 110 00 Prague, Czechia
pezlar@flu.cas.cz

Abstract. In this paper, we will be interested in the notion of a computable proposition. It allows for feasible computational semantics of empirical sentences, despite the fact that it is in general impossible to get to the truth value of a sentence through a series of effective computational steps. Specifically, we will investigate two approaches to the notion of a computable proposition based on constructive type theory and transparent intensional logic. As we will see, the key difference between them is their accounts of denotations of empirical sentences.

Keywords: Computable proposition · Sense–denotation distinction · Algorithmic theory of meaning · Procedural semantics · Constructive type theory · Transparent intensional logic

1 Introduction

In this paper, we will be interested in the following issue:

- Can we make sense of empirical sentences in computational terms given that it is generally impossible to get to the truth value of a sentence through a sequence of effective computational steps?

The answer we will put forward is positive, but it requires adoption of the notion of a computable proposition, i.e., a proposition that yields another proposition upon execution, not a truth value. These computable propositions will be understood as meanings of empirical sentences. We examine two possible approaches to this notion, namely, an approach based on constructive type theory (CTT, [14]) and an approach based on transparent intensional logic (TIL, [30]), and we try to clarify the relation between their respective notions of a computable proposition. But first we will look more closely at the general relationship between meaning and computing.

Work on this paper was supported by Grant Nr. 19-12420S from the Czech Science Foundation, GA ČR.

Let us start by considering the following mathematical object:¹ $2 + 2$. Using a common sense notion of computation, we can compute it and get the number 4. In what kind of a relationship does the object 4 stand to the object $2 + 2$? The standard answer is that the relation between $2 + 2$ and 4 can be understood in terms of an evaluation procedure. The number $2 + 2$ is a non-canonical form of the canonical number 4 and the latter can be reduced to the former by following appropriate reduction rules associated with the non-canonical operator $+$ used in constructing this non-canonical number. Thus, generally speaking, we can view the non-canonical objects as programs and their corresponding canonical objects as their values. For example, the program $2 + 2$ terminates at the value 4. Can we then conclude that the *meaning* of the object $2 + 2$ is its canonical value, i.e., the object 4 in this case? We can, but then we would also have to concede that all other programs that terminate at the same value (e.g., 2×2 , $5 - 1$ or $16 \div 4$, ...) have the same meaning, but this would be a hard pill to swallow for many. So what is the meaning of $2 + 2$, if not its canonical value? Whatever it is, it seems to be connected with the way we are computing its value, i.e., reducing it as a non-canonical object to a canonical one, not with the value itself.

Behind every non-canonical object a there is an unspoken question: ‘Can a be reduced to a canonical object?’ Could we, perhaps, view this question as roughly equivalent to the question: ‘Does a mean anything?’ In other words, are the non-canonical objects meaningful only insofar as they are reducible to their corresponding canonical forms?² This, however, seems to be too strong of a requirement. Generally, we seem to be understanding non-canonical objects just fine even in cases where we do not know (or cannot know) their corresponding canonical objects. Take, e.g., the following non-canonical object $16547 + 3^4$. It seems clear that we can understand it even if we do not know its value. If we compute it, we learn something new (the canonical form of this object), but the meaning of the original object seems to remain unchanged by this discovery – it does not seem to imbue $16547 + 3^4$ with more meaning than it had before the computation.

Probably the easiest way to make sense of this situation is to accept some form of Fregean meaning-denotation dichotomy: non-canonical objects are meanings ‘in themselves’ and they do not need to lead to any denotations in order to be intelligible for us. In other words, non-canonical objects do not become meaningful insofar as they are computable to canonical ones, they stand on their own, so to speak.

So far, we have talked about computation and meaning only in regards to mathematical objects, but can we adopt an analogous approach to the empirical

¹ For simplicity, we assume we are working directly with the mathematical objects, thus ignoring the syntactic layer for the sake of the semantic one. For example, consider the difference between a mathematical expression ‘ $2 + 2$ ’ and a mathematical object $2 + 2$: while the former can be reduced to the numeral ‘4’, the latter can be computed to the number 4.

² As is the case in, e.g., the Dummett–Prawitz-style proof-theoretic semantics (see [6, 19]).

discourse as well? Consider, e.g., the proposition: *Charles is a bachelor*. In what sense, if any, can we compute it? Clearly, we cannot compute it to its truth value since there is no general method how to compute propositions to their truth values. But maybe there is another way we can go about computing it, and thus carrying over the non-canonical and canonical object distinction?

Let us start with a simple example from intuitionistic logic.³ A proposition $\neg A$ is usually defined as $A \rightarrow \perp$ where \perp denotes absurdity. Note that the meaning of the proposition $\neg A$ is explained indirectly, since we have no proof conditions for it, strictly speaking, only for $A \rightarrow \perp$. In other words, if we want to inquire into the meaning of $\neg A$, first we have to transform it into $A \rightarrow \perp$. Arguably, this definitional transformation can be regarded as a basic computational step itself. And if so, then we can view $\neg A$ as a non-canonical proposition that can be computed to its canonical form $A \rightarrow \perp$.

The same approach, we believe, can be applied to empirical propositions as well. For example, the empirical proposition *Charles is a bachelor* can be viewed as a non-canonical proposition, i.e., a proposition that has no direct truth conditions. And if we want to inquire into its meaning, or rather its truth conditions, we have to transform it to its canonical form. In this case, it might be, for example: *Charles is a man* \wedge \neg (*Charles is married*). Thus, the empirical proposition *Charles is a bachelor* can be computed to *Charles is a man* \wedge \neg (*Charles is married*). The proposition *Charles is a bachelor* can then be understood as the meaning of the sentence ‘Charles is a bachelor’, while the proposition *Charles is a man* \wedge \neg (*Charles is married*) as its denotation. So, in this approach, canonical propositions are propositions that cannot be computed any further.⁴

From a historical perspective, the idea of entertaining computable propositions can be traced back to two different philosophical and logical sources: a constructive (intuitionistic) tradition concerned with the language of mathematics and logic that starts with [1] and a non-constructive (platonist) tradition concerned with natural language that starts later with [28]. In the rest of the paper, we examine two type-theoretic frameworks that – we believe – best represent these two traditions: Per Martin-Löf’s constructive type theory [14] and Pavel Tichý’s transparent intensional logic [30]. Specifically, we will look at how these systems approach the notion of a computable proposition, a task which can be split into two further questions: ‘What is a proposition?’ and ‘What is a computation?’⁵ In other words, our main aim will be to discuss the notion of a computable proposition from the perspectives of CTT and TIL and to try

³ This example as well as the whole idea of computing propositions to canonical forms not to truth values was proposed in [15], a paper presented at a workshop on Frege at the University of Leiden in August 25, 2001, transcribed by Bjørn Jespersen.

⁴ Since we are mainly interested in the notion of a computable proposition, we intentionally choose very basic examples of propositions such as ‘ $\neg A = A \supset \perp$ ’ or ‘Charles is a bachelor’ to keep the focus on the computability aspect rather than on the propositional aspect. To learn how to analyze more complex sentences in CTT, consult, e.g., [2, 12, 25]. For TIL, see, e.g., [7, 21, 23].

⁵ This investigation can be viewed as a follow up to the paper [18] that explores how these two systems approach mathematical and logical propositions.

to conceptually clarify the relation between their respective approaches to this notion. We will not aim to provide new technical developments for CTT or TIL.

Terminological note. Since we will be discussing two frameworks with different terminological backgrounds, we will try to simplify the vocabulary whenever reasonably possible. For example, we will call constructive sets of constructive type theory as types and hyperpropositions of transparent intensional logic as propositions. Deviations from the standard terminology such as these will always be pointed out throughout the text.

2 Constructive Tradition

Although it was probably Aarne Ranta [25] who first applied constructive type theory to a systematic study of natural language semantics,⁶ the general meaning as a computation approach can be traced much further, at least to the work of L. E. J. Brouwer on constructive mathematics, specifically on constructive logic and the proof-based account of logical constants, also known as the Brouwer-Heyting-Kolmogorov (or simply BHK) interpretation. The general idea is to explain the meaning of logical connectives not in terms of their truth conditions but in terms of their proof conditions. For the constructive propositional logic, we get the following explanations (see Table 1).

Table 1. The BHK interpretation of logical constants

A proof of the proposition	Consists of	Which can be formalized as
$A \wedge B$	a proof of A and a proof of B	$(a, b) : A \wedge B$
$A \vee B$	a proof of A or a proof of B	$i(a) : A \vee B$ or $j(b) : A \vee B$
$A \rightarrow B$	an effective method which takes any proof of A into a proof of B	$\lambda x.b(x) : A \rightarrow B$ (where $b(a)$ is a proof of B provided a is a proof of A)
\perp	there is no proof (absurdity)	–

This interpretation of logical constants, particularly of implication, served as a basis for another important discovery, specifically the observed analogy between proofs and programs and proposition and types.

2.1 Proofs as Programs

The intuition that logic and computing are somehow related has a long tradition – going back at least to Leibniz’s concept of the calculus ratiocinator – however,

⁶ Following Michael Dummett’s exploration of constructive principles outside the scope of mathematics [5] and Göran Sundholm’s analysis of donkey sentences using constructive type theory [27].

it hasn't been made precise until the discovery of the propositions as types principle.⁷ Briefly put, it identifies constructive proofs with programs.⁸

As hinted above, the propositions as types principle is closely related to the BHK interpretation of constructive logic and in its most common form it refers to the recognized correspondence between Gerhard Gentzen's intuitionistic natural deduction [8] and Alonzo Church's simply typed λ -calculus [3],⁹ which can be understood as a rudimentary functional programming language (see, e.g., [11]). Specifically, we get the following correspondences (see Table 2).

Table 2. The propositions as types principle

Natural deduction	λ -calculus
assumption	free variable
implication introduction (= 'deduction theorem')	λ -abstraction (= function definition)
implication elimination (= modus ponens)	β -reduction (= function application)
proposition	type
proof	(functional) program

In practice, this means that, e.g., $A \rightarrow B$ can be interpreted as an implicational proposition where A is the antecedent and B the consequent and simultaneously as a type of a function from objects of type A to objects of type B . Proving this proposition then corresponds to constructing an object of the type $A \rightarrow B$. Of course, these correspondences can be expanded (e.g., simplification of proofs understood as evaluation of programs, provability as a problem of type inhabitation, etc.) as well as generalized (introduction rules as constructors, elimination rules as destructors, etc.).

2.2 Constructive Type Theory

The proposition as types principle is one of the fundamental principles of constructive type theory (CTT, [14]) and it will help us to explain what is understood as a proposition in this framework. Specifically, under the propositions as types principle, the question 'What is a proposition?' becomes synonymous

⁷ Also known as the Curry-Howard correspondence or isomorphism ([4, 10]), although this name is rather unfortunate as it omits two other key figures of the discovery, namely N. G. de Bruijn and Per Martin-Lf.

⁸ For an excellent overview, see, e.g., [31].

⁹ For simplicity, we omit the Haskell Curry's side of the discovery – the fact that combinators from combinatory logic correspond to axioms in Hilbert-style calculus. For example, the combinator K (i.e., $Kxy = x$) corresponds to the axiom $A \rightarrow B \rightarrow A$.

with the question ‘What is a type?’¹⁰ In CTT, we specify a type by specifying what constitutes its canonical objects and when two canonical objects are equal. For example, the type N of natural numbers is specified by the following rules introducing canonical objects and equal canonical objects of type N :

$$0 : N \qquad 0 = 0 : N \qquad \frac{n : N}{s(n) : N} \qquad \frac{n = m : N}{s(n) = s(m) : N}$$

where $0 : N$ is a judgment stating that ‘0 is an object of type N ’. Thus, the type N is inhabited by canonical objects of the form $s(n)$ and 0. Furthermore, any object that reduces to a canonical object of a certain type is considered an object of that type as well.

This brings us to the notion of a computation. From the perspective of CTT, a computation is a process of reducing non-canonical objects to their canonical forms via the corresponding computation rules. For example, assuming we have defined 1 as $s(0)$, 2 as $s(1)$, etc., and $+$ as $a+0 = a : N$ and $a+s(b) = s(a+b) : N$, we can form a non-canonical natural number $2 + 2$ that can be computed to $s(2 + 1)$ (lazy evaluation) or fully evaluated to 4.¹¹

For the basic semantic scheme of CTT (based on [20]), see the Fig. 1. It implements the Fregean idea that meaning is a mode of presentation for picking out denotation (in modern terms, a program for computing denotation). Note that since we are able to state things like $2 + 2 = 4 : N$, it means that the result of a computation, i.e., 4 in this case, is an object of the same type N as the computation itself, i.e., $2 + 2$ (see Fig. 2). In other words, the levels of meanings and denotations are conflated, or rather, viewed as entities of the same category.¹²

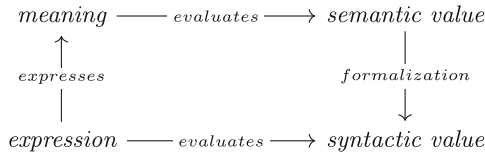


Fig. 1. Semantic scheme of CTT

¹⁰ Or more precisely, ‘What is a constructive set?’ since in CTT the word ‘type’ is typically reserved for the higher-order presentation of CTT, which we will not use here. In CTT, one typically starts with the notion of a set and defines a proposition in terms of it. The category of sets is then identified with the category of propositions, which is the way the propositions as types principle is adopted in CTT. For more, see, e.g., [16].

¹¹ The natural number $2 + 2$ is considered non-canonical because it does not follow the forms prescribed by the introduction rules for the type N , i.e., it is neither 0 nor does it have the form $s(n)$.

¹² See, e.g., [20], pp. 21–26. This, as we will see, is in contrast with TIL, where meanings and denotations are kept apart and viewed as entities of distinct kinds.

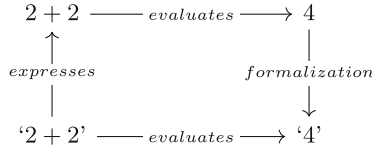


Fig. 2. An example of CTT semantics of non-empirical expressions

Now that we have briefly acquainted ourselves with the notions of a type and a computation, let us return back to propositions. So what exactly is a proposition in CTT? Analogously to what has been said above, to be able to judge that some A is a proposition, we have to know how to form its canonical objects and under what conditions two canonical objects are equal, i.e., provide rules for constructing its canonical proofs and equal canonical proofs.

Furthermore, a proposition A is considered true when we have a proof a of A . And in order to be able to judge that we have a proof a of A , we have to know that A is a proposition and that a is a method that yields upon execution a canonical proof of A as a value.¹³ For example, the canonical proofs (objects) and equal canonical proofs of the proposition $A \rightarrow B$ are specified by the following pair of rules:

$$\frac{[x : A] \quad b(x) : B}{\lambda x.b(x) : A \rightarrow B} \qquad \frac{[x : A] \quad b(x) = c(x) : B}{\lambda x.b(x) = \lambda x.c(x) : A \rightarrow B}$$

What is the canonical proof $\lambda x.b(x)$ of the proposition $A \rightarrow B$? It is an object – a lambda coding – that codes a function which takes a proof a of A and transforms it into a proof $b(a)$ of B (compare this with the BHK interpretation for implication in Fig. 1).

Analogously to the case above with non-canonical natural numbers such as, e.g., $2 + 2 : N$, we can have non-canonical proofs of propositions as well. For example, consider the following derivation:

$$\frac{f : (A \rightarrow B) \rightarrow (C \rightarrow D) \quad g : (A \rightarrow B)}{ap(f, g) : C \rightarrow D}$$

The proof $ap(f, g)$ of $C \rightarrow D$ is non-canonical because it does not have the form prescribed by the introduction rules for implicational propositions, i.e., $\lambda x.b(x)$. However, the non-canonical object $ap(f, g)$ is a method of obtaining a canonical object of $C \rightarrow D$. How to execute it? We have $f : (A \rightarrow B) \rightarrow (C \rightarrow D)$ which means that f is a method which yields a canonical object $\lambda x.c(x)$ of $(A \rightarrow B) \rightarrow (C \rightarrow D)$. Now, let us take $g : (A \rightarrow B)$ and substitute it for x in $c(x)$. Thus we get $c(g) : C \rightarrow D$ and when we execute $c(g)$ it will yield

¹³ It is an open question how to best carry over this approach to empirical discourse. See, e.g., [26, 27, 32].

a canonical object of $C \rightarrow D$, i.e., something of the form $\lambda y.d(y)$. For a more thorough exposition, see [14].

Note, however, that all the computations discussed so far were concerned exclusively with objects, specifically, with reducing non-canonical objects into canonical ones. For example, $2 + 2 : N$ was computed to $s(2 + 1) : N$ or $ap(f, g) : C \rightarrow D$ was computed to $c(g) : C \rightarrow D$. But what about computing with propositions themselves? This is, after all, our main interest. To extend computations towards propositions, we utilize the notion of a non-canonical type introduced in [9].

Through the propositions as types principle, this naturally applies to propositions as well. Following [9] (p. 91, Definition 6), let us define a non-canonical proposition A as a proposition that has a canonical proposition as the value.¹⁴ The fact that the non-canonical proposition A has the canonical proposition B as value will be denoted as

$$A \Rightarrow B : prop$$

and can be read as ‘a proposition A computes to a proposition B ’. The meaning of these proposition-computability judgments will be specified inferentially by the corresponding computation rules whose conclusions will take the general form $A \Rightarrow B : prop$ (see, e.g., the computation rule for $\neg comp$ below). Furthermore, again following [9] (p. 97), we adopt explicit definitions of non-canonical propositions of the following form: $D =_{def} A : prop$ where A is a proposition (not necessarily canonical) and D is a new undefined non-canonical proposition. Basically, it tells us that A is the value for the non-canonical proposition D . For example, our motivating case involving the definition of intuitionistic negation can be formalized as $\neg A =_{def} A \rightarrow \perp : prop$ which justifies a computational step from $\neg A$ to $A \rightarrow \perp$ that can be captured as follows $\neg A \Rightarrow A \rightarrow \perp : prop$. The corresponding proposition computation rule (read bottom-up) will be then as follows:

$$\frac{A \rightarrow \perp \Rightarrow A \rightarrow \perp : prop}{\neg A \Rightarrow A \rightarrow \perp : prop} \neg comp$$

with $\neg A$ being a non-canonical proposition and $A \rightarrow \perp$ being a canonical one. In other words, $A \rightarrow \perp$ is the value of the computable proposition $\neg A$. From the above considerations, it also follows that $\neg A = A \rightarrow \perp : prop$, i.e., that they are equal propositions.¹⁵

¹⁴ For a proper specification, see [9], especially sections §4. *Noncanonical sets and elements* and §5. *Nominal definitions*. It is also worth to note that non-canonical propositions/sets are already considered in [13], however, as opposed to [9], no dedicated proposition-computability judgments of the form $A \Rightarrow B : prop$ are used.

¹⁵ Of course, more complex reductions for other logical and/or mathematical propositions can be introduced. For example, [24] (pp. 41–42) shows how we can in CTT define, and thus reduce the propositional function $prime(x)$ (assuming $x : N$) into more basic concepts. Formulating the corresponding computation rule based on the provided definition is then straightforward.

Now, returning to our empirical case, let us assume the following definition: $bachelor(x) =_{def} man(x) \wedge \neg married(x)$ which can be unpacked, analogously as above, into the following three steps. First, we postulate that $bachelor(x) : prop$ assuming $x : A$, i.e., that $bachelor(x)$ is a propositional function taking as arguments objects of some type A (analogously with $married(x)$ and $man(x)$). Then we add the corresponding computation rule (again, assuming $x : A$):

$$\frac{man(x) \wedge \neg married(x) \Rightarrow man(x) \wedge \neg married(x) : prop}{bachelor(x) \Rightarrow man(x) \wedge \neg married(x) : prop}$$

Note that here we are taking man and $married$ as basic, further non-computable concepts.

And finally, we have to show that $bachelor(x)$ and $man(x) \wedge \neg married(x)$ are equal propositional functions, which follows from our computation rule: since both $bachelor(x)$ and $man(x) \wedge \neg married(x)$ have the same canonical forms, we can judge that they are equal propositional functions producing equal propositions. In other words, non-canonical propositions are equal, if their corresponding canonical propositions are equal. To sum it up, the proposition $man(Charles) \wedge \neg married(Charles) : prop$ is a canonical, i.e., further irreducible proposition and $bachelor(Charles)$ is a non-canonical proposition that can be computed to $man(Charles) \wedge \neg married(Charles) : prop$. For an example of the corresponding expanded semantic scheme for computable propositions, see Fig. 3.¹⁶

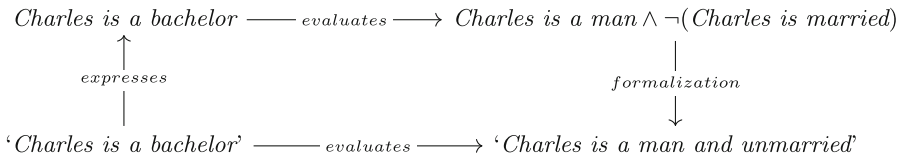


Fig. 3. An example of CTT semantics of empirical expressions

To conclude, in CTT we can approach the notion of a computable proposition via the notion of a non-canonical type introduced by [9], i.e., non-canonical constructive sets in a more standard terminology. This explication rests on two main principles: identification of propositions with types (i.e., the Curry-Howard isomorphism) and capturing computation in terms of the reduction of non-canonical objects to canonical ones.

In the following section, we will examine how TIL approaches the notion of a computable proposition.

¹⁶ Note that our approach has nothing further to say about the meaning of the conjuncts of this canonical proposition. More specifically, the meaning of atomic empirical propositions such as $man(Charles)$ is assumed to be given externally.

3 Non-constructive Tradition

As far as we know, it was Pavel Tichý [28] who first explicitly suggested to understand meanings of natural language expressions in terms of computations in the late 1960s, specifically in his paper *Intension in terms of Turing machines*:

... the fundamental relationship between sentence and procedure is obviously of a semantic nature; namely, the purpose of sentences is to record the outcome of various procedures. Thus e.g. the sentence ‘The liquid X is an acid’ serves to record that the outcome of a definite chemical testing procedure applied to X is positive ([28], p. 7).

Tichý saw in Turing machines an opportunity to finally explicate Frege’s notion of sense in rigorous terms. Eventually, however, Tichý replaced Turing machines with constructions. Constructions were understood as abstract computations codifying the procedures for computing denotations of the corresponding natural language expressions. Tichý subsequently developed his ideas into a system called transparent intensional logic presented in [30], which is still being actively developed (see, e.g., [7, 21, 23]).

3.1 Transparent Intensional Logic

As we have seen in CTT, the notion of a proposition is ultimately grounded in the notions of a type and a computation. In transparent intensional logic (TIL, [30]), the notion of a proposition can also be explained in these notions, however, the notions of a type and a computation in TIL differ from those of CTT.

First of all, the fundamental notion of computation in TIL, i.e., a construction, is much broader in comparison with the stricter constructive notion of effective computation found in CTT. It encompasses even non-computable and ill-specified procedures. As Tichý puts it:

[N]ot every construction is an algorithmic computation. An algorithmic computation is a sequence of *effective* steps, steps which consist in subjecting a manageable object (usually a symbol or a finite string of symbols) to a feasible operation. A construction, on the other hand, may involve steps which are not of this sort. [...]. As distinct from an algorithmic computation, a construction is an *ideal* procedure, not necessarily a mechanical routine for a clerk or a computing machine ([29], p. 526).

Thus, constructions should be viewed as idealized, abstract, not necessarily effective computations that need not be realizable either by a human or a machine. This more general approach towards computations is reflected in the treatment of types as well. In comparison with CTT, very little is required of them, no canonical objects have to be presented, no criterion of identity is required.¹⁷

¹⁷ From this perspective, TIL types are much closer to categories in CTT (i.e., types in proper CTT terminology), but even a category is a stricter notion as it has to come with a criterion of application and identity, which is not the case with TIL types.

Specifically, types are understood simply as non-empty pairwise disjoint collections ([30], p. 65). No further stipulations are given of what can and cannot constitute such a collection.

So what is a proposition in TIL?¹⁸ In order to understand what a proposition is we have to better understand Tichý’s notion of a construction. First, it is important to make clear what is really meant in TIL when it is said that propositions can be computed to yield their denotations (for the basic Fregean semantic scheme of TIL, see Fig. 4). In most current instances of TIL when one talks about denotations of propositions, they do not mean truth values (understood as references of sentences) but functions from possible worlds to truth values.¹⁹ Actual truth values cannot be, of course, computed and the task of determining them is delegated to empirical investigations. Thus, in TIL, we can compute the denotations of empirical sentences, but we cannot compute their references (i.e., truth values). Hence, the problem of computing truth values of empirical sentences is effectively sidestepped by distinguishing between denotations and references. This, however, does not apply to non-empirical expressions, where denotations and references are typically conflated.

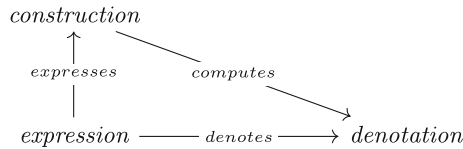


Fig. 4. Semantic scheme of TIL

For example, the non-empirical expression ‘ $2 + 2$ ’ is understood as expressing the arithmetical construction $[2 + 2]$ that computes (or constructs, in standard terminology) the number 4, i.e., an object of type ν , where ν is the type of natural numbers, which can be regarded as both the denotation and the reference of ‘ $2+2$ ’ (see Fig. 5).²⁰ In contrast, the empirical sentence ‘Charles is a bachelor’ expresses the proposition $\lambda w[[\mathbf{Bachelor} w] \mathbf{Charles}]$ which computes its denotation, i.e., a function of type $(o\omega)$, where o is the type of truth values and ω is the type of

¹⁸ In standard TIL terminology, the term ‘hyperproposition’ is used instead and the term ‘proposition’ is reserved for functions from possible worlds and time moments to truth values. We will diverge from this terminology.

¹⁹ In standard TIL, denotations of propositions are understood as functions from possible world *and* time moments to truth values, however, we omit the time parameter for simplicity here.

²⁰ The purpose of the bold font is, simply put, to distinguish between the constructional level and non-constructional level. Let us take, e.g., $\mathbf{2}$ and 2. What is the difference between them? The former is a construction, the latter is a constructed object. In other words, $\mathbf{4}$ can be understood as a trivial computation that yields the number 4 as a result. Furthermore, we now switch to the standard TIL notation with square brackets to better distinguish its expressions from those of CTT.

possible worlds. And only an empirical investigation can tell us what its reference is, i.e., whether it is true or false (see Fig. 6). Thus, when we are computing propositions in this sense, we are not searching for truth values, but for the corresponding functions from possible worlds to truth values.

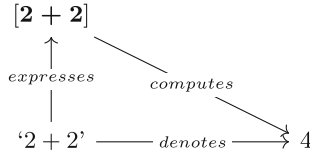


Fig. 5. An example of TIL semantics of non-empirical expressions

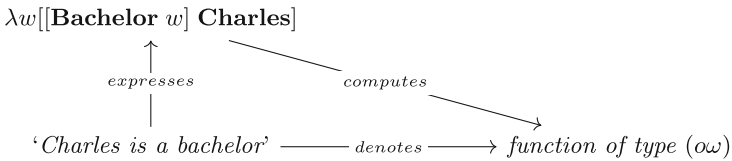


Fig. 6. An example of TIL semantics of empirical expressions

This notion of computation is, however, rather informal as it comes with no general instructions or computation rules telling us how should the evaluation of such computations proceed. Furthermore, it takes us from the level of constructions to a different level, specifically, a level of denotations, i.e., non-constructions (e.g., natural numbers, truth values, individuals).²¹ Consequently, this notion of a computation does not seem to be providing a satisfactory ground for explaining the notion of a computable proposition.

Fortunately for us, in TIL, another notion of a computation is identifiable that is much more similar to the notion of a computation in CTT. It comes with explicit computation rules (β -reductions) and the results of these computations do not leave the level of constructions, i.e., the results of such computations have the same type as the computations themselves. Following [17], let us call this new notion of a computation as a *syntactic* (‘machine-oriented’) notion of a computation and let us refer to the notion of a computation we have considered so far as a *semantic* (‘human-oriented’) notion of a computation. The main conceptual difference between these two notions is that a semantic computation takes us from constructions to their denotations, while a syntactic computation

²¹ Strictly speaking, in TIL we can have higher-order constructions that yield lower-order constructions as their denotations, but for simplicity of presentation, we omit these cases here.

takes us from constructions to other constructions, but never to their denotations. If we add the syntactic computation layer to the original semantic scheme of TIL presented earlier, we get a scheme closer to the one of CTT (see Fig. 7).²²

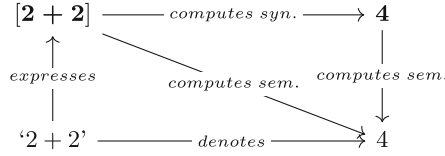


Fig. 7. An example of an expanded TIL semantics of non-empirical expressions

How is this relevant to computable propositions? When we are computing propositions in this syntactic sense, we are not searching for truth values (references) or functions from possible worlds to truth values (denotations) but rather for the simplest possible procedure for evaluating the truth conditions of the corresponding sentence. To better explain this, let us consider the following example. To mirror the initial intuitionistic case of defining $\neg A$ as $A \rightarrow \perp$, let us examine a classical case of defining $A \rightarrow B$ as $\neg A \vee B$. In TIL, this could be formalized as: $\lambda A B[A \rightarrow B] =_{def} \lambda A B[\neg A \vee B]$. Analogously to the CTT approach, the implicational proposition $\lambda A B[A \rightarrow B]$ can be understood as a defined proposition computable to its canonical form $\lambda A B[\neg A \vee B]$. The form of the corresponding computation rules would then be similar to those of CTT:²³

$$\frac{\lambda A B[\neg A \vee B] \Rightarrow \lambda A B[\neg A \vee B]}{\lambda A B[A \rightarrow B] \Rightarrow \lambda A B[\neg A \vee B]} \rightarrow comp$$

The same approach could also be applied to empirical cases. For example, when we compute the proposition $\lambda w[[\mathbf{Bachelor} w] \mathbf{Charles}]$ in this syntactic sense, the result we should expect is not a truth value, but rather the canonical form of the procedure for determining the truth value of the corresponding sentence, which might be, e.g., set by the following definition (**B** is an abbreviation of **Bachelor**, **C** of **Charles**, etc.):

$$\lambda w[[\mathbf{B} w] \mathbf{C}] =_{def} \lambda w[\lambda x[[\mathbf{M} w] x] \wedge [\neg[\mathbf{Mar} w] x]] \mathbf{C}]$$

which is, arguably, a more perspicuous test procedure (assuming it is simpler to check whether someone is a man and unmarried than that they are a bachelor).

²² Note that in comparison with CTT’s semantic scheme, here we have three levels of objects: syntactic (the expression ‘2 + 2’), semantic (the construction $[2 + 2]$), and denotational (the natural number 4). Recall that in CTT, there are only two levels: syntactic and semantic. In other words, in TIL the semantic level is distinguished further into constructional and denotational levels.

²³ This is only a sketch, for proper accounts of definitions/computation rules in TIL, see, e.g., [7], section 2.2.2 *Concepts and definitions* or [21], section 3.2 *Matches, sequents and rules*.

The computation rule will then be as follows (to save some space, let M represent the proposition $\lambda w[\lambda x[[\mathbf{M} w] x] \wedge [\neg[\mathbf{Mar} w] x]] \mathbf{C}]$):

$$\frac{M \Rightarrow M}{\lambda w[[\mathbf{B} w] \mathbf{C}] \Rightarrow M}$$

So, how can we compute with propositions in TIL? Analogously to CTT, we can approach this issue through the idea of defined and primitive propositions. However, in TIL the key difference between these two kinds of propositions will not be the presence or absence of direct proof conditions, but rather whether or not they contain primitive or derived concepts with respect to some conceptual system. In other words, by a canonical proposition we will understand a proposition that contains no derived concepts.

What are conceptual systems? Conceptual systems are, roughly put, applied instances of TIL tailored for a specific purpose (see [7, 22]). Thus, we can have, e.g., conceptual systems for propositional logic, conceptual systems for predicate logic, conceptual systems for reasoning about multiagent systems, etc. Formally, a conceptual system is a tuple $\langle Pr, Type, Var, C, Der \rangle$ where Pr is a finite class of primitive concepts P_1, \dots, P_k , i.e., basic objects of the system, $Type$ is an infinite class of types generated over a finite collection of base types (e.g., o, ι, ν for truth values, individuals, and natural numbers, respectively), Var is an infinite set of variables, countably infinite for each type from $Type$, C is the definition of kinds of basic TIL constructions, and Der is an infinite class of compound concepts derived from Pr and Var utilizing C . From this perspective, assuming a conceptual system CS_1 where \mathbf{B} and \mathbf{C} are simple, further undefined concepts, i.e., $Pr_1 = \{\mathbf{B}, \mathbf{C}\}$, the proposition $\lambda w[[\mathbf{B} w] \mathbf{C}]$ would be considered as a canonical one. However, assuming some other conceptual system CS_2 where \mathbf{B} is treated as a derived concept with $Pr_2 = \{\mathbf{M}, \mathbf{Mar}, \wedge, \neg\}$, then the proposition $\lambda w[[\mathbf{B} w] \mathbf{C}]$ will be non-canonical and computable to the canonical proposition $\lambda w[\lambda x[[\mathbf{M} w] x] \wedge [\neg[\mathbf{Mar} w] x]] \mathbf{C}]$ (see Fig. 8), which represents the most direct procedure for evaluating the truth conditions of the corresponding sentence (in the given conceptual system). For our logical example, we could use a conceptual system CS_3 with $Pr_3 = \{\neg, \vee\}$, which would render the proposition $\lambda A B[A \rightarrow B]$ non-canonical and the proposition $\lambda A B[\neg A \vee B]$ canonical. Thus, the status of canonicity of propositions will depend on the choice of the underlying conceptual system.

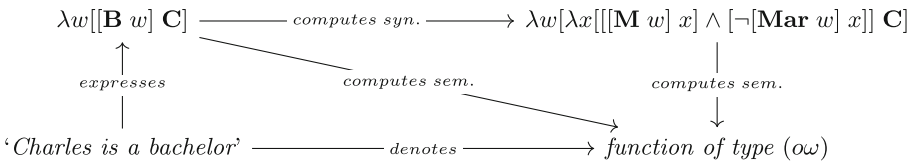


Fig. 8. An example of an expanded TIL semantics of empirical expressions

To conclude, in TIL we can approach the notion of a computable proposition via the notion of a (hyper)proposition, i.e., a construction that yields upon execution a function from possible worlds to truth values. This explication rests on three main principles: distinguishing between denotations and referents of empirical sentences, discerning between syntactic and semantic notions of computations, and capturing the notion of syntactic computation in terms of a reduction of non-canonical proposition to canonical ones within a scope of a given conceptual system.

4 Conclusion

In this paper, we have discussed two computational approaches to the semantics of empirical sentences that are based on the Fregean sense-denotation distinction. However, since the truth values – denotations in Frege’s approach – of sentences cannot be in general computed, these approaches had to modify the original scheme by changing what should be regarded as a denotation of an empirical sentence. Both approaches agree that it cannot be truth values and propose that it should be propositions. However, their respective notions of propositions differ. When a CTT-based approach proposes that a denotation of an empirical sentence is a proposition, what is meant is a canonical proposition and a proposition is understood intuitionistically, i.e., as a constructive set of its truth makers. On the other hand, when a TIL-based approach proposes that a denotation of an empirical sentence is a proposition, a proposition is understood as a function from possible worlds to truth values. Thus, in CTT the relation between meaning and denotation holds between objects of different types than in TIL. In the former it holds between propositions, in the latter it holds between propositions and functions.

This is not the only difference. Their respective notions of denoting understood as the relation between sense and denotation also diverge. CTT essentially identifies the notion of denoting with the notion of effective (syntactic) computation. In TIL, however, denoting is rather identified with the notion of constructing, i.e., the notion of not necessarily effective (semantic) computation. However, that does not mean that we cannot make sense of the notion of effectively (syntactically) computable propositions in TIL. It just means that if we compute with propositions in this sense, we can never get to their denotations, just to their canonical forms. This is in contrast to CTT, where canonical propositions and denotations are identified. The relation between propositions and their corresponding canonical forms in TIL is best seen as analogous to the same distinction in CTT, i.e., it should be viewed in terms of effective (syntactic) computability based on the underlying notion of definitional equality. So, from the TIL perspective, CTT conflates the notions of semantic and syntactic computability. Or, from the CTT perspective, TIL muddies the notion of computing by splitting it into two further notions of syntactic and semantic computability. It remains, however, an open question which of these approaches might generally prove to be more productive when dealing with natural language analysis.

To conclude, the key difference between CTT's and TIL's understanding of meaning of empirical sentences, i.e., computable propositions, lies in their respective ideas of what should constitute their corresponding denotations and how we should be able to reach them. CTT identifies denotations with canonical propositions, while TIL keeps them separate. In CTT, denoting has to be an effective procedure, while no such requirement is present in TIL. However, even though in TIL the procedure of getting from a proposition to its denotation is ineffective, the process of getting from a proposition to its canonical form is effective.

References

1. Brouwer, L.E.J.: *Over de Grondslagen der Wiskunde*. Ph.D. thesis, Universiteit van Amsterdam (1907)
2. Chatzikiyakidis, S., Luo, Z.: Adjectival and adverbial modification: the view from modern type theories. *J. Log. Lang. Inform.* **26**(1), 45–88 (2017). <https://doi.org/10.1007/s10849-017-9246-2>
3. Church, A.: A formulation of the simple theory of types. *J. Symb. Log.* **5**(02), 56–68 (1940). <https://doi.org/10.2307/2266170>
4. Curry, H.: Functionality in combinatory logic. *Proc. Natl. Acad. Sci.* **20**, 584–590 (1934). <https://doi.org/10.1073/pnas.20.11.584>
5. Dummett, M.: The philosophical basis of intuitionistic logic. *Stud. Log. Found. Math.* **80**(C), 5–40 (1975). [https://doi.org/10.1016/S0049-237X\(08\)71941-4](https://doi.org/10.1016/S0049-237X(08)71941-4)
6. Dummett, M.: *The Logical Basis of Metaphysics*. Duckworth, London (1991)
7. Duží, M., Jespersen, B., Materna, P.: *Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic*. Logic, Epistemology, and the Unity of Science, Springer, Dordrecht (2010). <https://doi.org/10.1007/978-90-481-8812-3>
8. Gentzen, G.: Untersuchungen über das logische Schließen. I. *Math. Z.* **39**(1), 176–210 (1935). <https://doi.org/10.1007/BF01201353>
9. Granström, J.G.: *Treatise on Intuitionistic Type Theory*. Logic, Epistemology, and the Unity of Science, Springer, Dordrecht (2011)
10. Howard, W.A.: The formulae-as-types notion of construction. In: Curry, H.B., Hindley, J.R., Seldin, J.P. (eds.) *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, London (1980)
11. Landin, P.J.: Correspondence between ALGOL 60 and Church's lambda-notation: part I. *Commun. ACM* **8**(2), 89–101 (1965). <https://doi.org/10.1145/363744.363749>
12. Luo, Z.: Formal semantics in modern type theories with coercive subtyping. *Linguist. Philos.* **35**(6), 491–513 (2012). <https://doi.org/10.1007/s10988-013-9126-4>
13. Martin-Löf, P.: Constructive mathematics and computer programming. In: Cohen, J.L., et al. (eds.) *Logic, Methodology and Philosophy of Science VI, 1979*, pp. 153–175. North-Holland, Amsterdam (1982)
14. Martin-Löf, P.: *Intuitionistic type theory: Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980*. Bibliopolis, Napoli (1984)
15. Martin-Löf, P.: *The sense/reference distinction in constructive semantics (manuscript)* (2001)
16. Nordström, B., Petersson, K., Smith, J.M.: *Programming in Martin-Löf's Type Theory: An Introduction*. Clarendon Press, Oxford (1990)

17. Pezlar, I.: On two notions of computation in transparent intensional logic. *Axiomathes* **29**(2), 189–205 (2018). <https://doi.org/10.1007/s10516-018-9401-7>
18. Pezlar, I.: Algorithmic theories of problems. a constructive and a non-constructive approach. *Log. Log. Philos.* **26**(4), 473–508 (2017). <https://doi.org/10.12775/LLP.2017.010>
19. Prawitz, D.: Meaning Approached Via Proofs. *Synthese* **148**(3), 507–524 (2006). <https://doi.org/10.1007/s11229-004-6295-2>
20. Primiero, G.: *Information and Knowledge*. Springer, Dordrecht (2008)
21. Raclavský, J.: *Belief Attitudes, Fine-Grained Hyperintensionality and Type-Theoretic Logic*. College Publications, London (2020)
22. Raclavský, J., Kuchyňka, P.: Conceptual and derivation systems. *Log. Log. Philos.* **20**(1–2), 159–174 (2011). <https://doi.org/10.12775/LLP.2011.008>
23. Raclavský, J., Kuchyňka, P., Pezlar, I.: *Transparentní intenzionální logika jako charakteristica universalis a calculus ratiocinator*. Masaryk University Press (Muni-press), Brno (2015)
24. Rahman, S., McConaughy, Z., Klev, A., Clerbout, N.: A brief introduction to constructive type theory. In: *Immanent Reasoning or Equality in Action*. LAR, vol. 18, pp. 17–55. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91149-6_2
25. Ranta, A.: *Type-Theoretical Grammar*. Clarendon Press, Oxford (1994)
26. Stovall, P.: Proof-theoretic semantics and the interpretation of atomic sentences. In: Sedlár, I., Blichá, M. (eds.) *The Logica Yearbook 2019*. College Publications, London (2020)
27. Sundholm, G.: Proof theory and meaning. In: Gabbay, D., Guentner, F. (eds.) *Handbook of Philosophical Logic*, vol. 166, pp. 471–506. Springer, Dordrecht (1986). https://doi.org/10.1007/978-94-009-5203-4_8
28. Tichý, P.: Intension in terms of Turing machines. *Stud. Log.* **24**(1), 7–21 (1969). <https://doi.org/10.1007/BF02134290>
29. Tichý, P.: Constructions. *Philos. Sci.* **53**(4), 514–534 (1986). <https://doi.org/10.1086/289338>
30. Tichý, P.: *The Foundations of Frege’s Logic*. de Gruyter, Berlin (1988)
31. Wadler, P.: Propositions as Types. *Commun. ACM* **58**(12), 75–84 (2015). <https://doi.org/10.1145/2699407>
32. Więckowski, B.: A constructive type-theoretical formalism for the interpretation of subatomically sensitive natural language constructions. *Stud. Log.* **100**(4), 815–853 (2012). <https://doi.org/10.1007/s11225-012-9431-x>



Modal Logic via Global Consequence

Xuefeng Wen^(✉)

Institute of Logic and Cognition, Department of Philosophy, Sun Yat-sen University,
Guangzhou 510275, China

Abstract. In modal logic, semantic consequence is usually defined locally by truth preservation at all worlds in all models (with respect to a class of frames). It can also be defined globally by truth preservation in all models (with respect to a class of frames). The latter is called global consequence, which is much less studied than the standard local one. In this paper we first study the relationship between local and global consequence. Then we give some correspondence results for global consequence. Finally, we illustrate two applications of global consequence, connecting it with informational consequence and update consequence proposed in formal semantics. Some results in the paper are already known, which are collected in the paper for the sake of completeness. The others appear to be new. We suggest that global consequence is not only interesting theoretically, but also useful for applications.

Keywords: Modal logic · Local consequence · Global consequence · Domain semantics · Update semantics

1 Introduction

Given a class of frames \mathbb{F} , the inference from Γ to φ is valid with respect to \mathbb{F} , if for every world w in every model $\mathfrak{M} = (W, R, V)$ such that $\mathfrak{F} = (W, R)$ is a frame in \mathbb{F} , if all formulas in Γ are true at w in \mathfrak{M} then φ is also true at w in \mathfrak{M} . This is called the local consequence (or local validity) in modal logic, which is the standard one. Another notion called global consequence (or global validity) in modal logic is also defined in the literature (e.g. in [1]). The inference from Γ to φ is globally valid with respect to \mathbb{F} , if for every model $\mathfrak{M} = (W, R, V)$ such that $\mathfrak{F} = (W, R)$ is a frame in \mathbb{F} , if all formulas in Γ are true in \mathfrak{M} then φ is also true in \mathfrak{M} , where a formula is true in a model if it is true at all worlds in the model.¹ Compared to local consequence, global consequence is much less

¹ Note that local and global consequence are polysemous in the literature. For example, in [15, p. 37], global consequence is defined as the preservation of frame validity, whereas local consequence is defined as the preservation of frame validity at every world. We are not talking about local and global consequence in this sense. In [4, 5], the authors also contrast the local and global perspective in modal logic, where ‘local’ means truth at a world, and ‘global’ means truth in a model. Our use of locality and globality is in line with theirs. But we are interested in global *consequence*, namely, the preservation of global truth, whereas they studied (modal definability by) global *truth* itself.

studied. Notable exceptions include, [7, 11] and [12]. Kracht [11] studied global consequence from an algebraic point of view systematically, obtaining a lot of meta properties (like finite model property, interpolation, etc.) of modal logics with global consequence. Fitting [7] integrated local and global consequence into a ternary relation, and proved completeness for various kinds of proof systems. Ma and Chen [12] presented Gentzen-style sequent calculi for global consequence. In this paper, we study global consequence within the standard relational semantics of modal logic, emphasizing its connection to local consequence and to some other consequence notions, which are popular in philosophical logic and formal semantics.

The remaining part of the paper is organized as follows. Section 2 shows the relationship between local consequence and global consequence. Section 3 gives a general correspondence result for global consequence and its typical instances. Section 4 illustrates two applications of global consequence, connecting it with informational consequence and update consequence. Section 5 concludes the paper and suggests future work. Some results in the paper are already known, which are collected in the paper for the sake of completeness. The others are supposed to be new. All the proofs for the results in Sect. 2 and Sect. 3 are put in the appendix.

2 Relationship Between Local and Global Consequence

In the sequel, we consider only normal modal logics. Let \mathcal{L}_0 be the classical propositional language, \mathcal{L}_\square the basic modal language. We use \Vdash for satisfaction relation. We write $\mathfrak{M}, w \Vdash \Gamma$ if $\mathfrak{M}, w \Vdash \varphi$ for all $\varphi \in \Gamma$. We write $\mathfrak{M} \Vdash \varphi$ if $\mathfrak{M}, w \Vdash \varphi$ for all w in \mathfrak{M} , and $\mathfrak{F} \Vdash \varphi$ if $\mathfrak{M} \Vdash \varphi$ for all models \mathfrak{M} based on the frame \mathfrak{F} . We denote by \mathbf{K} be the class of all frames, and \mathbf{M} the class of all models for \mathcal{L}_\square . We assume the readers are familiar with notations for typical classes of frames and axiomatic systems. For example, $\mathbf{K4}$ refers to the class of transitive frames, and $\mathbf{S5}$ the class of frames with equivalent relations; $\mathbf{K4}$ and $\mathbf{S5}$ denote their corresponding axiomatic systems, respectively.

Some other notations: $\Box^0\varphi := \varphi$, $\Box^{n+1}\varphi := \Box\Box^n\varphi$, $\Box_r\varphi := \varphi \wedge \Box\varphi$, $\Box\Gamma := \{\Box\varphi \mid \varphi \in \Gamma\}$, $\Box_r\Gamma := \{\Box_r\varphi \mid \varphi \in \Gamma\}$, $\Box^\omega\Gamma := \{\Box^n\psi \mid n \in \mathbb{N}, \psi \in \Gamma\}$, $\Box^\omega\varphi := \Box^\omega\{\varphi\}$. Let us recall the definitions of local semantic consequence and global semantic consequence in modal logic.

Definition 1. Given a class of (Kripke) frames \mathbf{F} ,

- (1) φ is a *local semantic consequence* of Γ , denoted, $\Gamma \models_{\mathbf{F}} \varphi$, if for any \mathfrak{F} in \mathbf{F} , for any model \mathfrak{M} based on \mathfrak{F} , and for any world w in \mathfrak{M} , $\mathfrak{M}, w \Vdash \Gamma$ implies $\mathfrak{M}, w \Vdash \varphi$;
- (2) φ is a *global semantic consequence* of Γ , denoted $\Gamma \models_{\mathbf{F}}^g \varphi$, if for any \mathfrak{F} in \mathbf{F} , for any model \mathfrak{M} based on \mathfrak{F} , if $\mathfrak{M} \Vdash \Gamma$ (i.e. for all w in \mathfrak{M} , $\mathfrak{M}, w \Vdash \Gamma$), then $\mathfrak{M} \Vdash \varphi$ (i.e. for all w in \mathfrak{M} , $\mathfrak{M}, w \Vdash \varphi$).

We summarize the results in this section as follows. Those with bold fonts are supposed to be new.

	Local by Global	Global by Local
restricting \mathcal{L}_\square , for all \mathbf{F}	Fact 2	Fact 2, Proposition 5
beyond \mathcal{L}_\square , for all \mathbf{F}	Proposition 7	Proposition 1, Proposition 2
within \mathcal{L}_\square , for some \mathbf{F}		Theorem 1, Theorem 2, Proposition 3, Corollary 1, Corollary 2, Corollary 3, Proposition 4

For a start, the following are well known results connecting local and global consequence.

Fact 1. For any class of frames \mathbf{F} , for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$,

- (1) $\vDash_{\mathbf{F}}^g \varphi$ iff $\vDash_{\mathbf{F}} \varphi$;
- (2) $\Gamma \vDash_{\mathbf{F}} \varphi$ implies $\Gamma \vDash_{\mathbf{F}}^g \varphi$.

Clause 1 says that locally and globally valid formulas coincide. Because of this, we are more interested in global consequence rather than globally valid formulas. Clause 2 says that local consequence is stronger than global consequence.

The following fact can be easily verified, which says that local consequence and global consequence coincide for modal-free formulas. This may be the reason why for modal-free reasoning, we do not distinguish local and global consequence.

Fact 2. Let $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_0$. Then for any class of frames \mathbf{F} , $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\Gamma \vDash_{\mathbf{F}} \varphi$.

The following two known results show that if we add some global operators in the language, then global consequence can always be defined by local consequence. This might be the reason why global consequence is somewhat neglected in the study of modal logic. Before that we need two definitions for the global operators.

Definition 2. Given a model $\mathfrak{M} = (W, R, V)$, define the operator \boxplus as follows,

$$\mathfrak{M}, w \Vdash \boxplus \varphi \text{ iff for all } u \in R^*(w), \mathfrak{M}, u \Vdash \varphi,$$

where R^* is the reflexive and transitive closure of R .

Definition 3. Given a model $\mathfrak{M} = (W, R, V)$, define the *universal operator* A as follows,

$$\mathfrak{M}, w \Vdash A\varphi \text{ iff for all } u \in W, \mathfrak{M}, u \Vdash \varphi.$$

Proposition 1 ([18], p. 159). For any class of frames \mathbf{F} , $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\boxplus \Gamma \vDash_{\mathbf{F}} \varphi$.

Proposition 2 ([9], Proposition 2.1). For any class of frames \mathbf{F} , $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $A\Gamma \vDash_{\mathbf{F}} \varphi$ iff $A\Gamma \vDash_{\mathbf{F}} A\varphi$.

If we consider only the class of frames \mathbf{K} , then global consequence can be defined by local consequence within the basic modal language, as the following proposition shows.

Proposition 3 ([1], p. 32). *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_K^g \varphi$ iff $\square^\omega \Gamma \vDash_K \varphi$.*

The proposition appears as an exercise in [1]. Instead of proving it directly, we generalize it as follows.

Theorem 1. *Let \mathbf{F} be any class of frames that is closed under point generated subframes. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\square^\omega \Gamma \vDash_{\mathbf{F}} \varphi$.*

Note that the direction from right to left does not require \mathbf{F} to be closed under point generated subframes. The other direction, however, does not hold for all \mathbf{F} , as the following fact shows.

Fact 3. *There exist a class of frames \mathbf{F} and formulas $\Gamma \cup \{\varphi\}$ such that $\Gamma \vDash_{\mathbf{F}}^g \varphi$ but $\square^\omega \Gamma \not\vDash_{\mathbf{F}} \varphi$.*

In [6, p. 425], the authors claim that the equivalence between $\Gamma \vDash_{\mathbf{F}}^g \varphi$ and $\square^\omega \Gamma \vDash_{\mathbf{F}} \varphi$ holds for all \mathbf{F} , which is incorrect by the above fact. But the closure under point generated subframes is not a necessary condition for the equivalence in Theorem 1, as the following fact shows.

Fact 4. *There exists a class of frames \mathbf{F} that is not closed under point generated subframes such that for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\square^\omega \Gamma \vDash_{\mathbf{F}} \varphi$.*

If we consider transitive frames, then the biconditional between local consequence and global consequence can be further simplified, as the following corollary shows. Recall that $\square_r \varphi := \varphi \wedge \square \varphi$ and $\square_r \Gamma := \{\square_r \varphi \mid \varphi \in \Gamma\}$.

Corollary 1. *Let \mathbf{F} be any class of transitive frames that is closed under point generated subframes. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\square_r \Gamma \vDash_{\mathbf{F}} \varphi$.*

To define global consequence by local consequence using only \square rather than \square_r , we could add another constraint for the class of frames.

Definition 4. A class of frames \mathbf{F} is *closed under irreflexive point extension*, if for any frame $\mathfrak{F} = (W, R)$ in \mathbf{F} , for any $w \in W$ with $\neg Rww$, any point extension $\mathfrak{F}' = (W', R')$ of \mathfrak{F} for w by $u \notin W$ is also in \mathbf{F} , where \mathfrak{F}' is defined as follows:

$$\begin{aligned} W' &= W \cup \{u\} \\ R' &= R \cup \{(u, w)\} \cup \{(u, w') \mid (w, w') \in R\} \end{aligned}$$

Theorem 2. *Let \mathbf{F} be any class of transitive frames that is closed under point generated subframes and irreflexive point extensions. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\square \Gamma \vDash_{\mathbf{F}} \square \varphi$.*

Corollary 2. *Let \mathbf{F} be any class of reflexive and transitive frames that is closed under point generated subframes. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{\mathbf{F}}^g \varphi$ iff $\square \Gamma \vDash_{\mathbf{F}} \square \varphi$ iff $\square \Gamma \vDash_{\mathbf{F}} \varphi$.*

The above corollary can also be derived from Theorem 1, noting that in any reflexive frame \mathfrak{F} , $\mathfrak{F} \Vdash \square_r \varphi \leftrightarrow \square \varphi$. The following result indicates a definition of global consequence in terms of local consequence for some familiar modal logics.

Corollary 3. *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, for any F in $\{K4, KD4, S4, S5\}$, $\Gamma \vDash_F^g \varphi$ iff $\square\Gamma \vDash_F \square\varphi$.*

The following proposition shows that to define global consequence by local consequence as above, sometimes various classes of frames are attainable.

Proposition 4. *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_{S5}^g \varphi$ iff $\square\Gamma \vDash_{S5} \square\varphi$ iff $\square\Gamma \vDash_{S5} \varphi$ iff $\square\Gamma \vDash_{K45} \square\varphi$ iff $\square\Gamma \vDash_{KD45} \square\varphi$*

If we restrict premises to be modal-free formulas, then global consequence can always be defined by local consequence (within the basic modal language), as the following proposition shows.

Proposition 5. *Let $\Gamma \subseteq \mathcal{L}_0$ and $\varphi \in \mathcal{L}_\square$. Then for any class of frames F , $\Gamma \vDash_F^g \varphi$ iff $\square^\omega\Gamma \vDash_F \varphi$.*

Some of the above results can also be given syntactically. Before that, we need some definitions. We denote by \vdash_S the (local) syntactic consequence for the axiomatic system S . We \vdash_S for modal logics in an eliminational way, as in most textbooks in modal logic (e.g. [3] and [1]), i.e. $\Gamma \vdash_S \varphi$ iff there is a finite subset $\Delta \subseteq \Gamma$ such that $\vdash_S \bigwedge \Delta \rightarrow \varphi$. The gist of this definition is to prevent the application of the rule of necessitation to the premises in Γ , since the inference from φ to $\square\varphi$ is generally not valid under local consequence. On the contrary, since we have $\varphi \vDash_F^g \square\varphi$ for any class of frames F , given a standard axiomatic system, the global syntactic consequence \vdash_S^g can be defined in the same way as in classical propositional logic, i.e. $\Gamma \vdash_S^g \varphi$ iff there is finite sequence of formulas $\varphi_1, \dots, \varphi_n$ such that $\varphi_n = \varphi$ and for each $i \leq n$ either $\varphi_i \in \Gamma$, or φ_i is an instance of an axiom scheme, or φ_i is obtained from previous formulas in the sequence by applying the rule(s) of the system. As a result, under global syntactic consequence, the rule of necessitation is applicable to the premises.

We say that S' is an axiomatic extension of S , if S' and S have the same inference rules, and all axioms of S are also axioms of S' . Now we have the following results.

Proposition 6. *Let S be any axiomatic extension of K . Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vdash_S^g \varphi$ iff $\square^\omega\Gamma \vdash_S \varphi$.*

The following two corollaries are straightforward.

Corollary 4. *Let S be any axiomatic extension of $K4$. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vdash_S^g \varphi$ iff $\square_r\Gamma \vdash_S \varphi$.*

Corollary 5. *Let S be any axiomatic extension of $S4$. Then for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vdash_S^g \varphi$ iff $\square\Gamma \vdash_S \varphi$ iff $\square\Gamma \vdash_S \square\varphi$.*

Can local syntactic consequence also be defined by global consequence? Yes, but much harder. We need to augment the language with both the universal operator and a special local operator.

Definition 5. Given a model \mathfrak{M} , define the ‘only here’ operator as follows:

$$\mathfrak{M}, w \Vdash O\varphi \text{ iff } \mathfrak{M}, w \Vdash \varphi \text{ and for all } w' \neq w, \mathfrak{M}, w' \nVdash \varphi.$$

Venema [18] presented the following result.

Proposition 7 ([18], p. 159). *For any class of frames F , for any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_{\square AO}$, for any $p \notin \text{Var}(\Gamma \cup \{\varphi\})$*

$$\Gamma \models_F \varphi \text{ iff } \{EOp\} \cup \{p \rightarrow \gamma \mid \gamma \in \Gamma\} \models_F^g p \rightarrow \varphi,$$

where E is the dual of the universal operator A in Definition 3, $\mathcal{L}_{\square AO}$ is the language extended by adding the operators A and O to \mathcal{L}_{\square} , and $\text{Var}(\Gamma \cup \{\varphi\})$ is the set of all propositional variables in all formulas in $\Gamma \cup \{\varphi\}$.

Though within the basic modal language global consequence cannot be reduced to local consequence generally, many properties (e.g. completeness, decidability, interpolation, etc.) for local consequence are preserved for global consequence for most logics. We do not discuss it here but refer the readers to [10, 11].

3 Global Correspondence

If we consider the correspondence between modal formulas and first-order frame properties, then there is nothing new for global consequence, since globally valid formulas coincide with locally valid formulas. But if we consider the correspondence between modally valid inferences and first-order frame properties, then it turns out to be much different for global consequence.

First, we have the following obvious fact.

Fact 5. $\varphi \models_F^g \Box\varphi$ for any class of frames F , in particular, we have

- (1) $\Box\varphi \models_F^g \Box\Box\varphi$
- (2) $\Diamond\varphi \models_F^g \Box\Diamond\varphi$

By contrast, $\Box\varphi \models_F \Box\Box\varphi$ if and only if F is transitive, and $\Diamond\varphi \models_F \Box\Diamond\varphi$ if and only if F is Euclidean.

Fact 6. $\Diamond\varphi \models_F^g \varphi$ iff F is globally isolated, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall x \exists y \forall z (Ryz \rightarrow z = x)$.

Fact 7. $\Diamond\Diamond\varphi \models_F^g \Diamond\varphi$ iff F is globally transitive, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall w \exists x \forall y \forall z (Rxy \wedge Ryz \rightarrow Rwx)$.

Fact 8. $\Diamond\Box\varphi \models_F^g \Box\varphi$ iff F is globally Euclidean, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall w \forall x \exists y \forall z (Rwx \wedge Ryz \rightarrow Rzx)$.

Fact 9. $\Box\varphi \models_F^g \varphi$ iff F is globally reflexive, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall x \exists y Ryx$.

Fact 10. $\varphi \models_F^g \Diamond\varphi$ iff F is globally inverse reflexive, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall x \exists y Rxy$.

Fact 11. $\Box\varphi \models_F^g \Diamond\varphi$ iff F is globally serial, i.e. for every $\mathfrak{F} = (W, R)$ in F , $\forall x \exists y \exists z (Ryz \wedge Rxz)$.

Fact 12. $\varphi \vDash_{\mathbb{F}}^g \Box \Diamond \varphi$ iff \mathbb{F} is globally symmetric, i.e. for every $\mathfrak{F} = (W, R)$ in \mathbb{F} , $\forall x \forall y \exists z (Rxy \rightarrow Ryz)$.

Fact 13. $\Diamond \Box \varphi \vDash_{\mathbb{F}}^g \varphi$ iff \mathbb{F} is globally inverse symmetric, i.e. for every $\mathfrak{F} = (W, R)$ in \mathbb{F} , $\forall x \exists y \forall z (Ryz \rightarrow Rzx)$.

Parallel to a famous general correspondence result for local consequence, we give a general correspondence result for global consequence, of which the above facts are all instances.

Theorem 3. $\Diamond^i \Box^j \varphi \vDash_{\mathbb{F}}^g \Box^k \Diamond^l \varphi$ iff every frame $\mathfrak{F} = (W, R)$ in \mathbb{F} satisfies the following condition

$$\forall w \forall x \exists y \forall z \exists u (R^k wx \wedge R^i yz \rightarrow R^l xu \wedge R^j zu).$$

Note that for local consequence, a valid inference often has an equivalent dual version. For example, $\Box \varphi \vDash \varphi$ iff $\varphi \vDash \Diamond \varphi$. This equivalence, however, does not hold for global consequence generally. For example, though $\Box \varphi \vDash^g \Box \Box \varphi$ holds for any class of frames, its dual $\Diamond \Diamond \varphi \vDash^g \Diamond \varphi$ holds only for globally transitive frames. This is a notable contrast between local and global consequence. We will indicate in the conclusion how this theoretical asymmetry might be used to explain our intuition concerning epistemic modal operators.

4 Applications

4.1 Informational Consequence

In [21] Yalcin advocated a non-classical consequence relation, called informational consequence. Yalcin noticed that if \Diamond denotes epistemic ‘might’ or ‘may’, then saying both φ and $\Diamond \neg \varphi$ seems inconsistent, which is not reflected in standard modal logic. So he proposed domain semantics and informational consequence (details below) to formalize this phenomenon. We will soon find that informational consequence is intimately related to global consequence.

Definition 6. A *domain model* is a pair $\mathfrak{D} = (W, V)$, where $W \neq \emptyset$ and $V : PV \rightarrow \wp(W)$ is a valuation on W . Given a domain model $\mathfrak{D} = (W, V)$, that φ is true at $(w, i) \in W \times \wp(W)$ in \mathfrak{D} , denoted $\mathfrak{D}, w, i \Vdash \varphi$, is inductively defined as follows, where $\mathfrak{D}, i \Vdash \varphi$ means for all $w \in i$, $\mathfrak{D}, w, i \Vdash \varphi$:

- $\mathfrak{D}, w, i \Vdash p$ iff $w \in V(p)$
- $\mathfrak{D}, w, i \Vdash \neg \varphi$ iff $\mathfrak{D}, w, i \not\Vdash \varphi$
- $\mathfrak{D}, w, i \Vdash \varphi \wedge \psi$ iff $\mathfrak{D}, w, i \Vdash \varphi$ and $\mathfrak{D}, w, i \Vdash \psi$
- $\mathfrak{D}, w, i \Vdash \Box \varphi$ iff $\mathfrak{D}, i \Vdash \varphi$

Definition 7 (Informational consequence). The inference from Γ to φ is *informationally valid*, denoted $\Gamma \vDash_I \varphi$, if for all domain models $\mathfrak{D} = (W, V)$ and $i \subseteq W$, $\mathfrak{D}, i \Vdash \Gamma$ implies $\mathfrak{D}, i \Vdash \varphi$.

It can be easily shown that under domain semantics, $\varphi \wedge \Diamond \neg \varphi \vDash_I \perp$. But this can also be achieved by global consequence for free.

Fact 14. $\varphi \wedge \Diamond \neg \varphi \vDash_F^g \perp$ for any class of frames F .

Proof. Suppose $\mathfrak{F}, V \Vdash \varphi \wedge \Diamond \neg \varphi$. Then $\mathfrak{F}, V \Vdash \varphi$ and $\mathfrak{F}, V \Vdash \Diamond \neg \varphi$. The former implies that $\mathfrak{F}, V \Vdash \Box \varphi$, which contradicts the latter.

In [2], Bledin convincingly argued that the rule of reduction to absurdity and constructive dilemma are not generally valid for natural language arguments. Rather, their correct forms should add some modal operators. More precisely, Bledin suggests that

- $\Gamma, \varphi \vDash_I \perp$ does not imply $\Gamma \vDash_I \neg \varphi$, instead we have $\Gamma, \varphi \vDash_I \perp \implies \Gamma \vDash_I \Diamond \neg \varphi$;
- $\Gamma, \alpha \vDash_I \varphi$ and $\Gamma, \beta \vDash_I \psi$ do not imply $\Gamma, \alpha \vee \beta \vDash_I \varphi \vee \psi$, instead we have $\Gamma, \alpha \vDash_I \varphi$ and $\Gamma, \beta \vDash_I \psi \implies \Gamma, \Box \alpha \vee \Box \beta \vDash_I \Box \varphi \vee \Box \psi$.

Bledin argued that informational consequence can perfectly predict the above desiderata. But global consequence can do the same job as well.

Fact 15. $\Gamma, \varphi \vDash_F^g \perp$ does not imply $\Gamma \vDash_F^g \neg \varphi$, instead for any reflexive and transitive F , we have $\Gamma, \varphi \vDash_F^g \perp \implies \Gamma \vDash_F^g \Diamond \neg \varphi$.

Proof. By Fact 14, we have $\Diamond \neg \varphi, \varphi \vDash_F^g \perp$ for any class of frames F . But by Fact 6, $\Diamond \neg \varphi \vDash_F^g \neg \varphi$ holds only for F that is globally isolated. For the remaining part, suppose $\Gamma \not\vDash_F^g \Diamond \neg \varphi$. Then there exists a model \mathfrak{M} with its underlying frame in F such that $\mathfrak{M} \Vdash \Gamma$ and $\mathfrak{M} \not\vDash \Diamond \neg \varphi$. By the latter there exists w in \mathfrak{M} such that $\mathfrak{M}, w \not\vDash \Diamond \neg \varphi$, i.e. $\mathfrak{M}, w \Vdash \Box \varphi$. Let \mathfrak{M}_w be the subframe of \mathfrak{M} generated by w . Then $\mathfrak{M}_w, w \Vdash \Box \varphi$. Since \mathfrak{M}_w is reflexive and transitive, we have $\mathfrak{M}_w \Vdash \varphi$. Thus $\Gamma, \varphi \not\vDash_F^g \perp$.

Fact 16. $\Gamma, \alpha \vDash_F^g \varphi$ and $\Gamma, \beta \vDash_F^g \psi$ do not imply $\Gamma, \alpha \vee \beta \vDash_F^g \varphi \vee \psi$, instead for any reflexive and transitive F , we have $\Gamma, \alpha \vDash_F^g \varphi$ and $\Gamma, \beta \vDash_F^g \psi \implies \Gamma, \Box \alpha \vee \Box \beta \vDash_F^g \Box \varphi \vee \Box \psi$.

Proof. By Fact 5, we have $p \vDash_F^g \Box p$ and $\neg p \vDash_F^g \Box \neg p$ for any class of frames F . But it is easily verified that $p \vee \neg p \not\vDash_{\{\mathfrak{F}\}}^g \Box p \vee \Box \neg p$, where $\mathfrak{F} = (\{1, 2\}, \{(1, 2), (2, 1)\})$. For the remaining part, suppose $\Gamma, \alpha \vDash_F^g \varphi$ and $\Gamma, \beta \vDash_F^g \psi$. Let \mathfrak{M} be any model with its underlying frame in F . Suppose $\mathfrak{M} \Vdash \Gamma$ and $\mathfrak{M} \Vdash \Box \alpha \vee \Box \beta$. Given any w in \mathfrak{M} , we have $\mathfrak{M}, w \Vdash \Box \alpha \vee \Box \beta$. Then either $\mathfrak{M}, w \Vdash \Box \alpha$ or $\mathfrak{M}, w \Vdash \Box \beta$. Since \mathfrak{M} is reflexive and transitive, if the former holds, then $\mathfrak{M}_w \Vdash \alpha$. By $\Gamma, \alpha \vDash_F^g \varphi$, we have $\mathfrak{M}_w \Vdash \varphi$. Thus $\mathfrak{M}, w \Vdash \Box \varphi$. If the latter holds, then $\mathfrak{M}_w \Vdash \beta$. By $\Gamma, \beta \vDash_F^g \psi$, we have $\mathfrak{M}_w \Vdash \psi$. Thus $\mathfrak{M}, w \Vdash \Box \psi$. Hence, $\mathfrak{M}, w \Vdash \Box \varphi \vee \Box \psi$. Since w is arbitrary, we have $\mathfrak{M} \Vdash \Box \varphi \vee \Box \psi$, as required.

Indeed, Schulz proved the following general result.

Theorem 4 ([14], Theorem 2.1). *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\Box$, $\Gamma \vDash_I \varphi$ iff $\Box \Gamma \vDash_{S5} \Box \varphi$.*

By Proposition 4, the following corollary easily follows.

Corollary 6. *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_\square$, $\Gamma \vDash_I \varphi$ iff $\Gamma \vDash_{S5}^g \varphi$.*

Compared to Theorem 4, it appears that Corollary 6 better characterizes informational consequence, since the former uses local consequence and by Proposition 4, with local consequence not only S5 can be used, but also K45 and KD45 are attainable. But with global consequence, such multiple correspondence disappears. On the other hand, Facts 15 and 16 show that if we just need to satisfy the desiderata above proposed by Yalcin and Bledin, it is possible to consider only S4 instead of S5, as far as global consequence is used.

4.2 Update Consequence

Update semantics proposed by Veltman in [17] is also a popular semantics for natural languages. In update semantics, two conjunctions can be defined. One is static (as in [17]), the other dynamic (as in [13, 19, 20]). To differentiate them, we consider the following language.

Given the set of propositional variables PV , the language \mathcal{L}_\square is defined as follows:

$$\mathcal{L}_\square; \exists \varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi; \varphi) \mid \square\varphi,$$

where $p \in PV$, \wedge is the static conjunction and $;$ the dynamic one. We stipulate that both \wedge and $;$ are left associated, so that $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ abbreviates $(\varphi_1 \wedge \varphi_2) \wedge \varphi_3$, and $\varphi_1; \varphi_2; \varphi_3$ abbreviates $(\varphi_1; \varphi_2); \varphi_3$, etc.

Definition 8. An *update model* is a pair $\mathfrak{U} = (W, V)$, where $W \neq \emptyset$ and $V : PV \rightarrow \wp(W)$ is a valuation on W . Given an update model $\mathfrak{U} = (W, V)$, define the update function $[\cdot]_{\mathfrak{U}} : \wp(W) \times \mathcal{L}_\square \rightarrow \wp(W)$ on \mathfrak{U} as follows.

- $s[p]_{\mathfrak{U}} = s \cap V(p)$
- $s[\neg\varphi]_{\mathfrak{U}} = s - s[\varphi]_{\mathfrak{U}}$
- $s[\varphi \wedge \psi]_{\mathfrak{U}} = s[\varphi]_{\mathfrak{U}} \cap s[\psi]_{\mathfrak{U}}$
- $s[\varphi; \psi]_{\mathfrak{U}} = s[\varphi]_{\mathfrak{U}}[\psi]_{\mathfrak{U}}$
- $s[\square\varphi]_{\mathfrak{U}} = \{w \in s \mid s[\varphi]_{\mathfrak{U}} = s\}$

We say that s supports φ in \mathfrak{U} , denoted $\mathfrak{U}, s \Vdash_U \varphi$, if $s[\varphi]_{\mathfrak{U}} = s$. We write $\mathfrak{U}, s \Vdash_U \Gamma$ iff $\mathfrak{U}, s \Vdash_U \varphi$ for all $\varphi \in \Gamma$.

It is easily seen that for any \mathfrak{U} and s in \mathfrak{U} , for any $\varphi \in \mathcal{L}_\square$, $s[\varphi]_{\mathfrak{U}} \subseteq s$.

Definition 9 (Update consequence). We say that φ is an *update consequence* of Γ , denoted $\Gamma \vDash_U \varphi$, if for all update models $\mathfrak{U} = (W, V)$, for all information states $s \subseteq W$, $\mathfrak{U}, s \Vdash_U \Gamma$ implies $\mathfrak{U}, s \Vdash_U \varphi$. We say that φ is a *sequential update consequence* of the sequence $\gamma_1, \dots, \gamma_n$, denoted $\gamma_1, \dots, \gamma_n \vDash_{SU} \varphi$, if for all update models $\mathfrak{U} = (W, V)$, for all information states $s \subseteq W$, $s[\gamma_1]_{\mathfrak{U}} \cdots [\gamma_n]_{\mathfrak{U}} \Vdash_U \varphi$.

Sometimes, another operator \rightarrow for indicative conditionals is also defined in update semantics (e.g. [8]), whose update function is given below.

- $s[\varphi \rightarrow \psi]_{\mathfrak{U}} = \{w \in s \mid s[\varphi]_{\mathfrak{U}}[\psi]_{\mathfrak{U}} = s[\varphi]_{\mathfrak{U}}\}$

It follows that \rightarrow can be defined by \Box and $;$ as the following fact shows.

Fact 17. For all \mathfrak{M} and s in \mathfrak{M} , $s[\varphi \rightarrow \psi]_{\mathfrak{M}} = s[\Box\neg(\varphi; \neg\psi)]_{\mathfrak{M}}$.

Now with \rightarrow , sequential update consequence can be reduced to update consequence.

Lemma 1. For any $\gamma_1, \dots, \gamma_n, \varphi \in \mathcal{L}_{\Box};$ $\gamma_1, \dots, \gamma_n \models_{SU} \varphi$ iff $\models_U (\gamma_1; \dots; \gamma_n) \rightarrow \varphi$ iff $\models_U \Box\neg(\gamma_1; \dots; \gamma_n; \neg\varphi)$.

Proof. Straightforward from the definitions.

Now we prove that update consequence can be defined by global consequence.

Definition 10. Given a relational model $\mathfrak{M} = (W, R, V)$, define the truth condition for $\varphi; \psi$ as follows.

- $\mathfrak{M}, w \Vdash \varphi; \psi$ iff $\mathfrak{M}, w \Vdash \varphi$ and $\mathfrak{M}^\varphi, w \Vdash \psi$, where $\mathfrak{M}^\varphi = (W^\varphi, R^\varphi, V^\varphi)$ is given below:

$$\begin{aligned} W^\varphi &= \{w \in W \mid \mathfrak{M}, w \Vdash \varphi\} \\ R^\varphi &= R \cap (W^\varphi \times W^\varphi) \\ V^\varphi(p) &= W^\varphi \cap V(p), \text{ for all } p \in PV. \end{aligned}$$

Given a relational model $\mathfrak{M} = (W, R, V)$, we write $\llbracket \varphi \rrbracket^{\mathfrak{M}}$ for the truth set of φ in \mathfrak{M} , i.e. $\llbracket \varphi \rrbracket^{\mathfrak{M}} = \{w \in W \mid \mathfrak{M}, w \Vdash \varphi\}$.

Lemma 2. For any update models $\mathfrak{M} = (W, V)$ and $\mathfrak{M}' = (W', V')$ such that $W \subseteq W'$ and $V = V' \upharpoonright_W$, for any $s \subseteq W$,

$$s[\varphi]_{\mathfrak{M}} = s[\varphi]_{\mathfrak{M}'}$$

Proof. By induction on φ .

- $\varphi = p$. Then $s[\varphi]_{\mathfrak{M}} = s[p]_{\mathfrak{M}} = s \cap V(p) = s \cap V'(p) = s[p]_{\mathfrak{M}'} = s[\varphi]_{\mathfrak{M}'}$.
- The Boolean cases are easily verified.
- $\varphi = \psi; \chi$. Then $s[\varphi]_{\mathfrak{M}} = s[\psi; \chi]_{\mathfrak{M}} = s[\psi]_{\mathfrak{M}}[\chi]_{\mathfrak{M}} = s[\psi]_{\mathfrak{M}}[\chi]_{\mathfrak{M}'} = s[\psi]_{\mathfrak{M}'}[\chi]_{\mathfrak{M}'} = s[\psi; \chi]_{\mathfrak{M}'} = s[\varphi]_{\mathfrak{M}'}$.
- $\varphi = \Box\psi$. Then $s[\varphi]_{\mathfrak{M}} = s[\Box\psi]_{\mathfrak{M}} = \{w \in s \mid s[\varphi]_{\mathfrak{M}} = s\} = \{w \in s \mid s[\varphi]_{\mathfrak{M}'} = s\} = s[\Box\psi]_{\mathfrak{M}'} = s[\varphi]_{\mathfrak{M}'}$.

Lemma 3. For any relational model $\mathfrak{M} = (W, R, V)$ with $R = W \times W$ and its underlying update model $\mathfrak{M}^{\mathfrak{M}} = (W, V)$, for any $\varphi \in \mathcal{L}_{\Box}$,

$$W[\varphi]_{\mathfrak{M}^{\mathfrak{M}}} = \llbracket \varphi \rrbracket^{\mathfrak{M}}$$

Hence, $\mathfrak{M}^{\mathfrak{M}}, W \Vdash_U \varphi$ iff $\mathfrak{M} \Vdash \varphi$.

Proof. By induction on φ .

- $\varphi = p \in PV$. Then $W[\varphi]_{\mathfrak{M}^{\mathfrak{M}}} = W[p]_{\mathfrak{M}^{\mathfrak{M}}} = W \cap V(p) = V(p) = \llbracket \varphi \rrbracket^{\mathfrak{M}}$.
- The Boolean cases are easily verified.

- $\varphi = \psi; \chi$. Then $W[\varphi]_{\mathfrak{U}^{\mathfrak{M}}} = W[\psi]_{\mathfrak{U}^{\mathfrak{M}}}[\chi]_{\mathfrak{U}^{\mathfrak{M}}} = \llbracket \psi \rrbracket^{\mathfrak{M}}[\chi]_{\mathfrak{U}^{\mathfrak{M}}} = W'[\chi]_{\mathfrak{U}^{\mathfrak{M}}} = W'[\chi]_{\mathfrak{U}'} = \llbracket \chi \rrbracket^{\mathfrak{M}^\psi} = \llbracket \psi; \chi \rrbracket^{\mathfrak{M}} = \llbracket \varphi \rrbracket^{\mathfrak{M}}$, where $W' = \llbracket \varphi \rrbracket^{\mathfrak{M}}$ and $\mathfrak{U}' = (W', V \upharpoonright_{W'})$. Note that the fourth identity follows from Lemma 2.
- $\varphi = \Box\psi$. Then $W[\varphi]_{\mathfrak{U}^{\mathfrak{M}}} = W[\Box\psi]_{\mathfrak{U}^{\mathfrak{M}}} = \begin{cases} W & \text{if } W[\psi]_{\mathfrak{U}^{\mathfrak{M}}} = W \\ \emptyset & \text{otherwise} \end{cases}$
 $= \begin{cases} W & \text{if } \llbracket \psi \rrbracket^{\mathfrak{M}} = W \\ \emptyset & \text{otherwise} \end{cases} = \begin{cases} \llbracket \Box\psi \rrbracket^{\mathfrak{M}} & \text{if } \llbracket \psi \rrbracket^{\mathfrak{M}} = W \\ \llbracket \Box\psi \rrbracket^{\mathfrak{M}} & \text{otherwise} \end{cases} = \llbracket \Box\psi \rrbracket^{\mathfrak{M}} = \llbracket \varphi \rrbracket^{\mathfrak{M}}$.

Lemma 4. *Given an update model $\mathfrak{U} = (W, V)$ and an information state $s \subseteq W$, define $\mathfrak{M}^s = (s, s \times s, V^s)$, where $V^s(p) = s \cap V(p)$. Then for any $\varphi \in \mathcal{L}_{\Box}$,*

$$s[\varphi]_{\mathfrak{U}} = \llbracket \varphi \rrbracket^{\mathfrak{M}^s}.$$

Hence, $\mathfrak{U}, s \Vdash_U \varphi$ iff $\mathfrak{M}^s \Vdash \varphi$.

Proof. By induction on φ .

- $\varphi = p \in PV$. Then $s[\varphi]_{\mathfrak{U}} = s[p]_{\mathfrak{U}} = s \cap V(p) = V^s(p) = \llbracket p \rrbracket^{\mathfrak{M}^s}$.
- The Boolean cases are easily verified.
- $\varphi = \psi; \chi$. Then $xs[\varphi]_{\mathfrak{U}} = s[\psi; \chi]_{\mathfrak{U}} = s[\psi]_{\mathfrak{U}}[\chi]_{\mathfrak{U}} = \llbracket \psi \rrbracket^{\mathfrak{M}^s}[\chi]_{\mathfrak{U}} = s'[\chi]_{\mathfrak{U}} = \llbracket \chi \rrbracket^{\mathfrak{M}^{s'}} = \llbracket \chi \rrbracket^{(\mathfrak{M}^s)^\psi} = \llbracket \psi; \chi \rrbracket^{\mathfrak{M}^s}$, where $s' = \llbracket \psi \rrbracket^{\mathfrak{M}^s}$.
- $\varphi = \Box\psi$. Then $s[\varphi]_{\mathfrak{U}} = s[\Box\psi]_{\mathfrak{U}} = \begin{cases} s & \text{if } s[\psi]_{\mathfrak{U}} = s \\ \emptyset & \text{otherwise} \end{cases} = \begin{cases} s & \text{if } \llbracket \psi \rrbracket^{\mathfrak{M}^s} = s \\ \emptyset & \text{otherwise} \end{cases}$
 $= \begin{cases} \llbracket \Box\psi \rrbracket^{\mathfrak{M}^s} & \text{if } \llbracket \psi \rrbracket^{\mathfrak{M}^s} = s \\ \llbracket \Box\psi \rrbracket^{\mathfrak{M}^s} & \text{otherwise} \end{cases} = \llbracket \Box\psi \rrbracket^{\mathfrak{M}^s} = \llbracket \varphi \rrbracket^{\mathfrak{M}^s}$.

Theorem 5. *For any $\Gamma \cup \{\varphi\} \subseteq \mathcal{L}_{\Box}$, $\Gamma \vDash_U \varphi$ iff $\Gamma \vDash_{S5}^g \varphi$.*

Proof. \Rightarrow) Suppose $\Gamma \not\vDash_{S5}^g \varphi$. Then there exists an S5 model \mathfrak{M} such that $\mathfrak{M} \Vdash \Gamma$ and $\mathfrak{M} \not\vDash \varphi$. By the latter, there exists w in \mathfrak{M} such that $\mathfrak{M}, w \not\vDash \varphi$. Let $\mathfrak{M}_w = (W_w, R_w, V_w)$ be the submodel of \mathfrak{M} generated by w . Then $\mathfrak{M}_w \Vdash \Gamma$ and $\mathfrak{M}_w \not\vDash \varphi$. Since \mathfrak{M}_w is a universal model, by Lemma 3, we have $\mathfrak{U}^{\mathfrak{M}_w}, W_w \Vdash_U \Gamma$ and $\mathfrak{U}^{\mathfrak{M}_w}, W_w \not\vDash_U \varphi$. Hence, $\Gamma \not\vDash_U \varphi$.

\Leftarrow) Suppose $\Gamma \not\vDash_U \varphi$. Then there exist an update model \mathfrak{U} and an information state s in \mathfrak{U} such that $\mathfrak{U}, s \Vdash_U \Gamma$ and $\mathfrak{U}, s \not\vDash_U \varphi$. By Lemma 4, we have $\mathfrak{M}^s \Vdash \Gamma$ and $\mathfrak{M}^s \not\vDash \varphi$. Since \mathfrak{M}^s is an S5 model, it follows that $\Gamma \not\vDash_{S5}^g \varphi$.

Corollary 7. *For any $\gamma_1, \dots, \gamma_n, \varphi \in \mathcal{L}_{\Box}$,*

$$\gamma_1, \dots, \gamma_n \vDash_{SU} \varphi \text{ iff } \vDash_{S5}^g \Box \neg(\gamma_1; \dots; \gamma_n; \neg\varphi) \text{ iff } \vDash_{S5} \Box \neg(\gamma_1; \dots; \gamma_n; \neg\varphi).$$

Proof. Straightforward from Lemma 1 and Theorem 5.

Note that the truth condition for $\varphi; \psi$ is just the same as that for $\langle \varphi \rangle \psi$ in public announcement logic (PAL, henceforth. For an excellent overview of PAL and more generally dynamic epistemic logic, see [16]). Thus $\varphi \rightarrow \psi$ is just $\Box[\varphi]\psi$ in PAL. Hence, we can define the following translation from \mathcal{L}_{\Box} to \mathcal{L}_{PAL} .

Definition 11. Define $t : \mathcal{L}_\square \rightarrow \mathcal{L}_{PAL}$ as follows.

- $t(p) = p, p \in PV$
- $t(\neg\varphi) = \neg t(\varphi)$
- $t(\varphi \wedge \psi) = t(\varphi) \wedge t(\psi)$
- $t(\varphi; \psi) = \langle t(\varphi) \rangle t(\psi)$
- $t(\Box\varphi) = \Box t(\varphi)$

Now we can define \models_{SU} by the standard local or global consequence within \mathcal{L}_{PAL} .

Theorem 6. For any $\Gamma \cup \{\gamma_1, \dots, \gamma_n, \varphi\} \subseteq \mathcal{L}_\square$,

- (1) $\Gamma \models_U \varphi$ iff $t(\Gamma) \vdash_{\mathbf{PAL}}^g t(\varphi)$,
- (2) $\gamma_1, \dots, \gamma_n \models_{SU} \varphi$ iff $\vdash_{\mathbf{PAL}} [\langle \dots \langle t(\gamma_1) \rangle t(\gamma_2) \rangle t(\gamma_3) \dots \rangle t(\gamma_n)] t(\varphi)$.

Proof. For (1), by Theorem 5, we have $\Gamma \models_U \varphi$ iff $\Gamma \models_{S5}^g \varphi$. Since $\varphi; \psi$ has the same truth condition as $\langle \varphi \rangle \psi$ in PAL, we have $\Gamma \models_{S5}^g \varphi$ iff $t(\Gamma) \models_{S5}^g t(\varphi)$. Then by the completeness of **PAL** (for global consequence), we have $t(\Gamma) \models_{S5}^g t(\varphi)$ iff $t(\Gamma) \vdash_{\mathbf{PAL}}^g t(\varphi)$. Item (2) follows from Corollary 7 in the same way, noting that $[\varphi]\psi \leftrightarrow \neg \langle \varphi \rangle \neg \psi$ and $\vdash_{\mathbf{PAL}} \varphi$ iff $\vdash_{\mathbf{PAL}} \Box \varphi$.

It is well known that (single agent) **PAL** can be reduced to **S5**. It follows that sequential update consequence in \mathcal{L}_\square , can finally be defined by the local or global consequence of **S5** within \mathcal{L}_\square . This in turn implies that \mathcal{L}_\square has the same expressive power as \mathcal{L}_\square , for both update consequence and sequential update consequence.

Corollary 8. For any $\Gamma \cup \{\gamma_1, \dots, \gamma_n, \varphi\} \subseteq \mathcal{L}_\square$,

- (1) $\Gamma \models_U \varphi$ iff $\Gamma \vdash_{\mathbf{S5}}^g \varphi$,
- (2) $\gamma_1, \dots, \gamma_n \models_{SU} \varphi$ iff $\vdash_{\mathbf{PAL}} [\langle \dots \langle \gamma_1 \rangle \gamma_2 \rangle \gamma_3 \dots \rangle \gamma_n] \varphi$.

5 Conclusions and Future Work

Though global consequence can be defined by local consequence for some classes of frames, it has its independent value for applications. If domain semantics and update semantics are considered to be good formalizations of natural languages, then standard modal semantics with global consequence could be considered as a handy alternative. At least, it is more flexible than the former two, since we can consider various classes of frames, which is absent in the former two semantics. In particular, with the global correspondence results, we are more prepared to defend and choose various inference patterns concerning modality in natural languages. For example, one might accept that p entails *might* p , but refuse that *must* p entails p . This cannot be modeled either in domain semantics or in update semantics, or in (normal) modal logic with local consequence, since in such semantics, either both $p \models \Diamond p$ and $\Box p \models p$ are valid, or they are equivalent. With global consequence, however, this can be easily modeled by choosing frames

which are globally inverse reflexive but not globally reflexive. On the other hand, by the connection between local and global consequence, some standard theories of modal logic for local consequence can be reused in the application with global consequence.

This is only a first step in the study of global consequence. There are still a lot of technical issues unaddressed. For instance, is there a sufficient and necessary condition on frames for global consequence to be defined by local consequence? How does the augmentation by new operators systematically affect such definability? Is there a Sahlqvist-like correspondence between global consequence and first-order properties? How to compare the frame definability between global consequence and local consequence extended with the universal modality? We leave these technical issues for future research.

Acknowledgments. Thanks to three anonymous referees for their helpful comments. Thanks also to audiences at the 2019 logic seminar and the mathematical philosophy week & workshop on modal logic at Peking University, where I gave a talk on part of this paper and received valuable feedback. The paper was supported by National Social Science Foundation of China for key projects (No. 18ZDA033).

Appendix

Proof of Theorem 1

Proof. \Rightarrow) Suppose $\Box^\omega \Gamma \not\vdash_{\mathbb{F}} \varphi$. Then there exist a frame \mathfrak{F} in \mathbb{F} , a valuation V on \mathfrak{F} , and a world w in \mathfrak{F} such that $\mathfrak{F}, V, w \Vdash \Box^\omega \Gamma$ but $\mathfrak{F}, V, w \not\Vdash \varphi$. Let (\mathfrak{F}', V') be the model generated by w from (\mathfrak{F}, V) . Then $\mathfrak{F}', V', w \Vdash \Box^\omega \Gamma$ and $\mathfrak{F}', V', w \not\Vdash \varphi$. From the former, it follows that $\mathfrak{F}', V' \Vdash \Gamma$, since all worlds in \mathfrak{F}' are accessible from w in finite (including zero) steps. From the latter, it follows that $\mathfrak{F}', V' \not\Vdash \varphi$. Since \mathbb{F} is closed under point generated subframes, \mathfrak{F}' is also in \mathbb{F} . Thus, $\Gamma \not\vdash_{\mathbb{F}}^g \varphi$.

\Leftarrow) Suppose $\Gamma \not\vdash_{\mathbb{F}}^g \varphi$. Then there exist a frame \mathfrak{F} in \mathbb{F} and a valuation V on \mathfrak{F} such that $\mathfrak{F}, V \Vdash \Gamma$ but $\mathfrak{F}, V \not\Vdash \varphi$. From the latter, it follows that there exists a world w in \mathfrak{F} such that $\mathfrak{F}, V, w \not\Vdash \varphi$. From the former, it follows that every $\psi \in \Gamma$ is true at all worlds in (\mathfrak{F}, V) . Thereby, it can be easily verified by induction that $\Box^n \psi$ is true at all worlds in (\mathfrak{F}, V) for all $\psi \in \Gamma$ and $n \in \mathbb{N}$. In particular, $\mathfrak{F}, V, w \Vdash \Box^\omega \Gamma$. Hence, $\Box^\omega \Gamma \not\vdash_{\mathbb{F}} \varphi$.

Proof of Fact 3

Proof. Let $\mathbb{F} = \{\mathfrak{F}\}$ with $\mathfrak{F} = (\{w, u\}, \{(w, u)\})$. Then for any valuation V on \mathfrak{F} , $\mathfrak{F}, V \not\Vdash \Box \perp$, since $\mathfrak{F}, V, w \not\Vdash \Box \perp$. Hence, $\Box \perp \not\vdash_{\mathbb{F}}^g \perp$. On the other hand, given any valuation V on \mathfrak{F} , $\mathfrak{F}, V, u \Vdash \Box^n \Box \perp$ for all $n \in \mathbb{N}$, but $\mathfrak{F}, V, u \not\Vdash \perp$. Hence, $\Box^\omega \Box \perp \not\vdash_{\mathbb{F}} \perp$.

Proof of Fact 4

Proof. Consider $F = \{\mathfrak{F}\}$ with $\mathfrak{F} = (\{w, u\}, \emptyset)$. Obviously, F is not closed under point generated subframes. The direction from right to left is easy. For the other direction, suppose $\Box^\omega \Gamma \not\vdash_F \varphi$. Then there exists a valuation V on \mathfrak{F} such that either $\mathfrak{F}, V, w \Vdash \Box^\omega \Gamma$ and $\mathfrak{F}, V, w \not\Vdash \varphi$, or $\mathfrak{F}, V, u \Vdash \Box^\omega \Gamma$ and $\mathfrak{F}, V, u \not\Vdash \varphi$. W.l.o.g., suppose the former holds. Then $\mathfrak{F}, V, w \Vdash \Gamma$. Let V' be a valuation such that $V'(w) = V'(u) = V(w)$. It is easily verified that $\mathfrak{F}, V, w \Vdash \psi$ iff $\mathfrak{F}, V', w \Vdash \psi$ iff $\mathfrak{F}, V', u \Vdash \psi$ for all $\psi \in \mathcal{L}_\Box$. Hence, $\mathfrak{F}, V' \Vdash \Gamma$ and $\mathfrak{F}, V' \not\Vdash \varphi$. Thereby, $\Gamma \not\vdash_F^g \varphi$.

Proof of Theorem 2

Proof. \Rightarrow) Suppose $\Box \Gamma \not\vdash_F \Box \varphi$. Then there exist a frame \mathfrak{F} in F , a valuation V on \mathfrak{F} , and a world w in \mathfrak{F} such that $\mathfrak{F}, V, w \Vdash \Box \Gamma$ and $\mathfrak{F}, V, w \not\Vdash \Box \varphi$. From the latter, it follows that there exists $u \in R(w)$ such that $\mathfrak{F}, V, u \not\Vdash \varphi$. Then from the former, it follows that $\mathfrak{F}, V, u \Vdash \Gamma \cup \Box \Gamma$, noting that \mathfrak{F} is transitive. Let \mathfrak{M}_u be the submodel of (\mathfrak{F}, V) generated by u . Then $\mathfrak{M}_u, u \not\Vdash \varphi$ and hence $\mathfrak{M}_u \not\Vdash \varphi$. Since \mathfrak{M}_u is transitive, every world in \mathfrak{M}_u is either u or accessible from u . Thus $\mathfrak{M}_u, v \Vdash \Gamma$ for all v in \mathfrak{M}_u . Then we have $\mathfrak{M}_u \Vdash \Gamma$. Since F is closed under point generated subframes, the frame underlying \mathfrak{M}_u is also in F . Therefore, $\Gamma \not\vdash_F^g \varphi$.

\Leftarrow) Suppose $\Gamma \not\vdash_F^g \varphi$. Then there exist a frame \mathfrak{F} in F and a valuation V on \mathfrak{F} such that $\mathfrak{F}, V \Vdash \Gamma$ and $\mathfrak{F}, V \not\Vdash \varphi$. From the latter, it follows that there exists w in \mathfrak{F} such that $\mathfrak{F}, V, w \not\Vdash \varphi$. If Rww , then $\mathfrak{F}, V, w \not\Vdash \Box \varphi$. Since $\mathfrak{F}, V \Vdash \Gamma$, we also have $\mathfrak{F}, V, w \Vdash \Box \Gamma$. Hence $\Box \Gamma \not\vdash_F \Box \varphi$. If $\neg Rww$, let \mathfrak{F}' be a point extension of \mathfrak{F} for w by u . Then it can be verified that $\mathfrak{F}', V, u \Vdash \Box \Gamma$ and $\mathfrak{F}', V, u \not\Vdash \varphi$. Since F is closed under irreflexive point extensions, \mathfrak{F}' is also in F . Hence, $\Box \Gamma \not\vdash_F \Box \varphi$.

Proof of Proposition 4

Proof. The first two ‘iff’s follow from Corollary 2. The direction from right to left of the third ‘iff’ is easy. For the other direction, suppose $\Box \Gamma \not\vdash_{K45} \Box \varphi$. Then there exist a transitive and Euclidean model $\mathfrak{M} = (W, R, V)$ and a world $w \in W$ such that $\mathfrak{M}, w \Vdash \Box \Gamma$ and $\mathfrak{M}, w \not\Vdash \Box \varphi$. From the latter, it follows that there exists $u \in R(w)$ such that $\mathfrak{M}, u \not\Vdash \varphi$. Since \mathfrak{M} is transitive, we also have $\mathfrak{M}, u \Vdash \Box \Gamma$. Let \mathfrak{M}_u be the point generated submodel of \mathfrak{M} by u . Then it can be verified that \mathfrak{M}_u is reflexive, transitive and Euclidean. Moreover, $\mathfrak{M}_u, u \Vdash \Box \Gamma$ and $\mathfrak{M}_u, u \not\Vdash \varphi$. Therefore $\Box \Gamma \not\vdash_{S5} \varphi$. The last ‘iff’ can be proved analogously.

Proof of Proposition 5

Proof. \Rightarrow) Suppose $\Box^\omega \Gamma \not\vdash_F \varphi$. Then there exist a frame $\mathfrak{F} = (W, R)$ in F , a valuation V on \mathfrak{F} , and a world w in \mathfrak{F} such that $\mathfrak{F}, V, w \Vdash \Box^\omega \Gamma$ but $\mathfrak{F}, V, w \not\Vdash \varphi$. Let (\mathfrak{F}', V') be the model generated by w from (\mathfrak{F}, V) . Let $\mathfrak{F}' = (W', R')$. Then $\mathfrak{F}', V', w \Vdash \Box^\omega \Gamma$ and $\mathfrak{F}', V', w \not\Vdash \varphi$. From the former, it follows that $\mathfrak{F}', V' \Vdash \Gamma$, since all worlds in \mathfrak{F}' are accessible from w in finite (including zero) steps. From the latter, it follows that $\mathfrak{F}', V' \not\Vdash \varphi$. Noting that Γ is satisfiable and contains no modal formulas, we can define a valuation V'' on \mathfrak{F} such that for all worlds

in W' , V'' coincides with V' , and for all worlds u in $W - W'$, for every atom p , $u \in V''(p)$ iff $w \in V'(p)$. Then $\mathfrak{F}, V'' \Vdash \Gamma$, but $\mathfrak{F}, V'' \nVdash \varphi$. Thus, $\Gamma \not\vdash_{\mathbf{F}}^g \varphi$.

\Leftarrow) The same as that in the proof of Theorem 1.

Proof of Proposition 6

Proof. By induction on the length of the proof in \mathbf{S} .

\Rightarrow) The only interesting case is that $\varphi = \Box\psi$ is obtained from ψ by applying the rule of necessitation. By induction hypothesis, $\Box^\omega \Gamma \vdash_{\mathbf{S}} \psi$. Hence, $\Box\Box^\omega \Gamma \vdash_{\mathbf{S}} \Box\psi$, which implies that $\Box^\omega \Gamma \vdash_{\mathbf{S}} \varphi$.

\Rightarrow) The only interesting case is that $\varphi = \Box^n \psi \in \Box^\omega \Gamma$. Then $\psi \in \Gamma$. Hence, $\Gamma \vdash_{\mathbf{S}}^g \psi$. By applying the rule of necessitation n times, we obtain $\Gamma \vdash_{\mathbf{S}}^g \varphi$.

Proof of Theorem 3

Proof. \Leftarrow) Given any frame $\mathfrak{F} = (W, R)$ in \mathbf{F} that satisfies the above property, given any valuation V on \mathfrak{F} , suppose $\mathfrak{F}, V \Vdash \Diamond^i \Box^j \varphi$. Given any $w \in W$, suppose $R^k w x$. Then by the property of R , there exists $y \in W$ s.t. for all $z \in W$ if $R^i y z$ then there exists $u \in W$ s.t. $R^l x u$ and $R^j z u$. By $\mathfrak{F}, V \Vdash \Diamond^i \Box^j \varphi$, we have $\mathfrak{F}, V, y \Vdash \Diamond^i \Box^j \varphi$. Then it follows that there exists $z \in W$ s.t. $R^i y z$ and $\mathfrak{F}, V, z \Vdash \Box^j \varphi$. By the property of R , there exists $u \in W$ s.t. $R^l x u$ and $R^j z u$. Thus $\mathfrak{F}, V, u \Vdash \varphi$ and $\mathfrak{F}, V, x \Vdash \Diamond^l \varphi$. Hence, $\mathfrak{F}, V, w \Vdash \Box^k \Diamond^l \varphi$. Since w is arbitrary, we have $\mathfrak{F}, V \Vdash \Box^k \Diamond^l \varphi$, as required.

\Rightarrow) Suppose $\mathfrak{F} = (W, R)$ in \mathbf{F} does not satisfy the above property. Then there exists $w, x \in W$ s.t. $R^k w x$ and for all $y \in W$ there exists $z \in W$ s.t. $R^i y z$ and $R^l(x) \cap R^j(z) = \emptyset$. Let $V(p) = W - R^l(x)$. Then $\mathfrak{F}, V, x \Vdash \Box^l \neg p$ and $\mathfrak{F}, V, w \Vdash \Diamond^k \Box^l \neg p$. Hence, $\mathfrak{F}, V, w \nVdash \Box^k \Diamond^l p$ and $\mathfrak{F}, V \nVdash \Box^k \Diamond^l p$. Given any $y \in W$, by the property of R , there exists $z \in W$ s.t. $R^i y z$ and $R^l(x) \cap R^j(z) = \emptyset$. Thus $\mathfrak{F}, V, z \Vdash \Box^j p$ and $\mathfrak{F}, V, y \Vdash \Diamond^i \Box^j p$. Since y is arbitrary, we have $\mathfrak{F}, V \Vdash \Diamond^i \Box^j p$. Therefore, $\Diamond^i \Box^j p \not\vdash_{\mathbf{F}}^g \Box^k \Diamond^l p$.

References

1. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
2. Bledin, J.: Logic informed. *Mind* **123**(490), 277–316 (2014)
3. Chellas, B.F.: Modal Logic: An Introduction. Cambridge University Press, Cambridge (1980)
4. de Rijke, M., Sturm, H.: Global vs. local in basic modal logic. In: Proceedings of the Nicht-Klassische Formen Der Logik Im Rahmen Des XVIII. Deutschen Kongresses Fuer Philosophie (1999)
5. de Rijke, M., Sturm, H.: Global definability in basic modal logic. In: Wansing, H. (ed.) Essays on Non-Classical Logic, pp. 111–135. World Scientific (2001)
6. de Rijke, M., Wansing, H.: Proofs and expressiveness in alethic modal logic. In: Jacqueline, D. (ed.) A Companion to Philosophical Logic. Blackwell Publishing (2006)
7. Fitting, M.: Proof Methods for Modal and Intuitionistic Logics. Springer, Berlin, Heidelberg (1983)

8. Gillies, A.: Epistemic conditionals and conditional epistemics. *Noûs* **4**, 585–616 (2004)
9. Goranko, V., Passy, S.: Using the universal modality. *J. Log. Comput.* **2**(56), 203–233 (1992)
10. Kracht, M.: *Tools and Techniques in Modal Logic*. Elsevier, Amsterdam (1999)
11. Kracht, M.: Modal consequence relations. In: *Handbook of Modal Logic*, pp. 491–545. Elsevier (2007)
12. Ma, M., Chen, J.: Sequent calculi for global modal consequence relations. *Stud. Log.* **107**(4), 613–637 (2018). <https://doi.org/10.1007/s11225-018-9806-8>
13. Moss, L.S.: Three etudes on logical dynamics and the program of natural logic. In: Baltag, A., Smets, S. (eds.) *Johan van Benthem on Logic and Information Dynamics*. OCL, vol. 5, pp. 705–727. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06025-5_26
14. Schulz, M.: Epistemic modals and informational consequence. *Synthese* **174**(3), 385–395 (2010)
15. van Benthem, J.: *Modal Logic and Classical Logic*. Bibliopolis (1983)
16. van Ditmarsch, H.: Wiebe van der Hoek, Barteld Kooi: *Dynamic Epistemic Logic*, vol. 337. Springer, Netherlands, Dordrecht (2007)
17. Veltman, F.: Defaults in update semantics. *J. Philos. Log.* **25**(3), 221–261 (1996)
18. Venema, Y.: *Many-Dimensional Modal Logic*. Ph.D. thesis, University of Amsterdam (1992)
19. Willer, M.: Dynamics of epistemic modality. *Philos. Rev.* **122**(1), 45–92 (2013)
20. Willer, M.: An update on epistemic modals. *J. Philos. Log.* **44**(6), 835–849 (2015)
21. Yalcin, S.: Epistemic modals. *Mind* **116**(464), 983–1026 (2007)



Games for Hybrid Logic

From Semantic Games to Analytic Calculi

Robert Freiman^(✉) 

Institute of Logic and Computation, TU Wien, Vienna, Austria
robert@logic.at

Abstract. Game semantics and winning strategies offer a potential conceptual bridge between semantics and proof systems of logics. We illustrate this link for hybrid logic – an extension of modal logic that allows for explicit reference to worlds within the language. The main result is that the systematic search of winning strategies over all models can be finitized and thus reformulated as a proof system.

Keywords: Hybrid logic · Games · Proof systems

1 Introduction

Games have a long and varied tradition in logic (see, for example [1, 9, 12]). Building on the concepts of rational behavior and strategic thinking, they offer a fruitful natural approach to logic, complementing the common paradigm of model-theoretic semantics and proof systems. Game semantics goes back to Jaakko Hintikka [8], who designed a game for two players, usually called *Me* (or *I*) and *You*, seeking to establish the truth of a formula ϕ of first-order logic in a model \mathcal{M} . The game proceeds by rules for step-wise reducing ϕ to an atomic formula. The winning condition depends on the truth of this atomic formula in \mathcal{M} . It turns out that *I* have a winning strategy for this game if and only if ϕ is true in \mathcal{M} . A natural question is, whether there is an algorithm for searching for winning strategies for the game of ϕ over *all* models, which could thus establish (or refute) the validity ϕ . One such approach are *disjunctive winning strategies* allowing the players to keep track and – if necessary – revise their choices, depending on the truth values of the atomic sentences in the current model. This technique has been first demonstrated in [7] for Giles’ game for Lukasiewicz logic.

In this paper, we apply this method to the case of hybrid logic. Hybrid logic is an extension of the possible-world semantics of “orthodox” modal logic allowing one to explicitly refer to worlds within the object language, using nominals, while keeping many attractive features of modal logic intact. Obviously, this increased expressivity allows us to get a grip on many frame properties that are provably not expressible in orthodox modal logic [2]. Apart from this, using nominals can

Research supported by FWF projects P 32684 and W 1255.

© Springer Nature Switzerland AG 2021

A. Silva et al. (Eds.): WoLLIC 2021, LNCS 13038, pp. 133–149, 2021.

https://doi.org/10.1007/978-3-030-88853-4_9

be an advantage for modelling in temporal logic [5] or in making the link between modal logic and description logics more explicit [4]. The choice of hybrid logic over orthodox modal logic for the development of a game-theoretic approach is a natural one: Explicit reference of particular worlds within the language provides conceptual clarity for the lifting of the semantic game to disjunctive strategies.

The coherence of hybrid logic with games has been demonstrated by Patrick Blackburn, who designed a Lorenzen-style [10] dialogue game for hybrid logic [3]. Sara Negri presented the labeled proof-system **G3K** that resembles the semantics of modal logic [11]. Our approach brings together the best of two worlds: it supplements the clear semantic motivation of **G3K** with an accessible game-theoretic viewpoint of Blackburn’s dialogue game with a direct semantic link. Similar to **G3K**, a failed search for a disjunctive winning strategy directly gives rise to a countermodel.

The main result of this paper addresses the following difficulty: generally, while searching for a disjunctive winning strategy, one has to keep track of all infinitely many possible models and (in the case of modal operators) infinitely many possible choices. The perhaps surprising and highly non-trivial insight is, that this search can be finitized in a way that is coherent with respect to the semantics-provability-bridge mentioned above. This is achieved by a conceptual reduction of choices of the opponent to an *optimal choice*. Thus, the search for a disjunctive winning strategy can itself be formulated as a proof-system.

This paper is structured as follows: Sect. 2 is a recap of hybrid logic. In Sect. 3 we present the semantic game over a model and formalize strategies. The disjunctive game is introduced in Sect. 4, which also contains the main result. Finally, its finitized formulation as an analytic calculus is presented in Sect. 5.

2 Preliminaries

The language of hybrid modal logic is as follows: We start from two disjoint, countably infinite sets N (set of nominals) and P (set of propositional variables). Nominals are usually called “ i, j, k, \dots ” propositional variables are called “ p, q, \dots ” Formulas ϕ are built according to the following grammar:

$$\phi ::= p \mid i \mid R(i, j) \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \neg \phi \mid @_i \phi \mid \Box \phi \mid \Diamond \phi$$

Formulas of the form p , i and $R(i, j)$ are called *elementary*. Intuitively, we can think of the nominal i as being the name of a particular world in a model. Hence, i is true in exactly one world. The formula $@_i \phi$ stands for the fact that ϕ is true in the world with name i . The relational claim $R(i, j)$ says that the world with name j is accessible from the world with name i . It is usually defined as $@_i \Diamond j$. However, we leave it as an elementary formula in order to prevent circular definitions in the description of the game rules in the following sections.

We now define the semantics formally: A model \mathcal{M} for hybrid modal logic is a tuple (W, R, V, g) , where

1. W is a non-empty set. Its elements are called *worlds*.
2. $R \subseteq W \times W$ is called *accessibility relation*. As usual, we write wRv instead of $(w, v) \in R$. The set of *accessible worlds from w* are $wR := \{v \in W : wRv\}$.
3. $V : P \rightarrow \mathcal{P}(W)$ ¹ is called *valuation function*.
4. $g : N \rightarrow W$ is called *assignment*. If $g(i) = w$, we say that i is a *name* of w , or simply that w has a name.

Truth of formulas in the world w of W is defined recursively:²

$$\begin{aligned}
\mathcal{M}, w &\models p, \text{ iff } w \in V(p), \\
\mathcal{M}, w &\models i, \text{ iff } g(i) = w, \\
\mathcal{M}, w &\models R(i, j), \text{ iff } g(i)Rg(j), \\
\mathcal{M}, w &\models \phi \wedge \psi, \text{ iff } \mathcal{M}, w \models \phi \text{ and } \mathcal{M}, w \models \psi, \\
\mathcal{M}, w &\models \phi \vee \psi, \text{ iff } \mathcal{M}, w \models \phi \text{ or } \mathcal{M}, w \models \psi, \\
\mathcal{M}, w &\models \phi \rightarrow \psi, \text{ iff } \mathcal{M}, w \not\models \phi \text{ or } \mathcal{M}, w \models \psi, \\
\mathcal{M}, w &\models \neg\phi, \text{ iff } \mathcal{M}, w \not\models \phi, \\
\mathcal{M}, w &\models @_i\phi, \text{ iff } \mathcal{M}, g(i) \models \phi, \\
\mathcal{M}, w &\models \Box\phi, \text{ iff for all } v \in W, (w, v) \notin R \text{ or } \mathcal{M}, v \models \phi, \\
\mathcal{M}, w &\models \Diamond\phi, \text{ iff for some } v \in W, wRv \text{ and } \mathcal{M}, v \models \phi.
\end{aligned}$$

We say that a formula ϕ is true in the model \mathcal{M} and write $\mathcal{M} \models \phi$ iff for every world w , $\mathcal{M}, w \models \phi$. For a class \mathfrak{M} of models, we write $\mathfrak{M} \models \phi$ and say that ϕ is *valid over* \mathfrak{M} iff for all $\mathcal{M} \in \mathfrak{M}$, $\mathcal{M} \models \phi$. We say that ϕ is *valid* (we write $\models \phi$) iff ϕ is valid over the class of all models.

The most important class for what follows is the class of named models \mathfrak{N} . A model is called *named* iff the assignment is surjective, i.e. every world has a name. The accessibility relation in such named models is completely determined by the truth value of the relational formulas. For these models, we can thus state the following semantic facts about the modal operators without explicitly referring to the semantics of the accessibility relation. Let $w = g(i)$, then

$$\begin{aligned}
\mathcal{M}, w &\models \Box\phi, \text{ iff for all } j \in N, \mathcal{M}, g(j) \models \neg R(i, j) \vee \phi, \\
\mathcal{M}, w &\models \Diamond\phi, \text{ iff for some } j \in N, \mathcal{M}, g(j) \models R(i, j) \wedge \phi. \\
\mathcal{M} &\models \phi, \text{ iff for all } i \in N, \mathcal{M}, g(i) \models \phi.
\end{aligned}$$

Named models are important, since questions of validity can be reduced to questions of validity over the class of named models:

Lemma 1 (Theorem 7.29 in [2]). $\models \phi \iff \mathfrak{N} \models \phi$.

¹ $\mathcal{P}(W)$ denotes the power set of W .

² Our definitions of $\mathcal{M}, w \models \phi \rightarrow \psi$ and $\mathcal{M}, w \models \Box\phi$ is equivalent to the usual “if \mathcal{M}, w models ϕ , then \mathcal{M}, w models ψ ” and “for all $v \in W$, if wRv , then $\mathcal{M}, v \models \phi$ ”. Our formulations are more easily representable as game rules.

3 A Game for Truth

The *semantic game* is played over a model $\mathcal{M} = (W, R, V, \mathbf{g})$ by two players, *Me* and *You*, who argue about the truth of a formula ϕ at a world w . At each stage of the game, one player acts in the role of a proponent, while the other one acts as opponent of the claim that a formula ϕ is true at the world w . We represent the situation where *I* am the proponent (and *You* are the opponent) by the *game state* $\mathbf{P}, w : \phi$, and the situation where *I* am the opponent (and *You* are the proponent) by $\mathbf{O}, w : \phi$. We add another kind of game state of the form $\mathbf{P} : \phi$ or $\mathbf{O} : \phi$ representing the claim that ϕ is true in the whole model.

In order to completely lay down the rules of the game, we must define its *game tree*, i.e. a tree whose nodes are game states and are labeled either “I” or “Y”. Each run of the game is identified with a path through that game tree: in nodes labelled “I” that have children, it is *My* choice to decide which of the children the game continues at. It is *Your* choice in nodes labelled “Y”. Each run of a game ends in a leaf state: if it is labelled “I”, *I* win and *You* lose, if it is labelled “Y”, *I* lose and *You* win. We denote the tree rooted in the game state g by $\mathbf{G}(\mathcal{M}, g)$ ³. Below, we give a recursive definition of $\mathbf{G}(g)$ by specifying its immediate subtrees. We give the rules only for the case where *I* am in the role of proponent. The variants of the rules where *I* am in the role of the opponent is obtained by switching “I” and “Y” and “P” and “O” in the respective rule⁴. Note that the rules closely parallel the recursive definition of truth of the previous section:

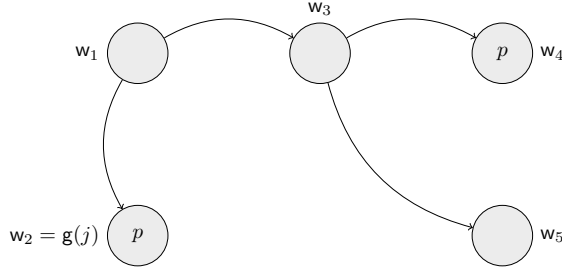
- (R_{\vee}) If $g = \mathbf{P}, w : \psi_1 \vee \psi_2$, then g is labelled “I” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, w : \psi_1)$ and $\mathbf{G}(\mathbf{P}, w : \psi_2)$.
- (R_{\wedge}) If $g = \mathbf{P}, w : \psi_1 \wedge \psi_2$, then g is labelled “Y” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, w : \psi_1)$ and $\mathbf{G}(\mathbf{P}, w : \psi_2)$.
- (R_{\rightarrow}) If $g = \mathbf{P}, w : \psi_1 \rightarrow \psi_2$, then g is labelled “I” and its immediate subtrees are $\mathbf{G}(\mathbf{O}, w : \psi_1)$ and $\mathbf{G}(\mathbf{P}, w : \psi_2)$.
- (R_{\neg}) If $g = \mathbf{P}, w : \neg\psi$, then g is labelled “I” and its immediate subtree is $\mathbf{G}(\mathbf{O}, w : \psi)$.
- $(R_{@})$ If $g = \mathbf{P}, w : @_i\psi$, then g is labelled “I” and its immediate subtree is $\mathbf{G}(\mathbf{P}, \mathbf{g}(i) : \psi)$.
- (R_{\square}) If $g = \mathbf{P}, w : \square\psi$ and $wR \neq \emptyset$, then g is labelled “Y” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, v : \psi)$, where v ranges over wR . If $wR = \emptyset$, then g is a leaf labelled “I”.
- (R_{\diamond}) If $g = \mathbf{P}, w : \diamond\psi$ and $wR \neq \emptyset$, then g is labelled “I” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, v : \psi)$, where v ranges over wR . If $wR = \emptyset$, then g is a leaf labelled “Y”.
- (R_p) If $g = \mathbf{P}, w : p$, then g is a leaf. It is labelled “I” iff $w \in V(p)$.
- (R_i) If $g = \mathbf{P}, w : i$, then g is a leaf. It is labelled “I” iff $\mathbf{g}(i) = w$.
- (R_R) If $g = \mathbf{P}, w : R(i, j)$, then g is a leaf. It is labelled “I” iff $\mathbf{g}(i)R\mathbf{g}(j)$.
- (R_U) If $g = \mathbf{P} : \psi$, then g is labelled “Y” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, w : \psi)$, where w ranges over W .

³ We often write $\mathbf{G}(g)$, if \mathcal{M} is clear from context.

⁴ For example, for (R_{\rightarrow}) , if $g = \mathbf{O}, w : \psi_1 \rightarrow \psi_2$, then g is labelled “Y” and its immediate subtrees are $\mathbf{G}(\mathbf{P}, w : \psi_1)$ and $\mathbf{G}(\mathbf{O}, w : \psi_2)$.

Since the degree of the involved formula strictly decreases with every child, game trees are always of finite height. This means that every run of the game lasts only finitely many rounds.

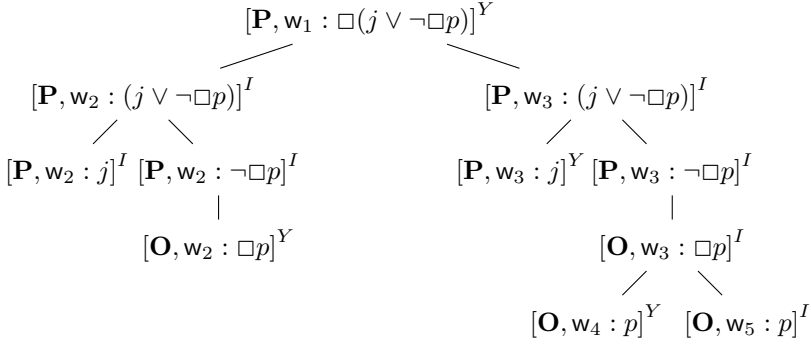
Example 1. Consider the following model \mathcal{M} with worlds w_1, \dots, w_5 and the accessibility relation represented by arrows. We write p inside the circle representing a world w iff $w \in V(p)$.



Let us consider the following run of the game $\mathbf{G}(\mathcal{M}, \mathbf{P}, w_1 : \Box(j \vee \neg\Box p))$. First, *You* must choose a neighboring world. Since *You* know that *I* can defend j at w_2 , let us say that *You* choose w_3 and *I* must then defend $j \vee \neg\Box p$ at w_3 . Clearly, *I* will choose the second disjunct. According to the rule of negation, now a role switch occurs: *I* am now the Opponent and *You* the Proponent of $\Box p$ at w_3 . Hence, *I* must choose a neighboring world and *You* must defend p there. As *My* choice is between the p -world w_4 and the non- p -world w_5 , *I* will choose w_5 and win the game. We can represent this run of the game as the following path:

$$\begin{array}{c}
 [\mathbf{P}, w_1 : \Box(j \vee \neg\Box p)]^Y \\
 | \\
 [\mathbf{P}, w_3 : (j \vee \neg\Box p)]^I \\
 | \\
 [\mathbf{P}, w_3 : \neg\Box p]^I \\
 | \\
 [\mathbf{O}, w_3 : \Box p]^Y \\
 | \\
 [\mathbf{O}, w_5 : p]^I
 \end{array}$$

The whole game tree represents all possible choices of the two players at any point:



I cannot win in the state $\mathbf{O}, w_2 : \Box p$, because there are no neighbors of w_2 . Although not all possible runs of a game end in a winning state for Me , I can make choices that will guarantee that the game will end in My victory. Generally, if I can always enforce a given game to end in a winning state for Me , we say that I have a *winning strategy* for that game. We will make this notion more formal in the following subsection.

Winning Strategies

To precisely describe the scenario where I can enforce that the game ends in a winning state for Me , we need the notion of a winning strategy:

Definition 1. A strategy⁵ for Me for the game $\mathbf{G}(\mathcal{M}, g)$ is a subtree obtained by removing from the game tree all but one children from every non-leaf node labelled “ I ”. A strategy for Me is winning if all leaf states are labelled “ I ”. (Winning) strategies for You are defined symmetrically.

Example 2. Continuing the game from Example 1, we can now make precise our observation that I can make choices such that the game will always end in winning states for Me . A winning strategy for Me for $\mathbf{G}(\mathbf{P}, w_1 : \Box(j \vee \neg \Box p))$ can be found in Fig. 1.

Theorem 1. Let \mathcal{M} be a model and w a world.

1. I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \phi)$ iff $\mathcal{M}, w \models \phi$.
2. I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{O}, w : \phi)$ iff $\mathcal{M}, w \not\models \phi$.
3. I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P} : \phi)$ iff $\mathcal{M} \models \phi$.

To prove the theorem we use the following basic lemma showing that having a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \phi)$ and $\mathbf{G}(\mathcal{M}, \mathbf{O}, w : \phi)$ is exclusive.

Lemma 2. I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \phi)$ iff I do not have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{O}, w : \phi)$.

⁵ This deviates from the standard, more general game-theoretic definition of a winning strategy. It is sufficient for our purposes.

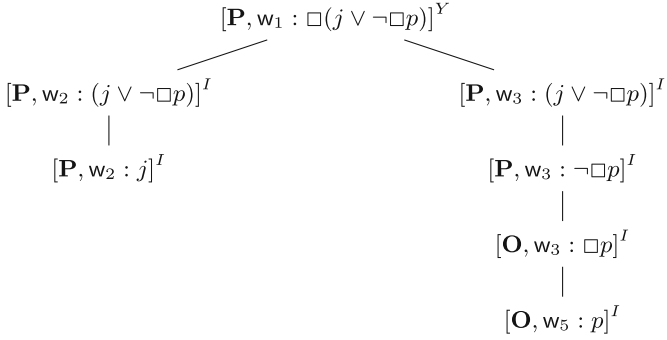


Fig. 1. A winning strategy for $\mathbf{G}(\mathbf{P}, w_1 : \Box(j \vee \neg\Box p))$

Proof. By induction on ϕ .

Proof (of Theorem 1). 2 follows from 1 in the obvious way. By Lemma 2, it suffices to show only 1. We show both directions simultaneously by induction on ϕ . If ϕ is atomic, everything follows from the definition of a winning state. Let us show some of the inductive steps: if ϕ is of the form $\psi \wedge \chi$, then I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \psi \wedge \chi)$ iff I have winning strategies for both $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \psi)$ and $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \chi)$. By induction hypothesis, this is the case iff $\mathcal{M}, w \models \psi$ and $\mathcal{M}, w \models \chi$, which in turn is equivalent to $\mathcal{M}, w \models \psi \wedge \chi$.

As for modal operators, I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \Box\psi)$ iff for all successors v of w , I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, v : \psi)$. By induction hypothesis, this is the case iff $\mathcal{M}, v \models \psi$. Since v is an arbitrary successor of w , this is equivalent to $\mathcal{M}, w \models \Box\psi$. Note that this also covers the trivial case, where there are no successors of w .

Lemma 2 is needed for the case of negation: I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \neg\psi)$ iff I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{O}, w : \psi)$. By the lemma, this is the case iff I do not have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, w : \psi)$. By induction hypothesis, this is the case iff $\mathcal{M}, w \not\models \psi$ iff $\mathcal{M}, w \models \neg\psi$.

We conclude this section with an important observation about the game for named models. Remember that in a named model, every world w has a name i , i.e. $\mathbf{g}(i) = w$. Therefore, it is unambiguous, if we write $\mathbf{Q}, i : \phi$ for the game state $\mathbf{Q}, w : \phi$. This, together with the fact that $\mathcal{M} \models R(i, j)$ iff $\mathbf{g}(i)\mathbf{R}\mathbf{g}(j)$ gives us the following equivalent formulations of the rules (R_\Box) , (R_\Diamond) and (R_U) :

- (R_\Box) If $g = \mathbf{P}, i : \Box\psi$, then g is labelled ‘‘Y’’ and its immediate subtrees are $\mathbf{G}(\mathbf{P}, j : \neg R(i, j) \vee \psi)$, where j ranges over the nominals.
- (R_\Diamond) If $g = \mathbf{P}, i : \Diamond\psi$, then it is labelled ‘‘I’’ and its immediate subtrees are $\mathbf{G}(\mathbf{P}, j : R(i, j) \wedge \psi)$, where j ranges over the nominals.
- (R_U) If $g = \mathbf{P} : \psi$, then it is labelled ‘‘Y’’ and its immediate subtrees are $\mathbf{G}(\mathbf{P}, i : \psi)$, where i ranges over the nominals.

From now on, we will tacitly assume that all models are named. This is justified by Lemma 1. Also, we will only use the reformulations from above. We will shortly see the advantages of these reformulations.

4 A Game for Validity

In this section we develop a so-called *disjunctive game* [7] to model validity of a formula ϕ . Intuitively, the idea is to play all possible games rooted in a game state g at once. This is only possible using the reformulations of the rules (R_{\square}), (R_{\diamond}) and (R_U): if we stick to those variants, the structure of the model does not affect the shape of the game tree anymore. To be more precise, if \mathcal{M}_1 and \mathcal{M}_2 are two models, then the game trees for $\mathbf{G}(\mathcal{M}_1, g)$ and $\mathbf{G}(\mathcal{M}_2, g)$ are identical, except maybe for the labelling of leaf states.

We have thus a uniform game tree and truth values become important only at the leafs. Consequently, we can design the disjunctive game as a two-player game. We might think of the two players as being the same as in the semantic game and correspondingly call them *Me* and *You*. In order to compensate *Me* for having to play “blindly” in a game state g , i.e. without being guided by truth within a particular model, we allow *Me* to add backtracking points $g \vee g$. The idea is that this allows *Me* to first play on the left copy of g and, in case of failure, change to right backup copy. In general, game states of the disjunctive game are thus of the form $D = g_1 \vee g_2 \vee \dots \vee g_n$, where the g_i are game states of the semantic game. According to our interpretation, such a disjunctive game state stands for the fact that for every model \mathcal{M} , I have a winning strategy for $\mathbf{G}(\mathcal{M}, g_1)$ or for $\mathbf{G}(\mathcal{M}, g_2)$... or for $\mathbf{G}(\mathcal{M}, g_n)$ ⁶.

Another ingredient is necessary: to determine who is to move at a disjunctive game state D , we introduce *regulation functions*. A regulation function ρ maps non-elementary⁷ disjunctive states to one of its non-elementary game states g_i .

Formally, we identify the disjunctive game specified by the regulation ρ and starting in the state D with its game tree $\mathbf{DG}(D, \rho)$ ⁸. It is defined recursively by specifying its immediate subtrees:

- If $\rho(D \vee g) = g$ and g is labelled “Y”, then so is $D \vee g$. Its immediate subtrees are $\mathbf{DG}(D \vee g', \rho)$, where g' ranges over all children of g in $\mathbf{G}(g)$.
- If $\rho(D \vee g) = g$ and g is labelled “I”, then so is $D \vee g$. Its immediate subtrees are the following:
 - $\mathbf{DG}(D \vee g', \rho)$, where g' ranges over all children of g in $\mathbf{G}(g)$.
 - The subtree $\mathbf{DG}(D \vee g \vee g)$ ⁹. We say that $D \vee g \vee g$ is obtained by *duplicating* g .

⁶ Unless noted otherwise, we identify a disjunctive state with the multiset of its game states. This implies that we consider two disjunctive states to be equal if they contain the same game states in the same numbers.

⁷ A disjunctive state is called *elementary* if all its game states are elementary, i.e. they involve only elementary formulas.

⁸ If ρ is clear from context (or not important), we conveniently write $\mathbf{DG}(D)$.

⁹ Due to this rule, the game may continue forever.

- If $D = g_1 \vee \dots \vee g_n$ is elementary, then it is labelled “I”, if for every model \mathcal{M} , there is some i such that g_i is labelled “I” in $\mathbf{G}(\mathcal{M}, g_i)$. Otherwise, it is labelled “Y”.

Example 3. The simplest example to illustrate and explain the idea of disjunctive states is the game $\mathbf{DG}(\mathbf{P} : p \vee \neg p)$. Obviously, I have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P} : p \vee \neg p)$ for every model \mathcal{M} .

In the first round of $\mathbf{DG}(\mathbf{P} : p \vee \neg p)$, *You* choose a nominal i and the game proceeds with the state $\mathbf{P}, i : p \vee \neg p$ (*You* made *Your* choice according to the rule (R_U)). Now I have two options: The first is to continue the game with either $\mathbf{P}, i : p$ or with $\mathbf{P}, i : \neg p$ (according to (R_V)). But both choices are bad: the first one, because there are models \mathcal{M} with $\mathcal{M}, \mathbf{g}(i) \not\models p$ and the second, because there are models \mathcal{N} with $\mathcal{N}, \mathbf{g}(i) \not\models \neg p$. This is where the second option comes in: I duplicate the state $\mathbf{P}, i : p \vee \neg p$ and the game continues with $(\mathbf{P}, i : p \vee \neg p) \vee (\mathbf{P}, i : p \vee \neg p)$.

A regulation function will choose one of the two (say the left one). Now I can choose to continue the game with $(\mathbf{P}, i : p) \vee (\mathbf{P}, i : p \vee \neg p)$. In the next round, the game continues with the right disjunct, and I play $(\mathbf{P}, i : p) \vee (\mathbf{P}, i : \neg p)$. Finally, the game arrives at $(\mathbf{P}, i : p) \vee (\mathbf{O}, i : p)$ and I win.

The rest of this section is dedicated to proving that this game indeed adequately models validity. Let us call a disjunctive state $D = g_1 \vee \dots \vee g_n$ *game valid*, if for every model \mathcal{M} , there is some i such that I have a winning strategy in $\mathbf{G}(\mathcal{M}, g_i)$. D is called *winning* if there is a regulation ρ such that I have a winning strategy for $\mathbf{DG}(D, \rho)$.

Theorem 2. *Every disjunctive state is game valid iff it is winning.*

In light of Theorem 1, we immediately get the following proposition:

Proposition 1. *The formula ϕ is valid iff $\mathbf{P} : \phi$ is winning.*

Theorem 2 suffers from what Johan van Benthem calls “ \exists -sickness” [1]. However, our proof will offer a more constructive formulation: (1) There is a direct construction of winning strategies witnessing the game validity of D from a winning strategy for Me in $\mathbf{DG}(D)$. (2) There is a construction of a particular strategy σ of Me and a regulation ρ with the following property: If σ is not winning in $\mathbf{DG}(D, \rho)$, then one can extract a model \mathcal{M} and winning strategies for *You* for all $\mathbf{G}(\mathcal{M}, g)$, where $g \in D$.¹⁰ We will now prove (1) and leave (2) for the next subsection.

Proof (of Theorem 2, right-to-left). Let σ be a winning strategy for $\mathbf{DG}(D_0, \rho)$. We show by upwards-tree-induction on σ that every D appearing in σ is game valid.

¹⁰ If one changes the disjunctive game such that infinite runs are considered winning for *You*, then one can extract the model and the strategies from *Your* winning strategy in $\mathbf{DG}(D, \rho)$.

If D is elementary, then it is valid, since σ is winning. Let us deal with some exemplary cases of the inductive step. Let us start with two cases, where it is *My* choice: If $D \vee g \vee g$ is valid and is obtained from $D \vee g$ by duplication of g , then clearly $D \vee g$ is valid too.

Let $D \vee (\mathbf{P}, i : \diamond\psi)$ appear¹¹ in σ and its child be $D \vee (\mathbf{P}, j : R(i, j) \wedge \psi)$. Let \mathcal{M} be a model. If there is a state g' of D such that I have a winning strategy in $\mathbf{G}(\mathcal{M}, g')$, then this state appears in $D \vee g$ and there is nothing to show. We will omit this trivial subcase in the other cases. If I have a winning strategy μ for $\mathbf{G}(\mathcal{M}, \mathbf{P}, j : R(i, j) \wedge \psi)$, then I have the following winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \diamond\phi)$: choose the (world corresponding to the) nominal j in the first round. The game then continues with $\mathbf{G}(\mathcal{M}, \mathbf{P}, j : R(i, j) \wedge \psi)$, where I can use μ to win.

We check three cases where it is *Your* move: if $D \vee (\mathbf{P} : \psi)$ appears in σ then its children are of the form $D \vee (\mathbf{P}, i : \psi)$, where i ranges over the nominals. If I have a winning strategy μ_i for every $\mathbf{G}(\mathbf{P}, i : \psi)$, then, since \mathcal{M} is named, I have a winning strategy for $\mathbf{G}(\mathbf{P} : \psi)$: By the rule (R_U), *You* must choose a nominal i in the first move. The game then proceeds with $\mathbf{G}(\mathbf{P}, i : \psi)$ and I can use μ_i to win the game.

If $D \vee (\mathbf{P}, i : \psi_1 \wedge \psi_2)$ appears in σ , then its children are $D \vee (\mathbf{P}, i : \psi_1)$ and $D \vee (\mathbf{P}, i : \psi_2)$. If \mathcal{M} is a model such that I have winning strategies μ_1 for $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \psi_1)$ and μ_2 for $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \psi_2)$, then I can use them to win $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \psi_1 \wedge \psi_2)$.

If $D \vee \mathbf{P}, i : \square\psi$ appears in σ , then its children are $D \vee (\mathbf{P}, j : \neg R(i, j) \vee \psi)$, where j ranges over the nominals. If I have a winning strategy μ_i for every $\mathbf{G}(\mathcal{M}, \mathbf{P}, j : \neg R(i, j) \vee \psi)$, then I can use them similarly to the above to win $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \square\psi)$.

The Best Way to Play

In this subsection we will construct a regulation ρ_0 and a strategy σ for Me for the game $\mathbf{DG}(D_0, \rho_0)$ such that σ is guaranteed to be winning, if D_0 is game valid. Intuitively, the idea is as follows: In order to prevent bad choices, every time I must move I will use the duplication rule first. This way I can always come back to have another shot. The trick is now to systematically exploit all possible choices. Playing this way ensures Me to always take a good choice, if there is one. Here is an exact formulation of this construction:

Construction of the Strategy σ . Let D^{el} be the disjunctive state obtained from D by removing all non-elementary game states. Let $\#$ be an enumeration of triples $\langle D, g, i \rangle$, where D is a non-elementary disjunctive state, g is a non-elementary game state and i is a nominal, and such that every triple appears in that enumeration infinitely often. We build a tree rooted in D_0 according to the following procedure:

¹¹ In this proof, we conveniently write $D \vee g$ to indicate that $\rho(D \vee g) = g$.

1. Add D_0 as a root of the tree.
2. For every $n = \#(D, g, i)$, if $D \vee g$ is a leaf:
 - (a) If *You* move in $\mathbf{G}(g)$, then add as successors to $D \vee g$ all of *Your* possible moves $D \vee g'$. For example, the successors of $D \vee (\mathbf{P}, i : \phi \wedge \psi)$ are $D \vee \mathbf{P}, i : \phi$ and $\mathbf{P}, i : \psi$. The successors of $D \vee (\mathbf{P} : \phi)$ are $D \vee (\mathbf{P}, i : \phi)$, for every nominal i .
 - (b) If *I* move in $\mathbf{G}(g)$ and D^{el} is game valid, then add to $D \vee g$ an arbitrary move $D \vee g'$ of Me .
 - (c) If *I* move in $\mathbf{G}(g)$ and D^{el} is not valid, then add to $D \vee g$ the child $D \vee g \vee g$. Afterwards, if there are only two options g_1 and g_2 for Me in $\mathbf{G}(g)$, then add as a child $D \vee g_1 \vee g$ and as its child $D \vee g_1 \vee g_2$. For example, for $D \vee (\mathbf{P}, i : \phi \vee \psi)$, the tree looks as follows:

$$\begin{array}{c}
 D \vee (\mathbf{P}, i : \phi \vee \psi) \\
 | \\
 D \vee (\mathbf{P}, i : \phi \vee \psi) \vee (\mathbf{P}, i : \phi \vee \psi) \\
 | \\
 D \vee (\mathbf{P}, i : \phi) \vee (\mathbf{P}, i : \phi \vee \psi) \\
 | \\
 D \vee (\mathbf{P}, i : \phi) \vee (\mathbf{P}, i : \psi)
 \end{array}$$

If there are infinitely many options in $\mathbf{G}(g)$, then they are parametrized by the nominals. For example, in $\mathbf{P}, j : \diamond\phi$, *I* must choose between $\mathbf{P}, k : R(i, k) \wedge \phi$, where k ranges over the nominals. We first add $D \vee g \vee g$ as a child to $D \vee g$. Afterwards we add as a child $D \vee g \vee g(i)$, where $g(i)$ is the game state given by nominal i . In our example, for $D \vee (\mathbf{P}, j : \diamond\phi)$, we add $D \vee (\mathbf{P}, i : \diamond\phi) \vee (\mathbf{P}, j : \diamond\phi)$ and $D \vee (\mathbf{P}, j : \diamond\phi) \vee (\mathbf{P}, i : R(i, j) \wedge \phi)$.

The outline of the rest of the proof is as follows: If σ , as constructed above, is not a winning strategy, then there is at least one path π through σ rooted at D_0 , such that π does not end in a game valid elementary leaf. This means that π either ends in a non-valid leaf or is infinite. We will now define a model \mathcal{M}_π with the property that *I* do not have a winning strategy for $\mathbf{G}(\mathcal{M}_\pi, g)$, for any game state g appearing along π ¹². We call a model with this property a π -countermodel.

Definition 2. Let D_π^n be the n -th node in the path π and let us define the relation $i \sim_\pi^n j$ between two nominals i and j iff $\mathbf{O}, i : j$ or $\mathbf{O}, j : i$ appear in D_π^n . Let \approx_π^n be the symmetric, reflexive and transitive closure of \sim_π^n . Let $i \approx_\pi j$, iff for some n , $i \approx_\pi^n j$. We write $[i]$ for the equivalence class of i . Since the elementary parts of the nodes accumulate in the course of π , we have for each n , $\approx_\pi^n \subseteq \approx_\pi^{n+1} \subseteq \approx_\pi$ (if the length of π is at least $n + 1$). We define the following named model \mathcal{M}_π :

¹² We say that a game state g appears along π , if it occurs as part of a disjunctive state in π .

- *Worlds: Equivalence classes of nominals,*
- *Accessibility relation R_π : We have $[i]R_\pi[j]$ iff for some $i' \in [i]$ and $j' \in [j]$, $k \in N$, $\mathbf{O}, k : R(i', j')$ appears along π .*
- *Valuation function V_π : $[i] \in V_\pi(p)$ iff for some $i' \in [i]$, $\mathbf{O}, i' : p$ appears in π .*
- *Assignment g_π : $g_\pi(i) = [i]$.*

The following lemma shows that the definition of the equivalence relation is not arbitrary: In every π -countermodel of D_π^i equivalent nominals must name the same worlds.

Lemma 3. *Let \mathcal{M} be a π -countermodel of D_π^n . Then \mathcal{M} respects the equivalence \approx_π^n : If $i \approx_\pi^n j$, then $g(i) = g(j)$.*

Proof. Let $i \approx_\pi^n j$. Then either $\mathbf{O}, i : j$ or $\mathbf{O}, j : i$ appear in D_π^n . Either way, to be a π -countermodel, \mathcal{M} must satisfy $g(i) = g(j)$. The result for the symmetric, reflexive and transitive closure follows from the corresponding properties of $=$.

Lemma 4. *If g appears along π , then You have a winning strategy for $\mathbf{G}(\mathcal{M}_\pi, g)$.*

Proof. We show the lemma by induction on $\mathbf{G}(g)$. The elementary cases where g is of the form $\mathbf{O}, i : \phi$ are clear. Assume $g = \mathbf{P}, i : p$ appears along π , but $\mathcal{M}_\pi, [i] \models p$. The latter implies that for some $j \in [i]$, $\mathbf{O}, j : p$ appears along π . Since the elementary states of π are accumulative, there is a disjunctive state D in π containing the three states $\mathbf{P}, i : p$, $\mathbf{O}, j : p$ and one of $\mathbf{O}, i : j$ or $\mathbf{O}, j : i$. Clearly, there is no model \mathcal{M} satisfying $\mathcal{M}, g(i) \not\models p$, $\mathcal{M}, g(j) \models p$ and $g(i) = g(j)$ at the same time. Thus, I could have easily won the game starting at D , a contradiction to the fact that σ is not a winning strategy for Me in $\mathbf{DG}(D, \rho)$. The cases for $\mathbf{P}, k : R(i, j)$ and $\mathbf{P}, i : j$ are similar.

Let us show some of the inductive steps: For the inductive step, suppose $g = \mathbf{P}, i : \psi_1 \vee \psi_2$ appears along π . By construction, both $\mathbf{O}, i : \psi_1$ and $\mathbf{O}, i : \psi_2$ appear along π . By inductive hypothesis, *You* have winning strategies μ_1 and μ_2 for $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_1)$ and $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_2)$ respectively. *You* can easily combine them to obtain a winning strategy for $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_1 \vee \psi_2)$: in the first round I choose to continue the game with $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_k)$ and *You* can use μ_k to win.

If $g = \mathbf{P}, i : \psi_1 \wedge \psi_2$ appears along π , then by construction, at least one of $\mathbf{P}, i : \psi_1$ and $\mathbf{P}, i : \psi_2$ appears along π . Without loss of generality, let $\mathbf{P}, i : \psi_1$ appear along π . By inductive hypothesis, *You* have a winning strategy μ for $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_1)$. Hence, *You* can win $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi_1 \wedge \psi_2)$ by choosing $\mathbf{P}, i : \psi_1$ in the first round and continuing along μ_1 .

If $\mathbf{P} : \psi$ appears along π , then by construction, there is a nominal i such that $\mathbf{P}, i : \psi$ appears along π . By inductive hypothesis, *You* have a winning strategy μ for $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, i : \psi)$. But then *You* can also win $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, \psi)$ by choosing i in the first round and continuing according to μ .

If $\mathbf{P}, i : \diamond\psi$ appears along π , then for every j , also $\mathbf{P}, j : R(i, j) \wedge \psi$ appears along π . By inductive hypothesis, *You* have a winning strategy μ_j for $\mathbf{G}(\mathcal{M}_\pi, \mathbf{P}, j : R(i, j) \wedge \psi)$ for every j . Clearly, *You* can combine them to a winning strategy for $\mathbf{G}(\mathcal{M}_\pi, i : \diamond\psi)$.

Proof (of Theorem 2, left-to-right). By contraposition: If D_0 is not winning, then, in particular, σ from above is not a winning strategy for $\mathbf{DG}(D_0, \rho_0)$. Let π be a path through σ rooted at D_0 , such that π does not end in a valid elementary leaf. By Lemma 4, *You* have a winning strategy for $\mathbf{G}(\mathcal{M}_\pi, g)$ for every g appearing along π . In particular, *You* have a winning strategy for all $\mathbf{G}(\mathcal{M}_\pi, g)$, where g is in D_0 . Hence, D_0 is not game valid.

5 From Strategies to Proof Systems

Proposition 1 shows that a regulation ρ and a winning strategy for *Me* in the game $\mathbf{DG}(\mathbf{P} : \phi)$ can be interpreted as a proof of ϕ . However, due to the rules (R_U) , (R_\square) and (R_\diamond) , this proof may be infinitely branching. In the following we will remedy this and give a finitized version of the disjunctive game.

5.1 Your Optimal Choices or How to Achieve Finite Branching

The idea is to use eigenvariables, similar to sequent calculi: for example, *I* should be able to win $\mathbf{DG}(D \vee (\mathbf{P}, i : \square\phi), \rho)$ iff *I* can win $\mathbf{DG}(D \vee (\mathbf{P}, j : \neg R(i, j) \vee \phi), \rho)$, if j stands for “an arbitrary nominal”. We may interpret this j as *Your* optimal choice. In technical terms, this translates into the condition that j does not appear in the state $D \vee (\mathbf{P}, i : \square\phi)$.

Proposition 2. *Let D be any disjunctive state.*

1. $\mathbf{P} : \phi$ is winning iff $\mathbf{P}, i : \phi$ is winning, where i is a nominal not occurring in ϕ .
2. $D \vee (\mathbf{P}, i : \square\phi)$ is winning iff $D \vee (\mathbf{P}, j : \neg R(i, j) \vee \phi)$ is winning, where j is a nominal not occurring in $D \vee (\mathbf{P}, i : \square\phi)$.
3. $D \vee (\mathbf{O}, i : \diamond\phi)$ is winning iff $D \vee (\mathbf{O}, j : R(i, j) \wedge \phi)$ is winning, where j is a nominal not occurring in $D \vee (\mathbf{O}, i : \diamond\phi)$.

Proof (sketch). Let us show 2: Let σ be a winning strategy for *Me* for the game $\mathbf{DG}(D \vee (\mathbf{P}, j : \neg R(i, j) \vee \phi), \rho)$. Then the tree rooted in $D \vee (\mathbf{P}, i : \square\phi)$ with σ as its immediate subtree defines a winning strategy for *Me* for $\mathbf{DG}(D \vee (\mathbf{P}, i : \square\phi), \rho')$, where ρ' is the same as ρ except that $\rho'(D \vee (\mathbf{P}, i : \square\phi)) = \mathbf{P}, i : \square\phi$.

If $D \vee (\mathbf{P}, i : \square\phi)$ is not winning, then, by Theorem 2, there is a model $\mathcal{M} = (W, R, V, g)$ such that *You* have winning strategies for all $\mathbf{G}(\mathcal{M}, g)$ for $g \in D$ and for $\mathbf{G}(\mathcal{M}, \mathbf{P}, i : \square\phi)$. In particular, *You* have a winning strategy for $\mathbf{G}(\mathcal{M}, \mathbf{P}, k : \neg R(i, k) \vee \phi)$ for some nominal k . Since truth of a formula in a model does not depend on nominals not appearing in that formula, we may assume that *You* have winning strategies for all $\mathbf{G}(\mathcal{M}', g)$ for $g \in D$ and for $\mathbf{G}(\mathcal{M}', \mathbf{P}, i : \square\phi)$. Here $\mathcal{M}' = (W, R, V, g')$ is the same as \mathcal{M} , except for g' : it agrees with g on all nominals appearing in D and ϕ , but $g'(j) = g(k)$. Now *You* have a winning strategy for $\mathbf{G}(\mathcal{M}', \mathbf{P}, j : \neg R(i, j) \vee \phi)$.

5.2 The Proof System DS

We are now ready to formulate the sequent calculus **DS** (Fig. 2). We say that a string $i : \phi$ consisting of a hybrid logic formula ϕ and a nominal i is a *labelled formula* and we call an object $\Gamma \vdash \Delta$, where Γ and Δ are multisets of formulas and labelled formulas, a *sequent*. A disjunctive state D can be rewritten as a sequent $\Gamma \vdash \Delta$ in the following way: Γ comprises all (labeled) formulas with the prefix **O** in D and Δ comprises all (labeled) formulas with the prefix **P** in D . For example, the disjunctive game state $\mathbf{O}, i : \Box p \vee \mathbf{P} : p$ becomes the sequent $i : \Box p \vdash p$. There is thus a 1-1 correspondence between sequents and disjunctive states.

Apart from the encoding of disjunctive states as sequents and the traditional bottom-up notation of proof trees, proofs in **DS** exactly correspond to *My* winning strategies in the disjunctive game: the order of rule application defines a regulation function and determines *My* moves in the strategy, branching in the proof tree corresponds to branching in the winning strategy, i.e. *Your* possible moves. Infinitary branching is modified according to the discussion in the previous subsection. Duplication in the game takes the form of left and right contraction rules. The base rules are exactly the (encoding¹³ of) valid disjunctive elementary states and thus winning for *Me*. Using this correspondence we immediately get the following results:

Proposition 3. $\vdash \phi$ is provable in **DS** iff $\mathbf{P} : \phi$ is winning.

Theorem 3. $\vdash \phi$ is provable in **DS** iff $\models \phi$.

The calculus **DS** takes a familiar form: The rules for the logical connectives are the (labeled) versions of the usual propositional rules of sequent calculus for classical logic. The modal operator rules come in the form of their first-order translations. Apart from the axioms, **DS** can be therefore seen as a fragment of the usual sequent system for first-order logic. In turn, **DS** is an extension of the sequent calculus **G3K** [11] to hybrid logic. Similarly to **G3K**, a failed proof search in **DS** directly gives rise to a countermodel. This follows from our proof of the left-to-right direction of Theorem 2, where the explicit countermodel \mathcal{M}_π was constructed.

Example 4. The rule

$$\frac{\Gamma, i : \phi \vdash i : \psi, \Delta}{\Gamma \vdash i : \phi \rightarrow \psi, \Delta} (R_{\rightarrow})$$

is derivable. This can be seen using the rules (R_{\rightarrow}^1) , (R_{\rightarrow}^2) and (CR) . Let us show how to prove $i : \Box(p \wedge q) \vdash i : \Box p$ in **DS**:

¹³ We say that $\Gamma \vdash \Delta$ is *elementary* iff its associated disjunctive state is. Similarly, $\Gamma \vdash \Delta$ is called *valid*, if its associated disjunctive state is game valid.

Base rules

$$\frac{}{\Gamma \vdash \Delta} (V) \text{ where } \Gamma \vdash \Delta \text{ is elementary and valid}$$

Structural Rules

$$\frac{\Gamma, i : \phi, i : \phi \vdash \Delta}{\Gamma, i : \phi \vdash \Delta} (CL)$$

$$\frac{\Gamma \vdash i : \phi, i : \phi, \Delta}{\Gamma \vdash i : \phi, \Delta} (CR)$$

$$\frac{\vdash i : \phi}{\vdash \phi} (U)$$

Logical connectives

$$\frac{\Gamma, i : \phi \vdash \Delta \quad \Gamma, i : \psi \vdash \Delta}{\Gamma, i : \phi \vee \psi \vdash \Delta} (L_{\vee}) \quad \frac{\Gamma \vdash i : \phi, \Delta}{\Gamma \vdash i : \phi \vee \psi, \Delta} (R_{\vee}^1)$$

$$\frac{\Gamma, i : \phi \vdash \Delta}{\Gamma, i : \phi \wedge \psi \vdash \Delta} (L_{\wedge}^1)$$

$$\frac{\Gamma \vdash i : \psi, \Delta}{\Gamma \vdash i : \phi \vee \psi, \Delta} (R_{\vee}^2)$$

$$\frac{\Gamma, i : \psi \vdash \Delta}{\Gamma, i : \phi \wedge \psi \vdash \Delta} (L_{\wedge}^2)$$

$$\frac{\Gamma \vdash i : \phi, \Delta \quad \Gamma \vdash i : \psi, \Delta}{\Gamma \vdash i : \phi \wedge \psi, \Delta} (R_{\wedge})$$

$$\frac{\Gamma, i : \phi \vdash \Delta}{\Gamma \vdash i : \phi \rightarrow \psi, \Delta} (R_{\rightarrow}^1)$$

$$\frac{\Gamma \vdash i : \phi, \Delta \quad \Gamma, i : \psi \vdash \Delta}{\Gamma, i : \phi \rightarrow \psi \vdash \Delta} (L_{\rightarrow})$$

$$\frac{\Gamma \vdash i : \psi, \Delta}{\Gamma \vdash i : \phi \rightarrow \psi, \Delta} (R_{\rightarrow}^2)$$

$$\frac{\Gamma \vdash i : \phi, \Delta}{\Gamma, i : \neg \phi \vdash \Delta} (L_{\neg})$$

$$\frac{\Gamma, i : \phi \vdash \Delta}{\Gamma \vdash i : \neg \phi, \Delta} (R_{\neg})$$

Modal operators

$$\frac{\Gamma, j : \neg R(i, j) \vee \phi \vdash \Delta}{\Gamma, i : \Box \phi \vdash \Delta} (L_{\Box})$$

$$\frac{\Gamma \vdash j : \neg R(i, j) \vee \phi, \Delta}{\Gamma \vdash i : \Box \phi, \Delta} (R_{\Box})$$

$$\frac{\Gamma, j : R(i, j) \wedge \phi \vdash \Delta}{\Gamma, i : \Diamond \phi \vdash \Delta} (L_{\Diamond})$$

$$\frac{\Gamma \vdash j : R(i, j) \wedge \phi, \Delta}{\Gamma \vdash i : \Diamond \phi, \Delta} (R_{\Diamond})$$

$$\frac{\Gamma, j : \phi \vdash \Delta}{\Gamma, i : @_j \phi \vdash \Delta} (L_{@})$$

$$\frac{\Gamma \vdash j : \phi, \Delta}{\Gamma \vdash i : @_j \phi, \Delta} (R_{@})$$

In the rules (R_{\Box}) , (L_{\Diamond}) and (U) , the nominal j must not occur in the lower sequent. This condition corresponds to *Your* optimal choices.

Fig. 2. The proof system **DS**

$$\begin{array}{c}
\frac{j : R(i, j) \vdash j : R(i, j)}{\vdash j : \neg R(i, j), j : R(i, j)} (R_{\neg}) \\
\frac{\vdash j : \neg R(i, j) \vdash j : \neg R(i, j)}{j : \neg R(i, j) \vdash j : \neg R(i, j)} (L_{\neg}) \\
\frac{j : \neg R(i, j) \vdash j : \neg R(i, j) \vee p}{j : \neg R(i, j) \vdash j : \neg R(i, j) \vee p} (R_{\vee}^2) \\
\frac{j : p \vdash j : p}{j : p \vdash j : \neg R(i, j) \vee p} (R_{\vee}^1) \\
\frac{j : p \wedge q \vdash j : \neg R(i, j) \vee p}{j : p \wedge q \vdash j : \neg R(i, j) \vee p} (L_{\wedge}^1) \\
\frac{j : \neg R(i, j) \vee (p \wedge q) \vdash j : \neg R(i, j) \vee p}{j : \neg R(i, j) \vee (p \wedge q) \vdash j : \neg R(i, j) \vee p} (L_{\vee}) \\
\frac{i : \Box(p \wedge q) \vdash j : \neg R(i, j) \vee p}{i : \Box(p \wedge q) \vdash j : \neg R(i, j) \vee p} (L_{\Box}) \\
\frac{i : \Box(p \wedge q) \vdash i : \Box p}{i : \Box(p \wedge q) \vdash i : \Box p} (R_{\Box})
\end{array}$$

The proof of $i : \Box(p \wedge q) \vdash i : \Box p$ looks similar. We can use both to give a proof of $\Box(p \wedge q) \rightarrow (\Box p \wedge \Box q)$:

$$\begin{array}{c}
\frac{i : \Box(p \wedge q) \vdash i : \Box p \quad i : \Box(p \wedge q) \vdash i : \Box q}{i : \Box(p \wedge q) \vdash i : \Box p \wedge \Box q} (R_{\wedge}) \\
\frac{i : \Box(p \wedge q) \vdash i : \Box p \wedge \Box q}{\vdash i : \Box(p \wedge q) \rightarrow (\Box p \wedge \Box q)} (R_{\rightarrow}) \\
\frac{\vdash i : \Box(p \wedge q) \rightarrow (\Box p \wedge \Box q)}{\vdash \Box(p \wedge q) \rightarrow (\Box p \wedge \Box q)} (R_U)
\end{array}$$

Remark 1. The above example demonstrates the importance of the regulation function ρ for the disjunctive game: the proof of $i : \Box(p \wedge q) \vdash i : \Box p$ relies on the possibility to expand the left-hand side before the right one. In game-theoretic terms, I have a winning strategy for $\mathbf{DG}(D, \rho)$, where $D = \mathbf{O}, i : \Box(p \wedge q) \vee \mathbf{P}, i : \Box p$ and $\rho(D) = \mathbf{O}, i : \Box(p \wedge q)$. If, on the other hand, ρ does never pick $\mathbf{O}, i : \Box(p \wedge q)$ (unless it has to), then I do not have a winning strategy in the corresponding game. This complication does not arise in disjunctive games where the set of actions is finite, for example [7]. Intuitively the idea there is for Me to duplicate and exhaustively make one choice after the other. After finitely many rounds, all the possible actions have been played and I can be sure to have played a good one, if there is any. It is clear that in general, this strategy does not work in the infinite case.

6 Conclusion and Future Work

In this paper, we developed a semantic game for hybrid logic and proved its adequacy: I have a winning strategy in the game of ϕ at world w over the model \mathcal{M} iff $\mathcal{M}, w \models \phi$. We proved that a version of the disjunctive game [7] models validity in hybrid logic adequately.

In this paper, we have only looked at game-theoretic modelling of validity in basic hybrid logic. However, hybrid language allows us to characterize many classes of frames that lie beyond the expressivity of orthodox modal logic [2]. In the future, we plan to make use of this feature and present an interesting extension of the disjunctive game for hybrid logic to model validity over classes of frames characterizable in hybrid language and investigate the connections to Blackburn's dialogue game [2].

Our aim, however, was not only to give game-theoretic characterizations of truth in a model and validity, but to do so in a way that allows for a natural lifting

of the first to the second: the disjunctive game corresponds to simultaneously playing the semantic game over all possible models. The search for a winning strategy, in turn, can be straightforwardly formulated as an analytic calculus.

The lifting of semantic games to analytic calculi via a disjunctive game has been demonstrated on the propositional level for classical logic, Łukasiewicz logic and Gödel logic [6, 7]. Our approach is the first to deal with infinitely many possible actions and we are keen to extend the known results to the first-order case of the above logics. Ultimately, the liftings rely on similar game-theoretic properties of the semantic and the corresponding disjunctive game. We plan to develop a powerful general lifting algorithm by exploiting, on an abstract level, the relevant features of these similarities.

References

1. van Benthem, J.: *Logic in Games*. The MIT Press, Cambridge (2014)
2. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2002). <https://books.google.at/books?id=gFEidNVDWVoC>
3. Blackburn, P.: Modal logic as dialogical logic. *Synthese* **127**, 57–93 (2001). <https://doi.org/10.1023/A:1010358017657>
4. Blackburn, P., Tzakova, M.: Hybridizing concept languages. *Ann. Math. Artif. Intell.* **24**, 23–49 (2000). <https://doi.org/10.1023/A:1018988913388>
5. Blackburn, P., Tzakova, M., Gargov, I.: Hybrid languages and temporal logic. *Log. J. IGPL* **7**, 27–54 (1999). <https://doi.org/10.1093/jigpal/7.1.27>
6. Fermüller, C., Lang, T., Pavlova, A.: From truth degree comparison games to sequents-of-relations calculi for Gödel logic. In: Reformat, M.-J., et al. (eds.) *IPMU 2020*. CCIS, vol. 1237, pp. 257–270. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50146-4_20
7. Fermüller, C.G., Metcalfe, G.: Giles’s game and the proof theory of Łukasiewicz logic. *Stud. Log.: Int. J. Symb. Log.* **92**(1), 27–61 (2009). <http://www.jstor.org/stable/40269050>
8. Hintikka, J.: *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic*. Clarendon Press, Oxford (1973)
9. Hodges, W., Väänänen, J.: Logic and games. In: Zalta, E.N. (ed.) *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2019 edn. (2019)
10. Lorenzen, P., Lorenz, K.: *Dialogische Logik*. Wissenschaftliche Buchgesellschaft (1978)
11. Negri, S.: Kripke completeness revisited. In: Primiero, G. (ed.) *Acts of Knowledge: History, Philosophy and Logic*, pp. 233–266. College Publications (2009)
12. Väänänen, J.: *Models and Games*. Cambridge Studies in Advanced Mathematics, Cambridge University Press, Cambridge (2011). <https://doi.org/10.1017/CBO9780511974885>



Verifying the Conversion into CNF in Dafny

Viorel Iordache and Ștefan Ciobâcă^(✉)

Alexandru Ioan Cuza University, Iași, Romania

Abstract. We present two computer-verified implementations of the CNF conversion for propositional logic. The two implementations are fully verified: functional correctness and termination is machine-checked using the Dafny language for both. The first approach is based on repeatedly applying a set of equivalences and is often presented in logic textbooks. The second approach is based on Tseitin's transformation and is more efficient. We present the main ideas behind our formalization and we discuss the main difficulties in verifying the two algorithms.

1 Introduction

Several computer-checked solvers for the boolean satisfiability problem have emerged relatively recently [5, 9, 18, 22]. Most SAT solvers work with CNF-SAT, where the input formula is known to be in conjunctive normal form. This is not a limitation, since efficient algorithms to find the CNF of any formula are known. In this article, we address the problem of finding the CNF of a formula and we verify in the Dafny [15] language¹ two such algorithms.

The first approach that we verify is based on the standard textbook algorithm of applying a series of equivalences from left to right as long as possible. We work with the following nine equivalences:

$$\begin{aligned} (1) \quad \varphi_1 \Leftrightarrow \varphi_2 &\equiv (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1); & (2) \quad \varphi_2 \Rightarrow \varphi_1 &\equiv \neg\varphi_1 \vee \varphi_2 \\ (3) \quad \varphi_1 \vee (\varphi_2 \wedge \varphi_3) &\equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3); \\ (4) \quad (\varphi_2 \wedge \varphi_3) \vee \varphi_1 &\equiv (\varphi_2 \vee \varphi_1) \wedge (\varphi_3 \vee \varphi_1); \\ (5) \quad \varphi_1 \vee (\varphi_2 \vee \varphi_3) &\equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3; & (6) \quad \varphi_1 \wedge (\varphi_2 \wedge \varphi_3) &\equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3; \\ (7) \quad \neg(\varphi_1 \vee \varphi_2) &\equiv \neg\varphi_1 \wedge \neg\varphi_2; & (8) \quad \neg(\varphi_1 \wedge \varphi_2) &\equiv \neg\varphi_1 \vee \neg\varphi_2; & (9) \quad \neg\neg\varphi &\equiv \varphi. \end{aligned}$$

The first two equivalences remove implications and double implications, equivalences three and four distribute disjunctions over conjunctions, equivalences five and six associate parentheses in a standard form and the last three are used to push negations towards the leaves.

This approach has two advantages: it is simple and it does not introduce new variables. However, the disadvantage is that certain classes of formulae, such as $(x_1 \wedge x'_1) \vee (x_2 \wedge x'_2) \vee \dots \vee (x_n \wedge x'_n)$ ($n \geq 1$), lead to exponentially large CNFs.

¹ We assume some familiarity with a system such as Dafny [15] or Why3 [6]. We give a very brief overview of Dafny in Appendix A for readers unfamiliar with Dafny.

First Contribution. We verify in Dafny that an algorithm based on applying the nine equivalences above is functionally correct. The most difficult part is to prove termination, for which we use a carefully designed 5-tuple as a variant. To our knowledge, this is incidentally the first proof (paper or computer-checked) of termination of the nine rules. Indeed, in all logic textbooks that we surveyed, termination is only proved for a certain strategy (first applying rules 1 and 2, then finding a NNF using rules 7–9, etc.) of applying the rules. The interest into this is of theoretical interest, since other strategies (such as bringing the formula into NNF first) are easier to prove.

The second approach that we verify is a so-called definitional CNF [12] based on Tseitin’s² transformation [24]. The idea is to create fresh boolean variables for each subformula. Each such fresh variable is constrained, by using carefully chosen clauses in the resulting CNF, to be equivalent to its associated subformula. This approach produces a CNF that is linear in the size of the given formula, with the theoretical disadvantage that the CNF is only equisatisfiable with the initial formula (not equivalent in general).

Second Contribution. We verify an implementation of Tseitin’s transformation in Dafny. The main difficulty is to find the right inductive invariant. There are also some technical difficulties with the verification: our implementation pushes the prover to its limits and requires carefully designed lemmas, helper predicates, and assertions in order to verify successfully. For this approach, termination is established by Dafny automatically, since the function is recursive on the formula ADT. To our knowledge, this is the first auto-active proof of Tseitin’s transformation.

Paper Structure. In Sect. 2 we present our verified implementation of the textbook-based CNF transformation and in Sect. 3 we present our verified implementation of Tseitin’s transformation. In Sect. 4 we discuss related work and in Sect. 5 we conclude and discuss potential future work. There are two appendices: in Appendix A we give a very short overview of Dafny and in Appendix B we discuss the structure of our Dafny development. The Dafny development is available at <https://github.com/iordacheviorel/cnf-dafny>. We only present the most important parts of the development, which are necessary in order to understand our approach.

2 The Textbook Conversion into CNF

We describe our verified implementation of the textbook CNF transformation.

2.1 Data Structures

We represent boolean formulas as instances of the following algebraic data type:

```
datatype FormulaT = Var(val : int) | Not(f1 : FormulaT)
| And(f1 : FormulaT, f2 : FormulaT) | Implies(f1 : FormulaT, f2 : FormulaT)
| Or(f1 : FormulaT, f2 : FormulaT) | DImplies(f1 : FormulaT, f2 : FormulaT)
```

² Also spelled *Tseytin*.

We note that all standard logical connectives are available, that the connectives have a fixed arity, and that variables are represented by *non-negative integers*. The fact that variables are represented by non-negative integers is not encoded into the datatype; instead, we follow the standard Dafny practice of checking this as a precondition to all functions/methods computing with formulae by using a predicate that we call `validFormulaT`. We often require to know not merely that a formula is syntactically valid, but also to know that it contains at most n variables. For this purpose, we use a predicate `variablesUpTo`. We define `truthValue` to compute the truth value of a formula in a given assignment:

```
function method truthValue(f: FormulaT, assignment : seq<bool>) : bool
  decreases f; requires variablesUpTo(f, |assignment|);
{ match f {
  case Var(val) => assignment[val]
  case Not(f1) => ¬truthValue(f1, assignment)
  case And(f1,f2) => truthValue(f1,assignment) ∧ truthValue(f2,assignment)
  [...] } }
```

Truth assignments are represented as sequences of booleans (`seq<bool>`) that have sufficiently many elements to account for all variables in the formula, hence the `requires variablesUpTo(f, |assignment|)` precondition. As the truth value of a formula is used both in the specification and in the implementation of the algorithm, we declare it as a `function method`.

We define the predicate `equivalent` to check equivalence of two formulae:

```
predicate equivalent(f1 : FormulaT, f2 : FormulaT)
  requires validFormulaT(f1) ∧ validFormulaT(f2);
{ ∀ tau : seq<bool> • variablesUpTo(f1,|tau|) ∧ variablesUpTo(f2,|tau|)
  ⇒ truthValue(f1, tau) = truthValue(f2, tau) }
```

The predicate checks that the truth values of the two formulae are the same in any (sufficiently large) truth assignment.

2.2 Algorithm

We implement the CNF conversion algorithm using three methods:

- The method `applyAtTop` takes a formula and tries to apply one of the nine equivalence rules *at the root of the formula*. If it fails, the formula is returned unchanged.

```
method applyAtTop(f:FormulaT, ghost orsAbvLft:int, ghost andsAbvLft:int)
  returns (r : FormulaT) decreases f;
  requires orsAbvLft ≥ 0 ∧ andsAbvLft ≥ 0 ∧ validFormulaT(f);
  ensures validFormulaT(r) ∧ equivalent(f, r);
  ensures f = r ⇒ ¬f.Implies? ∧ f = r ⇒ ¬f.DImplies?;
  ensures r = f ∨ Utils.smaller(measure(r, orsAbvLft, andsAbvLft),
    measure(f, orsAbvLft, andsAbvLft));
{ match f {
  case DImplies(f1, f2) => { r := applyRule1(f, orsAbvLft, andsAbvLft); }
  case Implies(f1, f2) => { r := applyRule2(f, orsAbvLft, andsAbvLft); }
  case Or(f1, f2) => { if (f2.And?) {
    r := applyRule3(f, orsAbvLft, andsAbvLft); } [...]
  } [...] } [...] }
```

We present the entire specification (pre- and post-conditions), but we skip some implementation details (replaced by `[...]`). The two ghost parameters are used in the termination proof, as discussed in Sect. 2.3. The first `ensures` clause is used for functional correctness. The last `ensures` clauses are used to prove termination of the main algorithm. In particular, the function `measure` returns a tuple that acts as the variant of the main algorithm.

We also define the methods `applyRule1`, `applyRule2`, `...`, `applyRule9`, which apply one of the nine rules. We present the first of these methods:

```
method applyRule1(f : FormulaT, ghost orsAbvLft : int,
  ghost andsAbvLft : int)
  returns (r : FormulaT) requires validFormulaT(f) ∧ f.DImplies?;
  requires orsAbvLft ≥ 0 ∧ andsAbvLft ≥ 0;
  ensures validFormulaT(r) ∧ equivalent(f, r);
  ensures weightOfAnds(r) ≤ weightOfAnds(f);
  ensures countDImplies(r) < countDImplies(f);
  ensures smaller(measure(r, orsAbvLft, andsAbvLft),
    measure(f, orsAbvLft, andsAbvLft));
{ var DImplies(f1, f2) := f;
  r := And(Implies(f1, f2), Implies(f2, f1));
  [...] }
```

Again, we show the entire specification, most of which is needed for the termination proof. The missing part in the implementation, denoted by `[...]`, are helper assertions and lemma calls that are required to prove the postconditions.

- The method `applyRule` takes a formula, traverses its tree in preorder, and calls `applyAtTop` to transform the first subformula where it is possible to do so at the root. Therefore, the method `applyRule` makes exactly one effective call to `applyAtTop`. We present the entire specification and the implementation of one of the cases for `applyRule`:

```
method applyRule(f : FormulaT, ghost orsAbvLft : int, ghost andsAbvLft : int)
  returns (r : FormulaT) decreases f;
  requires validFormulaT(f) ∧ orsAbvLft ≥ 0 ∧ andsAbvLft ≥ 0;
  ensures validFormulaT(r) ∧ equivalent(f, r);
  ensures r = f ∨ Utils.smaller(measure(r, orsAbvLft, andsAbvLft),
    measure(f, orsAbvLft, andsAbvLft));
{ var res : FormulaT := applyAtTop(f, orsAbvLft, andsAbvLft);
  if (res ≠ f) { return res; } else if (f.Or?) {
    var f1_step := applyRule(f.f1, orsAbvLft, andsAbvLft);
    if (f.f1 = f1_step) {
      var f2_step := applyRule(f.f2, orsAbvLft + 1, andsAbvLft);
      assert equivalent(f.f2, f2_step);
      assert equivalent(Or(f.f1, f.f2), Or(f.f1, f2_step));
      res := Or(f.f1, f2_step);
      if (weightOfAnds(f2_step) < weightOfAnds(f.f2)) {
        Rule30r(f.f1, f.f2, f2_step); }
      return res;
    } else { [...] }
  } else if (f.And?) { [...] } else if (f.Not?) { [...] } }
```

Note again that the two ghost parameters are only used to help prove termination of the main algorithm and that the pre- and post-conditions are very similar to `applyRule`. Also note that there are no cases for `f.Implies?` and `f.DImplies?` since in these two cases `applyAtTop` is forced to make progress. The lemma `Rule30r` is used to propagate a termination variant upwards in the tree of the formula and is explained in Sect. 2.3.

- The main method in the algorithm is `convertToCNF`. It takes a formula and calls `applyRule` on it in a recursive loop until there are no more changes.

```

method convertToCNF(f : FormulaT) returns (r : FormulaT)
  decreases weightOfAnds(f); /*3,4,7,8,9*/ decreases countDImplies(f); /*1*/
  decreases countImplies(f); /*2*/      decreases countOrPairs(f,0); /*5*/
  decreases countAndPairs(f, 0); /*6*/   requires validFormulaT(f);
  ensures validFormulaT(r) ∧ equivalent(f, r);
{ var res := applyRule(f, 0, 0); assert equivalent(f, res);
  if(res ≠ f) { r := convertToCNF(res);
               assert equivalent(res, r); [...]
  } else { r := res; } }

```

The main difficulty here is to prove the termination of this fixed-point method. For this purpose, we use as a variant a 5-tuple **measure**, whose definition we unfold in the five **decreases** clauses of `convertToCNF`. The numbers in the comments represent the equivalences among the set of nine that ensure a strict decrease of the particular element of the tuple.

2.3 Proof of Termination

In this section, we discuss in more depth the proof of termination which *seems* intuitively easy: (I1) it *seems* that the first two equivalences strictly decrease the number of double implications and implications, respectively; (I2) it *seems* that equivalences three and four strictly decrease the number of disjunctions that sit above conjunctions in the tree of the formula; (I3) it *seems* that rule five (resp. six) strictly decrease the number of **ors** just above and to the left of another **or** (resp. **and**); (I4) it *seems* that rules 7–9 strictly decrease the number of negations above conjunctions and disjunctions. The above intuition works when using a particular strategy to compute a CNF: first, remove double implications; secondly, remove implications; thirdly, compute the NNF, etc.

Difficulties. Unfortunately, all of the above intuition breaks when the rules can be applied in any order. There are two main difficulties with the variant candidates above: (D1) Equivalences one, three, and four are not right-linear; they might duplicate subformulae. Therefore, the variants suggested by intuitions I1, I2, I3, I4 (e.g., number of double implications) might actually increase when applying one of these equivalences. The apparent solution of counting the number of distinct subformulae rooted in a double implication instead of the number of double implications does not work either: after the initial duplication, the two subformulae might be transformed in different ways. (D2) The variants suggested by intuitions I1–I4 do not in general extend homomorphically to the root of a formula when the transformation is performed in a proper subformula. Take for example the number of **ands** directly above and to the left of another **and** node (intuition I3). If we apply rule three in a context of the form $\varphi \wedge \square$, we obtain $\varphi \wedge ((\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3))$ from $\varphi \wedge (\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$ and therefore our variant candidate actually increases at the root.

The Main Variant. In order to prove termination, we rely instead on a numerical interpretation of formulae that we call **weightOfAnds**. It decreases strictly in equivalences 3, 4, 7, 8, and 9 and it does not increase in the other equivalences.

```

function weightOfAnds(f : FormulaT) : (res : int)
  decreases f; ensures res ≥ 2;
{ match f {
  case Var(val)           ⇒ 2
  case Not(f1)            ⇒ pow(2, weightOfAnds(f1))
  case And(f1, f2)       ⇒ weightOfAnds(f1) + weightOfAnds(f2) + 1
  case Or(f1, f2)        ⇒ weightOfAnds(f1) * weightOfAnds(f2)
  case Implies(f1, f2)   ⇒ pow(2, weightOfAnds(f1)) * weightOfAnds(f2)
  case DImplies(f1,f2) ⇒ (pow(2,weightOfAnds(f1)) * weightOfAnds(f2)) +
                        (pow(2,weightOfAnds(f2)) * weightOfAnds(f1)) + 1 } }

```

Intuitively (although we admit this is not a perfect intuition), this function computes the number of conjunctions that a formula might have when brought into CNF, hence the $+ 1$ in the `And(f1, f2)` case. For disjunctions, intuitively the `Or` needs to be distributed over all `And`s and therefore we use multiplication. For negation, the number might increase exponentially (negation “turns” `Or`s into `And`s and vice-versa). The `pow` function is not builtin; it performs exponentiation and is defined in the Dafny development along with several helper lemmas. For technical reasons, for leaves the counting starts at 2 (case `Var(val) => 2`).

The cases for implication and double implication are handled as if an implication $\varphi_1 \Rightarrow \varphi_2$ is just syntactic sugar for $\neg\varphi_1 \vee \varphi_2$ and therefore applying equivalences 1 and 2 trivially does not change the value of `weightOfAnds`. Equivalences 5 and 6 also do not change the value of `weightOfAnds`, as the numerical interpretations of \vee and of \wedge are associative. This numerical interpretation is homomorphic, and therefore difficulty D2 is handled.

Secondary Variants. The main variant establishes termination of rules 3, 4, 7, 8, and 9. To establish termination of the entire system, we require 4 more secondary variants. The second and third elements of the tuple (`countDImplies(f)` and `countImplies(f)`) unsurprisingly count the number of double implications and implications in a formula, respectively. These decrease strictly when applying equivalences 1 and 2, respectively.

More interestingly, we discuss the last two components: `countOrPairs(f, 0)` and `countAndPairs(f, 0)`. These are used to establish termination of the rules for associating parentheses to the left (equivalences 5 and 6). As explained in difficulty D2 above, simply counting the number of `And` nodes directly above and to the left of another `And` node does not work, since the inequality required of such a variant does not propagate from a subformula towards the root. Therefore, we count instead the number of pairs of nodes labeled `And` such that one is a (possibly indirect) ancestor of the other towards the left. We display `countOrPairs`, as the other function is similar:

```

function countOrPairs(f : FormulaT, orsAbvLft : int) : (res : int)
  decreases f; requires orsAbvLft ≥ 0; ensures res ≥ 0;
{ match f {
  case Or(f11,f12) ⇒ countOrPairs(f11, orsAbvLft) +
                    countOrPairs(f12, orsAbvLft + 1) + orsAbvLft // note the “+ 1”
  case Var(val) ⇒ 0
  case And(f11,f12) ⇒ countOrPairs(f11, orsAbvLft) +
                    countOrPairs(f12, orsAbvLft)
  [...] } }

```


The helper parameter represents the number of `Ors` above and to the left of the current subformula and should therefore be 0 in the initial call. This variant propagates as required.

In order to implement this variant, we require to keep track in the `applyRule` method of the number of `Ors` and `Ands` that are possibly indirect ancestors towards the left of the current subformula. These numbers account for the two ghost parameters `orsAbvLft` and `andsAbvLft` of the `applyRule` and `applyAtTop` methods mentioned above and left for the current subsection. Initially, when `applyRule` is called in `convertToCNF`, both are initialized to 0.

The first three components of the variant do not commute, but the last two could be interchanged without affecting the termination proof.

3 The Tseitin Conversion

We discuss our formalization of the Tseitin conversion into CNF [24], also called definitional CNF [12]. We first briefly recall the Tseitin transformation via a short example.

Example 1. Consider the formula $\varphi = \neg x_1 \vee x_2$. Choose fresh variables y_1, y_2 for the two subformulae $\neg x_1$ and $\neg x_1 \vee x_2$, respectively. Add to the resulting CNF clauses that encode that each of the two variables is equivalent to the corresponding subformula: 1. for $y_1 \equiv \neg x_1$, add the clauses $y_1 \vee x_1, \neg y_1 \vee \neg x_1$; 2. for $y_2 \equiv \neg x_1 \vee x_2$, add the clauses: $\neg y_1 \vee y_2, \neg x_2 \vee y_2, \neg y_2 \vee y_1 \vee x_2$.

Finally, add a clause consisting of a single literal: the variable corresponding to the initial formula³. The final result is: $(y_1 \vee x_1) \wedge (\neg y_1 \vee \neg x_1) \wedge (\neg y_1 \vee y_2) \wedge (\neg x_2 \vee y_2) \wedge (\neg y_2 \vee y_1 \vee x_2) \wedge y_2$. The result of applying our implementation of Tseitin's algorithm on the above formula is the sequence of sequences $[[3, 1], [-3, -1], [-3, 4], [-2, 4], [-4, 3, 2], [4]]$, where the numbers 1, 2, 3, and 4 represent the variables x_1, x_2, y_1 , and y_2 , respectively.

The guarantee offered by this transformation is that the resulting formula is equisatisfiable to the initial formula, as opposed to equivalent for the textbook transformation discussed in Sect. 2. The advantage is that the size is at most $O(1)$ times bigger than the initial formula.

3.1 Data Structures

In verifying the definitional CNF transformation, we use the same data type `FormulaT` for the input formula as in the previous formalization. However, for the output formula, we choose to represent literals as integers, clauses as sequences of integers and CNF formulae as sequences of clauses, similar to the well-known DIMACS format:

³ Tseitin [24] proposes to add the negation of this literal, the initial formula being valid iff the resulting formula is unsatisfiable; modern treatments diverge [12].

```

predicate method validLiteral(lit : int) { lit ≤ -1 ∨ lit ≥ 1 }
predicate validClause(clause : seq<int>) {
  ∀ lit • lit in clause ⇒ validLiteral(lit) }
predicate validCnfFormula(f : seq<seq<int> >) {
  ∀ clause : seq<int> • clause in f ⇒ validClause(clause) }

```

Note that `valid` in the predicates above reflects standard Dafny use rather than semantical validity in the logical sense. As variables are represented by non-negative integers, we consistently use the following function (methods) to convert between variables and literals:

```

predicate validVariable(v : int) { v ≥ 0 }
function method posVarToLit(v : int) : int
  requires validVariable(v);
  ensures posVarToLit(v) ≥ 1 ∧ validLiteral(posVarToLit(v));
{ v + 1 }
function method negVarToLit(v : int) : int
  requires validVariable(v);
  ensures negVarToLit(v) ≤ -1 ∧ validLiteral(negVarToLit(v));
{ (-v) - 1 }
function method litToVar(l : int) : int
  requires validLiteral(l);
  { if (l ≤ -1) then (-l) - 1 else l - 1 }

```

We represent assignments as sequences of booleans, just like in the textbook transformation. To compute truth values, we use the following predicates:

```

predicate truthValueLiteral(lit : int, tau : seq<bool>)
  requires validLiteral(lit) ∧ variablesUpToLiteral(lit, |tau|);
{ if lit < 0 then ¬tau[litToVar(lit)] else tau[litToVar(lit)] }
predicate truthValueClause(clause : seq<int>, tau : seq<bool>)
  requires validClause(clause) ∧ variablesUpToClause(clause, |tau|);
{ ∃ lit • lit in clause ∧ truthValueLiteral(lit, tau) }
predicate truthValueCnfFormula(rf : seq<seq<int> >, tau : seq<bool>)
  requires validCnfFormula(rf) ∧ variablesUpToCnfFormula(rf, |tau|);
{ ∀ clause | clause in rf • truthValueClause(clause, tau) }

```

The predicates `variablesUpTo*` (with $* \in \{\text{CnfFormula}, \text{Clause}, \text{Literal}\}$), check that the assignment is sufficiently large to account for all variables occurring in the formulae. We consistently use the following convention: program variables such as `f`, `f1`, `f2` stand for formulae of type `FormulaT` and program variables such `rf`, `rf1` stand for *resulting formulae* of type `seq<seq<int> >`.

As the guarantee of the Tseitin transformation is equisatisfiability between the initial formula and the resulting formula, we model this by using the following predicates:

```

predicate satisfiable(f : FormulaT) requires validFormulaT(f);
{ ∃ tau | |tau| = maxVar(f) • truthValue(f, tau) }
predicate satisfiableCnfFormula(rf : seq<seq<int> >) requires validCnf[..](rf);
{ ∃ tau | |tau| = maxVarCnfFormula(rf) • truthValueCnfFormula(rf, tau) }
predicate equiSatisfiable(f : FormulaT, rf : seq<seq<int> >)
  requires validFormulaT(f); requires validCnfFormula(rf);
{ satisfiable(f) ⇔ satisfiableCnfFormula(rf) }

```

The function methods `maxVar*` compute the maximum natural number that represents a variable inside the formulae, plus one. Therefore an assignment of size `maxVar*` is sufficiently large to compute the truth value.

3.2 The Algorithm

The entry point to the algorithm is the method `tseitin`.

```
method tseitin(f : FormulaT) returns (result : seq<seq<int> >)
  requires validFormulaT(f);
  ensures validCnfFormula(result) ^ equiSatisfiable(f, result);
{ var n := maxVar(f); var v : int; var end : int; var rf : seq<seq<int> >;
  rf, v, end := tseitinCnf(f, n, n);
  result := rf + [[posVarToLit(v)]]; [...] }
```

The guarantee is that the resulting CNF formula is equisatisfiable to the initial one. The main work is performed by the method `tseitinCnf`, which traverses the input formula recursively and adds the right clauses to the result:

```
method tseitinCnf(f : FormulaT, n : int, start : int)
  returns (rf : seq<seq<int> >, v : int, end : int)
  requires variablesUpTo(f, n) ^ start ≥ n ≥ 0;
  ensures valid(f, rf, v, n, start, end);
  ensures tseitinSameValue(f, rf, v, n, start, end);
  ensures tseitinCanExtend(f, rf, v, n, start, end);
{ match f {
  case Or(f1, f2) ⇒ {
    var rf1 : seq<seq<int> >; var rf2 : seq<seq<int> >;
    var v1 : int; var v2 : int;
    var mid : int;
    rf1, v1, mid := tseitinCnf(f1, n, start);
    rf2, v2, v := tseitinCnf(f2, n, mid);
    end := v + 1;
    rf := rf1 + rf2 + orClauses(v1, v2, v);
    proveCanExtendOr(f1, rf1, v1, f2, rf2, v2, n, start, mid, v, end, rf);
    proveSameValueOr(f1, rf1, v1, f2, rf2, v2, n, start, mid, v, end, rf);
  }
  case And(f1, f2) ⇒ [...] case Implies(f1, f2) ⇒ [...]
  case DImplies(f1, f2) ⇒ [...] case Not(f1) ⇒ [...]
  case Var(val) ⇒ [...]
} }
```

The method `tseitinCnf` takes as input: 1. a formula `f` to transform into CNF, which might be a subformula of the initial formula given to `tseitin`; 2. a natural number `n` with the meaning that all variables in the initial formula are between 0 and $n - 1$; 3. a natural number `start`, with the meaning that the variables `start`, `start + 1`, `start + 2`, ... are not used and can be safely used as fresh variables by the method. Variables between `n` (inclusively) and `start` (exclusively) might have been used for some other subformulae.

The method `tseitinCnf` returns as output: 1. A set of clauses `rf` encoding that the freshly chosen variables are equivalent to the corresponding subformulae; 2. A variable `v` that corresponds to the input formula `f`; 3. A number `end` with the meaning that the recursive call used fresh variables between `start` (inclusively) and `end` (exclusively) and therefore the variables `end`, `end + 1`, `end + 2`, ... can be used safely as fresh after the call is finished. The predicate `valid` is used to account for the validity of the entire state of the algorithm:

```
predicate valid(f : FormulaT, rf : seq<seq<int> >, v : int,
  n : int, start : int, end : int)
{ 0 ≤ n ≤ start ≤ end ^ variablesUpTo(f, n) ^ validCnfFormula(rf) ^
  validVariable(v) ^ variableInInterval(v, n, start, end) ^
  variablesInInterval(rf, n, start, end) }
```

The predicate `variablesInInterval`(`rf`, `n`, `start`, `end`) checks that `rf` uses only the initial variables (between 0 and $n - 1$) and fresh variables between `start` (inclusively) and `end` (exclusively).

We discuss in more detail the implementation of the `Or` case in `tseitinCnf` presented above. Note how the recursive call on `f1` uses fresh variables between `start` (inclusively) and `mid` (exclusively), while the recursive call on `f2` uses fresh variables between `mid` (inclusively) and `v` (exclusively). The variable `v` is therefore used as the fresh variable corresponding to the entire formula `f = Or(f1, f2)`. The final set of clauses is then the union of `rf1` (set of clauses corresponding to `f1`), `rf2` (set of clauses corresponding to `f2`) and `orClauses(v1, v2, v)`, which encodes that `v` should be equivalent to `Or(v1, v2)`:

```
function method orClauses(v1 : int, v2 : int, v : int) : seq<seq<int> >
  requires validVariable(v1) ^ validVariable(v2) ^ validVariable(v);
  { [[negVarToLit(v), posVarToLit(v1), posVarToLit(v2)],
    [negVarToLit(v1), posVarToLit(v)], [negVarToLit(v2), posVarToLit(v)]] }
```

3.3 The Proof

The main difficulty in verifying the algorithm is coming up with the right invariants. Assuming that `tseitinCnf(f, n, start)` returns `rf`, `v`, `end`, we find that the following two invariants explain the functional correctness of the algorithm: 1. any truth assignment `tau` to the initial `n` variables can be extended (uniquely) to a truth assignment `tau'` that makes `rf` true and such that the value of `f` in `tau` is the same as the value of `v` in `tau'`; 2. vice-versa, any truth assignment to all `end` variables that makes `rf` true also makes `v` and `f` have the same truth value. We formalize the two invariants above in the predicates `tseitinCanExtend` and `tseitinSameValue`:

```
predicate tseitinCanExtend(f : FormulaT, rf : seq<seq<int> >,
  v : int, n : int, start : int, end : int)
  requires valid(f, rf, v, n, start, end);
  { ∀ tau : seq<bool> | |tau| = n • canExtend(tau, f, rf, v, n, start, end) }
```

```
predicate canExtend(tau : seq<bool>, f : FormulaT, rf : seq<seq<int> >,
  v : int, n : int, start : int, end : int)
  requires |tau| = n ^ valid(f, rf, v, n, start, end);
  { ∃ tau' : seq<bool> | tau ≤ tau' ^ |tau'| = end •
    truthValueCnfFormula(rf, tau') ^ truthValue(f, tau) =
    truthValueLiteral(posVarToLit(v), tau') }
```

```
predicate tseitinSameValue(f : FormulaT, rf : seq<seq<int> >,
  v : int, n : int, start : int, end : int)
  requires valid(f, rf, v, n, start, end);
  { ∀ tau : seq<bool> | |tau| ≥ end ^ truthValueCnfFormula(rf, tau) •
    [...] truthValueLiteral(posVarToLit(v), tau) = truthValue(f, tau) }
```

We find that because `tseitinCanExtend` is of the form $\forall \dots \exists \dots$ (nested quantifiers), it is useful for verification performance to give a name to the $\exists \dots$ part, hence the predicate `canExtend`. Dafny cannot prove the two predicates automatically, and therefore we design helper lemma for each of the two invariants and for each of the cases (`Or`, `And`, `Not`, \dots). The main idea behind these proof is to combine two assignments `tau1` and `tau2`, which necessarily agree on the first `n` variables (the variables in the initial formula), into a single assignment `tau'`.

This is possible since the *interesting* assignments in `tau1` range from `start` to `mid` and the *interesting* assignments in `tau2` range from `mid` to `v`; that is they are disjoint. We elide the computer-checked proof for space reasons.

4 Related Work

The work closest to ours is by Barroso et al. [3], who verify a CNF transformation for propositional logic in the Why3 [6] verification platform. They use the textbook approach, but they rely on a particular strategy (first, remove implication, then: compute the negation normal form, etc.) This makes their proof much simpler, especially w.r.t. termination. One theoretical difference is that they model truth assignments as functions from variables to truth values and therefore their specification is closer to the mathematical treatment of logic (we instead model truth assignment as finite sequences, but we ensure they are sufficiently large for the context in which they are used). Barroso et al. emphasize the verification of continuation-passing style of the CNF transformation, which is out of the scope of our paper.

Michaelis and Nipkow [20] mechanize and prove Tseitin’s transformation in Isabelle/HOL as part of the formalization [21] of a series of propositional proof systems. The implementation is functional, based on first generating fresh names for all distinct subformulae and then adding the corresponding clauses for each internal node of the input formula. The fact that the fresh names are generated at the very beginning seems to make the proof simpler. As the emphasis is placed on metatheoretical considerations, efficiency is not the main concern. In our Dafny approach, the implementation is more efficient and is compositional: each subformula is recursively translated into a set of clauses.

Gäher and Kunze [11] implement and verify Tseitin’s transformation in the Coq proof assistant as part of the proof of the Cook-Levin theorem. The algorithm is implemented as a fixpoint (terminating, pure, recursive function) in the functional language of the Coq proof assistant. The function is very similar to our implementation: it takes a subformula and a natural starting from which fresh identifiers can be chosen. It returns the set of clauses for the subformula, the new variable associated to the subformula and a new number to be used for freshness. The inductive invariant `tseytin_formula_repr` used for the proof is also very similar to what we have independently found. Since the implementations are in very different proof environments, isolating Tseitin’s transformation in both developments and performing a more detailed comparison could be used to understand the pros and cons of the two proof assistants (Dafny and Coq).

Verified transformation into CNF should be a first step in verified SAT solvers that take as input arbitrary formulae. The SAT solver `versat` [22] was implemented and verified in the Guru programming language using dependent types. The solver is verified to be sound: if it produces an `UNSAT` answer, then the input formula truly is unsatisfiable. Blanchette and others [5] present a certified SAT solving framework verified in the Isabelle/HOL proof assistant. The proof effort is part of the *Isabelle Formalization of Logic* project. The framework is based on

refinement: at the highest level sit several calculi like CDCL and DPLL, which are formally proved. Depending on the strategy, the calculi are also shown to be terminating. Another SAT solver verified in Isabelle/HOL is by Marić [18]. In contrast to previous formalization, the verification methodology is not based on refinement. Instead, the Hoare triples associated to the solver pseudo-code are verified in Isabelle/HOL. In subsequent work [19], Marić and Jančić prove in Isabelle the functional correctness of a SAT solver represented as an abstract transition system. Andrici and Ciobâcă [1, 2] verify an implementation of DPLL in Dafny. Another formalization of a SAT solver (extended with linear arithmetic) is by Lescuyer [17], who verifies a DPLL-based decision procedure for propositional logic in Coq and exposes it as a reflexive tactic. Finally, a decision procedure based on DPLL is also verified by Shankar and Vaucher [23] in the PVS system. For the proof, they rely on subtyping and dependent types. Berger et al. have used the Minlog proof assistant to extract a certified SAT solver [4]. **None of the verified SAT solvers described above perform a CNF conversion**, with the exception of the reflexive procedure by Lescuyer [17]. Lescuyer notes that implementing Tseitin’s procedure in Coq proved to be *much more challenging* and therefore implements a lazy CNF transformation.

5 Discussion

Compiling (including verification time) the entire development (the two CNF transformations) takes about one minute on a standard laptop. The following table contains a summary of the entire development in numbers.

Lines of code	2440	Methods	32
Preconditions	318	Postconditions	176
Predicates	26	Functions	26
Assertions	227	Lemmas	51
Ghost variables	31	Verification time	~1 min

We find that Dafny can be used successfully to complete this case study. However, the degree of proof automation is small and most of the interesting verified proofs require significant assistance from the user in the form of helper assertions and lemmas. Additionally, we found that the verified proofs of the Tseitin algorithm push Dafny to the edge, in the sense that a particular organization of the proofs is required in order for the development to verify in reasonable time. To this purpose, we propose the following *verification patterns* that help achieve good verification performance and that are portable to other Dafny developments:

- (1) Do not use nested quantifiers. Instead, whenever a formula like $\forall_.\exists_.$ occurs, create a predicate $Q \equiv \exists_.$ and use \forall_Q instead of the initial formula. We use this verification pattern in the verification of the Tseitin transformation in the context of the `tseitinCanExtend` predicate. This transformation

could be automated and could serve as an improvement in deductive verifiers, but further investigation into its merits on more case studies should be performed first.

- (2) Do *not* inline even simple predicates. Make sure inlining is consistent. For example, we prefer to use the following trivial predicate:

```
predicate validVariable(v : int) { v ≥ 0 }
```

instead of inlining it. Additionally, consistency is required: mixing `v >= 0` and `validVariable(v)` will generally result in a potential performance degradation in verification (e.g., mixing `v >= 0` and `validVariable(v)` in one file results in a verification time increases by approx. 16% in our project).

- (3) Consistently add post-conditions, even if they are trivial. For example, if we remove the two post-conditions in the following function method:

```
function method negVarToLit(v : int) : int    requires validVariable(v);
    ensures negVarToLit(v) ≤ -1 ∧ validLiteral(negVarToLit(v));
{ (-v) - 1 }
```

our development still verifies, but takes approx. 50% more time to do so (approx. 1m30s instead of approx. 1m). On more complex functions, this can be the difference between verification succeeding and failing (for no apparent reason).

Incidentally, the above patterns do not only help Dafny verify the development faster, but also often clarify invariants for the programmer by forcing them to consistently give names to certain recurring formulae and enabling them to essentially create a mini-DSL for proofs.

Our case study also suggests a few areas where Dafny and other similar verifiers could be improved. For example, numerical functions such as `pow` (exponentiation) could be built in, possibly with some associated helper lemmas. Termination measures could be improved, both in allowed syntax (e.g., allowing `decrease userdefinedfunction(...)`), but also in allowing other well-founded orders such as multiset orderings (although we have finally not needed such orders in our development). Another area that could be improved is predictable verification performance. We find that, especially when not following the patterns described above, performance is very difficult to predict.

Implementation Choices. For Tseitin’s algorithm, we use a different representation for the input formula (an ADT) and the output formula (a set of clauses). Not only is this representation of the output the most natural for implementing the algorithm, but it is also what a SAT solver takes as input. Therefore, it allows in principle to easily combine our CNF transformation with a SAT solver. It would be easy to convert the set of clauses into the ADT representation and prove equivalence of the two, or to directly work with ADTs as part of the transformation. For the textbook transformation, the output and the input have the same representation (ADTs). However, we do not prove explicitly that the result is in CNF (it follows implicitly from the fact that none of the nine equivalences can be applied anymore). Proving this explicitly would require to first specify what it means for a formula to be in CNF; it would be an interesting exercise to prove this and to extract the set of clauses from the CNF.

Future Work. Our case study opens several directions for future work. *Possible improvements.* The formulae could be represented as DAGs instead of trees. This could speed up both algorithms; more interestingly, this might simplify the termination proofs for the textbook algorithm as discussed in Difficulty D1 on Page 5. Several optimizations to Tseitin’s algorithm [8,12] could also be implemented and verified. For a high-performance conversion, literals and variables should be represented as machine integers, which would require proving bounds throughout the code. *Theoretical extensions.* It would be interesting to find improved (possibly tight) bounds on the termination measure for the first algorithm. Additionally, it would be useful to bridge the distance between *truth assignment* as defined in the Dafny development and the usual notion of *truth assignment* in logic: our assignments are finite (with sufficiently many elements to account for all propositional variables in the context where they are used), while *truth assignments* are usually countably infinite. *Verification improvements.* It would be useful to further simplify the variant used to prove termination for the first algorithm. As Dafny supports a limited form of refinement types [10] (only numerical types can be refined by a logical constraint), it might be useful for the verification time to use such types for variables and literals. *Finally*, our CNF transformations can be linked with a verified CNF-SAT solver to obtain an end-to-end verified solver for the general SAT problem.

A Short Overview of Dafny

The Dafny programming language supports auto-active verification, a technique known since the 70s, but which has only become practical in the past decade, thanks to advances in both the usability of verification tools and computer processing power. Here is an example of a simple typical auto-actively verified Dafny method implementing binary search:

```

method binarySearch(a: array<int>, key : int) returns (r : int)
  requires  $\forall j, k \bullet 0 \leq j < k < a.Length \implies a[j] \leq a[k]$ ;
  ensures  $r \geq 0 \implies 0 \leq r < a.Length \wedge a[r] = key$ ;
  ensures  $r < 0 \implies \forall k \bullet 0 \leq k < a.Length - 1 \implies a[k] \neq key$ ;
{
  var left : int := 0;
  var right : int := a.Length - 1;
  while (left  $\leq$  right)
    invariant  $0 \leq left \leq a.Length$ ;
    invariant  $-1 \leq right < a.Length$ ;
    invariant  $\forall k \bullet 0 \leq k < left \implies a[k] < key$ ;
    invariant  $\forall k \bullet right < k < a.Length \implies a[k] > key$ ;
    decreases right - left;
  {
    var mid : int := (left + right) / 2;
    if (key < a[mid]) {
      right := mid - 1;
    } else if (key > a[mid]) {
      left := mid + 1;
    } else {
      return mid;
    }
  }
  return -1;
}

```


Auto-active verification is a mix of *automatic* verification and *interactive* verification. It means that the program is annotated by the programmer with specifications (such as preconditions, introduced by **requires**, and postconditions, introduced by **ensures**), which are automatically checked by the system to hold. The system verifies that the program implements the specification. If the postcondition cannot be proven to hold whenever the precondition holds, then the Dafny compiler fails with an error message. For example, the code above compiles (and verifies) without any issues; however, if we had made any mistake, like initializing **right** by **a.Length** instead of **a.Length - 1** or reversing the comparison operators **<** and **>** inside the body of the while loop, then compilation would fail as the postcondition would not be provable.

For complicated post-conditions, Dafny cannot establish their validity automatically, and therefore additional help is required from the part of the programmer (the interactive part) in the form of invariants (in the example above, four invariants are given for the while loop), variants (introduced by the **decreases** keyword, used to establish termination), lemmas, or other helper annotations.

Auto-active verification is featured in frameworks such as Why3 [6], Dafny [16] or even programming languages such as Ada [14] or C [7]. It has been used successfully to develop trusted code for small and even average sized projects [13]. The main advantage of using auto-active verification in software development is that we obtain a high degree of confidence in the correctness of software projects that were developed in this style.

B The Dafny Development

The attached Dafny development consists of 8 source files:

- **utils.dfy** contains generally useful definitions and lemmas, such as the definition of exponentiation (**pow**);
- **formula.dfy** contains the definition of the **FormulaT** data type and related functions and lemmas such as **validFormulaT**, **truthValue**, **maxVar**;
- **cnf.dfy** contains the verified implementation of the textbook algorithm for finding the CNF (with functions/methods such as **convertToCNF** or **applyRule**);
- **cnfformula.dfy** contains various items concerning the representation of CNF formulae as elements of type **seq<seq<int> >**, such as the predicates **validLiteral**, **validCnfFormula**, **truthValueCnfFormula**;
- **tseitin.dfy** contains the entry point (**tseitin**) to Tseitin's transformation, together with the main implementatino **tseitinCnf**;
- it relies on **tseitinproofs.dfy**, which contains lemmas that prove the invariant of **tseitinCnf** for all cases;
- both of the modules above rely on **tseitincore.dfy**, which contains definitions useful in both the algorithm and its proof, such as the set of clauses **orClauses** to be added for disjunctions;
- **main.dfy** exercises the CNF transformation in a **Main** method and can be used to obtain an executable;

- `Makefile` to be used in the usual Unix-like manner.
To compile (and verify) the development, it is sufficient to run:
- `dafny /verifySeparately *.dfy`.

We have verified the source code with Dafny version 3.0.0.20820, but some earlier versions should work as well.

The Dafny development is available at

<https://github.com/iordacheviorel/cnf-dafny>.

References

1. Andrici, C.-C., Ciobăcă, Ș.: Verifying the DPLL algorithm in Dafny. In: Marin, M., Craciun, A. (eds.) *Proceedings Third Symposium on Working Formal Methods, FROM 2019*, Timișoara, Romania, 3–5 September 2019. EPTCS, vol. 303, pp. 3–15 (2019)
2. Andrici, C.-C., Ciobăcă, Ș.: Who verifies the verifiers? A computer-checked implementation of the DPLL algorithm in Dafny. *CoRR*, [arXiv:2007.10842](https://arxiv.org/abs/2007.10842) (2020)
3. Barroso, P., Pereira, M., Ravara, A.: Animated logic: correct functional conversion to conjunctive normal form. In: *PAAR 2020/SC-Square 2020*. CEUR Workshop Proceedings, vol. 2752, pp. 1–20. CEUR-WS.org (2020)
4. Berger, U., Lawrence, A., Forsberg, F.N., Seisenberger, M.: Extracting verified decision procedures: DPLL and resolution. *Log. Methods Comput. Sci.* **11**(1) (2015)
5. Blanchette, J.C., Fleury, M., Lammich, P., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. *J. Autom. Reason.* **61**(1–4), 333–365 (2018)
6. Bobot, F., Filliâtre, J.-C., Marché, C., Paskevich, A.: Why3: shepherd your herd of provers. In: *Boogie 2011: First International Workshop on Intermediate Verification Languages*, Wrocław, Poland, pp. 53–64, August 2011. <https://hal.inria.fr/hal-00790310>
7. Cohen, E., et al.: VCC: a practical system for verifying concurrent C. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) *TPHOLs 2009*. LNCS, vol. 5674, pp. 23–42. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03359-9_2
8. de la Tour, T.B.: An optimality result for clause form translation. *J. Symb. Comput.* **14**(4), 283–302 (1992)
9. Fleury, M.: Optimizing a verified SAT solver. In: Badger, J.M., Rozier, K.Y. (eds.) *NFM 2019*. LNCS, vol. 11460, pp. 148–165. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20652-9_10
10. Ford, R.L., Leino, K.R.M.: *Dafny Reference Manual* (2017)
11. Gäher, L., Kunze, F.: Mechanising complexity theory: the cook-Levin theorem in Coq. In: Cohen, L., Kaliszyk, C. (eds.) *12th International Conference on Interactive Theorem Proving (ITP 2021)*. Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, vol. 193, pp. 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
12. Harrison, J.: *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, Cambridge (2009)
13. Hawblitzel, C., et al.: Ironclad apps: end-to-end security via automated full-system verification. In: *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2014*, Broomfield, CO, USA, 6–8 October 2014, pp. 165–181 (2014)

14. Hoang, D., Moy, Y., Wallenburg, A., Chapman, R.: SPARK 2014 and gnatprove - a competition report from builders of an industrial-strength verifying compiler. *Int. J. Softw. Tools Technol. Transf.* **17**(6), 695–707 (2015)
15. Rustan, K., Leino, M.: Developing verified programs with Dafny. In: 35th International Conference on Software Engineering, ICSE 2013, San Francisco, CA, USA, 18–26 May 2013, pp. 1488–1490 (2013)
16. Rustan, K., Leino, M.: Accessible software verification with dafny. *IEEE Softw.* **34**(6), 94–97 (2017)
17. Lescuyer, S.: Formalizing and Implementing a Reflexive Tactic for Automated Deduction in Coq. Theses, Université Paris Sud - Paris XI, January 2011
18. Marić, F.: Formalization and implementation of modern SAT solvers. *J. Autom. Reason.* **43**(1), 81–119 (2009)
19. Marić, F., Janičić, P.: Formalization of abstract state transition systems for SAT. *Log. Methods Comput. Sci.* **7**(3) (2011)
20. Michaelis, J., Nipkow, T.: Formalized proof systems for propositional logic. In: TYPES 2017. LIPIcs, vol. 104, pp. 5:1–5:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
21. Michaelis, J., Nipkow, T.: Propositional proof systems. *Archive of Formal Proofs*, June 2017. Formal proof development. https://isa-afp.org/entries/Propositional_Proof_Systems.html
22. Oe, D., Stump, A., Oliver, C., Clancy, K.: versat: a verified modern SAT solver. In: Kuncak, V., Rybalchenko, A. (eds.) VMCAI 2012. LNCS, vol. 7148, pp. 363–378. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27940-9_24
23. Shankar, N., Vaucher, M.: The mechanical verification of a DPLL-based satisfiability solver. *Electr. Notes Theor. Comput. Sci.* **269**, 3–17 (2011)
24. Tseitin, G.S.: On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic, Part II*, pp. 115–125 (1968)



Analysis in a Formal Predicative Set Theory

Nissan Levi^(✉)  and Arnon Avron^(✉) 

School of Computer Science, Tel Aviv University, Tel Aviv, Israel
aa@cs.tau.ac.il

Abstract. We present correct and natural development of fundamental analysis in a predicative set theory we call PZF^U . This is done by using a delicate and careful choice of those Dedekind cuts that are adopted as real numbers. PZF^U is based on ancestral logic rather than on first-order logic. Its key feature is that it is definitional in the sense that every object which is shown in it to exist is defined by some closed term of the theory. This allows for a very concrete, computationally-oriented model of it, and makes it very suitable for MKM (Mathematical Knowledge Management) and ITP (Interactive Theorem Proving). The development of analysis in PZF^U does not involve coding, and the definitions it provides for the basic notions (like continuity) are the natural ones, almost the same as one can find in any standard analysis book.

Keywords: Foundation of mathematics · Predicativity · Computable set theories

1 Introduction

Axiomatic set theory is almost universally accepted as the basic theory which provides the foundations of mathematics, and in which the whole of present day mathematics can (and many say: should) be developed. As such it should be considered to be the most natural framework for MKM (Mathematical Knowledge Management) in general, and ITP (Interactive Theorem Proving) in particular (especially for goals like those of the AUTOMATH project ([7, 15, 22]) and the QED manifesto ([21]). Moreover: as is emphasized and demonstrated in [4], set theory has not only a great pragmatic advantage as a basic language for mathematical discourse, but it also has a great computational potential as a basis for specification languages, declarative programming, and proof verifiers. However, in order to be used for any of these tasks it is necessary to overcome the following serious gaps that exist between the “official” formulations of set theory (as given e.g. by Zermelo Fränkel Set Theory ZF; see e.g. [11]), and actual mathematical practice:

1. The official formalizations of axiomatic set theories in almost all textbooks are based on some *standard* first-order languages. In such languages terms are

variables, constants, and sometimes function applications (like $x \cap y$). What is *not* available in the *official* languages of those formalizations is the use of set terms of the form $(\{x \mid \varphi\})$. As a result, already the formulation of the axioms is quite cumbersome, and even the formalization of elementary proofs becomes something practically incomprehensible.

2. ZF treats all the mathematical objects on a par, and so hid the computational significance of many of them. Thus although certain functions are first-class citizens in many programming languages, in set theory they are just “infinite sets”, and ZF in its usual presentation is an extremely poor framework for computing with such sets (or handling them constructively).
3. Full ZF is far too strong for core mathematics, which practically deals only with a small fraction of the set-theoretical “universe”. It is obvious that much weaker (and easier to mechanize) systems should do.

The first of these three problems can be overcome by using the framework for formalizing mathematics that was developed in [1]. This framework is based on set theory and is close in many ways to ZF on one hand, but is definitional in spirit on the other. In particular: it makes an extensive use of abstract set terms of the form $\{x \mid \varphi\}$. One of its crucial features is that all abstract set terms that it allows to use are *statically* defined in a precise formal way (using the mechanism of safety relations). Therefore it preserves the very useful complete separation we have in first-order logic between the (easy) check whether a given expression is a well-formed term or formula, and the (difficult) check whether it is a theorem. This feature makes the framework particularly appropriate for mechanical manipulations and for interactive theorem proving.¹

The other two problems mentioned above have been tackled in [2,3,5] by employing *predicative* set theories. By this, these papers followed Poincaré ([16,17]), Weyl ([20]), and Feferman, who in [9,10] forcefully argued that predicative mathematics suffices for developing all of scientifically applicable mathematics, i.e. the mathematics that is actually indispensable to present-day natural science. Poincaré-Weyl-Feferman’s predicativist program is essentially based on the principle that higher-order constructs, such as sets or functions, are acceptable only when introduced through non-circular definitions. The main goal of [3,5] was to show that using the framework of [1], this definitional approach to mathematics can be implemented in a user friendly way, which is based on set theory, and has no essential conflicts with mathematical practice.

The main problem of predicative mathematics is how to introduce and handle the real numbers, and what is usually taken as their characteristic property: their completeness with respect to their standard ordering. This principle has in fact been abandoned (and replaced by a weaker principle) by Weyl and Feferman. In contrast, it is preserved in [5]. However, its applicability there is severely limited by the fact that most of the important collections of real numbers, (including \mathbb{R} itself and all intervals) are not available as sets according to the theory RST^{HF}

¹ This has already been demonstrated in an initial implementation made in Tel Aviv university.

used there, but only as proper classes. Moreover, because of this fact the development of analysis there is not natural, and involves a lot of coding (much like the development of analysis carried out in reverse mathematics [19]). What seemed to be a successful attempt to avoid this state of affairs was made in [3], using a (still predicative) extension of RST^{HF} . Unfortunately, there was a subtle, but rather difficult to repair, mistake in the proof given there of the completeness of what is taken there as \mathbb{R} . In fact, the theorem is wrong! (See Remarks 5 and 7 below.) Since all proofs that come later in [3] depend on the completeness of its \mathbb{R} , that mistake invalidates them as well.

In this paper we achieve the goals of [3] by presenting correct and natural development of fundamental analysis in a predicative set theory which is based on the framework given in [1]. We use for this an extension called PZF^{U} of the system PZF developed in [2] (which is based on ancestral logic rather than on first-order logic), and a more delicate and careful choice of those Dedekind cuts that we adopt as real numbers. Like the systems used in [3] and [5], the key feature of PZF^{U} is that it is definitional in the sense that every object which is shown in it to exist is defined by some closed term of the theory. This allows for a very concrete, computationally-oriented model of it. The development of analysis in PZF^{U} does not involve coding (like in [19] and [5]), and the definitions it provides for the basic notions (like continuity) are the natural ones, almost the same as one can find in any standard analysis book (e.g. [12]).

2 Preliminaries

2.1 Notations

We denote formulas by $\varphi, \psi, \chi, \theta, \alpha, \beta$, terms by r, s, t , and variables by x, y, z, w, a, b , indexed or not. Given an expression (i.e. a formula or a term) e , we denote by $\text{Fv}(e)$ ($\text{Bv}(e)$) its set of free variables (bound variables). Let x_1, \dots, x_k be k different variables, and t_1, \dots, t_k be k terms. We write $e\{t_1/x_1, \dots, t_k/x_k\}$ for the expression that is obtained from e by simultaneously substituting the terms t_i s for the free occurrences of the variables x_i s (we do not assume that $\text{Fv}(e) = \{x_1, \dots, x_k\}$, or even that $\text{Fv}(e) \subseteq \{x_1, \dots, x_k\}$). We always assume that during a substitution no free variable of the substituted terms is getting captured (renaming the bounded variables in e to fresh new ones if needed). We write \bar{x} as a shorthand for x_1, \dots, x_n .

2.2 The Theory PZF^{U}

In this section we define the theories PZF and PZF^{U} . The theory PZF was presented in [2].² (See there for more information.) We start by introducing $\mathcal{L}_{\text{PZF}^{\text{U}}}$, the language of PZF^{U} . $\mathcal{L}_{\text{PZF}^{\text{U}}}$ contains three major notions: the notion of term,

² In [3] the theory $\text{RST}^{\text{HF,U}}$ was presented. The theory PZF^{U} is a stronger version of it.

the notion of formula, and the notion of a safety relation \succ_{PZF^U} , which is a relation between formulas and sets of variables. The definitions of those notions are mutual recursive.

Definition 1. The terms, formulas, and the safety relation \succ_{PZF^U} of $\mathcal{L}_{\text{PZF}^U}$ are:

- Terms:
 1. Every variable is a term.
 2. The constant \mathbf{U} is a term.
 3. If x is a variable and φ is a formula s.t. $\varphi \succ_{\text{PZF}^U} \{x\}$, then $\{x \mid \varphi\}$ is a term.
- Formulas:
 4. If t and s are terms than $t \in s$, $t = s$ are (atomic) formulas.
 5. If φ , and ψ are formulas, and x is a variable, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, and $\exists x\varphi$ are formulas.
 6. If φ is a formula; s, t are terms; and x, y are distinct variables, then $(\text{TC}_{x,y}\varphi)(t, s)$ is a formula.
- The safety relation \succ_{PZF^U} :
 7. $\varphi \succ_{\text{PZF}^U} \emptyset$ if φ is atomic.
 8. $\varphi \succ_{\text{PZF}^U} \{x\}$ if $\varphi \in \{x \neq x, x = t, t = x, x \in t\}$, provided that $x \notin \text{Fv}(t)$.
 9. $\neg\varphi \succ_{\text{PZF}^U} \emptyset$ if $\varphi \succ_{\text{RST}} \emptyset$.
 10. $\varphi \vee \psi \succ_{\text{PZF}^U} X$ if $\varphi \succ_{\text{PZF}^U} X$ and $\psi \succ_{\text{PZF}^U} X$.
 11. $\varphi \wedge \psi \succ_{\text{PZF}^U} X \cup Y$ if $\varphi \succ_{\text{PZF}^U} X$; $\psi \succ_{\text{PZF}^U} Y$; and $Y \cap \text{Fv}(\varphi) = \emptyset$.
 12. $\exists y\varphi \succ_{\text{PZF}^U} X - \{y\}$ if $y \in X$ and $\varphi \succ_{\text{PZF}^U} X$.
 13. $(\text{TC}_{x,y}\varphi)(x, y) \succ_{\text{PZF}^U} X$ if $\varphi \succ_{\text{PZF}^U} X \cup \{x\}$ or $\varphi \succ_{\text{PZF}^U} X \cup \{y\}$.

Definition 2. The definition of \mathcal{L}_{PZF} (the language PZF) is (almost) identical to Definition 1, but with the constant \mathbf{U} omitted. (the safety relation of \mathcal{L}_{PZF} is denoted by \succ_{PZF} .)

Definition 3. A formula φ of $\mathcal{L}_{\text{PZF}^U}$ (\mathcal{L}_{PZF}) is *absolute* if $\varphi \succ_{\text{PZF}^U} \emptyset$ ($\varphi \succ_{\text{PZF}} \emptyset$).

Definition 4. The logic which underlies PZF^U and PZF is TC-logic (transitive closure logic), also called AL (ancestral logic). See [2, 6, 13, 14, 18].

Definition 5. PZF is the set-theory in \mathcal{L}_{PZF} that has the following axioms:

- Extensionality: $\forall z(z \in x \leftrightarrow z \in y) \leftrightarrow x = y$
- Comprehension: $\forall x.x \in \{x \mid \psi\} \leftrightarrow \psi$, provided $\psi \succ_{\text{PZF}} \{x\}$.
- \in -induction scheme: $\left(\forall x.(\forall y \in x.\varphi[y/x]) \rightarrow \varphi\right) \rightarrow \forall x\varphi$ (provided y is free for x in φ).³

Definition 6. PZF^U is the set-theory in $\mathcal{L}_{\text{PZF}^U}$ that has all the axioms of PZF (with \succ_{PZF} replaced by \succ_{PZF^U}) and the following schema for \mathbf{U} : \mathbf{U} -closure scheme: $\forall y_1 \dots y_n \in \mathbf{U}.t \in \mathbf{U}$, provided t is a term, $\text{Fv}(t) = \{y_1, \dots, y_n\}$, and \mathbf{U} does not occur in t . (i.e., t is a term of PZF.)

³ Although the \in -induction scheme is a part of PZF/PZF^U , we shall never use this scheme in this work.

Some notes concerning Definition 6:

- By straightforward structural induction it can be proved that if e is an (absolute) formula than $e\{t_1/x_1, \dots, t_k/x_k\}$ remains an (absolute) formula, and that if e is a term than $e\{t_1/x_1, \dots, t_k/x_k\}$ remains a term. It holds for PZF and for PZF^U .
- The constant U stands for “Universe”. In [2] it was shown that every rudimentary operation⁴ can be defined by a set-term of PZF (even without the use of the TC operator).⁵ Hence by the U -closure scheme (see Definition 6), the universe U is closed under the rudimentary operations. For example if $x, y \in U$ then $x \times y, x \cup y, x \cap y, \cup x$ are also elements of U .
- In the sequel, we shall always indicate which definitions and which claims take place in PZF and which in PZF^U . Usually we shall do it by writing in parenthesis whether it is PZF (for example “**Definition 1.1** (PZF)”) or PZF^U (for example “**Lemma 1.1** (PZF^U)”). We have a special concern with definitions of set-terms that belong to PZF because we can apply to them the U -closure scheme (see Definition 6).

2.3 Extending the Base Language

Although the official language of PZF^U (and of PZF) is the one defined above, it is a standard mathematical practice to introduce new symbols and notations as the work progresses. So practically we shall enrich our language, and this will be done by adding to the language two kinds of defined symbols: defined predicates, and defined operations.

Defined Predicates. This is done much like as adding new predicate symbols to a standard FOL language \mathcal{L} . In general, given a formula φ of \mathcal{L} s.t. $\text{Fv}(\varphi) = \{x_1, \dots, x_n\}$ we may extend \mathcal{L} with a new n -ary predicate symbol P^φ that abbreviates φ (this of course must come with appropriate axioms that “define” P^φ). In our work we shall do the same, but with the constraint that we shall add new predicate symbols only for *absolute* formulas. The reason for this is that we want to preserve property (7) of Definition 1 (atomic formulas need to be absolute).

Defined Operations. In a usual FOL one may add new operations symbols.⁶ Working in a theory T (in a language \mathcal{L}), if φ is a formula of \mathcal{L} s.t. $\text{Fv}(\varphi) = \{y, x_1, \dots, x_n\}$, and if $T \vdash \forall x_1, \dots, x_n \exists! y. \varphi$, we may extend \mathcal{L} with a new n -ary operation symbol f^φ , with the following intuitive meaning: given x_1, \dots, x_n , $f^\varphi(x_1, \dots, x_n)$ is that unique y . (Of course we need to add to T appropriate

⁴ In the literature these are referred as “rudimentary functions”. We prefer to call them “rudimentary operations”, because we reserve the word “function” only for objects that exist as sets.

⁵ For more information about the rudimentary operations see [8].

⁶ These are usually referred as functions symbols.

axioms that “defines” f^φ). Hence, in the process of adding a new operation symbol, there is usually a need to prove (inside the theory) the existence and uniqueness of the object “returned” by the operation. An exception is the case where the new operation symbol is introduced as an abbreviation of some term in the language. (More formally the above φ is of the form $y = t$ where t is a term. The existence and uniqueness of that y is obvious.) While working in PZF (PZF^U) this is the only way we shall use when we add new operation symbols. For example consider the term $t = \{a \mid a \in x_1 \vee a \in x_2\}$ (the union of x_1 and x_2). We add a new 2-ary operation symbol “ \cup ” corresponding to t (or more formally corresponding to the formula $y = t$). Given two terms r and s , we may view the term $r \cup s$ simply as an abbreviation of $\{a \mid a \in r \vee a \in s\}$.

Remark 1.

1. We stress that at any point in the future, our extended language will always satisfy all the properties specified in Definition 1. Specifically, an atomic formula will always be absolute, and given a term t of \mathcal{L}_{PZF} , $x \in t \succ_{\text{PZF}} \{x\}$, and $x = t \succ_{\text{PZF}} \{x\}$ (same for PZF^U).
2. If one wants to translate a formula that involves new predicate-symbols/operation-symbols to a formula of the original language, all he needs to do to is to unravel the definitions of the defined notions, which are merely abbreviations of formulas/terms in the original language.
3. One delicate point concerns the schemes in Definition 6 (the Comprehension scheme, the \in -induction scheme, and the U-closure scheme). The formulas and terms that appear there are expressions in the original language \mathcal{L}_{PZF} ($\mathcal{L}_{\text{PZF}^U}$), while we will use those schemes also with expressions that involve new defined symbols. Practically no real problem arises because the new symbols merely represent new ways of abbreviating formulas and terms. We shall be more careful with the U-closure scheme, when applying it to a term t . We shall always check that t is a term of the (extended) language of PZF.
4. Let t_1, \dots, t_n be n terms, let P be an n -ary predicate symbols, and f be an n -ary operation symbols. The standard convention of applying P to t_1, \dots, t_n , is $P(t_1, \dots, t_n)$ (and $f(t_1, \dots, t_n)$ when applying f to them). To enhance clarity, we leave the symbols “(,)” only for the case when we apply a function to an element. Instead of $P(t_1, \dots, t_n)$ we shall always write $P[t_1, \dots, t_n]$, and instead of $f(t_1, \dots, t_n)$ we shall always write $f\langle t_1, \dots, t_n \rangle$.

2.4 Basic Notations

In what follows we shall use the following.

1. In [2] it was shown how basic predicates and operations can be defined. In the sequel we shall use the following: $x \subseteq y$ (the usual meaning), $\text{func}[f]$ (f is a function), $\text{func}[f, x, y]$ (f is a function from x to y), $\text{bijection}[f, x, y]$, $\text{surjection}[f, x, y]$ (f is a bijection/surjection from x to y), $\text{seq}[s]$ (s is a function with domain \mathbb{N}), $\text{seqFin}[s]$ (s is a function from a proper initial segment of \mathbb{N}). $\emptyset, x \cup y, x \cap y, x \setminus y, x \times y, \cup x$ (the usual meaning), $\{x\}, \{x_1, \dots, x_n\}$ (the usual meaning), $\langle x, y \rangle$ (ordered pair), $\text{dom}\langle r \rangle, \text{rng}\langle r \rangle$ (the domain/range of

a relation r), $f \upharpoonright x$ (the restriction of a function f to x). We note that all the above take place in PZF.

The next four notations are merely abbreviations for set-terms. (For more information see [3]).

2. $\iota x.\varphi$ (provided $\varphi \succ_{\text{PZF}^U} \{x\}$) stands for the unique element x s.t. φ .
3. $\lambda x \in s.t$ (provided $x \notin \text{Fv}(s)$). Moreover, we note that if s and t do not involve the constant U , and if $\text{Fv}(s) \cup \text{Fv}(t) \subseteq \{x_1, \dots, x_n\}$ then $\text{PZF}^U \vdash \forall x_1, \dots, x_n \in U. (\lambda x \in s.t) \in U$.
4. If f is a function, $f(x)$ stands for $\iota y. \langle x, y \rangle \in f$, where y is a fresh new variable.
5. Restricted replacement: if t is a term s.t. $\{\bar{x}\} \subseteq \text{Fv}(t)$, $\psi \succ_{\text{PZF}^U} \{\bar{x}\}$, we write $\{t \mid \psi\}$ instead of $\{a \mid \exists \bar{x}. a = t \wedge \psi\}$.
6. Definition by cases: assume that t_1, t_2 are terms, and that φ_1, φ_2 are absolute formulas s.t. $\text{PZF}(\text{PZF}^U) \vdash \neg(\varphi_1 \wedge \varphi_2)$. Let a be a fresh new variable, and consider the following term: $t := \{a \mid a \in t_1 \wedge \varphi_1\} \cup \{a \mid a \in t_2 \wedge \varphi_2\}$. It is provable in PZF (PZF^U) that:

$$(\varphi_1 \rightarrow t = t_1) \wedge (\varphi_2 \rightarrow t = t_2) \wedge ((\neg\varphi_1 \wedge \neg\varphi_2) \rightarrow t = \emptyset).$$

In the sequel we introduce t simply by writing:

$$t := \begin{cases} t_1 & \text{if } \varphi_1 \\ t_2 & \text{if } \varphi_2 \\ \emptyset & \text{else} \end{cases}$$

Remark 2. For simplicity, we stated the above notation for a two-cases term, but we shall use this type of notations also for more than two terms. Also, if it holds that $\text{PZF}(\text{PZF}^U) \vdash \varphi_1 \vee \dots \vee \varphi_k$ we shall omit the “else”.

2.5 Mathematics in PZF

We summarize facts from [2,3,5] about the development of mathematics in PZF^U .

The Sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$. For every element of J_{ω^ω} , the ω^ω level of Jensen’s hierarchy, there exists a closed set-term that represents it. Specifically, there are closed set-terms that represent:

- The set of natural numbers, the set of integers, and the set of rational numbers. Those set-terms are denoted as usual by \mathbb{N} , \mathbb{Z} , and \mathbb{Q} .
- The following relations and functions (on \mathbb{Q}): $<_{\mathbb{Q}}, +_{\mathbb{Q}}, -_{\mathbb{Q}}, \times_{\mathbb{Q}}, /_{\mathbb{Q}}, |\cdot|_{\mathbb{Q}}$

It can be proved in PZF that the above sets, functions, and relations have their usual properties.

Induction Scheme. The full induction scheme can be proved, namely for *every* formula φ of PZF (PZF^U):

$$\frac{}{\vdash_{\text{PZF}(\text{PZF}^U)} (\varphi\{0/n\} \wedge \forall n(\varphi \rightarrow \varphi\{n+1/n\})) \rightarrow \forall n \in \mathbb{N}. \varphi.}$$

⁷ Actually, [3,5] deal with the systems RST^{HF} and $\text{RST}^{\text{HF},U}$. Since PZF is stronger than these theories, all the development done there is also available in PZF.

Dedekind Cuts. We define Dedekind cuts in the usual way.

Definition 7. r is a *Dedekind cut*:

$$\text{dedCut}[r] := (\emptyset \neq r \subsetneq \mathbb{Q}) \wedge (r \text{ doesn't have a maximum}) \wedge (\forall x, y \in \mathbb{Q}. x \in r \wedge y < x \rightarrow y \in r)$$

After defining Dedekind cuts, we can define the usual operations $+$, $-$, \times , $/$, $|\cdot|$, and the predicate $<$. For example the term that defines the operation $+$ is: $r + s := \{q +_{\mathbb{Q}} q' \mid q \in r \wedge q' \in s\}$. Similarly we define $-$, \times , $/$, $|\cdot|$, $<$. It is a standard matter to prove in PZF that these operations and relations have their usual properties. In addition, for every Dedekind-cut r , and every $k \in \mathbb{N}$, the exponentiation term r^k is defined and it has its usual properties.

The following completeness theorem can be proved in PZF:

Theorem 1 (PZF)(Completeness). *Let X be a non empty set of Dedekind cuts. Assume also that there exists a Dedekind cut d s.t. $\forall x \in X. x \leq r$ (namely X is bounded from above). Then $\cup X$ is a Dedekind cut and it is the supremum of X . (Later we shall denote $\cup X$ by $\text{sup}\langle X \rangle$)*

2.6 Recursive Definitions

In the sequel we shall define several terms by recursion. In this section we prove that such definitions are indeed legitimate. The following theorem is actually a schema in PZF (PZF^U):

Theorem 2. *Let t_i, t_s be terms of PZF (PZF^U) s.t. $\text{Fv}(t_i) = \{\bar{x}\}$ and $\text{Fv}(t_s) = \{n, p, \bar{x}\}$.⁸ Then there exists a term t_{rec} of PZF (PZF^U) with $\text{Fv}(t_{\text{rec}}) = \{\bar{x}\}$ s.t. the following is provable in PZF (PZF^U):*

$$\forall \bar{x}. \text{func}[t_{\text{rec}}] \wedge \text{dom}\langle t_{\text{rec}} \rangle = \mathbb{N} \wedge t_{\text{rec}}(0) = t_i \wedge \tag{1}$$

$$\forall n \in \mathbb{N}^+ \forall p. p = t_{\text{rec}}(n-1) \rightarrow t_{\text{rec}}(n) = t_s \tag{2}$$

Informally, t_i is the initial value, and t_s is the “step” function that “gets” the previous value in the variable p , and outputs current value.

Remark 3. When using this theorem, we shall introduce t_{rec} by writing:

$$t_{\text{rec}}\langle \bar{x} \rangle := \lambda n \in \mathbb{N}. \begin{cases} t_i\langle \bar{x} \rangle & \text{if } n = 0 \\ t_s\langle n, t_{\text{rec}}\langle \bar{x} \rangle(n-1), \bar{x} \rangle & \text{if } n > 0 \end{cases}$$

Proof. See in the appendix.

2.7 Finite Sums

Using Theorem 2 we can define the 2-ary operation of finite sum $\sum_{i=0}^n a(i)$. It is provable in PZF that it has all its usual properties.

⁸ We remind the reader that \bar{x} stands for x_1, \dots, x_n .

3 Sequences of Dedekind Cuts

In this section we develop the notion of a limit of a sequence (of Dedekind cuts).

Definition 8 (PZF). *Define the following:*

- $\text{seq}^{\text{dc}}[s] := \text{seq}[s] \wedge \forall a \in \text{rng}\langle s \rangle. \text{dedCut}[a]$
- $\text{bounded}[a] := \text{seq}^{\text{dc}}[a] \wedge \exists q \in \mathbb{Q} \forall n \in \mathbb{N}. |a(n)| \leq q$.
- *The following term maps a bounded sequence a , to its limsup . Define first the term $t\langle a \rangle := \{q \in \mathbb{Q} \mid \forall n \in \mathbb{N} \exists k \geq n. q \in a(k)\}$. Note that $t\langle a \rangle$ is not necessarily a Dedekind cut, because it might contain a maximal element. In that case we simply remove it:*

$$\text{limsup}\langle a \rangle = \begin{cases} t\langle a \rangle & \text{if } \neg \exists q \in t\langle a \rangle \forall q' \in t\langle a \rangle. q' \leq q \\ t\langle a \rangle \setminus \{iq. q \in t\langle a \rangle \wedge \forall q' \in t\langle a \rangle. q' \leq q\} & \text{else} \end{cases}$$

- $\text{liminf}\langle a \rangle$ is defined in a similar way.
- $\text{converge}[a] := \text{bounded}[a] \wedge \text{liminf}\langle a \rangle = \text{limsup}\langle a \rangle$.
- *The term $\text{limit}\langle a \rangle$ maps a convergent sequence to its limit:*

$$\text{limit}\langle a \rangle := \text{liminf}\langle a \rangle$$

- *The sequence a converges to (the Dedekind cut) r :*

$$\begin{aligned} \text{converge}[a, r] &:= \text{seq}^{\text{dc}}[a] \wedge \text{dedCut}[r] \wedge \\ &\forall \varepsilon \in \mathbb{Q}^+ \exists N \in \mathbb{N} \forall n \geq N. |a(n) - r| < \varepsilon \end{aligned}$$

It is straightforward to prove that $\text{converge}[a, r] \wedge \text{converge}[a, r'] \rightarrow r = r'$.

- *The sequence a is a Cauchy sequence if:*

$$\text{cauchySeq}[a] := \text{seq}^{\text{dc}}[a] \wedge \forall \varepsilon \in \mathbb{Q}^+ \exists N \in \mathbb{N} \forall n, m \geq N. |a(n) - a(m)| < \varepsilon$$

Proposition 1 (PZF).

$$\forall a. \text{seq}^{\text{dc}}[a] \wedge \text{bounded}[a] \rightarrow \text{dedCut}[\text{limsup}\langle a \rangle] \wedge \text{dedCut}[\text{liminf}\langle a \rangle]$$

Proof. The proof is straightforward and left for the reader.

Proposition 2 (PZF). *Assume that $\text{seq}^{\text{dc}}[a]$. The following holds:*

1. $\text{converge}[a] \rightarrow \exists r. \text{converge}[a, r] \wedge r = \text{limit}\langle a \rangle$
2. $\forall r. \text{converge}[a, r] \rightarrow (\text{converge}[a] \wedge \text{limit}\langle a \rangle = r)$
3. $\text{converge}[a] \leftrightarrow \text{cauchySeq}[a]$

Proof. The proof of the above propositions is straightforward. In the appendix we sketch for example the proof that $\text{cauchySeq}[a] \rightarrow \text{converge}[a]$.

Remark 4. It is tempting to define $\text{converge}[a]$ as $\exists r. \text{dedCut}[r] \wedge \text{converge}[a, r]$. But the last formula is not absolute, while our definition of $\text{converge}[a]$ is done by an absolute one.

4 The Real Line

Up to now all the terms, absolute formulas, and claim took place in PZF, namely we did not use the constant U in our definitions. Since we want to treat \mathbb{R} (the real numbers) as a set, we move to work in PZF^U and define \mathbb{R} as follows.

Definition 9 (PZF^U).

- The real line $\mathbb{R} := \{r \in U \mid \text{dedCut}[r]\}$
- r is a real number (or simply real) if $r \in \mathbb{R}$. Note that “ r is real” is not equivalent to $\text{dedCut}[r]$, since not all Dedekind cuts belong to U .

It is straightforward to prove the following theorem:

Theorem 3 (PZF^U)(**U-completeness**). *Let $X \subseteq \mathbb{R}$, $X \neq \emptyset$, and assume that X is bounded from above. If $X \in U$ then $\sup\langle X \rangle \in \mathbb{R}$.*

Remark 5. In [3] it was claimed that already in the theory $RST^{HF,U}$ (and so also in PZF^U , which is stronger) one can prove that every nonempty bounded subset of \mathbb{R} has a supremum in \mathbb{R} . As we noted in the introduction, this claim and its proof in [3] are wrong. The problem was that the need in Theorem 3 of the condition $X \in U$ had been missed.

In addition we have:

Theorem 4 (PZF^U). *\mathbb{R} is closed under $+$, $-$, \times , $/$ and $|\cdot|$. (In the case of $/$ we exclude division by 0, of course.)*

Notations: Henceforth we use the usual notations for closed/open intervals, like: $[a, b] := \{r \in \mathbb{R} \mid a \leq r \leq b\}$, and $(a, b) := \{r \in \mathbb{R} \mid a < r < b\}$.

Warning: Although \mathbb{R} is a subset of U , it is not necessarily an element of U . The same is true for $[a, b]$ and (a, b) in case $a < b$ (for every $a, b \in R$).

Proposition 3 (PZF^U)(**Convergence in U**). *Let $a \in U$ be a sequence. Then:*

$$\text{converge}[a] \rightarrow \exists r \in \mathbb{R}. \text{converge}[a, r] \wedge r = \text{limit}\langle a \rangle$$

Proof. See in the appendix.

5 Continuous Functions

In this section we define the notion of continuous functions. The major obstacle we need to overcome is the fact that we do not have a “full” completeness of \mathbb{R} , rather a completeness with respect to subsets of \mathbb{R} which are elements of U (see Theorem 3). To overcome this problem we only consider a restricted class of functions. Functions that are quasi element of U , in the sense that their parts whose domains are available in the universe U , are also in U . The definition of these functions is the following:

Definition 10 (PZF^U)(Ufunc).

1. $\text{Ufunc}[f] := \text{func}[f] \wedge \forall x \in \mathbf{U}. x \subseteq \text{dom}\langle f \rangle \rightarrow f \upharpoonright x \in \mathbf{U}$.
2. $\mathbf{P}^{\mathbb{R}} := \{x \in \mathbf{U} \mid x \subseteq \mathbb{R}\}$
3. $\text{Ufunc}^{\mathbb{R}}[f] := \text{func}[f] \wedge \text{dom}\langle f \rangle \subseteq \mathbb{R} \wedge \text{rng}\langle f \rangle \subseteq \mathbb{R} \wedge \forall x \in \mathbf{P}^{\mathbb{R}}. f \upharpoonright x \in \mathbf{U}$ (which is obviously equivalent to $\text{Ufunc}[f] \wedge \text{dom}\langle f \rangle \subseteq \mathbb{R} \wedge \text{rng}\langle f \rangle \subseteq \mathbb{R}$).

Proposition 4 (PZF^U).

- $\forall f, g. \text{Ufunc}^{\mathbb{R}}[f] \wedge \text{Ufunc}^{\mathbb{R}}[g] \wedge \text{rng}\langle g \rangle \subseteq \text{dom}\langle f \rangle \rightarrow \text{Ufunc}^{\mathbb{R}}[f \circ g]$
- $\forall c \in \mathbb{R} \forall f, g. \text{Ufunc}^{\mathbb{R}}[f] \wedge \text{Ufunc}^{\mathbb{R}}[g] \rightarrow \text{Ufunc}^{\mathbb{R}}[c \cdot f] \wedge \text{Ufunc}^{\mathbb{R}}[f + g] \wedge \text{Ufunc}^{\mathbb{R}}[f \cdot g] \wedge \text{Ufunc}^{\mathbb{R}}[\frac{1}{f}]$

Proof. The proof is straightforward and left for the reader.

Definition 11 (PZF^U)(continuous function). Assume that $\text{Ufunc}^{\mathbb{R}}[f]$.

1. Let $x \in \text{dom}\langle f \rangle$.
We say that f is continuous at x (and denote it by $\text{continuous}[f, x]$) if f is defined in a neighborhood of x (namely $\exists \delta \in \mathbb{R}^+. (x - \delta, x + \delta) \subseteq \text{dom}\langle f \rangle$) and:

$$\forall \varepsilon \in \mathbb{R}^+ \exists \delta \in \mathbb{R}^+ \forall x'. |x' - x| < \delta \rightarrow |f(x') - f(x)| < \varepsilon \quad (3)$$

2. The notion of left continuity and right continuity is defined in a similar way.
3. We say that f is continuous on a closed segment $[a, b]$ if $[a, b] \subseteq \text{dom}\langle f \rangle$, f is right-continuous at a , f is left-continuous at b , and f is continuous at every $x \in (a, b)$.
4. If the domain of f is a closed or open segment, or \mathbb{R} , we say that f is continuous if it is continuous on its entire domain. (In the case of closed segment $[a, b]$, we only require right/left continuity on a/b (respectively). We denote it by $\text{continuous}[f]$).

Remark 6. It is tempting to drop the condition $\text{Ufunc}[f]$ in the above definition. But it turns out that this approach falls too short. See Remarks 5 and 7.

In the following we show that Definition 11 is not void, and actually every polynomial is a continuous function at every $x \in \mathbb{R}$. We start with the definition of the set of all polynomials:

Definition 12 (PZF^U).

- $\text{seqFin}^{\mathbb{R}}[a] := \text{seqFin}[a] \wedge \forall i \in \text{dom}\langle a \rangle. a(i) \in \mathbb{R}$
- $\text{poly}\langle a \rangle := \lambda x \in \mathbb{R}. \sum_{i=0}^{\text{dom}\langle a \rangle - 1} a(i) \cdot x^i$
- $\text{Polynomials} := \{\text{poly}\langle a \rangle \mid a \in \text{seqFin}^{\mathbb{R}}\}$

Proposition 5 (PZF^U).

- $\text{continuous}[\lambda x \in \mathbb{R}. x]$, and $\forall c \in \mathbb{R}. \text{continuous}[\lambda x \in \mathbb{R}. c]$.
- Let f, g be functions from a subset of \mathbb{R} to \mathbb{R} , and let $c \in \mathbb{R}$.

- If $x \in \text{dom}\langle f \rangle \cap \text{dom}\langle g \rangle$ and $\text{continuous}[f, x] \wedge \text{continuous}[g, x]$ then

$$\text{continuous}[f + g, x] \wedge \text{continuous}[f \cdot g, x] \wedge \text{continuous}[c \cdot f, x]$$
- If $x \in \text{dom}\langle f \rangle \wedge f(x) \neq 0$ then $\text{continuous}[\frac{1}{f}, x]$
- If $x \in \text{dom}\langle f \rangle \wedge f(x) \in \text{dom}\langle g \rangle$ then $\text{continuous}[f \circ g, x]$.

Proof. For every $x \in \mathbb{R}$, condition (3) of Definition 11 is proved in the usual way (for all the above cases). The fact that the above functions are indeed $\text{Ufunc}^{\mathbb{R}}$, follows from Proposition 4.

Corollary 1 (PZF^U). *For every $p \in \text{Polynomials}$, p is continuous at every $x \in \mathbb{R}$.*

Proof. This is a straightforward induction using Proposition 5.

Power Series and Elementary Functions. Using theorem 2 (recursive definitions) it is possible to define the set of the elementary functions, and prove that they continuous. We sketch it in the appendix.

5.1 Intermediate Value Theorem

To demonstrate our definition of continuity, we prove the intermediate value theorem:

Theorem 5 (PZF^U)(intermediate value theorem). *Let f be a continuous function from $[0, 1]$ to \mathbb{R} , s.t. $f(0) < 0$ and $f(1) > 0$. Then there exists $r \in [0, 1]$ s.t. $f(r) = 0$.*

Proof. Define the following set:

$$X = \{q \in \mathbb{Q} \mid 0 \leq q \leq 1 \wedge f(q) < 0\} \quad (= \{q \in \mathbb{Q} \mid 0 \leq q \leq 1 \wedge (f \upharpoonright \mathbb{Q})(q) < 0\}).$$

By the closure properties of \mathbb{U} , and the fact that $f \upharpoonright_{\mathbb{Q}} \in \mathbb{U}$, we conclude that $X \in \mathbb{U}$. Obviously X is non-empty and bounded from above, hence by the completeness of \mathbb{R} it has a supremum - denote it by s . Obviously $s \in [0, 1]$. We prove that $f(s) = 0$. Assume that it is not the case, say $f(s) > 0$. By the continuity of f at s , there exists $\delta \in \mathbb{R}^+$ s.t.

$$\forall x. |x - s| < \delta \rightarrow |f(x) - f(s)| < \frac{f(s)}{2} \tag{4}$$

Since $s = \sup(X)$, there exists $q' \in (s - \delta, s) \cap X$. By the definition of X , $f(q') < 0$, but by (4), $f(q') > \frac{f(s)}{2} > 0$ - a contradiction.

Remark 7. As we noted in Remark 6, it is tempting to define the notion of continuity for every function f from (subset of) \mathbb{R} to \mathbb{R} , and not demand that $\text{Ufunc}[f]$. Indeed, that was the definition that was adopted in [3]. We note that the assumption $\text{Ufunc}[f]$ is crucial in the above proof, otherwise we could not deduce that $s = \sup\langle X \rangle \in \mathbb{R}$. In [3], although the definition of continuity did not assume that $\text{Ufunc}[f]$, the intermediate value theorem was “proved” in a very similar way. This is due to the mistake in the proof of Theorem 3 (completeness of \mathbb{R}),⁹ where the assumption that the bounded subset of \mathbb{R} is an element of \mathbb{U} was omitted.

A Appendix

Proof (proof of Theorem 2). We prove the case of PZF (the case of $\text{PZF}^{\mathbb{U}}$ is identical). In the following proof we argue in PZF .

Define first the formula φ to be:

$$\begin{aligned} (f = \emptyset \wedge g = \{\langle 0, t_i \rangle\}) \vee \\ (f \neq \emptyset \wedge g = f \cup \{\langle \text{dom}\langle f \rangle, t_s \{ \text{dom}\langle f \rangle / n, f(\text{dom}\langle f \rangle - 1) / p \} \}) \end{aligned}$$

Informally, we set g to be the finite sequence achieved from the finite sequence f by expanding the domain of f by 1 according to the “step function” t_s . Obviously $\varphi \succ_{\text{PZF}^{\mathbb{U}}} \{g\}$. Let

$$X = \{g \mid g = \{\langle 0, t_i \rangle\} \vee \exists f. f = \emptyset \wedge (\text{TC}_{f,g}\varphi)(f, g)\}$$

and define $t_{\text{rec}} = \cup X$. Obviously t_{rec} is a term of PZF.

Let x_1, \dots, x_n be n sets. We observe first that for all $g \in X$ it holds that:

$$\text{seqFin}[g] \wedge g(0) = t_i \wedge \forall n \in \text{dom}\langle g \rangle. n > 0 \rightarrow g(n) = t_s \{g(n-1)/p\} \quad (5)$$

The proof is by a straightforward application of the induction rule for the TC operator.

Next we prove that $\text{func}[t_{\text{rec}}]$. Since every $g \in X$ is a function s.t. $\text{dom}\langle g \rangle \subseteq \mathbb{N}$, it is obvious that $\text{rel}[t_{\text{rec}}]$, and that $\text{dom}\langle t_{\text{rec}} \rangle \subseteq \mathbb{N}$. Therefore it suffices to show by induction on n that

$$\forall n \in \mathbb{N} \forall g, g' \in X. n \in \text{dom}\langle g \rangle \cap \text{dom}\langle g' \rangle \rightarrow g(n) = g'(n)$$

If $n = 0$ then by (5), $g(0) = g'(0) = t_i$. Assume that $n > 0$. Since $n \in \text{dom}\langle g \rangle \cap \text{dom}\langle g' \rangle$ and since $\text{seqFin}[g] \wedge \text{seqFin}[g']$, $n-1 \in \text{dom}\langle g \rangle \cap \text{dom}\langle g' \rangle$. By the induction hypothesis, $g(n-1) = g'(n-1)$, hence by (5), $g(n) = g'(n) = t_s$, and overall t_{rec} is a function.

We prove now by induction on n that $\forall n \in \mathbb{N}. n \in \text{dom}\langle t_{\text{rec}} \rangle$ (and hence $\text{dom}\langle t_{\text{rec}} \rangle = \mathbb{N}$). Let $g_0 = \{\langle 0, t_i \rangle\}$. Since $\varphi\{\emptyset/f, g_0/g\}$, it holds that $g_0 \in X$ and hence $0 \in \text{dom}\langle t_{\text{rec}} \rangle$. Assume that $n \in \text{dom}\langle t_{\text{rec}} \rangle$. Then there exists $g \in X$ s.t. $n \in$

⁹ See also Remark 5.

$\text{dom}\langle g \rangle$. If $n + 1 \in \text{dom}\langle g \rangle$ we are done. Otherwise, let $g' := g \cup \{\langle n + 1, t_s \langle n + 1, g(n), \bar{x} \rangle \rangle\}$. Obviously $\varphi\{g/f, g'/g\}$, hence $g' \in X$, and hence $n + 1 \in \text{dom}\langle g \rangle$ as desired.

Overall we proved (1) (t_{rec} is a function with domain \mathbb{N}). Proving (2) is done by (5) and a straightforward TC-induction.

Proof (part of the proof of Proposition 2). We sketch the proof for $\text{cauchySeq}[a] \rightarrow \text{converge}[a]$. So assume that $\text{cauchySeq}[a]$. It is straightforward to show that a is bounded. We prove that $\text{liminf}\langle a \rangle = \text{limsup}\langle a \rangle$. It is straightforward to show that $\text{liminf}\langle a \rangle \subseteq \text{limsup}\langle a \rangle$ (even without any assumption on a). Assume for contradiction that $\exists q \in \text{limsup}\langle a \rangle \setminus \text{liminf}\langle a \rangle$. Since $\text{dedCut}[\text{limsup}\langle a \rangle]$ and $\text{dedCut}[\text{liminf}\langle a \rangle]$ it also holds that

$$\exists q, q', q'' \in \mathbb{Q}. q < q' < q'' \wedge q, q', q'' \in \text{limsup}\langle a \rangle \setminus \text{liminf}\langle a \rangle.$$

Let $\varepsilon = q'' - q'$. Since $\text{cauchySeq}[a]$, there exists $N \in \mathbb{N}$ s.t.

$$\forall m, k \geq N. |a(m) - a(k)| < \varepsilon. \quad (6)$$

Since $q' \notin \text{liminf}\langle a \rangle$, there exists $k' \geq N$ s.t. $q' \notin a(k')$, and hence $a(k') \leq q < q'$. Since $q'' \in \text{limsup}\langle a \rangle$, there exists $k'' \geq N$ s.t. $q'' \in a(k'')$, and hence $q'' < a(k'')$. Putting it together, it holds that

$$a(k') < q' < q'' < a(k'')$$

Hence $|a(k'') - a(k')| > q'' - q' = \varepsilon$, that contradicts (6).

Proof (proof of Proposition 3).

By Proposition 2-(1) it follows that $\text{converge}[a, \text{limit}\langle a \rangle]$. Since the term $\text{limit}\langle \cdot \rangle$ is a term of \mathcal{L}_{PZF} , and since $a \in \mathbb{U}$, by the \mathbb{U} -closure scheme it holds that $\text{limit}\langle a \rangle \in \mathbb{U}$. Hence not just $\text{dedCut}[\text{limit}\langle a \rangle]$ (Proposition 1), but also $\text{limit}\langle a \rangle \in \mathbb{R}$, and the claim follows.

A.1 Uniformly Convergence and Power Series

In this section we want to talk about sequences of (real) functions. We use the standard procedure of Curryng. Define the following:

Definition 13 ($\text{PZF}^{\mathbb{U}}$)(sequence of functions).

- $\text{seqFunc}[F] := \text{func}[F] \wedge \text{dom}[F] = \mathbb{N} \times \mathbb{R} \wedge \text{rng}\langle F \rangle \subseteq \mathbb{R}$.
- $\text{uniformlyConvergence}[F, f] := \text{seqFunc}[F] \wedge \text{func}[f, \mathbb{R}, \mathbb{R}] \wedge \forall \varepsilon \in \mathbb{R}^+ \exists N \in \mathbb{N} \forall x \in \mathbb{R} \forall n \geq N. |F(n, x) - f(x)| < \varepsilon$.

Proposition 6 ($\text{PZF}^{\mathbb{U}}$). *Assume that $\text{uniformlyConvergence}[F, f]$. Then:*

1. $\forall x \in \mathbb{R}. \text{converge}[\lambda n \in \mathbb{N}. F(n, x), f(x)]$.
2. $\forall x \in \mathbb{R}. \text{limit}\langle \lambda n \in \mathbb{N}. F(n, x) \rangle = f(x)$.

Proof. 1. Let $x \in \mathbb{R}$. From $\text{uniformlyConvergence}[F, f]$ it follows that $\forall \varepsilon \in \mathbb{R}^+ \exists N \in \mathbb{N} \forall x \in \mathbb{R} \forall n \geq N. |F(n, x) - f(x)| < \varepsilon$. Hence $\forall \varepsilon \in \mathbb{R}^+ \exists N \in \mathbb{N} \forall n \geq N. |F(n, x) - f(x)| < \varepsilon$, and hence $\text{converge}[\lambda n \in \mathbb{N}. F(n, x), f(x)]$ as desired.

2. It follows immediately from previous item and Proposition 2-(2).

Proposition 7 (PZF^U). *Assume that: $\text{uniformlyConvergence}[F, f]$, $\text{Ufunc}[F]$, and that $\forall n \in \mathbb{N}. \text{continuous}[\lambda x \in \mathbb{R}. F(n, x)]$. Then $\text{continuous}[f]$.*

Proof. For every $x \in \mathbb{R}$, condition (3) of Definition 11 is proved in the usual way. It remains to prove that $\text{Ufunc}^{\mathbb{R}}[f]$ holds. Since $\text{Ufunc}[F]$, and since for every $x \in \mathbb{R}$, $\mathbb{N} \times \{x\} \in \mathbf{U}$, it follows that $F \upharpoonright \mathbb{N} \times \{x\} \in \mathbf{U}$ and hence for every $x \in \mathbb{R}$,

$$\lambda n \in \mathbb{N}. F(n, x) = \lambda n \in \mathbb{N}. (F \upharpoonright \mathbb{N} \times \{x\})(n, x) \in \mathbf{U}. \quad (7)$$

Let $A \in \mathbf{P}^{\mathbb{R}}$. By Proposition 6-(2),

$$f \upharpoonright A = \lambda x \in A. f(x) = \lambda x \in A. \text{limit}\langle \lambda n \in \mathbb{N}. F(n, x) \rangle$$

and by Proposition 6-(1) for every $x \in A$, the sequence $\lambda n \in \mathbb{N}. F(n, x)$ converges. By (7), for every $x \in A$, $\lambda n \in \mathbb{N}. F(n, x) \in \mathbf{U}$, and since $\text{limit}\langle \cdot \rangle$ maps convergent sequences from \mathbf{U} to real numbers (in \mathbf{U}), we conclude that

$$\lambda x \in A. \text{limit}\langle \lambda n \in \mathbb{N}. F(n, x) \rangle \in \mathbf{U}$$

and the claim follows.

Remark 8. To ease notations, Definition 13 and hence Proposition 7 deal only with sequences of functions from \mathbb{R} to \mathbb{R} . However, it can easily be extended to deal with sequences of functions from open segments to \mathbb{R} .

Definition 14 (PZF^U)(Series). *Define the following term that maps sequences $a \in \mathbf{U}$ such that $\text{converge}[\lambda n \in \mathbb{N}. \sum_{k=0}^n a(k)]$ to their sum:*

$$\text{series}\langle a \rangle := \text{limit}\langle \lambda n \in \mathbb{N}. \sum_{k=0}^n a(k) \rangle.$$

We denote $\text{series}\langle a \rangle$ by the usual convention $\sum_{k=0}^{\infty} a(k)$.

Proposition 8 (PZF^U)(Power Series).

Let $a \in \mathbf{U}$ be a sequence, and let $c \in \mathbb{R}^+$. If $\text{converge}[\lambda n \in \mathbb{N}. |a(n)| \cdot c^n]$, then $\forall x \in (-c, c). \text{converge}[\lambda n \in \mathbb{N}. \sum_{k=0}^n a(k) \cdot x^k]$, and $\lambda x \in (-c, c). \sum_{k=0}^{\infty} a(k) \cdot x^k$ is continuous (on $(-c, c)$).

Proof (proof of Proposition 8).

The proof that $\forall x \in (-c, c). \text{converge}[\lambda n \in \mathbb{N}. \sum_{k=0}^n a(k) \cdot x^k]$ is standard hence omitted. Define the functions $f := \lambda x \in (-c, c). \sum_{k=0}^{\infty} a(k) \cdot x^k$, and $F = \lambda n \in \mathbb{N}. \lambda x \in (-c, c). \sum_{k=0}^n a(k) \cdot x^k$. We wish to apply Proposition 7 to F and f , and conclude that $\text{continuous}[f]$.

The proof for $\text{uniformlyConvergence}[F, f]$ is also standard and omitted. By Proposition 5 and straightforward induction, it holds that

$$\forall n \in \mathbb{N}. \text{continuous}[\lambda x \in (-c, c). \sum_{k=0}^n a(k) \cdot x^k].$$

It remains to prove that $\text{Ufunc}[F]$. Let $X \in \mathbb{U}$ s.t. $X \subseteq \mathbb{N} \times \mathbb{R}$. By the closure properties of \mathbb{U} , it is obvious that $(\lambda \langle n, r \rangle \in X. \sum_{k=0}^n a(k) \cdot x^k) \in \mathbb{U}$, and the claim follows.

Corollary 2 (PZF^U). *Let $a \in \mathbb{U}$ be a sequence of rational numbers. If for every $c \in \mathbb{R}^+$, $\text{converge}[\lambda n \in \mathbb{N}. |a(n)| \cdot c^n]$, then $\forall x \in \mathbb{R}. \text{converge}[\lambda n \in \mathbb{N}. \sum_{k=0}^n a(k) \cdot x^k]$, and $\lambda x \in \mathbb{R}. \sum_{k=0}^{\infty} a(k) \cdot x^k$ is continuous (on \mathbb{R}).*

Proof. It is straightforward from previous proposition.

A.2 Elementary Functions

Define the following sequence:

$$a^{\sin} = \lambda n \in \mathbb{N}. \begin{cases} \frac{1}{n!} & \text{if } n \equiv 1 \pmod{4} \\ -\frac{1}{n!} & \text{if } n \equiv 3 \pmod{4} \\ 0 & \text{else} \end{cases}$$

It is straightforward to show that for every $c \in \mathbb{R}^+$, $\text{converge}[\lambda n \in \mathbb{N}. |a^{\sin}(n)| \cdot c^n]$. Hence by Corollary 2 the following function is continuous:

$$\sin = \lambda x \in \mathbb{R}. \sum_{k=0}^{\infty} a^{\sin}(k) \cdot x^k$$

Similarly we can define the other trigonometric functions, the inverse trigonometric functions, the hyperbolic/inverse hyperbolic functions, the exponentiation, and the logarithmic functions. Using Corollary 2 or Proposition 8, we can then prove that those functions are continuous. (Note that not all of these functions have domain \mathbb{R} .) Next, we define the elementary functions to be all the functions obtained by adding, subtracting, multiplying, dividing, and composing any of the previously mentioned functions. By Proposition 5, we conclude that every elementary function is a continuous function.

References

1. Avron, A.: A framework for formalizing set theories based on the use of static set terms. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 87–106. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78127-1_6
2. Avron, A.: A new approach to predicative set theory. In: Schindler, R. (ed.) Ways of Proof Theory, Onto Series in Mathematical Logic, vol. 2, pp. 31–63. Onto Verlag (2010)

3. Avron, A., Cohen, L.: Formalizing scientifically applicable mathematics in a definitional framework. *J. Formaliz. Reason.* **9**(1), 53–70 (2016)
4. Cantone, D., Omodeo, E., Policriti, A.: *Set Theory for Computing: From Decision Procedures to Declarative Programming with Sets*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-1-4757-3452-2>
5. Cohen, L., Avron, A.: Applicable mathematics in a minimal computational theory of sets. *Log. Methods Comput. Sci.* **14** (2018)
6. Cohen, L., Avron, A.: The middle ground-ancestral logic. *Synthese* **196**(7), 2671–2693 (2015). <https://doi.org/10.1007/s11229-015-0784-3>
7. De Bruijn, N.G.: A survey of the project AUTOMATH. In: *Studies in Logic and the Foundations of Mathematics*, vol. 133, pp. 141–161. Elsevier (1994)
8. Devlin, K.J.: *Constructibility*, vol. 6. Cambridge University Press, Cambridge (2017)
9. Feferman, S.: Systems of predicative analysis. *J. Symb. Log.* **29**(1), 1–30 (1964)
10. Feferman, S.: A more perspicuous formal system for predicativity. *Konstruktionen versus Positionen* **1**, 68–93 (1978)
11. Fraenkel, A.A., Bar-Hillel, Y., Levy, A.: *Foundations of Set Theory*. Elsevier, Amsterdam (1973)
12. Landau, E.: *Foundations of Analysis*. Chelsea Publishing Company, New York (1951). Translated from German ‘Grundlagen der Analysis’ by F. Steinhardt
13. Martin, R.M.: A homogeneous system for formal logic. *J. Symb. Log.* **8**(1), 1–23 (1943)
14. Myhill, J.: A derivation of number theory from ancestral theory. *J. Symb. Log.* **17**(3), 192–197 (1952)
15. Nederpelt, R.P., Geuvers, J.H., de Vrijer, R.C.: *Selected Papers on Automath*. Elsevier, Amsterdam (1994)
16. Poincaré, H.: *Les mathématiques et la logique*. *Revue de Métaphysique et de Morale* **14**(3), 294–317 (1906). <http://www.jstor.org/stable/40893278>
17. Poincaré, H.: *Dernière pensées*. Flammarion, Paris (1913). Trans. by J. Bolduc as *Mathematics and Science: Last Essays* (1963)
18. Shapiro, S.: *Foundations Without Foundationalism: A Case for Second-Order Logic*, vol. 17. Clarendon Press, Oxford (1991)
19. Simpson, S.G.: *Subsystems of Second Order Arithmetic*, vol. 1. Cambridge University Press, Cambridge (2009)
20. Weyl, H.: *Das Kontinuum: Kritische Untersuchungen über die Grundlagen der Analysis* (1918)
21. Wiedijk, F.: The QED manifesto revisited. *Stud. Log. Gramm. Rhetor.* **10**(23), 121–133 (2007)
22. Zucker, J.: Formalization of classical mathematics in AUTOMATH. In: *Studies in Logic and the Foundations of Mathematics*, vol. 133, pp. 127–139. Elsevier (1994)



Coherence via Focusing for Symmetric Skew Monoidal Categories

Niccolò Veltri^(✉) 

Tallinn University of Technology, Tallinn, Estonia
niccolo@cs.ioc.ee

Abstract. The symmetric skew monoidal categories of Bourke and Lack are a weakening of Mac Lane’s symmetric monoidal categories where: (i) the three structural laws of left and right unitality and associativity are not required to be invertible, they are merely natural transformations with a specific orientation; (ii) the structural law of symmetry is a natural isomorphism involving three objects rather than two. In this paper we study the structural proof theory of symmetric skew monoidal categories, progressing the project initiated by Uustalu et al. on deductive systems for categories with skew structure. We discuss three equivalent presentations of the free symmetric skew monoidal category on a set of generating objects: a Hilbert-style categorical calculus; a cut-free sequent calculus; a focused subsystem of derivations, corresponding to a sound and complete goal-directed proof search strategy for the cut-free sequent calculus. Focusing defines an effective normalization procedure for maps in the free symmetric skew monoidal category, as such solving the coherence problem for symmetric skew monoidal categories.

Keywords: Symmetric skew monoidal categories · Focused sequent calculus · Coherence · Substructural logic · Agda

1 Introduction

Skew monoidal categories are a weakening of Mac Lane’s monoidal categories in which the structural laws λ , ρ and α are not required to be invertible, they are merely natural transformations with a specific orientation. They were introduced by Szlachányi in his study of bialgebroids [27] and have subsequently been investigated by many (mostly Australian) category theorists [3, 5, 7, 19, 20]. In programming language semantics, skew monoidal categories prominently appear as a categorical framework for the study of relative monads [1]: similarly to monads on a category \mathbb{C} , which are characterized as monoids in the monoidal category of endofunctors on \mathbb{C} , relative monads on a functor $J : \mathbb{J} \rightarrow \mathbb{C}$ are monoids in the skew monoidal category of functors between \mathbb{J} and \mathbb{C} .

This work was supported by the Estonian Research Council grant PSG659 and by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001).

Skew monoidal categories have received notable attention in recent years in connection to their coherence problem. For (normal, non-skew) monoidal categories, Mac Lane [22] showed that each homset in the free monoidal category generated by a set of objects contains at most one map, and exactly one when the domain and codomain have the same frontier of generating objects, i.e. generating objects appear in the same order and with same multiplicity. The same is not true in the free skew monoidal category: there exist pairs of objects with the same frontier but with either no maps or multiple maps between them. This peculiarity spawned the search for concrete presentations of the free skew monoidal category, or searching for sufficient conditions for the existence of a unique map between two objects. Taking a rewriting approach, Uustalu [28] showed that there is at most one map between an object and an object in a certain normal form, and exactly one map between an object and that object's normal form. Lack and Street [20], and successively also Bourke and Lack [4], addressed the problem of determining equality of maps by proving that there is a faithful, structure-preserving functor from the free skew monoidal category on one generating object to the category Δ_{\perp} of finite non-empty ordinals and first-element-and-order-preserving functions.

Uustalu et al. [29] solved the coherence problem for skew monoidal categories by constructing a focused (in the sense of Andreoli [2]) sequent calculus in which sequent derivations are in one-to-one correspondence with maps in the free skew monoidal category. From the focused calculus, one can extract algorithms solving the following problems in the free skew monoidal category: (1) deciding equality of parallel maps; (2) enumerating all maps in a certain homset, which is a meaningful problem only in the skew case in which not all parallel maps are equal. This proof-theoretic approach to coherence is inspired by the pioneering work of Lambek on deductive systems for residuated categories [21], which many authors have successfully applied in subsequent years to other categories with structure, such as Szabo [26], Mints [24], and Dosen and Petrić [12]. Recently, Zeilberger revived this line of work in his study of the Tamari order [33].

In this paper, we continue following Lambek's footsteps and study the proof theory of symmetric skew monoidal categories, again for the purpose of coherence. An appropriate notion of symmetry (and, more generally, braiding) for skew monoidal categories have recently been introduced by Bourke and Lack [6] (which we recollect in Sect. 2): the original symmetry of Mac Lane [22], typed $B \otimes C \rightarrow C \otimes B$, is replaced by a symmetry typed $(A \otimes B) \otimes C \rightarrow (A \otimes C) \otimes B$, exclusively allowing the swapping of the second and third objects B and C , leaving the first object A in its place. Mac Lane's coherence for symmetric (normal, non-skew) monoidal categories states that two parallel maps in the free symmetric monoidal category are equal if and only if they have the same underlying permutation of generating objects. Analogously to the skew monoidal case, this is not true in the free symmetric skew monoidal category, and this peculiarity leads again to a more sophisticated solution to the coherence problem.

The central contribution of this paper resides in the introduction of three equivalent presentations of the free symmetric skew monoidal category on a set of generators: a Hilbert-style categorical calculus (Sect. 3), a cut-free sequent

calculus (Sect. 4) and a focused subsystem of the latter (Sect. 5), implementing a sound and complete backward proof search strategy attempting to build a derivation in the sequent calculus. Similarly to the skew monoidal case of Uustalu et al. [29], the sequent calculus has peculiar sequents of the form $S \mid \Gamma \vdash C$, where the antecedent is split into an optional formula S , called the stoup, and a list of formulae Γ , called the context. The symmetry $(A \otimes B) \otimes C \rightarrow (A \otimes C) \otimes B$ is modelled via a restrained exchange rule, which allows the permutation of formulae in the context but leaves the formula in the stoup unchanged. Our sequent calculus can be seen as a restricted variant of the $\text{!}, \otimes$ fragment of intuitionistic linear logic [16]. Focusing defines a normalization procedure for maps in the free symmetric skew monoidal category, as such solving the coherence problem for symmetric skew monoidal categories. Following Uustalu et al. [30], we also discuss an extension of the focused sequent calculus, defining a concrete presentation of the free normal symmetric monoidal category and recover the original Mac Lane coherence theorem for symmetric monoidal categories (Sect. 6).

The material presented in Sects. 3, 4 and 5 have been formalized in the Agda proof assistant, the code is available at <https://github.com/nicoloveltri/cohsymmskewmon>.

2 Braided/Symmetric Skew Monoidal Categories

A category \mathbb{C} is *(left-)skew monoidal* [27] if it is equipped with a distinguished object ! , a functor $\otimes : \mathbb{C} \times \mathbb{C} \rightarrow \mathbb{C}$ and three natural transformations

$$\lambda_A : \text{!} \otimes A \rightarrow A \quad \rho_A : A \rightarrow A \otimes \text{!} \quad \alpha_{A,B,C} : (A \otimes B) \otimes C \rightarrow A \otimes (B \otimes C)$$

satisfying the equations

$$\begin{array}{ccc} \text{(m1)} & \begin{array}{c} \text{!} \otimes \text{!} \\ \rho_{\text{!}} \nearrow \quad \searrow \lambda_{\text{!}} \\ \text{!} \end{array} & \text{(m2)} \quad \begin{array}{ccc} (A \otimes \text{!}) \otimes B & \xrightarrow{\alpha_{A,\text{!},B}} & A \otimes (\text{!} \otimes B) \\ \rho_{A \otimes B} \uparrow & & \downarrow A \otimes \lambda_B \\ A \otimes B & \xlongequal{\quad\quad\quad} & A \otimes B \end{array} \end{array}$$

$$\begin{array}{ccc} \text{(m3)} \quad \begin{array}{ccc} (\text{!} \otimes A) \otimes B & \xrightarrow{\alpha_{\text{!},A,B}} & \text{!} \otimes (A \otimes B) \\ \lambda_{A \otimes B} \searrow & & \nearrow \lambda_{A \otimes B} \\ & A \otimes B & \end{array} & \text{(m4)} \quad \begin{array}{ccc} (A \otimes B) \otimes \text{!} & \xrightarrow{\alpha_{A,B,\text{!}}} & A \otimes (B \otimes \text{!}) \\ \rho_{A \otimes B} \swarrow & & \nearrow A \otimes \rho_B \\ & A \otimes B & \end{array} \end{array}$$

$$\begin{array}{ccc} \text{(m5)} \quad (A \otimes (B \otimes C)) \otimes D & \xrightarrow{\alpha_{A,B \otimes C,D}} & A \otimes ((B \otimes C) \otimes D) \\ \alpha_{A,B,C \otimes D} \uparrow & & \downarrow A \otimes \alpha_{B,C,D} \\ ((A \otimes B) \otimes C) \otimes D & \xrightarrow{\alpha_{A \otimes B,C,D}} (A \otimes B) \otimes (C \otimes D) & \xrightarrow{\alpha_{A,B,C \otimes D}} A \otimes (B \otimes (C \otimes D)) \end{array}$$

The latter equations are directed versions of the original Mac Lane axioms [22] for monoidal categories. Kelly [18] observed that, in the monoidal case, equations (m1), (m3), and (m4) follow from (m2) and (m5). In the skew situation, this is not the case.

A skew monoidal category is *braided* [6] if it is additionally equipped with a natural isomorphism

$$s_{A,B,C} : (A \otimes B) \otimes C \rightarrow (A \otimes C) \otimes B$$

satisfying the equations

$$(b1) \quad \begin{array}{ccccc} ((A \otimes B) \otimes C) \otimes D & \xrightarrow{s_{A \otimes B, C, D}} & ((A \otimes B) \otimes D) \otimes C & \xrightarrow{s_{A, B, D \otimes C}} & ((A \otimes D) \otimes B) \otimes C \\ s_{A, B, C \otimes D} \downarrow & & & & \downarrow s_{A \otimes D, B, C} \\ ((A \otimes C) \otimes B) \otimes D & \xrightarrow{s_{A \otimes C, B, D}} & ((A \otimes C) \otimes D) \otimes B & \xrightarrow{s_{A, C, D \otimes B}} & ((A \otimes D) \otimes C) \otimes B \end{array}$$

$$(b2) \quad \begin{array}{ccc} ((A \otimes B) \otimes C) \otimes D & \xrightarrow{s_{A, B, C \otimes D}} & ((A \otimes C) \otimes B) \otimes D \xrightarrow{s_{A \otimes C, B, D}} ((A \otimes C) \otimes D) \otimes B \\ \alpha_{A \otimes B, C, D} \downarrow & & \downarrow \alpha_{A, C, D \otimes B} \\ (A \otimes B) \otimes (C \otimes D) & \xrightarrow{s_{A, B, C \otimes D}} & (A \otimes (C \otimes D)) \otimes B \end{array}$$

$$(b3) \quad \begin{array}{ccc} ((A \otimes B) \otimes C) \otimes D & \xrightarrow{s_{A \otimes B, C, D}} & ((A \otimes B) \otimes D) \otimes C \xrightarrow{s_{A, B, D \otimes C}} ((A \otimes D) \otimes B) \otimes C \\ \alpha_{A, B, C \otimes D} \downarrow & & \downarrow \alpha_{A \otimes D, B, C} \\ (A \otimes (B \otimes C)) \otimes D & \xrightarrow{s_{A, B \otimes C, D}} & (A \otimes D) \otimes (B \otimes C) \end{array}$$

$$(b4) \quad \begin{array}{ccc} ((A \otimes B) \otimes C) \otimes D & \xrightarrow{\alpha_{A, B, C \otimes D}} & (A \otimes (B \otimes C)) \otimes D \xrightarrow{\alpha_{A, B \otimes C, D}} A \otimes ((B \otimes C) \otimes D) \\ s_{A \otimes B, C, D} \downarrow & & \downarrow A \otimes s_{B, C, D} \\ ((A \otimes B) \otimes D) \otimes C & \xrightarrow{\alpha_{A, B, D \otimes C}} & (A \otimes (B \otimes D)) \otimes C \xrightarrow{\alpha_{A, B \otimes D, C}} A \otimes ((B \otimes D) \otimes C) \end{array}$$

The braiding s is a *symmetry* if it is its own inverse, i.e. $s^{-1} = s$. The structural law s is a restricted version of the usual braiding of Joyal and Street [17] in normal braided monoidal categories typed $B \otimes C \rightarrow C \otimes B$. Now the leftmost object (A in the type of s above) is not allowed to be swapped with the other objects B and C . This implies the existence of a map between any two objects of the form $(\dots((A \otimes B_1) \otimes B_2) \dots) \otimes B_n$ and $(\dots((A \otimes B_{i_1}) \otimes B_{i_2}) \dots) \otimes B_{i_n}$, where i_1, \dots, i_n is a permutation of $1, \dots, n$, and the object A is required to stay

in its place. At first sight, equations (b1)-(b4) look very different from the usual equations of normal braided monoidal categories, but they are reminiscent of an alternative presentation by Davydov and Runkel via b-structures [10]. They also appear, with minor variations, as the coherence conditions for operadic trees of Curien et al. [9]. Bourke and Lack [6] showed that, when the left unitor λ is invertible, a braiding typed $B \otimes C \rightarrow C \otimes B$ is derivable:

$$B \otimes C \xrightarrow{\lambda_B^{-1} \otimes C} (1 \otimes B) \otimes C \xrightarrow{s_{1,B,C}} (1 \otimes C) \otimes B \xrightarrow{\lambda_C \otimes B} C \otimes B$$

and moreover the braided skew monoidal structure is normal braided monoidal. In particular, ρ and α are also invertible.

Example 1. The category **Ptd** of pointed sets and point-preserving maps has the following symmetric skew monoidal structure. The unit is $1 = (1, \star)$, where 1 is the singleton set with unique element \star . The tensor of two pointed sets (A, a) and (B, b) is $(A, a) \otimes (B, b) = (A + B, \text{inl}(a))$, where $A + B$ is the disjoint union of A and B , and inl is the injection of A into $A + B$. The structural laws λ and ρ are not invertible, while α is an isomorphism. The natural isomorphism $s : ((A + B) + C, \text{inl}(\text{inl}(a))) \rightarrow ((A + C) + B, \text{inl}(\text{inl}(a)))$ is defined using the symmetry and associativity of disjoint union.

This example is an instance of a more general phenomenon discussed by Bourke and Lack [6]: given a braided (resp. symmetric) monoidal category \mathbb{C} and a monoid M in \mathbb{C} , then the category of left M -modules is braided (resp. symmetric) skew monoidal. The example above arises by taking \mathbb{C} as the category of sets and functions with the symmetric monoidal structure given by disjoint union, and the monoid M as the singleton set 1 . The category of left 1 -modules is isomorphic to **Ptd**.

Example 2. Given a braided (skew) monoidal category $(\mathbb{C}, 1, \otimes)$ and a comonad (D, ε, δ) on \mathbb{C} . Suppose D is *lax braided monoidal*, i.e., comes with a map $e : 1 \rightarrow D1$ and a natural transformation $m : DA \otimes DB \rightarrow D(A \otimes B)$ agreeing suitably with $\lambda, \rho, \alpha, s, \varepsilon, \delta$. The category \mathbb{C} has another braided skew monoidal structure $(1, \otimes^D)$ where $A \otimes^D B = A \otimes DB$. The unitors, associator and braiding are:

$$\begin{aligned} \lambda_A^D &= 1 \otimes DA \xrightarrow{1 \otimes \varepsilon_A} 1 \otimes A \xrightarrow{\lambda_A} A \\ \rho_A^D &= A \xrightarrow{\rho_A} A \otimes 1 \xrightarrow{A \otimes e} A \otimes D1 \\ \alpha_{A,B,C}^D &= (A \otimes DB) \otimes DC \xrightarrow{(A \otimes DB) \otimes \delta_C} (A \otimes DB) \otimes D(DC) \\ &\xrightarrow{\alpha_{A,DB,D(DC)}} A \otimes (DB \otimes D(DC)) \xrightarrow{A \otimes m_{B,DC}} A \otimes D(B \otimes DC) \\ s_{A,B,C}^D &= (A \otimes DB) \otimes DC \xrightarrow{s_{A,DB,DC}} (A \otimes DC) \otimes DB \end{aligned}$$

If the braiding s of \mathbb{C} is a symmetry and D is lax symmetric monoidal, then s^D is a symmetry as well.

3 The Free Symmetric Skew Monoidal Category

The free symmetric skew monoidal category $\mathbf{Fssk}(\text{At})$ over a set At (of atoms) can be defined as a Hilbert-style deductive system, which we call the *categorical calculus*. Objects of $\mathbf{Fssk}(\text{At})$ are *formulae* inductively generated as follows: either an *atom* $X \in \text{At}$; \mathbf{l} ; or $A \otimes B$ where A, B are formulae. We write Fma for the set of formulae. Maps between two formulae A and C are *derivations* of (singleton-antecedent, singleton-succedent) sequents $A \Longrightarrow C$, constructed using the following inference rules:

$$\begin{array}{c}
 \frac{}{A \Longrightarrow A} \text{id} \quad \frac{B \Longrightarrow C \quad A \Longrightarrow B}{A \Longrightarrow C} \circ \quad \frac{A \Longrightarrow C \quad B \Longrightarrow D}{A \otimes B \Longrightarrow C \otimes D} \otimes \\
 \frac{\mathbf{l} \otimes A \Longrightarrow A}{A \Longrightarrow A} \lambda \quad \frac{A \Longrightarrow A \otimes \mathbf{l}}{A \Longrightarrow A \otimes \mathbf{l}} \rho \quad \frac{}{(A \otimes B) \otimes C \Longrightarrow A \otimes (B \otimes C)} \alpha \quad (1) \\
 \frac{}{(A \otimes B) \otimes C \Longrightarrow (A \otimes C) \otimes B} s
 \end{array}$$

and identified up to the congruence \doteq induced by the equations in Fig. 1.

The categorical calculus defines the free symmetric skew monoidal category on At in a direct way. Given another symmetric skew monoidal category \mathbb{C} with function $G : \text{At} \rightarrow \mathbb{C}$, we can easily define mappings $\overline{G}_0 : \text{Fma} \rightarrow \mathbb{C}_0$ and $\overline{G}_1 : (A \Longrightarrow C) \rightarrow \mathbb{C}(\overline{G}_0(A), \overline{G}_0(C))$ by induction. These specify a strict symmetric monoidal functor, in fact the only existing one satisfying $\overline{G}_0(X) = G(X)$.

Define the *frontier* $\delta(A)$ of a formula A as the ordered list of atoms contained in A . Given an ordered list l with length n and an element σ of the symmetric group on n elements, i.e. a permutation of n elements, we write $\sigma \cdot l$ for the action of σ on l .

Mac Lane [22] proved a coherence theorem for (normal, non-skew) symmetric monoidal categories, which can be phrased as follows: given two formulae A and C in the free symmetric monoidal category, there exists a map typed $A \Longrightarrow C$ iff there exists a permutation σ such that $\delta(C) = \sigma \cdot \delta(A)$ (which is to say that $\delta(A)$ and $\delta(C)$ are equal as multisets) and, if this is the case, two parallel maps in $A \Longrightarrow C$ are equal iff they have the same underlying permutation σ .

For the free symmetric skew monoidal category, this is not the case. First, there exist pairs of formulae with the exact same frontier but with no maps between them: there are no maps typed $X \Longrightarrow \mathbf{l} \otimes X$, no maps typed $X \otimes \mathbf{l} \Longrightarrow X$ and no maps typed $X \otimes (Y \otimes Z) \Longrightarrow (X \otimes Y) \otimes Z$. There are also no maps typed $X \otimes Y \Longrightarrow Y \otimes X$. Moreover, there are multiple maps between formulae with the exact same frontier: there are two maps $\text{id} \neq \alpha \circ \rho \otimes \lambda : X \otimes (\mathbf{l} \otimes Y) \Longrightarrow X \otimes (\mathbf{l} \otimes Y)$ and two maps $\text{id} \neq \rho \otimes \lambda \circ \alpha : (X \otimes \mathbf{l}) \otimes Y \Longrightarrow (X \otimes \mathbf{l}) \otimes Y$. Bourke and Lack [6] showed that there are two maps $(\text{id} \otimes \lambda) \otimes \text{id} \circ \alpha \otimes \text{id} \neq \text{id} \otimes \lambda \circ \alpha \circ s \otimes \text{id} : ((X \otimes \mathbf{l}) \otimes Y) \otimes Z \Longrightarrow (X \otimes Y) \otimes Z$. Postcomposing the latter two maps with s , we obtain two distinct maps typed $((X \otimes \mathbf{l}) \otimes Y) \otimes Z \Longrightarrow (X \otimes Z) \otimes Y$ which have underlying permutation of frontiers $\sigma = (132)$.

$$\begin{array}{ll}
\text{(category laws)} & \text{id} \circ f \doteq f \quad f \doteq f \circ \text{id} \quad (f \circ g) \circ h \doteq f \circ (g \circ h) \\
(\otimes \text{ functorial}) & \text{id} \otimes \text{id} \doteq \text{id} \quad (h \circ f) \otimes (k \circ g) \doteq h \circ k \circ f \otimes g \\
(\lambda, \rho, \alpha, s \text{ nat. trans.}) & \lambda \circ \text{id} \otimes f \doteq f \circ \lambda \quad \rho \circ f \doteq f \otimes \text{id} \circ \rho \\
& \alpha \circ (f \otimes g) \otimes h \doteq f \otimes (g \otimes h) \circ \alpha \\
& s \circ (f \otimes g) \otimes h \doteq (f \otimes h) \otimes g \circ s \\
\text{(m1-m5)} & \lambda \circ \rho \doteq \text{id} \quad \text{id} \doteq \text{id} \otimes \lambda \circ \alpha \circ \rho \otimes \text{id} \\
& \lambda \circ \alpha \doteq \lambda \otimes \text{id} \quad \alpha \circ \rho \doteq \text{id} \otimes \rho \\
& \alpha \circ \alpha \doteq \text{id} \otimes \alpha \circ \alpha \otimes \text{id} \\
\text{(b1-b4)} & s \otimes \text{id} \circ s \circ s \otimes \text{id} \doteq s \circ s \otimes \text{id} \circ s \\
& s \circ \alpha \doteq \alpha \otimes \text{id} \circ s \circ s \otimes \text{id} \quad s \circ \alpha \otimes \text{id} \doteq \alpha \circ s \otimes \text{id} \circ s \\
& \alpha \circ \alpha \otimes \text{id} \circ s \doteq \text{id} \otimes s \circ \alpha \circ \alpha \otimes \text{id} \\
\text{(s symmetry)} & s \circ s \doteq \text{id}
\end{array}$$

Fig. 1. Equivalence of derivations in the categorical calculus

4 Symmetric Skew Monoidal Sequent Calculus

The free symmetric skew monoidal category $\mathbf{Fssk}(\text{At})$ admits an equivalent presentation as a cut-free sequent calculus. Formulae are again elements of \mathbf{Fma} . Similarly to the non-symmetric case worked out in a previous paper [29], sequents are triples of the form $S \mid \Gamma \vdash C$. The antecedent of the sequent is split in two parts: S is an optional formula called the *stoup*, which can either be empty (which we denote $-$) or a single formula, and Γ is an ordered list of formulae, that we call the *context*. The succedent C is a single formula. *Derivations* of sequents $S \mid \Gamma \vdash C$ are generated by the following inference rules:

$$\begin{array}{c}
\frac{A \mid \Gamma \vdash C}{- \mid A, \Gamma \vdash C} \text{pass} \quad \frac{- \mid \Gamma \vdash C}{\Gamma \vdash C} \text{IL} \quad \frac{A \mid B, \Gamma \vdash C}{A \otimes B \mid \Gamma \vdash C} \otimes\text{L} \\
\frac{}{A \mid \vdash A} \text{ax} \quad \frac{}{- \mid \vdash -} \text{IR} \quad \frac{S \mid \Gamma \vdash A \quad - \mid \Delta \vdash B}{S \mid \Gamma, \Delta \vdash A \otimes B} \otimes\text{R} \\
\frac{S \mid \Gamma, A, B, \Delta \vdash C}{S \mid \Gamma, B, A, \Delta \vdash C} \text{ex}_{A,B}
\end{array} \quad (2)$$

(pass for ‘passivate’, ex for ‘exchange’, L, R for introduction on the left (in the stoup) resp. right) and identified up to the congruence \doteq induced by the equations in Fig. 2.

The rules IL, $\otimes\text{L}$ and ex are invertible up to \doteq , but the passivation rule pass is not. General forms of exchange, swapping a formula with a list of formulae, are admissible by induction on the list of formulae Ξ :

$$\frac{S \mid \Gamma, \Xi, A, \Delta \vdash C}{S \mid \Gamma, A, \Xi, \Delta \vdash C} \text{ex}_{S,A,\Xi} \quad \frac{S \mid \Gamma, A, \Xi, \Delta \vdash C}{S \mid \Gamma, \Xi, A, \Delta \vdash C} \text{ex}_{S,\Xi,A} \quad (3)$$

(η -conversions)

$$\mathbf{ax}_I \stackrel{\circ}{=} \mathbf{IL} \mathbf{IR} \qquad \mathbf{ax}_{A \otimes B} \stackrel{\circ}{=} \otimes \mathbf{L} (\otimes \mathbf{R} (\mathbf{ax}_A, \mathbf{pass} \mathbf{ax}_B))$$

(commutative conversions)

$$\begin{aligned} \mathbf{ex}_{A',B'} (\mathbf{ex}_{A,B} f) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\mathbf{ex}_{A',B'} f) && (\text{for } f : S \mid \Gamma_1, A, B, \Gamma_2, A', B', \Gamma_3 \vdash C) \\ \mathbf{pass} (\mathbf{ex}_{A,B} f) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\mathbf{pass} f) && (\text{for } f : A' \mid \Gamma, A, B, \Delta \vdash C) \\ \mathbf{IL} (\mathbf{ex}_{A,B} f) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\mathbf{IL} f) && (\text{for } f : - \mid \Gamma, A, B, \Delta \vdash C) \\ \otimes \mathbf{L} (\mathbf{ex}_{A,B} f) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\otimes \mathbf{L} f) && (\text{for } f : A' \mid B', \Gamma, A, B, \Delta \vdash C) \\ \otimes \mathbf{R} (\mathbf{ex}_{A,B} f, g) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\otimes \mathbf{R} (f, g)) && (\text{for } f : S \mid \Gamma_1, A, B, \Gamma_2 \vdash A', g : - \mid \Delta \vdash B') \\ \otimes \mathbf{R} (f, \mathbf{ex}_{A,B} g) &\stackrel{\circ}{=} \mathbf{ex}_{A,B} (\otimes \mathbf{R} (f, g)) && (\text{for } f : S \mid \Gamma \vdash A', g : - \mid \Delta_1, A, B, \Delta_2 \vdash B') \\ \otimes \mathbf{R} (\mathbf{pass} f, g) &\stackrel{\circ}{=} \mathbf{pass} (\otimes \mathbf{R} (f, g)) && (\text{for } f : A' \mid \Gamma \vdash A, g : - \mid \Delta \vdash B) \\ \otimes \mathbf{R} (\mathbf{IL} f, g) &\stackrel{\circ}{=} \mathbf{IL} (\otimes \mathbf{R} (f, g)) && (\text{for } f : - \mid \Gamma \vdash A, g : - \mid \Delta \vdash B) \\ \otimes \mathbf{R} (\otimes \mathbf{L} f, g) &\stackrel{\circ}{=} \otimes \mathbf{L} (\otimes \mathbf{R} (f, g)) && (\text{for } f : A' \mid B', \Gamma \vdash A, g : - \mid \Delta \vdash B) \end{aligned}$$

(Yang-Baxter equation)

$$\mathbf{ex}_{B,D} (\mathbf{ex}_{A,D} (\mathbf{ex}_{A,B} f)) \stackrel{\circ}{=} \mathbf{ex}_{A,B} (\mathbf{ex}_{A,D} (\mathbf{ex}_{B,D} f)) \qquad (\text{for } f : S \mid \Gamma, A, B, D, \Delta \vdash C)$$

(ex isomorphism)

$$\mathbf{ex}_{A,B} (\mathbf{ex}_{A,B} f) \stackrel{\circ}{=} f \qquad (\text{for } f : S \mid \Gamma, A, B, \Delta \vdash C)$$

Fig. 2. Equivalence of derivations in the sequent calculus

Two forms of cut are admissible, satisfying a large number of $\stackrel{\circ}{=}$ -equations (see [29, Figures 5 and 6] for the list of such equations not involving the exchange rule \mathbf{ex}).

$$\frac{S \mid \Gamma \vdash A \quad A \mid \Delta \vdash C}{S \mid \Gamma, \Delta \vdash C} \text{scut} \qquad \frac{- \mid \Gamma \vdash A \quad S \mid \Delta_1, A, \Delta_2 \vdash C}{S \mid \Delta_1, \Gamma, \Delta_2 \vdash C} \text{ccut} \qquad (4)$$

The inference rules in (2) are reminiscent of the rules of the \mathbf{l}, \otimes fragment of intuitionistic linear logic, but there are some crucial differences/restrictions:

1. The left rules \mathbf{IL} and $\otimes \mathbf{L}$ act only on the formula in the stoup. In particular, there are no rules for decomposing a unit \mathbf{l} or a tensor $A \otimes B$ in the context. This allows the derivability of sequents corresponding to the right unitor $\rho : A \Longrightarrow A \otimes \mathbf{l}$ and the associator $\alpha : (A \otimes B) \otimes C \Longrightarrow A \otimes (B \otimes C)$, but not their inverses.

$$\frac{\frac{\frac{\overline{A \mid \vdash A} \text{ ax} \quad \overline{- \mid \vdash \perp} \text{ IR}}{A \mid \vdash A \otimes \perp} \otimes \text{R}}{\frac{\frac{\overline{A \mid \vdash A} \text{ ax} \quad \frac{\frac{\overline{B \mid \vdash B} \text{ ax} \quad \frac{\overline{C \mid \vdash C} \text{ ax}}{- \mid C \vdash C} \text{ pass}}{B \mid C \vdash B \otimes C} \otimes \text{R}}{- \mid B, C \vdash B \otimes C} \text{ pass}}{A \mid B, C \vdash A \otimes (B \otimes C)} \otimes \text{R}}{A \otimes B \mid C \vdash A \otimes (B \otimes C)} \otimes \text{L}}{(A \otimes B) \otimes C \mid \vdash A \otimes (B \otimes C)} \otimes \text{L}}$$

2. There is a distinction between antecedents of the form $A \mid \Gamma$, where the formula A is in the stoup, and antecedents of the form $- \mid A, \Gamma$, where A is out of the stoup. This distinction is crucial in the right rule $\otimes \text{R}$, which always sends the formula in the stoup to the first premise (when the rule is read bottom-up). This allows the derivability of a sequent corresponding to the left unitor $\lambda : \perp \otimes A \Longrightarrow A$, but not its inverse, since the passivation rule pass is not invertible.

$$\frac{\frac{\frac{\overline{A \mid \vdash A} \text{ ax}}{- \mid A \vdash A} \text{ pass}}{\perp \mid A \vdash A} \text{ IL}}{\perp \otimes A \mid \vdash A} \otimes \text{L}$$

3. The exchange rule $\text{ex}_{A,B}$ permits the swap of two adjacent formulae A and B in the context, but there is no way to generally swap a formula in the stoup with a formula in the context. This allows the derivability of a sequent corresponding to the symmetry $s : (A \otimes B) \otimes C \Longrightarrow (A \otimes C) \otimes B$ involving three formulae, but not of a symmetry involving two formulae typed $B \otimes C \Longrightarrow C \otimes B$.

$$\frac{\frac{\frac{\overline{A \mid \vdash A} \text{ ax} \quad \frac{\overline{C \mid \vdash C} \text{ ax}}{- \mid C \vdash C} \text{ pass}}{A \mid C \vdash A \otimes C} \otimes \text{R} \quad \frac{\overline{B \mid \vdash B} \text{ ax}}{- \mid B \vdash B} \text{ pass}}{\frac{A \mid C, B \vdash (A \otimes C) \otimes B}{A \mid B, C \vdash (A \otimes C) \otimes B} \text{ ex}_{B,C}}{A \otimes B \mid C \vdash (A \otimes C) \otimes B} \otimes \text{L}}{(A \otimes B) \otimes C \mid \vdash (A \otimes C) \otimes B} \otimes \text{L}$$

The sequent calculus rules in (2) accurately match the categorical calculus rules in (1), and the $\stackrel{=}{\dashv}$ -equations in Fig. 2 match the $\stackrel{\doteq}{\dashv}$ -equations in Fig. 1, in very a precise sense. There exists an effective procedure $\text{sound} : (S \mid \Gamma \vdash C) \rightarrow (\llbracket S \mid \Gamma \rrbracket \Longrightarrow C)$ turning a sequent calculus derivation into a categorical calculus derivation, where the interpretation of an antecedent as a formula $\llbracket S \mid \Gamma \rrbracket$ is defined as $\llbracket S \mid \Gamma \rrbracket = \llbracket S \ll \llbracket \Gamma \rrbracket \rrbracket$ with

$$\llbracket - \rrbracket \langle = \mid \quad \llbracket A \rrbracket \langle = A \quad A \langle \llbracket \rrbracket = A \quad A \langle \llbracket B, \Gamma \rrbracket = (A \otimes B) \langle \llbracket \Gamma \rrbracket$$

which means that $A \langle \llbracket A_1, A_2, \dots, A_n \rrbracket = (\dots (A \otimes A_1) \otimes A_2) \dots \otimes A_n$. The function sound is well-defined, in the sense that it sends $\dot{=}$ -related derivations to $\dot{=}$ -related derivations. There exists also an effective procedure $\text{cmplt} : (\llbracket S \mid \Gamma \rrbracket \Longrightarrow C) \rightarrow (S \mid \Gamma \vdash C)$ which is the inverse of sound up to the equivalences $\dot{=}$ and $\dot{=}$, i.e. $\text{sound}(\text{cmplt}(f)) \dot{=} f$, for all $f : \llbracket S \mid \Gamma \rrbracket \Longrightarrow C$, and $\text{cmplt}(\text{sound}(g)) \dot{=} g$, for all $g : S \mid \Gamma \vdash C$. Composition $f \circ g$ in the categorical calculus is interpreted by cmplt as $\text{scut}(\text{cmplt}(g), \text{cmplt}(f))$. The function cmplt is also well-defined, i.e. it sends $\dot{=}$ -related derivations to $\dot{=}$ -related derivations.

Theorem 1. *The set of derivations of the sequent $S \mid \Gamma \vdash C$, quotiented by the equivalence relation $\dot{=}$, is isomorphic to the set of derivations of the sequent $\llbracket S \mid \Gamma \rrbracket \Longrightarrow C$, quotiented by the equivalence relation $\dot{=}$.*

This shows that the cut-free sequent calculus is an equivalent presentation of the free symmetric skew monoidal category $\mathbf{Fssk}(\text{At})$.

5 A Focused Subsystem for the Symmetric Skew Case

The free symmetric skew monoidal category admits a more concrete presentation as a focused sequent calculus. Derivations in this calculus correspond to canonical representatives of equivalence classes of the relation $\dot{=}$. They are generated by the following inference rules:

$$\begin{array}{c} \frac{S \mid \Omega \dot{=} \Gamma, A, \Delta \vdash_C C}{S \mid \Omega, A \dot{=} \Gamma, \Delta \vdash_C C} \text{exs}_{A, \Gamma} \quad \frac{S \mid \Gamma \vdash_L C}{S \mid \dot{=} \Gamma \vdash_C C} \text{sw}_{LC} \\ \frac{- \mid \Gamma \vdash_L C}{\mid \mid \Gamma \vdash_L C} \text{ll} \quad \frac{A \mid B \dot{=} \Gamma \vdash_C C}{A \otimes B \mid \Gamma \vdash_L C} \otimes_L \quad \frac{A \mid \Gamma \vdash_L C}{- \mid A, \Gamma \vdash_L C} \text{pass} \quad \frac{T \mid \Gamma \vdash_R C}{T \mid \Gamma \vdash_L C} \text{sw}_{RL} \quad (5) \\ \frac{}{X \mid \vdash_R X} \text{ax} \quad \frac{}{- \mid \vdash_R \mid} \text{IR} \quad \frac{T \mid \Gamma \vdash_R A \quad - \mid \Delta \vdash_L B}{T \mid \Gamma, \Delta \vdash_R A \otimes B} \otimes_R \end{array}$$

(T is always an optional atom, i.e. either empty or an atomic formula.) As in Andreoli's original formulation for linear logic [2], the focused calculus defines a goal-directed proof search strategy which attempts to build a derivation of a sequent in the cut-free sequent calculus of Sect. 4, starting from the root:

- We start in phase C (for ‘context’) by permuting the formulae in the context. This is performed step-by-step using the rule exs , moving one formula at the time, in a way reminiscent of the insertion-sort algorithm. In this phase, contexts are split in two parts $\Omega \dot{=} \Gamma$, where Γ consists of formulae that have already been moved by exs and the formulae in Ω are yet to be moved. Once all the formulae have been moved, so the antecedent is of the form $\dot{=} \Gamma$, we switch to phase L using the rule sw_{LC} .

- In phase L (for ‘left’), the formula in the stoup is eagerly decomposed using the invertible left rules IL and $\otimes\text{L}$. The premise of the $\otimes\text{L}$ rule is a derivation of a sequent in phase C, which gives the chance to further move the formula B in a different position of the context. If the stoup is empty we have the possibility of applying the **pass** rule, i.e. moving the leftmost formula A in the context to the stoup, and continue the decomposition of A . When the formula in the stoup is fully decomposed, i.e. either the stoup is empty or it contains an atomic formula, we switch to phase R using the rule sw_{RL} . Notice that, when the stoup is empty, we are not obliged to use the **pass** rule, so we have a choice between applying **pass** and sw_{RL} .
- In phase R (for ‘right’), we focus on the succedent formula. Depending on its shape, only one among the rules ax , IR , $\otimes\text{R}$ can be applied. The second premise of the $\otimes\text{R}$ rule is a derivation of a sequent in phase L, which gives the chance of applying the **pass** rule and subsequently the invertible left rules IL and $\otimes\text{L}$. Different ways of splitting the context in an application of the rule $\otimes\text{R}$ can lead to different successful derivations, which is another source of nondeterminism.

By dropping the phase annotations (also turning $\dot{\vdash}$ into a comma), we can define three functions emb_{C} , emb_{L} and emb_{R} embedding focused sequent calculus derivations in the unfocused sequent calculus. The function $\text{emb}_{\text{C}} : (S \mid \Omega \dot{\vdash} \Gamma \vdash_{\text{C}} C) \rightarrow (S \mid \Omega, \Gamma \vdash C)$ interprets the focused rule $\text{exs}_{A,\Gamma}$ in (5) as the admissible unfocused rule $\text{exs}_{A,\Gamma}$ in (3). We can also define a normalization function $\text{focus} : (S \mid \Omega \vdash C) \rightarrow (S \mid \Omega \dot{\vdash} \vdash_{\text{C}} C)$, which maps $\overset{\circ}{=}$ -related derivations to *equal* focused derivations. For the definition of focus , all the unfocused rules in (2) are proved admissible in phase C. The functions focus and emb_{C} establish a bijection between the sequent calculus and its focused subsystem: $\text{emb}_{\text{C}}(\text{focus } f) \overset{\circ}{=} f$ and $\text{focus}(\text{emb}_{\text{C}} g) = g$, for all $f : S \mid \Gamma \vdash C$ and $g : S \mid \Omega \dot{\vdash} \vdash_{\text{C}} C$.

Theorem 2. *The following are isomorphic:*

- (i) *the set of derivations of the sequent $S \mid \Omega \dot{\vdash} \vdash_{\text{C}} C$;*
- (ii) *the set of derivations of the sequent $S \mid \Omega \vdash C$, quotiented by the equivalence relation $\overset{\circ}{=}$;*
- (iii) *the set of derivations of the sequent $\llbracket S \mid \Gamma \rrbracket \Longrightarrow C$, quotiented by the equivalence relation $\overset{\circ}{=}$.*

The focused sequent calculus is peculiar in that it gives the ability of having a “change of mind” regarding the position to which a formula is moved during phase C. To explain this phenomenon, consider for example the two derivations of $\vdash X, \text{I} \otimes Y \dot{\vdash} \vdash_{\text{C}} X \otimes Y$, which in the categorical calculus correspond to distinct maps $\lambda_X \otimes \lambda_Y \neq \lambda_X \otimes \text{id}_Y \circ s_{\text{I},Y,X} \circ \lambda_{\text{I} \otimes Y} \otimes \text{id}_X \circ s_{\text{I},X,\text{I} \otimes Y} : (\text{I} \otimes X) \otimes (\text{I} \otimes Y) \Longrightarrow X \otimes Y$:

$$\begin{array}{c}
\frac{}{Y \mid \vdash_R Y} \text{ax} \\
\frac{}{Y \mid \vdash_L Y} \text{ax} \\
\frac{}{- \mid Y \vdash_L Y} \text{pass} \\
\frac{}{\mid Y \vdash_L Y} \text{IL} \\
\frac{}{\mid \vdots Y \vdash_C Y} \text{sw}_{\text{LC}} \\
\frac{}{\mid Y \vdots \vdash_C Y} \text{ex}_{\text{SY},()}\otimes\text{L} \\
\frac{}{\mid \otimes Y \mid \vdash_L Y} \text{pass} \\
\frac{}{X \mid \vdash_R X} \text{ax} \quad \frac{}{- \mid \otimes Y \vdash_L Y} \otimes\text{R} \\
\frac{}{X \mid \mid \otimes Y \vdash_R X \otimes Y} \text{sw}_{\text{RL}} \\
\frac{}{X \mid \mid \otimes Y \vdash_L X \otimes Y} \text{pass} \\
\frac{}{- \mid X, \mid \otimes Y \vdash_L X \otimes Y} \text{sw}_{\text{LC}} \\
\frac{}{- \mid \vdots X, \mid \otimes Y \vdash_C X \otimes Y} \text{ex}_{\text{X},()}\otimes\text{L} \\
\frac{}{- \mid X \vdots \mid \otimes Y \vdash_C X \otimes Y} \text{ex}_{\text{S}\mid\otimes\text{Y},()}\otimes\text{L} \\
\frac{}{- \mid X, \mid \otimes Y \vdots \vdash_C X \otimes Y} \text{pass}
\end{array}
\quad
\begin{array}{c}
\frac{}{Y \mid \vdash_R Y} \text{ax} \\
\frac{}{Y \mid \vdash_L Y} \text{ax} \\
\frac{}{- \mid Y \vdash_L Y} \text{pass} \\
\frac{}{X \mid \vdash_R X} \text{ax} \quad \frac{}{- \mid Y \vdash_L Y} \otimes\text{R} \\
\frac{}{X \mid Y \vdash_R X \otimes Y} \text{sw}_{\text{RL}} \\
\frac{}{X \mid Y \vdash_L X \otimes Y} \text{pass} \\
\frac{}{- \mid X, Y \vdash_L X \otimes Y} \text{IL} \\
\frac{}{\mid X, Y \vdash_L X \otimes Y} \text{sw}_{\text{LC}} \\
\frac{}{\mid \vdots X, Y \vdash_C X \otimes Y} \text{ex}_{\text{SY},\text{X}}\otimes\text{L} \\
\frac{}{\mid Y \vdots X \vdash_C X \otimes Y} \otimes\text{L} \\
\frac{}{\mid \otimes Y \mid X \vdash_L X \otimes Y} \text{pass} \\
\frac{}{- \mid \otimes Y, X \vdash_L X \otimes Y} \text{sw}_{\text{LC}} \\
\frac{}{- \mid \vdots \mid \otimes Y, X \vdash_C X \otimes Y} \text{ex}_{\text{X},\mid\otimes\text{Y}}\otimes\text{L} \\
\frac{}{- \mid X \vdots \mid \otimes Y \vdash_C X \otimes Y} \text{ex}_{\text{S}\mid\otimes\text{Y},()}\otimes\text{L} \\
\frac{}{- \mid X, \mid \otimes Y \vdots \vdash_C X \otimes Y} \text{pass}
\end{array}
\tag{6}$$

On the left, the formulae in the context are never swapped. On the right, first X is swapped with $\mid \otimes Y$ (the **blue** `exs` rule), then Y is swapped with X (the **green** `exs` rule). The second exchange is necessary for completing the derivation. This means that we are allowed to move the atom X past the atom Y (initially inside the composite formula $\mid \otimes Y$) and subsequently change our mind and swap the positions of X and Y again. The construction of two such distinct focused derivations with the same underlying permutation of atoms (in this case the identity permutation fixing X and Y) is possible since the atom Y is wrapped in the composite formula $\mid \otimes Y$, whose leftmost formula \mid is *closed*, i.e. free of atoms: in the right derivation in (6), when $\mid \otimes Y$ is moved to the stoup, after an application of the rule $\otimes\text{L}$ we have the possibility of swapping Y and X again and subsequently the unit \mid in the stoup is removed with an application of IL . This “change of mind”, in which we apply the rule `exs` on the same atom multiple times, is only possible in the skew case, the same is not doable in the focused sequent calculus of symmetric monoidal categories (see the next section).

The separation of the context in two parts $\Omega \vdots \Gamma$ in phase C takes inspiration from Chaudhuri and Pfenning’s focused sequent calculus for linear logic [8], and it also appears in the design of focused sequent calculi for right-normal and associative-normal skew monoidal categories [30].

6 Recovering Coherence for the Non-Skew Case

The focused sequent calculus in (5) can be expanded and modified in order to obtain a concrete presentation of the free symmetric monoidal category and recover Mac Lane’s coherence theorem [22]. This is in analogy with Uustalu et al.’s recovery of the coherence theorem for monoidal categories starting from a focused sequent calculus for skew monoidal categories [30].

$$\begin{array}{c}
\frac{S \mid \Omega \vdash_C C}{S \mid \Omega, I \vdash_C C} \text{IC} \quad \frac{S \mid \Omega, A, B \vdash_C C}{S \mid \Omega, A \otimes B \vdash_C C} \otimes\text{C} \quad \frac{S \mid \Omega \vdash_C \Gamma, X, \Delta \vdash_C C}{S \mid \Omega, X \vdash_C \Gamma, \Delta \vdash_C C} \text{exs}_{X, \Gamma} \\
\frac{S \mid \Gamma \vdash_L C \quad S \neq X}{S \mid \vdash_C C} \text{sw}_{\text{LC}} \quad \frac{- \mid \Gamma, X, \Delta \vdash_L C}{X \mid \vdash_C C} \text{exs}_{X, \Gamma}^S \\
\frac{- \mid \Gamma \vdash_L C}{I \mid \Gamma \vdash_L C} \text{IL} \quad \frac{A \mid B \vdash_C C}{A \otimes B \mid \Gamma \vdash_L C} \otimes\text{L} \\
\frac{A \mid \Gamma \vdash_L C}{- \mid A, \Gamma \vdash_L C} \text{pass} \quad \frac{T \mid \Gamma \vdash_R C \quad T = - \rightarrow \Gamma = ()}{T \mid \Gamma \vdash_L C} \text{sw}_{\text{RL}} \\
\frac{}{X \mid \vdash_R X} \text{ax} \quad \frac{}{- \mid \vdash_R I} \text{IR} \\
\frac{T \mid \Gamma \vdash_R A \quad - \mid \Delta \vdash_L B}{T \mid \Gamma, \Delta \vdash_R A \otimes B} \otimes\text{R} \quad \frac{- \mid \vdash_R A \quad X \mid \Delta \vdash_R B}{X \mid \Delta \vdash_R A \otimes B} \otimes\text{R}_2
\end{array} \tag{7}$$

(Condition $S \neq X$ in rule sw_{LC} requires the stoup S to not be an atomic formula. Condition $T = - \rightarrow \Gamma = ()$ in rule sw_{RL} requires the context Γ to be empty whenever the stoup T is empty.)

The differences with the focused sequent calculus in (5) are:

- In phase C, the new rules IC and $\otimes\text{C}$ allow the decomposition of units and tensors in the context, and correspond to adding inverses for ρ and α in the categorical calculus (1) (see Sects. 3.2 and 3.3 of [30] for a discussion on this correspondence). The modified exs rule now acts only on atomic formulae. This implies that each sequent $S \mid \Omega \vdash_C C$ appearing in the derivation of a valid sequent $S_0 \mid \Omega_0 \vdash_C C_0$ has the context Γ consisting only of atomic formulae. Once all atoms have been moved using exs and the antecedent is of the form $\vdash_C \Gamma$, we check whether the formula in the stoup is an atom: if it is, we move it in the context using the new rule exs^S , otherwise we switch to phase L using the rule sw_{LC} as before.
- Rules IL, $\otimes\text{L}$ and pass in phase L are unchanged, but the condition in sw_{RL} for switching from phase L to phase R is more stringent: if the stoup is empty, we are allowed to switch phase only when the context is empty as well. This restriction forces all formulae in the context to be reduced to atoms using the rules IC, $\otimes\text{C}$, IL and $\otimes\text{L}$. In particular, in a successful derivation of a sequent $S_0 \mid \Omega_0 \vdash_C C_0$, it is possible to switch to phase R only if the antecedent is completely empty or all the formulae in the antecedents are atoms and the stoup is non-empty.
- Phase R contains a new rule $\otimes\text{R}_2$, which allows to send the atom in the stoup to the second premise, provided that all of the context is also sent to the second premise (so the antecedent of the first premise is left completely empty). The rule $\otimes\text{R}_2$, together with the restriction in sw_{RL} discussed above, correspond to adding an inverse for λ in the categorical calculus (1) (as explained in Sect. 3.1 of [30]).

The “change of mind” of the focused sequent calculus for symmetric skew monoidal categories, exemplified in the derivation on the right in (6), where the

showed how to extend the focused sequent calculus in order to obtain a calculus of normal forms for symmetric monoidal categories and retrieve Mac Lane’s coherence theorem using techniques from structural proof theory.

The story narrated in Sects. 3 and 4 applies with minor modifications to the more general case of braided skew monoidal categories, which we also formalized in Agda. The free braided skew monoidal category on a set of atoms has the same objects of $\mathbf{Fssk}(\mathbf{At})$, but the grammar of derivations (1) is augmented with a new rule s^{-1} inverse (up to \doteq) of s . A sequent calculus for braided skew monoidal categories is obtained by including in the proof system (2) a new rule $\text{ex}_{A,B}^{-1}$ inverse (up to \doteq) of $\text{ex}_{A,B}$. As in the symmetric case, one can define effective procedures `cmplt` and `sound` translating between categorical calculus and sequent calculus, preserving the congruences \doteq and $\overset{\circ}{\doteq}$, and exhibiting a bijection between the two deductive systems. However, the construction of a focused subsystem in the braided case does not seem obtainable as a simple generalization of the focused sequent calculus of Sect. 5. This complication is related to the representation of normal forms for braids in Artin braid groups, which is more convoluted than the representation of normal forms in symmetric groups [11, 14]. The rule `exs` in (5) outlines an algorithm for constructing permutations of formulae in the context: starting from a context $\Omega \vdash \cdot$, repeated applications of `exs` construct a context $\vdash \Gamma$, where Γ is a permutation of Ω , and all permutations can be represented in this way. A focused sequent calculus for the braided case will instead construct an action of the braid group on the formulae in the context, but such actions admit normal forms which are more arduous to formalize. After properly sorting out the proof theory of braided skew monoidal categories, it would be interesting to also understand the relationship between the resulting deductive systems and existing calculi for categories with braided structure [13, 15, 23].

In Sect. 6, we introduced a focused sequent calculus for normal symmetric monoidal categories and retrieve Mac Lane-style coherence. We plan to understand the relation between the latter proof system and Shulman’s practical type theory for symmetric monoidal categories [25]. We also plan to investigate the proof theory of *partially* normal symmetric skew monoidal categories in the style of [30], when one or both structural laws among ρ and α are invertible (but not λ , whose invertibility implies the invertibility of all structural laws [6]).

Finally, we want to study deductive systems for (symmetric) skew monoidal closed categories, which will include a linear implication \multimap as in the proof systems for skew prounital closed categories [31]. The resulting calculi should allow the representation of interesting classes of concurrent computations, a skew variant of the concurrent logical framework of Watkins et al. [32].

References

1. Altenkirch, A., Chapman, J., Uustalu, T.: Monads need not be endofunctors. *Log. Methods Comput. Sci.* **11**(1), Article 3 (2015). [https://doi.org/10.2168/lmcs-11\(1:3\)](https://doi.org/10.2168/lmcs-11(1:3))
2. Andreoli, J.-M.: Logic programming with focusing proofs in linear logic. *J. Log. Comput.* **2**(3), 297–347 (1992). <https://doi.org/10.1093/logcom/2.3.297>

3. Bourke, J.: Skew structures in 2-category theory and homotopy theory. *J. Homotopy Relat. Struct.* **12**(1), 31–81 (2015). <https://doi.org/10.1007/s40062-015-0121-z>
4. Bourke, J., Lack, S.: Free skew monoidal categories. *J. Pure Appl. Alg.* **222**(10), 3255–3281 (2018). <https://doi.org/10.1016/j.jpaa.2017.12.006>
5. Bourke, J., Lack, S.: Skew monoidal categories and skew multicategories. *J. Alg.* **506**, 237–266 (2018). <https://doi.org/10.1016/j.jalgebra.2018.02.039>
6. Bourke, J., Lack, S.: Braided skew monoidal categories. *Theor. Appl. Categ.* **35**(2), 19–63 (2020). <http://www.tac.mta.ca/tac/volumes/35/2/35-02abs.html>
7. Buckley, M., Garner, R., Lack, S., Street, R.: The Catalan simplicial set. *Math. Proc. Cambridge Philos. Soc.* **158**(12), 211–222 (2014). <https://doi.org/10.1017/s0305004114000498>
8. Chaudhuri, K., Pfenning, F.: Focusing the inverse method for linear logic. In: Ong, L. (ed.) *CSL 2005*. LNCS, vol. 3634, pp. 200–215. Springer, Heidelberg (2005). https://doi.org/10.1007/11538363_15
9. Curien, P.L., Obradović, J., Ivanović, J.: Syntactic aspects of hypergraph polytopes. *J. Homotopy Relat. Struct.* **14**(1), 235–279 (2019). <https://doi.org/10.1007/s40062-018-0211-9>
10. Davydov, A., Runkel, I.: An Alternative Description of Braided Monoidal Categories. *Appl. Categ. Struct.* **23**(3), 279–309 (2013). <https://doi.org/10.1007/s10485-013-9338-3>
11. Dehornoy, P.: Efficient solutions to the braid isotopy problem. *Discret. Appl. Math.* **156**(16), 3091–3112 (2008). <https://doi.org/10.1016/j.dam.2007.12.009>
12. Dosen, K., Petrić, Z.: *Proof-Theoretical Coherence*. King’s College Publications (2004)
13. Fleury, A.: Ribbon braided multiplicative linear logic. *Mat. Contemp.* **24**, 39–70 (2003). <https://www.mat.unb.br/~matcont/24.3.pdf>
14. Garside, F.A.: The braid group and other groups. *Quart. J. Math. Oxford* **20**(1), 235–254 (1969). <https://doi.org/10.1093/qmath/20.1.235>
15. Hasegawa, M.: A braided lambda calculus. Paper presented at Joint Workshop on Linearity & TLLA 2020. https://www.cs.unibo.it/~dallago/TLLALINEARITY2020/A_Braided_Lambda_Calculus.pdf
16. Hyland, M., de Paiva, V.: Full intuitionistic linear logic (extended abstract). *Ann. Pure Appl. Log.* **64**(3), 273–291 (1993). [https://doi.org/10.1016/0168-0072\(93\)90146-5](https://doi.org/10.1016/0168-0072(93)90146-5)
17. Joyal, A., Street, R.: Braided tensor categories. *Adv. Math.* **102**(1), 20–78 (1993). <https://doi.org/10.1006/aima.1993.1055>
18. Kelly, G.M.: On Mac Lane’s conditions for coherence of natural associativities, commutativities, etc. *J. Alg.* **1**(4), 397–402 (1964). [https://doi.org/10.1016/0021-8693\(64\)90018-3](https://doi.org/10.1016/0021-8693(64)90018-3)
19. Lack, S., Street, R.: Skew monoidales, skew warpings and quantum categories. *Theor. Appl. Categ.* **26**, 385–402 (2012). <http://www.tac.mta.ca/tac/volumes/26/15/26-15abs.html>
20. Lack, S., Street, R.: Triangulations, orientals, and skew monoidal categories. *Adv. Math.* **258**, 351–396 (2014). <https://doi.org/10.1016/j.aim.2014.03.003>
21. Lambek, J.: Deductive systems and categories I: syntactic calculus and residuated categories. *Math. Syst. Theory* **2**(4), 287–318 (1968). <https://doi.org/10.1007/bf01703261>
22. Mac Lane, S.: Natural associativity and commutativity. *Rice Univ. Stud.* **49**(4), 28–46 (1963). <http://hdl.handle.net/1911/62865>

23. Mellies, P.-A.: Ribbon tensorial logic. In: Dawar, A., Grädel, E. (eds.) Proceedings of 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, pp. 689–698. ACM (2018). <https://doi.org/10.1145/3209108.3209129>
24. Mints, G.E.: Closed categories and the theory of proofs, Zap. Nauchn. Sem. LOMI **68**, 83–114 (1977). (in Russian). Translated in 1981 in J. Sov. Math. **15**, pp. 45–62. <https://doi.org/10.1007/bf01404107>. Reprinted in 1992 in Selected Papers in Proof Theory, Studies in Proof Theory **3**, 183–212. Bibliopolis/North-Holland
25. Shulman, M.: A practical type theory for symmetric monoidal categories. arXiv eprint 1911.00818 (2019). <https://arxiv.org/abs/1911.00818>
26. Szabo, M.E.: Algebra of Proofs. Studies in Logic and the Foundations of Mathematics 88. North-Holland (1978)
27. Szlachányi, K.: Skew-monoidal categories and bialgebroids. Adv. Math. **231**(3–4), 1694–1730 (2012). <https://doi.org/10.1016/j.aim.2012.06.027>
28. Uustalu, T.: Coherence for skew-monoidal categories. In: Levy, P., Krishnaswami, N. (eds.). Proceedings of 5th Workshop on Mathematically Structured Programming, MSFP 2014, Electronic Proceedings in Theoretical Computer Science, vol. 153, pp. 68–77. Open Publishing Assoc. (2014). <https://doi.org/10.4204/eptcs.153.5>
29. Uustalu, T., Veltri, N., Zeilberger, N.: The sequent calculus of skew monoidal categories. In: Casadio, C., Scott, P.J. (eds.) Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics. OCL, vol. 20, pp. 377–406. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-66545-6_11
30. Uustalu, T., Veltri, N., Zeilberger, N.: Proof theory of partially normal skew monoidal categories. In: Spivak, D. I., Vicary, J. (eds.). Proceedings of 3rd Applied Category Theory Conference, ACT 2020. Electronic Proceedings in Theoretical Computer Science, vol. 333, pp. 230–246. Open Publishing Association (2020). <https://doi.org/10.4204/eptcs.333.16>
31. Uustalu, T., Veltri, N., Zeilberger, N.: Deductive systems and coherence for skew prounital closed categories. In: Sacerdoti Coen, C., Tiu, A. (eds.). Proceedings of 15th International Workshop on Logical Frameworks and Metalanguages: Theory and Practice, LFMTP 2020. Electronic Proceedings in Theoretical Computer Science, vol. 332, pp. 35–53. Open Publishing Association (2020). <https://doi.org/10.4204/eptcs.332.3>
32. Watkins, K., Cervesato, I., Pfenning, F., Walker, D.: A concurrent logical framework: the propositional fragment. In: Berardi, S., Coppo, M., Damiani, F. (eds.) TYPES 2003. LNCS, vol. 3085, pp. 355–377. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24849-1_23
33. Zeilberger, N.: A sequent calculus for a semi-associative law. Log. Methods Comput. Sci. **15**(1), Article 9 (2019). [https://doi.org/10.23638/lmcs-15\(1:9\)2019](https://doi.org/10.23638/lmcs-15(1:9)2019)



On the Subtle Nature of a Simple Logic of the Hide and Seek Game

Dazhu Li^{1,2}, Sujata Ghosh^{3(✉)}, Fenrong Liu^{1,2}, and Yaxin Tu⁴

¹ Department of Philosophy, Tsinghua University, Beijing, China

² ILLC, University of Amsterdam, Amsterdam, The Netherlands

³ Indian Statistical Institute, Chennai, India

sujata@isichennai.res.in

⁴ Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Abstract. We discuss a simple logic to describe one of our favourite games from childhood, hide and seek, and show how a simple addition of an equality constant to describe the winning condition of the seeker makes our logic undecidable. There are certain decidable fragments of first-order logic which behave in a similar fashion and we add a new modal variant to that class of logics. We also discuss the relative expressive power of the proposed logic in comparison to the standard modal counterparts.

1 From Games to Logic

Everyone remembers the pleasure of playing hide and seek in her or his childhood. After calling out “I am ready, you can come to find me”, the fun part is to stay at your secret spot, not making any noise, and to expect that the other player would not discover you. Once you are found, the other wins. Let us consider a two-player setting, use E to denote the hider, and A the seeker. Following the research program of [7], the game of hide and seek is naturally seen as a graph game, where A and E are located at two different nodes, and are allowed to move around. The goal of A is to meet E , while the goal of E is to avoid A . For the game that many of us played in childhood, the player E (one who hides) basically stays at one place, whereas player A (one who seeks) moves from one node to another. We can describe such graph games using the basic modal logic. However, if we consider a simple modification by allowing moves for both the players (akin to the game of cops and robber [23]), the setting becomes quite diverse. On one hand, these graph games are natural candidates for modelling computational search problems, on the other hand, the nuanced interaction between the players playing hide and seek is a showcase of interactive players having their goals entangled, which is a popular phenomenon in social networks. In other words, the graph game of hide and seek provides us with an ideal arena where we can study reasoning about social interaction and challenges therein arising from such intertwined objectives of players. In the following we

will make these games more precise and provide a language to express strategic reasoning and winning conditions of players.

However, before going into the logic details, let us first get a feel about the hide and seek game regarding the information available to the players. That will also lead us to understand the kind of reasoning that we plan to explore for such games. Essentially, it is an imperfect information game where the seeker is not aware of the position of hider, whereas the hider may or may not know the exact position of the seeker. Both the players know the game graph where they move about and are aware of their own positions and moves. Now, the modification that we talk about makes the setting even more interesting information-wise, as then we can consider different levels of information available to both the players. However, to keep things simple we start off from a high-level modeller's perspective, that is, we reason *about* such games. Thus, we reason about players' observations and moves with the assumption that the whole graph and the players' positions at each stage of the game are available to us. We leave the players' perspectives for future work.

Coming back to the game proper, we have the two players located at two different nodes. To model their moves we consider a pair of states as an evaluation point rather than a single state in a Kripke model (a pointed model), and consider distinct modalities to express the moves of the players. The evaluation of these two different modalities, one for each player, can then be assessed coordinate-wise with respect to the pair of states. In addition, a winning condition for the hide and seek game corresponding to the seeker finding the hider can be modelled by considering a pair of states whose first and second elements are the same. This basically gives us the identity relation which can be expressed by introducing a special identity proposition. We first note that using standard modal logic arguments, one can show the decidability of the satisfaction problem of the two-dimensional modal logic mentioned above, without the special proposition. Interestingly enough, such a simple addition, viz. incorporating the identity proposition, transforms a decidable modal logic into an undecidable one. In fact, there are various elegant examples of logics that suggest that taking this identity relation into account may change previously decidable logics (without equality) into undecidable ones, e.g., the Gödel class of first-order formulas with identity (cf. [16]). A more recent example is the logic of functional dependence with function symbols (see [4] and [24]). We add one more logic to this class, and that constitutes the main technical result of this paper. This result also refutes a claim mentioned in [7] which stated that the extended logic with the identity proposition will remain decidable. The related notion of expressive power of the proposed logic is also discussed here.

We finally note that this modified version of hide and seek game played on graphs is a special case of cops and robber game [23], a classic pursuit-evasion game played on graphs, where several cops attempt to catch a robber. The hide and seek game corresponds to the game having a single cop chasing a robber. Thus, this study opens up the possibility of a logical analysis of these cops and robber games with all their generality (cf. [23]) which have been well-studied

from algorithmic and combinatorial perspectives. We are currently working on this idea and exploring an extension of the logic proposed here with modal substitution operators [26].

In Sect. 2 we introduce a logic (LHS) to reason about plays and winning conditions in the hide and seek game. Section 3 deals with the relative expressive power of the language and relevant notions of bisimulation are introduced to facilitate the discussion. Section 4 gives the main result of this work, viz. the satisfaction problem of LHS is undecidable. Section 5 provides a discussion on related work, and Sect. 6 gives pointers to further research.

2 Logic of Hide and Seek (LHS)

Let us first introduce a logic to describe the game of hide and seek, LHS, followed by an informal discussion about the expressivity of the proposed logic.

Definition 1 (Language). *Let P_E denote a countable set of propositional variables for player E , and P_A for player A . The two dimensional modal language is given as follows:*

$$\varphi ::= p_A \mid p_E \mid I \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle \text{left} \rangle \varphi \mid \langle \text{right} \rangle \varphi$$

where $p_E \in P_E$, $p_A \in P_A$, and I is a propositional constant. Other Boolean connectives are defined in the usual way, and so are the corresponding box modalities $[\text{left}]$ and $[\text{right}]$.

Without loss of generality, the modal operator representing player E 's moves is given by $\langle \text{left} \rangle$ and that representing A 's moves is given by $\langle \text{right} \rangle$. Formulas are evaluated in *standard relational models* $\mathbf{M} = (W, R, V)$, where W is a non-empty set of vertices, $R \subseteq W \times W$ is a set of edges, and $V : P_E \cup P_A \rightarrow 2^W$ is a valuation function. Moreover, for any $s, t \in W$, we call (\mathbf{M}, s, t) a *pointed graph model for two players* (for simplicity, *graph model*): intuitively, s and t represent respectively the positions of players E and A . To simplify notations, we also employ \mathbf{M}, s, t for (\mathbf{M}, s, t) . Semantics for LHS is given by the following:

Definition 2 (Semantics). *Let $\mathbf{M} = (W, R, V)$ be a model and $s, t \in W$. Truth of formulas φ at the graph model (\mathbf{M}, s, t) , written as $\mathbf{M}, s, t \models \varphi$, is defined recursively as follows:*

$$\begin{aligned} \mathbf{M}, s, t \models p_E &\Leftrightarrow s \in V(p_E) \\ \mathbf{M}, s, t \models p_A &\Leftrightarrow t \in V(p_A) \\ \mathbf{M}, s, t \models I &\Leftrightarrow s = t \\ \mathbf{M}, s, t \models \neg\varphi &\Leftrightarrow \mathbf{M}, s, t \not\models \varphi \\ \mathbf{M}, s, t \models \varphi \wedge \psi &\Leftrightarrow \mathbf{M}, s, t \models \varphi \text{ and } \mathbf{M}, s, t \models \psi \\ \mathbf{M}, s, t \models \langle \text{left} \rangle \varphi &\Leftrightarrow \exists s' \in W \text{ s.t. } Rss' \text{ and } \mathbf{M}, s', t \models \varphi \\ \mathbf{M}, s, t \models \langle \text{right} \rangle \varphi &\Leftrightarrow \exists t' \in W \text{ s.t. } Rtt' \text{ and } \mathbf{M}, s, t' \models \varphi \end{aligned}$$

As mentioned earlier, the above language has two modalities, one for each player, viz. $\langle \text{left} \rangle$ for player E and $\langle \text{right} \rangle$ for player A . Accordingly, all the formulas are evaluated in a graph model. The constant I denotes the identity relation in a game graph to describe the meeting of two players, signifying the fact that the seeker has found the hider. Let us denote LHS_{-I} to be the logic LHS without the constant I .

Here are some useful notions. Given a model \mathbf{M} and a set $U \subseteq W$ of states, define $R(U) := \{t \in W \mid \text{there is } s \in U \text{ with } Rst\}$, denoting the set of successors of the points in U . For simplicity, we usually write $R(s)$ for $R(\{s\})$ when U is a singleton $\{s\}$. We can introduce the logical notions such as *satisfiability* and *modal equivalence* in the usual way, and we will omit the details here.

Going back to the hide and seek game itself, one can consider different variants played on the game graph model, e.g., the players can move simultaneously or sequentially. In a sequential play, one can also consider different orders of play. In this paper, we assume that the players move sequentially, and that the hider E starts the game. Local one-step winning positions (pairs of states describing the current positions of the players) for each player can be expressed in our language as follows:

- $E : \langle \text{left} \rangle [\text{right}] \neg I$
- $A : [\text{left}] \langle \text{right} \rangle I$

More generally, winning positions for E and A can be described as:

- $E : \forall n (\langle \text{left} \rangle [\text{right}])^n \neg I$
- $A : \exists n ([\text{left}] \langle \text{right} \rangle)^n I$

Note that the above conditions involve countable conjunction/disjunction of finite iterations of interactions between two players. The interactions $\langle \text{left} \rangle [\text{right}] / [\text{left}] \langle \text{right} \rangle$ are expressed with two separate modalities, but they are considered as a single unit. These are not expressible in our language. As mentioned in the introduction, we are currently exploring an extension of this language with modal substitution operators which would also provide a finitary way to express such countable boolean operations.

Remark 1. There are other ways to give suitable logics capturing the hide and seek game. For instance, one can replace identity constant I with C , denoting ‘catching’: $\mathbf{M}, s, t \models C$ iff $R(s) \subseteq R(t)$. From the perspective of the game, constant C describes that all states accessible to the hider are accessible to the seeker as well. In contrast to I which states that the seeker has already won, C indicates that she can win in the next round. They amount to the same condition for games of perfect information: if the seeker has the ability to meet the hider she will actually do that, if she is rational. However, from a logical perspective, their interpretations are entirely different, leading to distinct expressive features. For an illustration, let us note that C can be defined as $[\text{left}] \langle \text{right} \rangle I$ in LHS, but I is not definable in the logic extending LHS_{-I} with C . The constant proposition C with the given interpretation is also useful in describing cop-win graphs in the cops and robber game involving a single cop [23], see more details in [26].

In the next two sections we will explore some logical properties of LHS regarding its expressiveness on one hand and satisfiability on the other hand.

3 Bisimulation and Expressive Power

The notion of bisimulation is an important tool for studying the expressive power of modal logics. We are now going to explore a suitable notion tailored to our logic. We usually need to be careful when introducing the conditions: on one hand, the definition should ensure that the logic cannot distinguish bisimilar models (i.e., the desired notion is strong enough), but on the other hand, it should also hold between two models whenever they cannot be distinguished by the logical language (thus, it is weak enough). In what follows, we take the standard bisimulation [9] as the benchmark and investigate the relations between expressiveness of basic modal logic \mathbf{M} , LHS_{-I} and LHS. Let us start by comparing that for LHS_{-I} and \mathbf{M} .

The standard bisimulation, denoted by \leftrightarrow^s , provides us a semantic characterization of the expressiveness of the basic modal language. And at a first glance, the semantic design of logic LHS_{-I} is similar to that of the basic modal logic, except that we now need to consider two states simultaneously when evaluating formulas. So, is logic LHS_{-I} invariant under the standard notion? First, we provide a positive answer in the following sense:

Proposition 1. *If $(\mathbf{M}, w) \leftrightarrow^s (\mathbf{M}', w')$ and $(\mathbf{M}, v) \leftrightarrow^s (\mathbf{M}', v')$, then (\mathbf{M}, w, v) and (\mathbf{M}', w', v') satisfy the same formulas of LHS_{-I} .*

Proof. The proof is straightforward by applying induction on formulas of LHS_{-I} . We leave the details to the reader. □

Therefore, the standard bisimulation is strong enough to measure the expressive power of LHS_{-I} . But meanwhile, to behave properly, is it also weak enough? Unfortunately, we have the following negative result:

Proposition 2. *There are (\mathbf{M}, w, v) and (\mathbf{M}', w', v') s.t. they satisfy the same LHS_{-I} -formulas but at least one of $(\mathbf{M}, w) \leftrightarrow^s (\mathbf{M}', w')$, $(\mathbf{M}, v) \leftrightarrow^s (\mathbf{M}', v')$ may not hold.¹*

Proof. It suffices to give a counterexample. Consider the models \mathbf{M} and \mathbf{M}' depicted in Fig. 1. It holds that (\mathbf{M}, w_1, w_2) and (\mathbf{M}', v_1, v_2) satisfy the same LHS-formulas, but we do not have $(\mathbf{M}, w_1) \leftrightarrow^s (\mathbf{M}', v_1)$. □

¹ Strictly speaking, a negative result holds even for the basic modal logic (see [9]). However, it is still ideal if the notion of bisimulation can behave well in a large class of models (e.g., *image-finite models*). This is also one of our guiding spirits. But, as illustrated by the counterexample used to show the result, the standard notion even excludes situations that are very simple but cannot be distinguished by LHS_{-I} .

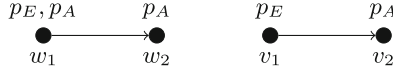


Fig. 1. Two graph models (\mathbf{M}, w_1, w_2) and (\mathbf{M}', v_1, v_2) satisfying same LHS-formulas.

Intuitively, the failure originates from the ‘evaluation-gap’ between the two worlds in our graph models (\mathbf{M}, s, t) : when considering atomic properties of s , both LHS_{-I} and LHS can only describe those in P_E , but not the ones in P_A .²

Now, it is time to introduce the notion of bisimulation for LHS, from which we can easily obtain that for LHS_{-I} . Here is the definition:

Definition 3 (Bisimulation for LHS models). *Let $\mathbf{M} = (W, R, V)$, $\mathbf{M}' = (W', R', V')$ be two models and let $s, t \in W$ and $s', t' \in W'$. We say, (\mathbf{M}, s, t) is bisimilar to (\mathbf{M}', s', t') (denoted by $(\mathbf{M}, s, t) \Leftrightarrow (\mathbf{M}', s', t')$) if*

Atom: (\mathbf{M}, s, t) and (\mathbf{M}', s', t') satisfy the same propositional letters.

Meet: $s = t$ iff $s' = t'$.

Zig_{left}: if there exists $u \in W$ such that Rsu , then there exists $u' \in W'$ such that $R's'u'$ and $(\mathbf{M}, u, t) \Leftrightarrow (\mathbf{M}', u', t')$.

Zig_{right}: if there exists $v \in W$ such that Rtv , then there exists $v' \in W'$ such that $R't'v'$ and $(\mathbf{M}, s, v) \Leftrightarrow (\mathbf{M}', s', v')$.

Zag_{left}, Zag_{right}: those analogous clauses in the converse direction of **Zig_{left}** and **Zig_{right}** respectively.³

With this definition, it is now easy to check that (\mathbf{M}, w_1, w_2) and (\mathbf{M}', v_1, v_2) in Fig. 1 are bisimilar. Although the clauses above look rather routine, it is instructive to notice some subtle aspects of the definition that are in line with our previous observation: the condition **Atom** in effect just requires that $V(s) \cap \text{P}_E = V'(s') \cap \text{P}_E$ and $V(t) \cap \text{P}_A = V'(t') \cap \text{P}_A$, but s and s' may satisfy different properties p_A and p'_A , say, from P_A , and t and t' may satisfy different properties p_E and p'_E , say, from P_E . Moreover, the clause **Meet** aims to deal with the constant I , and the others are analogous to the zigzag conditions in standard situations.

By dropping the clause **Meet** above, we get the notion for LHS_{-I} , and by $(\mathbf{M}, s, t) \Leftrightarrow^- (\mathbf{M}', s', t')$ we denote the case that (\mathbf{M}, s, t) and (\mathbf{M}', s', t') are LHS_{-I} -bisimilar. With Definition 3, it holds that:

² From the perspective of games, the evaluation-gap suggests a way to handle situations where the two players have different observations even when they are at the same position. For example, the gap might allow us to consider further enrichments so that the states in the playing arena can encode different properties for the players: a crowded street reducing the possible moves of the escaping robber is helpful for a chasing cop, meanwhile, it is definitely a disaster to the robber.

³ One may also like to treat LHS as a product logic over models containing two binary relations R_{left} and R_{right} on domain $W \times W$, and then explore expressive power or other properties of LHS with respect to the new setting. We leave a systematic study of relations between our logic and existing combined logics for future inquiry.

Proposition 3. *If $(\mathbf{M}, s, t) \Leftrightarrow (\mathbf{M}', s', t')$, then (\mathbf{M}, s, t) and (\mathbf{M}', s', t') satisfy the same LHS-formulas. Also, if $(\mathbf{M}, s, t) \Leftrightarrow^- (\mathbf{M}', s', t')$, then they satisfy the same LHS $_{-I}$ -formulas.*

It can be proved by induction on the structure of LHS-formulas. Therefore, the language cannot distinguish between bisimilar models. However, our previous discussion indicates that having a very strong notion is never the final goal: it is equally important to ask whether the notion is also weak enough. This time we are going to present a positive result w.r.t. a class of models that are LHS-saturated:

Definition 4 (LHS-saturation). *A model $\mathbf{M} = (W, R, V)$ is said to be LHS-saturated, if for any set Φ of formulas and states $w, v \in W$, it holds that:*

- *If Φ is finitely satisfiable in $R(w) \times \{v\}$, then the whole set Φ is satisfiable in $R(w) \times \{v\}$, and*
- *If Φ is finitely satisfiable in $\{w\} \times R(v)$, then the whole set Φ is satisfiable in $\{w\} \times R(v)$.*

The notion is essentially obtained by adapting the so-called *m-saturation* [9] to fit into our logics. As usual, any finite model is LHS-saturated. Furthermore, in terms of infinite \mathbf{M} , it intuitively requires that \mathbf{M} contains ‘enough’ states: for instance, if every finite subset of Φ can be satisfied by some pairs in $R(w) \times \{v\}$, then there must also be a pair satisfying Φ itself. By restricting Φ to the fragment without I , we have a notion for LHS $_{-I}$, called LHS $_{-I}$ -saturation. Now we have enough background to show that:

Proposition 4. *For all \mathbf{M} and \mathbf{M}' that are LHS-saturated, if (\mathbf{M}, s, t) and (\mathbf{M}', s', t') satisfy the same formulas of LHS, then it holds that $(\mathbf{M}, s, t) \Leftrightarrow (\mathbf{M}', s', t')$. Moreover, when \mathbf{M} and \mathbf{M}' are LHS $_{-I}$ -saturated, if (\mathbf{M}, s, t) and (\mathbf{M}', s', t') satisfy the same formulas of LHS $_{-I}$, then it holds that $(\mathbf{M}, s, t) \Leftrightarrow^- (\mathbf{M}', s', t')$.*

It can be proved by showing that the modal equivalence relation itself is a bisimulation, but due to the page-limit constraints, details are omitted. Therefore, just as the usual case, w.r.t. the class of models that are LHS/LHS $_{-I}$ -saturated, our notion of bisimulation coincides with the corresponding notion of modal equivalence.

Having shown that our novel notions behave well, we end this section with the following result concerning the relations among aforementioned varieties of bisimulations:

Proposition 5. *With respect to the three varieties of bisimulations \Leftrightarrow^s , \Leftrightarrow and \Leftrightarrow^- , we have the following:*

- (1) *Both \Leftrightarrow^s and \Leftrightarrow are strictly stronger than \Leftrightarrow^- : \Leftrightarrow^s entails \Leftrightarrow^- and \Leftrightarrow entails \Leftrightarrow^- , but the converse directions do not hold.*
- (2) *\Leftrightarrow^s and \Leftrightarrow are incomparable: they do not entail each other.*

Proof. We show the two claims one by one.

(1) The relation between \leftrightarrow^s and \leftrightarrow^- follows from Proposition 1, 2 and 4. Also, it is obvious that \leftrightarrow is stronger than \leftrightarrow^- . For an example, consider the two models given in Fig. 2: it holds $(\mathbf{M}, w_1, w_1) \leftrightarrow^- (\mathbf{M}', v_1, v_1)$, but $\mathbf{M}, w_1, w_1 \models \langle \text{left} \rangle \langle \text{right} \rangle \neg I$ and $\mathbf{M}', v_1, v_1 \not\models \langle \text{left} \rangle \langle \text{right} \rangle \neg I$. Now, by Proposition 3, we do not have $(\mathbf{M}, w_1, w_1) \leftrightarrow (\mathbf{M}', v_1, v_1)$.

(2) Consider the models in Fig. 2. It is not hard to see that the states w_1 and v_1 cannot be distinguished by the basic modal language, but this would not be the case when we consider the logic LHS. Thus, standard bisimulations need not be bisimulations of LHS. On the other hand, using the models in Fig. 1, it is not hard to see that bisimulations of LHS may also be excluded by the notion of standard bisimulation. This completes the proof. \square

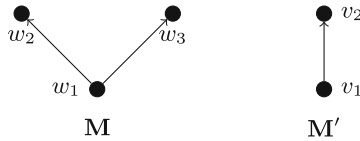


Fig. 2. $(\mathbf{M}, w_1, w_1) \leftrightarrow^- (\mathbf{M}', v_1, v_1)$, but not $(\mathbf{M}, w_1, w_1) \leftrightarrow (\mathbf{M}', v_1, v_1)$.

Properties of LHS- and LHS_{-I}-bisimulation explored here are very basic, and several further questions are worth studying. For instance,

Open Problem. What is the computational complexity of checking for bisimulation of LHS or LHS_{-I}? Are they as complex as each other?

4 Towards Undecidability of the Satisfaction Problem

Essentially, LHS introduces a propositional constant to deal with equality in a modal logic framework. This universally accepted relation of indiscernibility is simple in nature. However, as we mentioned in Sect. 1, there are various elegant examples of logics that suggest that taking this relation into account may change previously decidable logics (without equality) into undecidable ones. In this section, we are going to contribute one more instance to this class: in what follows, we first show that LHS does not have the tree model property or the finite model property, and then prove that the satisfiability problem for LHS is undecidable.

Usually, the tree model property and the finite model property are positive signals for the computational behaviors of a logic (cf. e.g., [9]). However, in what follows, we will show that our logic LHS lacks both the properties. Let us begin with a simple result concerning the tree model property:

Proposition 6. *The logic LHS does not have the tree model property.*

Proof. Consider the following formula:

$$\varphi_r := I \wedge \langle \text{left} \rangle \top \wedge [\text{left}] I$$

It is easy to see that it is satisfiable. Also, let $\mathbf{M} = (W, R, V)$ and $u, v \in W$ such that $\mathbf{M}, u, v \models \varphi_r$. From I it follows that $u = v$. Also, the conjunct $\langle \text{left} \rangle \top$ indicates that the state u has successors, i.e., $R(u) \neq \emptyset$. Moreover, for all $s \in R(u)$, we have $s = v$. Therefore, $R(u) = \{u\}$. Consequently, the model \mathbf{M} cannot be a tree. The proof is completed. \square

Moreover, by constructing a ‘spy-point’ [10], i.e., all states that are reachable from u in n -steps can also be reached in one step, we can also establish the following:

Theorem 1. *The logic LHS lacks the finite model property.*

Proof. Let φ_∞ be the conjunction of the following formulas:

- (F1) $I \wedge [\text{left}] \neg I$
- (F2) $\langle \text{left} \rangle [\text{left}] \perp$
- (F3) $[\text{left}] \langle \text{right} \rangle (\neg I \wedge \langle \text{right} \rangle \top \wedge [\text{right}] I)$

Let us briefly comment on the intuition underlying these formulas. First, (F1) shows that the two states in the current graph model are the same and the point is irreflexive. Also, formula (F2) states that the point can reach a state that is a dead end having no successors. Additionally, the last formula, motivated by [20], indicates that the point has more than one successor and for all its successors i , there is also another successor j of i such that i has j as its only successor.

After presenting the basic ideas of those formulas, we show that the formula φ_∞ is satisfiable. Consider the model $\mathbf{M}_\infty = (W, R, V)$ that is defined as follows:

- $W := \{s\} \cup \mathbb{N}$
- $R := \{\langle s, i \rangle \mid i \in \mathbb{N}\} \cup \{\langle i + 1, i \rangle \mid i \in \mathbb{N}\}$
- For all $p \in P_A \cup P_E$, $V(p) := \emptyset$.

See Fig. 3 for an illustration. By construction, it can be easily checked that the formula holds at (s, s) , i.e., $\mathbf{M}_\infty, s, s \models \varphi_\infty$.

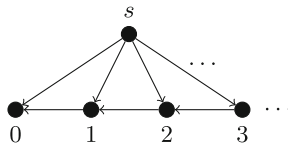


Fig. 3. The model \mathbf{M}_∞ .

Next, let $\mathbf{M} = (W, R, V)$ be an arbitrary model such that $u \in W$ and $\mathbf{M}, u, u \models \varphi_\infty$. We are going to show that W is infinite. To do so, we claim that the model contains the following sequence of states of \mathbf{M} :

$$w_0, w_1, w_2, w_3, w_4, \dots$$

such that for all $i \in \mathbb{N}$, the following conditions hold:

- P1. $\mathbf{M}, w_i, w_{i+1} \models \neg I \wedge \langle \text{right} \rangle \top \wedge [\text{right}] I$
- P2. $\langle u, w_i \rangle \in R$
- P3. $R(w_0) = \emptyset$, and for $1 \leq i$, $R(w_i) = \{w_{i-1}\}$

By making an induction on $i \in \mathbb{N}$, we show that there is always such a sequence of those w'_i s.

First, let us consider the basic case that $i = 0$. As $\mathbf{M}, u, u \models (F2)$, we know that there is $w_0 \in W$ such that Ruw_0 and $\mathbf{M}, w_0, u \models [\text{left}] \perp$. Therefore, $R(w_0) = \emptyset$, i.e., we have already obtained the dead end. Moreover, by formula (F3), it holds $\mathbf{M}, w_0, u \models \langle \text{right} \rangle (\neg I \wedge \langle \text{right} \rangle \top \wedge [\text{right}] I)$. Therefore, there exists $w_1 \in W$ such that Ruw_1 , $w_0 \neq w_1$ and $R(w_1) = \{w_0\}$. Now, it is not hard to see that the clauses P1-P3 hold for both w_0 and w_1 .

Now, suppose that we have already had all those states $w_{i \leq n}$, and we proceed to show that there exists w_{n+1} satisfying the conditions P1-P3. By the induction hypothesis, we have Ruw_n . Now, from the formula (F3), it follows that $\mathbf{M}, w_n, u \models \langle \text{right} \rangle (\neg I \wedge \langle \text{right} \rangle \top \wedge [\text{right}] I)$. So, there is a state $w_{n+1} \in W$ such that Ruw_{n+1} and $\mathbf{M}, w_n, w_{n+1} \models \neg I \wedge \langle \text{right} \rangle \top \wedge [\text{right}] I$. This indicates that w_{n+1} satisfies the requirements P1 and P2. Also, as $\neg I$, $w_n \neq w_{n+1}$. Furthermore, from $\mathbf{M}, w_n, w_{n+1} \models \langle \text{right} \rangle \top \wedge [\text{right}] I$, we know that $R(w_{n+1}) = \{w_n\}$, which indicates that the node satisfies P3 as well.

Moreover, by the property P3, we have $w_i \neq w_j$ whenever $i \neq j$. To be more specific, we have $\mathbf{M}, w_i, w_i \models \langle \text{left} \rangle^i \top \wedge [\text{left}]^{i+1} \perp$ for each i . Therefore, we have infinitely many states w_i . So, the model \mathbf{M} is infinite. The proof is completed. \square

4.1 Undecidability

Now, by encoding the $\mathbb{N} \times \mathbb{N}$ *tiling problem*, we show that the logic LHS is undecidable. A *tile* t is a 1×1 square, of the fixed orientation, with colored edges $right(t)$, $left(t)$, $up(t)$ and $down(t)$. The $\mathbb{N} \times \mathbb{N}$ tiling problem is: given a finite set $T = \{t^1, \dots, t^n\}$ of tile types, is there a function $f : \mathbb{N} \times \mathbb{N} \rightarrow T$ such that $right(f(n, m)) = left(f(n+1, m))$ and $up(f(n, m)) = down(f(n, m+1))$? The problem is known to be undecidable (see [8]).

Theorem 2. *The satisfiability problem of logic LHS is undecidable.*

The proof is given by reduction of the tiling problem to the satisfaction problem of LHS, and is provided in the Appendix. It is worth noting that the proof essentially indicates the undecidability of the class of logics generalizing our framework to capture the games with $2 \leq n \in \mathbb{N}$ players.

A closely related problem is that of model-checking, and it is important to study the complexity of the model-checking problem for our logic. In contrast to the satisfaction problem, we believe that the model checking problem for LHS can be solved efficiently, that is, the problem lies in P.

Finally, it is not hard to see that logic LHS can be translated into first-order logic,⁴ which then suggests that the logic itself is effectively axiomatizable. Consequently, a crucial direction is to explore the following:

Open Problem. Is there a complete proof calculus for the logic LHS?

5 Related Works

Graph Games and Modal Logics. Motivated by a simple graph game of hide and seek, this paper belongs to a broader program [7] that promotes a study of graph game design in tandem with matching new modal logics. In recent years, several interesting new graph games have been studied. For instance, in sabotage games [6], a player moves along a link available to her on a graph to reach some fixed goal region, while her opponent removes an arbitrary link in each round to prevent her from reaching her goal. The games are captured by the sabotage modal logic [3], extending the basic modal logic with a link deletion operator. Further, games in which links are removed *locally* according to certain conditions which were expressed explicitly in the language have been studied in [21]. Moreover, several variants of sabotage games were applied to the learning/teaching scenarios [15], and their computational behaviors were analyzed. Following this direction, a new game setting allowing both link deletion and link addition was developed in [5] to capture some further features of the learning process. Closely related to the logic of [5], a class of *relation-changing logics*, containing operators to swap, delete or add links, was well explored in [1, 2]. Instead of modifying links, in poison games [13], a player can poison a node to make it unavailable to the opponent. These games have been studied with diverse modal approaches in [11] and [17]. Additionally, by updating valuation functions of models, a dynamic logic of local fact change was studied in [25], which captures a class of graph games in which properties of states might get affected by those of others.

Product Logics with Diagonal Constant. Technically, our work is close to that of many-dimensional modal logics [14, 22]. In particular, a class of product logics was studied in [18–20] with the so-called *diagonal constant* δ .⁵ In [18, 20] it was shown that $\mathbf{K} \times^\delta \mathbf{K}$, the product logic augmenting $\mathbf{K} \times \mathbf{K}$ with δ , lacks the finite model property and is undecidable, which seem very similar to our

⁴ Although details of the translation are not described in the article, it is instructive to notice that unlike usual situations, LHS is not a fragment of the first-order logic with two free variables.

⁵ For instance, in two dimensional models δ holds at a state (s, t) just in the case that $s = t$.

results at a first glance. However, our logic differs from those both conceptually and technically.

First, our formulas are evaluated at *pairs* of states, where each of the states can occur by itself (and, not just as a constituent of an ordered pair), which makes it possible for us to study the relationship between two states directly. In $\mathbf{K} \times^\delta \mathbf{K}$, even though formulas are evaluated at pairs of states, these pairs themselves form nodes in the domain. As a result, product logic cannot express the more fine-grained relation (i.e., identity) between the two components forming a pair. In [18, 20], δ is interpreted as a special subset of the domain, not necessarily consisting of pairs formed by the same components from those dimensions. Therefore, we can say that constant I explored in this article works at a meta level. In contrast, δ in [18, 20] is an object level notion.⁶

Next, techniques adopted to establish the undecidability of LHS are very different. Similar to all other product logics, various relations representing transitions of states in different dimensions are considered in [18, 20]. Moreover, the product nature endows the relations with possible interactions: say, *commutativity* and *confluence*. With such interactions, product logics obtain grid-like structures automatically. However, as illustrated in our proofs, a crucial step in proving undecidability of LHS was exactly to build such a shape. In other words, these extra efforts make our proof technically non-trivial.

6 Conclusion and Future Work

Summary. Motivated by the meeting/avoiding game, this paper studies a modal logic LHS that allows us to talk about moves for each player, as well as the situation of meeting. More specifically, formulas in this logic are evaluated at two states of the domain, representing positions of different players. A constant I expressing the meeting of two players is explored in depth, which adds a natural and novel treatment of equality in modal logics. We establish a series of results concerning its expressive power and computational behavior. A new notion of bisimulation for LHS is proposed, and is compared systematically with those of related logics. Further, we have proved that the logic does not enjoy the tree model property or the finite model property, and that the satisfiability problem of the logic is undecidable, which refutes a conjecture made by van Benthem and Liu in their recent paper [7].

Further Directions. We mention a few directions that we would like to pursue immediately. Though we have obtained some basic results about LHS, more properties of the logic need to be explored. Several open problems have been formulated along the way, including the axiomatization of LHS, and issues regarding its expressive power. Regarding the language, the constant I seems rather simple and innocent, but surprisingly, our logic turned out to be undecidable. It

⁶ But this does not exclude possible ‘mixture’ of the two lines of the frameworks: on one hand, technically LHS can be reduced to product logics with δ , and on the other hand, product models themselves can also be viewed as special models (with two relations) for LHS (and then I denotes the identity of *two* pairs).

makes sense to understand this phenomenon better, and possibly by investigating some alternative logics (e.g., the logic mentioned in Remark 1). In Sect. 5, we have seen the differences between our work and product logics, and a systematic comparison is needed. As stated earlier, we have taken a high-level modeller’s perspective to study the hide and seek game in this paper. We reason about players’ observations and moves with the assumption that the whole graph and the players’ positions at each stage of the game are available to us. For the next step, we will pursue strategic reasoning from the players’ perspectives *in* the game. We will focus on technical issues like the epistemic aspects of the players and extend the current language with epistemic modalities to deal with those concepts.

Finally, as mentioned in various places, our work has a natural connection with the game of cops and robber in the vast literature of graph games (see, e.g., [12, 23]). We are exploring richer versions of these games, focusing on different characterization results of *cop-win graphs*. We have extended LHS with modal substitution operators [26] which enable us to express winning positions of players in the general sense, as discussed in Sect. 2. We have also obtained some new results regarding cop-win characterizations. We will continue this line of research in the future.

Acknowledgements. We thank Johan van Benthem for his inspiring suggestions. Also, we wish to thank the three anonymous reviewers for their helpful comments for improvements. Sujata Ghosh acknowledges Department of Science and Technology, Government of India for financial support vide Reference No DST/CSRI/2018/202 under Cognitive Science Research Initiative (CSRI) to carry out this work. Dazhu Li, Fenrong Liu and Yaxin Tu are supported by the Major Program of the National Social Science Foundations of China [17ZDA026].

Appendix: Proof of Theorem 2

Proof. Let $T = \{T^1, \dots, T^n\}$ be a finite set of tile types. For each $T^i \in T$ we use $u(T^i), d(T^i), l(T^i), r(T^i)$ to represent the colors of its up, down, left and right edges, respectively. We are going to define a formula φ_{tile} such that:

$$\varphi_{\text{tile}} \text{ is satisfiable iff } T \text{ tiles } \mathbb{N} \times \mathbb{N}.$$

To do so, we will use three relations in models (W, R^s, R^r, R^u) in the proof to follow. In line with this, syntactically we have six operators $[\text{left}]^*$ and $[\text{right}]^*$ for $\star \in \{s, r, u\}$. Intuitively, all the relations describe the transitions of the left evaluation point and the right evaluation point of a graph model: in what follows, we are going to construct a spy point over relation R^s , and the relations R^u and R^r represent moving up and to the right, respectively, from the corresponding tile to the other.

These three relations are useful to present the underlying intuitions of the formulas that will be constructed. Also, they are helpful in making these formulas short and readable, facilitating a better understanding of the same. Crucially,

this does not change the computational behavior of the original LHS: the three relations can be reduced to one relation as that of our standard models. For instance, given that we have three relations and the evaluation gap, we can mimic the three relations with a singular relation and 3×2 fresh propositional letters encoding, e.g., $[\text{left}]^*$ and $\langle \text{right} \rangle^*$ as $[\text{left}](p_E^* \rightarrow \dots)$ and $\langle \text{right} \rangle(p_A^* \wedge \dots)$, respectively. Definitely, to preserve the structure of those relations and truth of formulas, we need to be careful when defining the new relation and the valuation function. However, due to page-limit constraints, we forego those details here. Now, we proceed to present the details of φ_{tile} , whose components will be divided into four groups. Let us begin with the first one.

Group 1: Infinite many states induced by R^s and their ‘scope’

- (U1) $I \wedge [\text{left}]^s \neg I$
- (U2) $\langle \text{left} \rangle^s [\text{left}]^s \perp$
- (U3) $[\text{right}]^s \langle \text{left} \rangle^s (\neg I \wedge \langle \text{left} \rangle^s \top \wedge [\text{left}]^s I)$
- (U4) $[\text{left}]^s [\text{right}]^s ([\text{left}]^s \perp \wedge [\text{right}]^s \perp \rightarrow I)$
- (U5) $[\text{left}]^s [\text{right}]^s (\langle \text{left} \rangle^s \langle \text{right} \rangle^s I \rightarrow I)$

Notice that formulas (U1)–(U3) are just the R^s -version of the formulas in Theorem 1 that are used to create infinite models. Immediately, there exists an infinite sequence of states as follows:

$$w_0, w_1, w_2, \dots$$

such that $R^s(w_{i+1}) = \{w_i\}$ and $R^s(w_0) = \emptyset$. Also, for the current evaluation pair (e.g., (s, s)), we have $\{w_i \mid i \in \mathbb{N}\} \subseteq R^s(s)$.

Now let us spell out what (U4) and (U5) express. Essentially, both the formulas establish a ‘border’ for the scope of nodes that are (directly or indirectly) reachable from s via relation R^s . Specifically, the formula (U4) shows that $R^s(s)$ contains only a dead end which is exactly w_0 listed above, and moreover, the formula (U5) indicates that for any $w_i, w_j \in R^s(s)$, if they can reach the same state, then we have $w_i = w_j$. See Fig. 4 for two counterexamples without the properties of (U4) or (U5). From the two formulas, we know that $R^s(s) = \{w_i \mid i \in \mathbb{N}\}$.

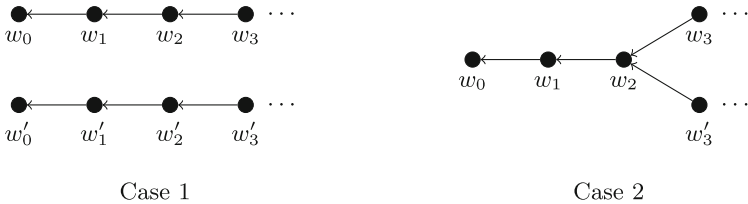


Fig. 4. Two impossible cases of the R^s -structure $R^s(s)$: Case 1 cannot satisfy (U4), while Case 2 cannot satisfy (U5).

Intuitively, we will use these w'_i 's to represent tiles. To make this precise, beyond the simple linear order of R^s among those states, we still need to structure them with R^r and R^u in a subtler way. Our next group of formulas concerns some basic features of the two relations:

Group 2: Basic features of R^u and R^r

- (U6) $[\text{left}]^s[\text{left}]^\dagger[\text{right}]^s I$ $\dagger \in \{u, r\}$
 (U7) $[\text{left}]^s(\langle \text{left} \rangle^u \top \wedge \langle \text{left} \rangle^r \top)$
 (U8) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^u \neg I \wedge [\text{left}]^r \neg I)$
 (U9) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^\dagger \neg \langle \text{left} \rangle^\dagger I)$ $\dagger \in \{u, r\}$

Before listing more formulas, let us briefly comment on these properties.

For all $i \in \mathbb{N}$, the formulas of (U6) essentially give $R^r(w_i)$ and $R^u(w_i)$ a ‘scope’. Specifically, they guarantee that $R^r(w_i), R^u(w_i) \subseteq \{w_0, w_1, \dots\}$. Therefore, when considering the two relations, we only need to consider those w'_i 's, and there do not exist other states that are involved.

The formula (U7) states that every w_i has successors via R^u and R^r , i.e., $R^u(w_i) \neq \emptyset$ and $R^r(w_i) \neq \emptyset$. Intuitively, this expresses that every tile has at least one tile above it and at least one tile to its right.

Also, the formula (U8) indicates that for all $i \in \mathbb{N}$, we do not have $R^r w_i w_i$ or $R^u w_i w_i$. Moreover, formulas in (U9) show that both the relations R^r and R^u are asymmetric.

Except those basic features captured by formulas of Group 2, what might be more important is our next group of formulas, which structure the states in a *grid* with R^r and R^u :

Group 3: Grid formed by R^u and R^r

- (U10) $[\text{left}]^s[\text{right}]^s(\langle \text{left} \rangle^\dagger I \rightarrow [\text{left}]^\dagger I)$ $\dagger \in \{u, r\}$
 (U11) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^u[\text{right}]^r \neg I)$
 (U12) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^u[\text{left}]^r \neg I \wedge [\text{left}]^r[\text{left}]^u \neg I)$
 (U13) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^u[\text{left}]^r[\text{left}]^u \neg I)$
 (U14) $[\text{left}]^s[\text{right}]^s(I \rightarrow [\text{left}]^u[\text{right}]^r \langle \text{left} \rangle^r \langle \text{right} \rangle^u I)$

Whereas (U7) tells us that all the w'_i 's have R^r - and R^u -successors, formulas in (U10) state that every w_i has at most one R^r -successor and at most one R^u -successor. Thus, both (U7) and (U10) ensure that the transitions between those w'_i 's via R^u and R^r are essentially *functions*: precisely, for all $i \in \mathbb{N}$ and $\dagger \in \{u, r\}$, $R^\dagger(w_i)$ is a singleton.

Moreover, formula (U11) suggests that the R^r -successor and the R^u -successor of a tile are different: for all $i \in \mathbb{N}$, $R^r(w_i) \cap R^u(w_i) = \emptyset$. That is, a tile cannot be above as well as to the right of another tile.

Additionally, (U12) shows that no tile can be both above/below and to the right/left of another tile, and (U13) disallows cycles following successive steps of the R^u , R^r and R^u relations, in this order. Formula (U14) states the property

of ‘confluence’: for all tiles w_i, w_j, w_k , if $R^u w_i w_j$ and $R^r w_i w_k$ hold, then there exists another tile w_n such that $R^r w_j w_n$ and $R^u w_k w_n$ hold. Now, the tiles are arranged in a grid.

Now, it remains to set a genuine tiling, which can be achieved by our fourth group of formulas. Very roughly, in usual cases this work is often routine when we have an infinite grid-like model (cf. [9]). Let us present the details here:

Group 4: Tiling the model

$$(U15) \quad [\text{left}]^s \left(\bigvee_{1 \leq i \leq n} t_E^i \wedge \bigwedge_{1 \leq i < j \leq n} \neg(t_E^i \wedge t_E^j) \right)$$

$$(U16) \quad [\text{right}]^s \left(\bigvee_{1 \leq i \leq n} t_A^i \wedge \bigwedge_{1 \leq i < j \leq n} \neg(t_A^i \wedge t_A^j) \right)$$

$$(U17) \quad [\text{left}]^s [\text{right}]^s \left(I \rightarrow \bigvee_{1 \leq i \leq n} (t_E^i \wedge t_A^i) \right)$$

$$(U18) \quad [\text{left}]^s \left(\bigwedge_{1 \leq i \leq n} (t_E^i \rightarrow \langle \text{left} \rangle^u \bigvee_{1 \leq j \leq n, u(T_i)=d(T_j)} t_E^j) \right)$$

$$(U19) \quad [\text{left}]^s \left(\bigwedge_{1 \leq i \leq n} (t_E^i \rightarrow \langle \text{left} \rangle^r \bigvee_{1 \leq j \leq n, r(T_i)=l(T_j)} t_E^j) \right)$$

Formulas (U15)–(U16) indicate that a node can be occupied ‘two’ tiles t_E^i and t_A^j . As one node can only be occupied by exactly one tile, the statement here may look a bit strange. However, we would like to argue that essentially there exists no problem, see our discussion on formula (U17) below.

By formula (U17), for every fixed i , when both t_E^i and t_A^j hold at a node, then we have $i = j$, i.e., they are of the same type T^i . In this sense, we can say that ‘E’ and ‘A’ are just ‘position-labels’ to refer to the evaluation nodes in the current graph model, and a node in the model is essentially occupied by exactly one tile. Moreover, for the same reason, although for each T^i , we have different propositional atoms t_A^i and t_E^i , all types of tiles we use are exactly those given by the original T , but not any extra ones.

Finally, the ideas of formulas (U18) and (U19) are routine: the former one states that colors match going up, while the latter expresses that they match going right.

Now, let φ_{tile} be the conjunctions of all formulas listed in the four groups. Based on our analyses above, any model satisfying φ_{tile} is a tiling of $\mathbb{N} \times \mathbb{N}$.

On the other hand, we still need to show the other direction. Now suppose that a function $f : \mathbb{N} \times \mathbb{N} \rightarrow T$ is a tiling of $\mathbb{N} \times \mathbb{N}$. Define a model $\mathbf{M}_t = (W, R^s, R^u, R^r, V)$ in the following:

- $W := \{s\} \cup (\mathbb{N} \times \mathbb{N})$
- R^s consists of the following:
 - For all $x \in \mathbb{N} \times \mathbb{N}$, $\langle s, x \rangle \in R^s$
 - For all $\langle n, 0 \rangle \in \mathbb{N} \times \mathbb{N}$ with $1 \leq n$, $\langle \langle n + 1, 0 \rangle, \langle 0, n \rangle \rangle \in R^s$
 - For all other $\langle n, m + 1 \rangle \in \mathbb{N} \times \mathbb{N}$, $\langle \langle n, m + 1 \rangle, \langle n + 1, m \rangle \rangle \in R^s$

- $R^u := \{ \langle \langle n, m \rangle, \langle n, m + 1 \rangle \rangle \mid n, m \in \mathbb{N} \}$
- $R^r := \{ \langle \langle n, m \rangle, \langle n + 1, m \rangle \rangle \mid n, m \in \mathbb{N} \}$
- $V(t_E^i) = V(t_A^i) = \{ \langle n, m \rangle \in \mathbb{N} \times \mathbb{N} \mid f(\langle n, m \rangle) = T^i \}$, for all $i \in \{1, \dots, n\}$
- $V(p_E) = V(q_A) = \emptyset$, for all other $p_E, q_A \in P_E \cup P_A$.

Figure 5 presents a crucial fragment of the model. By construction, it is not hard to check that $\mathbf{M}_t, s, s \models \varphi_{\text{tile}}$. This completes the proof. \square

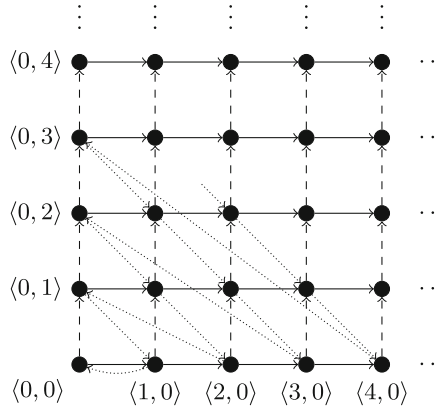


Fig. 5. The restriction of the structure of \mathbf{M}_t to $\mathbb{N} \times \mathbb{N}$, where the resulting R^s, R^u, R^r are represented by dotted-, dashed- and solid-arrows respectively. To obtain the whole structure, we just need to add the state s and draw a dotted-arrow from s to each of the members of $\mathbb{N} \times \mathbb{N}$.

References

1. Areces, C., Fervari, R., Hoffmann, G.: Moving arrows and four model checking results. In: Ong, L., de Queiroz, R. (eds.) WoLLIC 2012. LNCS, vol. 7456, pp. 142–153. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32621-9_11
2. Areces, C., Fervari, R., Hoffmann, G., Martel, M.: Satisfiability for relation-changing logics. *J. Log. Comput.* **28**, 1443–1470 (2018)
3. Aucher, G., van Benthem, J., Grossi, D.: Modal logics of sabotage revisited. *J. Logic Comput.* **28**(2), 269–303 (2018). <https://doi.org/10.1093/logcom/exx034>
4. Baltag, A., van Benthem, J.: A simple logic of functional dependence. *J. Philos. Logic* (2021)
5. Baltag, A., Li, D., Pedersen, M.Y.: On the right path: a modal logic for supervised learning. In: Blackburn, P., Lorini, E., Guo, M. (eds.) LORI 2019. LNCS, vol. 11813, pp. 1–14. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-60292-8_1
6. van Benthem, J.: *Logic in Games*. The MIT Press, Cambridge (2014)

7. van Benthem, J., Liu, F.: Graph games and logic design. In: Liu, F., Ono, H., Yu, J. (eds.) *Knowledge, Proof and Dynamics*. LASLL, pp. 125–146. Springer, Singapore (2020). <https://doi.org/10.1007/978-981-15-2221-5.7>
8. Berger, R.: *The Undecidability of the Domino Problem*. American Mathematical Society (1966)
9. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press, Cambridge (2001)
10. Blackburn, P., Seligman, J.: Hybrid languages. *J. Logic Lang. Inform.* **4**, 251–272 (1995)
11. Zaffora Blando, F., Mierzewski, K., Areces, C.: The modal logics of the poison game. In: Liu, F., Ono, H., Yu, J. (eds.) *Knowledge, Proof and Dynamics*. LASLL, pp. 3–23. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2221-5_1
12. Bonato, A., Nowakowski, R.J.: *The Game of Cops and Robbers on Graphs*. AMS (2011)
13. Duchet, P., Meyniel, H.: Kernels in directed graphs: a poison game. *Discret. Math.* **115**, 273–276 (1993)
14. Gabbay, D., Kurucz, A., Wolter, F., Zakharyashev, M.: *Many-Dimensional Modal Logics: Theory and Applications*, *Studies in Logic and the Foundations of Mathematics*, vol. 148. Elsevier (2003)
15. Gierasimczuk, N., Kurzen, L., Velázquez-Quesada, F.R.: Learning and teaching as a game: a sabotage approach. In: He, X., Horty, J., Pacuit, E. (eds.) *LORI 2009*. LNCS (LNAI), vol. 5834, pp. 119–132. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-04893-7.10>
16. Goldfarb, W.: The unsolvability of the Gödel class with identity. *J. Symb. Log.* **49**, 1237–1252 (1984)
17. Grossi, D., Rey, S.: Credulous acceptability, poison games and modal logic. In: Agmon, N., Taylor, M.E., Elkind, E., Veloso, M. (eds.) *Proceedings of AAMAS 2019*, pp. 1994–1996 (2019)
18. Hampson, C., Kikot, S., Kurucz, A.: The decision problem of modal product logics with a diagonal, and faulty counter machines. *Stud. Logica.* **104**, 455–486 (2016)
19. Kikot, S.P.: Axiomatization of modal logic squares with distinguished diagonal. *Math. Notes* **88**, 238–250 (2020)
20. Kurucz, A.: Products of modal logics with diagonal constant lacking the finite model property. In: Ghilardi, S., Sebastiani, R. (eds.) *FroCoS 2009*. LNCS (LNAI), vol. 5749, pp. 279–286. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-04222-5.17>
21. Li, D.: Losing connection: the modal logic of definable link deletion. *J. Log. Comput.* **30**, 715–743 (2020)
22. Marx, M., Venema, Y.: *Multi-Dimensional Modal Logic*. Kluwer Academic Publishers (1997)
23. Nowakowski, R., Winkler, R.P.: Vertex-to-vertex pursuit in a graph. *Discret. Math.* **13**, 235–239 (1983)
24. Pützstück, P.: *Decidability and Bisimulation for Logics of Functional Dependence*. Master’s thesis, Mathematical Foundations of Computer Science, RWTH-Aachen University (2020)
25. Thompson, D.: Local fact change logic. In: Liu, F., Ono, H., Yu, J. (eds.) *Knowledge, Proof and Dynamics*. LASLL, pp. 73–96. Springer, Singapore (2020). <https://doi.org/10.1007/978-981-15-2221-5.5>
26. Tu, Y., Ghosh, S., Li, D., Liu, F.: A new modal logic for cops-and-robber games. Manuscript (2021)



Orthogonal Frames and Indexed Relations

Philippe Balbiani and Saúl Fernández González^(✉)

Institut de Recherche en Informatique de Toulouse,
CNRS, Université de Toulouse, Toulouse, France
saul.fgonzalez@irit.fr

Abstract. We define and study the notion of an *indexed frame*. This is a bi-dimensional structure consisting of a Cartesian product equipped with relations which only relate pairs if they coincide in one of their components. We show that these structures are quite ubiquitous in modal logic, showing up in the literature as products of Kripke frames, subset spaces, or temporal frames for STIT logics. We show that indexed frames are completely characterised by their ‘orthogonal’ relations, and we provide their sound and complete logic. Using these ‘orthogonality’ results, we provide necessary and sufficient conditions for an arbitrary Kripke frame to be isomorphic to certain well-known bi-dimensional structures.

1 Introduction

This text is concerned with a certain type of bi-dimensional relational structure which shows up in multiple areas of modal logic. The ubiquity of these structures, we wish to argue, should motivate an independent study of their properties and their logic, towards which we take the first steps in this paper.

In the text we will call these structures *indexed frames*. Let us start off by providing two distinct (but ultimately equivalent) definitions of what we mean by that.

Definition 1. *By indexed frame we refer indistinctly to any of the following structures:*

(IF1) *Frames $(W_1 \times W_2, R_1, R_2)$ where R_1 and R_2 are binary relations on $W_1 \times W_2$ such that $(w_1, w_2)R_i(w'_1, w'_2)$ implies $w_j = w'_j$ for $i \neq j$;*

(IF2) *Tuples (W_1, W_2, R^1, R^2) where, for $i \neq j$, $R^i = \{R_w^i : w \in W_j\}$ is a family of binary relations on W_i indexed by the elements of W_j .*

It is straightforward to see how these two definitions refer to the same type of structure. Given a frame of the form (IF1), we define $w_2R_w^1w'_2$ iff $(w, w_2)R_1(w, w'_2)$ and $w_1R_w^2w'_1$ iff $(w_1, w)R_2(w'_1, w)$ to obtain a frame of the form (IF2); conversely, given a frame in the form (IF2) we obtain a (IF1) frame by setting $(w_1, w_2)R_i(w'_1, w'_2)$ iff $w_j = w'_j$ and $w_iR_w^iw'_i$.

Having these bi-dimensional structures at hand, one can interpret formulas over a bi-modal language

$$\phi ::= p \mid \perp \mid (\phi \wedge \phi) \mid \neg\phi \mid \Box_1\phi \mid \Box_2\phi$$

with respect to pairs in $W_1 \times W_2$ as follows:

- (IF1) $(w_1, w_2) \models \Box_i \phi$ iff $(w_1, w_2)R_i(w'_1, w'_2)$ implies $(w'_1, w'_2) \models \phi$;
 (IF2) $(w_1, w_2) \models \Box_1 \phi$ iff $w_1 R_{w_2}^1 w'_1$ implies $(w'_1, w_2) \models \phi$;
 $(w_1, w_2) \models \Box_2 \phi$ iff $w_2 R_{w_1}^2 w'_2$ implies $(w_1, w'_2) \models \phi$.

It can be easily seen how these semantics are equivalent.

We start this paper by illustrating that indexed frames show up quite often in the literature. In order to put forward this argument, we provide in the next section examples of well-known models in different areas of modal logic which are indexed frames. In Sect. 3 we show that the property of ‘orthogonality’ (i.e., the fact that each point in the model is uniquely determined by the pair of connected components to which it belongs) is necessary and sufficient to characterize indexed frames, and we use this property to provide their sound and complete logic. In Sect. 4 we enrich our language with modalities \blacksquare_1 and \blacksquare_2 which fix w_2 (resp. w_1) and quantify over all points in W_1 (resp. W_2). We provide the sound and complete logic for this extended language. In Sect. 5, we come back to the examples presented in Sect. 2 with the results on orthogonality previously discussed, showing necessary and sufficient conditions for a bi-relational Kripke frame to be isomorphic to several well-known types of indexed frames. We conclude in Sect. 6.

The proofs of some minor Propositions and Lemmas have been moved to the Appendix; this is indicated with the symbol \square .

2 Examples of Indexed Frames

Let us see some well-known structures that are either indexed frames or generated subframes thereof. We will use the term “indexed relation” to informally refer to a relation defined on a Cartesian product that respects one of the coordinates.

Example 1 (Products). [7, Chapter 3] The *product* of two Kripke frames (W_1, R_1) and (W_2, R_2) (wherein R_i is a binary relation defined on W_i for $i = 1, 2$) is the frame

$$(W_1, R_1) \times (W_2, R_2) = (W_1 \times W_2, R_H, R_V),$$

where R_H and R_V are binary relations on $W_1 \times W_2$ (called the ‘horizontal’ and ‘vertical’ relations respectively) defined as:

$$\begin{aligned} (w_1, w_2)R_H(w'_1, w'_2) &\text{ iff } w_2 = w'_2 \text{ and } w_1 R_1 w'_1, \text{ and} \\ (w_1, w_2)R_V(w'_1, w'_2) &\text{ iff } w_1 = w'_1 \text{ and } w_2 R_2 w'_2 \end{aligned}$$

Products very closely adjust to the (IF1) definition. In fact, indexed frames can be seen as a generalization of products. Indeed, a product can be seen as an (IF2) indexed frame (W_1, W_2, R^1, R^2) with the extra property that, for all $w_1, w'_1 \in W_1$ and $w_2, w'_2 \in W_2$, $R_{w_2}^1 = R_{w'_2}^1$ and $R_{w_1}^2 = R_{w'_1}^2$.

It is of note that, while the logic of bidimensional products of frames (as studied in [7]) contains axioms making both modalities interact (such as $\Diamond_1 \Diamond_2 p \leftrightarrow \Diamond_2 \Diamond_1 p$), this will not be the case for the logic of indexed frames.

Example 2 (Subset spaces). In its most basic form [12], a *subset space* is a tuple consisting of a set X and some collection \mathcal{O} of nonempty subsets of X .

One can interpret formulas of a bimodal language including \Box and K modalities on a subset space with respect to a pair (x, U) such that $x \in U$ and $U \in \mathcal{O}$ as follows:

$$\begin{aligned} x, U \models K\phi &\text{ iff } y, U \models \phi \text{ for all } y \in U \\ x, U \models \Box\phi &\text{ iff } x, V \models \phi \text{ for all } V \subseteq U \text{ such that } x \in V \ \& \ V \in \mathcal{O}. \end{aligned}$$

The semantics above naturally defines two indexed relations on the graph $\mathcal{O}_X := \{(x, U) : x \in U \ \& \ U \in \mathcal{O}\}$, namely:

$$\begin{aligned} (x, U) \equiv_K (y, V) &\text{ iff } U = V; \\ (x, U) \geq_{\Box} (y, V) &\text{ iff } x = y \text{ and } U \supseteq V \end{aligned}$$

Clearly, the standard Kripke semantics on the frame $(\mathcal{O}_X, \equiv_K, \geq_{\Box})$ (let us call this a *subset space frame*) are the exact semantics above, and moreover this subset space frame is (a generated subframe of) an indexed frame.

Example 3 (Social Epistemic Logic). *Social Epistemic Logic* (SEL) is a multi-modal framework to model knowledge within social networks, introduced in [15]. Its language contains, in addition to atomic propositional variables p, q, \dots , *nominal variables* n, m, \dots , an artefact borrowed from Hybrid Logic [1]. It has operators $K\phi$ and $F\phi$ to express things such as ‘‘I know ϕ ’’ and ‘‘all my friends ϕ ’’, and, in addition, it presents an operator $@_n\phi$ for each nominal n to express ‘‘ ϕ is true of the agent named by n ’’.

The models for SEL are of the form $(W, A, \{\sim_a\}_{a \in A}, \{\succ_w\}_{w \in W}, V)$, where each \sim_a is an ‘epistemic indistinguishability’ equivalence relation for agent a on the set of possible worlds W , and each \succ_w is a ‘social’ symmetric and irreflexive relation, representing which pairs of agents in the set A are ‘friends’ at world w . The valuation V assigns subsets of $W \times A$ to propositional variables p and, for a nominal n , $V(n)$ is of the form $W \times \{a\}$ for some a ; it is then said that ‘‘ n is the name of a ’’, denoted $a = \underline{n}_V$.

For the semantics, we read formulas with respect to a pair of a world and an agent as follows: $(w, a) \models K\phi$ iff $(v, a) \models \phi$ for all v such that $w \sim_a v$; $(w, a) \models F\phi$ iff $(w, b) \models \phi$ for all b such that $a \succ_w b$, and $(w, a) \models @_n\phi$ iff $(w, \underline{n}_V) \models \phi$.

$(W, A, \{\sim_a\}_{a \in A}, \{\succ_w\}_{w \in W})$ is clearly an (IF2) indexed frame and even the $@_n$ modality can be interpreted via the ‘‘indexed’’ relation: $(w, a)R_n(v, b)$ iff $w = v$ and $b = \underline{n}_V$.

Its equivalent (IF1) form is $(W \times A, \sim, \succ)$, where $(w, a) \sim (v, b)$ iff $a = b$ and $w \sim_a v$, and $(w, a) \succ (v, b)$ iff $w = v$ and $a \succ_w b$.

Example 4 (STIT logic). The logic of *seeing-to-it-that* or *STIT* was studied in [3] and has shown up in the literature with many variations; in most cases, the different models for STIT are quite explicitly indexed frames, or present indexed relations. The one we showcase here is (a slightly simplified version of) a *Kamp frame*, discussed in [5, 17].

A *Kamp frame* is a tuple $(W, \mathcal{O}, \{\sim_t\}_{t \in T}, \{\sim_{t,i}\}_{t \in T, i \in A_{gt}})$, where each world has a ‘timeline’ associated to it, this being a linear order $\mathcal{O}(w) = (T_w, <_w)$. T is

the union of all the T_w 's. For each t , the relations \sim_t and $\sim_{t,i}$ are equivalence relations defined on the set $\{w : t \in T_w\}$.

Sentences in a language including a necessity operator \Box , agency operators $[i]$ for $i \in \text{Agt}$ and a temporal operator G are read with respect to pairs (t, w) such that $t \in T_w$ as follows:

$$\begin{aligned} (t, w) \models \Box\phi &\text{ iff } (t, w') \models \phi \text{ for all } w' \sim_t w; \\ (t, w) \models [i]\phi &\text{ iff } (t, w') \models \phi \text{ for all } w' \sim_{t,i} w; \\ (t, w) \models G\phi &\text{ iff } (t', w) \models \phi \text{ for all } t' >_w t. \end{aligned}$$

While this does not exactly adjust to the definitions of indexed frame above, one sees how this structure can be defined as (a generated subframe of) the (IF2) indexed frame

$$(W, T, \{\sim_t\}_{t \in T}, \{\sim_{t,i}\}_{t \in T, i \in \text{Agt}}, \{<_w\}_{w \in W}).$$

(We are slightly bending our definition of ‘indexed frame’ here and allowing for multiple families of relations indexed by the elements of T .)

We can easily ‘rewrite’ these relations to be defined on (a subset of) $W \times T$ in the (IF1) way:

$$\begin{aligned} (t, w) \equiv_{\Box} (t', w') &\text{ iff } t = t' \ \& \ w \sim_t w' \\ (t, w) \equiv_i (t', w') &\text{ iff } t = t' \ \& \ w \sim_{t,i} w' \\ (t, w) <_G (t', w') &\text{ iff } w = w' \ \& \ t <_w t' \end{aligned}$$

All the frames showcased in this section share one property: namely that of *orthogonality*. We explain and study this in the next section.

3 Orthogonal Frames

The relations R_1 and R_2 in an indexed frame $(W_1 \times W_2, R_1, R_2)$ are “orthogonal” to each other, in the sense that there cannot be two distinct points connected by both R_1 and R_2 . Indeed, if there is an R_i path from (w_1, w_2) to (w'_1, w'_2) (i.e. if they belong to the same R_i -connected component), then $w_j = w'_j$ for $j \neq i$ and, in consequence, if there are both R_1 paths and R_2 paths between these pairs, then $(w_1, w_2) = (w'_1, w'_2)$. In the present section we shall see that this property fully characterises indexed frames.

For the rest of this paper, given a relation R , we let R^* denote the least equivalence relation containing R , i.e., the equivalence relation induced by the connected components of R . By Id_W we refer to the identity relation $\{(w, w) : w \in W\}$.

Definition 2. *A birelational Kripke frame (W, R_1, R_2) is orthogonal if there exist equivalence relations \equiv_1 and \equiv_2 on W satisfying:*

$$\begin{aligned} \text{(O1)} \quad R_i &\subseteq \equiv_i, \quad i = 1, 2; \\ \text{(O2)} \quad \equiv_1 \cap \equiv_2 &= \text{Id}_W. \end{aligned}$$

A frame (W, R_1, R_2) is said to be full orthogonal if there exist equivalence relations \equiv_1 and \equiv_2 on W satisfying (O1), (O2) and

$$\text{(O3)} \quad \equiv_1 \circ \equiv_2 = W^2.$$

We leave it to the reader to check that:

Lemma 1. (W, R_1, R_2) is orthogonal if and only if $R_1^* \cap R_2^* = \text{Id}_W$.

Note that, if such a pair of equivalence relations exists, it is not necessarily unique: consider the frame (W, R_1, R_2) where $W = \mathbb{N}^2$ and $R_1 = R_2 = \text{Id}_W$; the pair of equivalence relations (\equiv_1, \equiv_2) , where $(n_1, n_2) \equiv_i (m_1, m_2)$ iff $n_i = m_i$ satisfies properties O1 – O3; however, the pair (W^2, Id_W) does as well.

Proposition 1. (W, R_1, R_2) is isomorphic to an indexed frame if and only if it is full orthogonal.

Proof. Let $(W_1 \times W_2, R_1, R_2)$ be an indexed frame. Then the relations $(w_1, w_2) \equiv_i (w'_1, w'_2)$ iff $w_j = w'_j$ (where $\{i, j\} = \{1, 2\}$) satisfy O1, O2, and O3.

Conversely, suppose such relations exist, and let $[w]_i$ denote the equivalence class of w under \equiv_i . By O2 and O3, given any pair $(w, v) \in W^2$, there is exactly one element in the intersection $[w]_2 \cap [v]_1$: let $x_{w,v}$ denote this unique element. Consider the frame $(W/\equiv_2 \times W/\equiv_1, R_1, R_2)$, where

$$([w]_2, [v]_1)R_i([w']_2, [v']_1) \text{ if and only if } x_{w,v}R_ix_{w',v'}.$$

This is an indexed frame, for if $x_{w,v}R_1x_{w',v'}$, then $x_{w,v} \equiv_1 x_{w',v'}$, and since $v \equiv_1 x_{w,v} \equiv_1 x_{w',v'} \equiv_1 v'$, this gives $[v]_1 = [v']_1$. We reason analogously for R_2 . It is isomorphic to (W, R_1, R_2) via the map $f([w]_2, [v]_1) = x_{w,v}$. For injectivity, note that if $x_{w,v} \neq x_{w',v'}$, then either $w \not\equiv_2 w'$ or $v \not\equiv_1 v'$. For surjectivity, note that $w = x_{w,w}$ for all $w \in W$. Finally, note that we have defined the map in such a way that $([w]_2, [v]_1)R_i([w']_2, [v']_1)$ iff $f([w]_2, [v]_1)R_if([w']_2, [v']_1)$.

Definition 3. Given two Kripke-complete unimodal logics L_1 and L_2 we say that a birelational frame (W, R_1, R_2) is a $[L_1, L_2]$ -frame if $(W, R_i) \models L_i$, for $i = 1, 2$.

Recall that the fusion logic $L_1 \oplus L_2$ is the least normal modal logic containing the axioms and rules of L_1 for \Box_1 and of L_2 for \Box_2 and that:

Theorem [7, Thm. 4.1]. $L_1 \oplus L_2$ is the logic of $[L_1, L_2]$ -frames.

We have:

Proposition 2. An orthogonal $[L_1, L_2]$ -frame (W, R_1, R_2) is a generated sub-frame of a full orthogonal $[L_1, L_2]$ -frame. □

Proposition 3. The fusion logic $L_1 \oplus L_2$ is the logic of orthogonal $[L_1, L_2]$ -frames.

Proof. The proof in [7, Thm. 4.1] of the fact that

the logic of frames (W, R_1, R_2) such that $(W, R_i) \models L_i$ for $i = 1, 2$ is the fusion $L_1 \oplus L_2$

utilises the construction of a *dovetailed frame* in order to prove that any formula ϕ consistent in $L_1 \oplus L_2$ is satisfiable in an $[L_1, L_2]$ -frame. It is a recursive process done as follows: first, one obtains a consistent formula in the language of L_1 by rewriting ϕ with ‘surrogate’ propositional variables $p_{\Diamond_2\psi_1}^1, \dots, p_{\Diamond_2\psi_n}^1$ in place of its maximal subformulas preceded by \Diamond_2 . Then one constructs a rooted L_1 -frame satisfying ϕ . Whenever a point in this frame satisfies a surrogate variable $p_{\Diamond_2\psi}^1$, one rewrites ψ in the language of L_2 by using surrogates $q_{\Diamond_1\theta_1}^2, \dots, q_{\Diamond_1\theta_n}^2$ and makes this point the root of an L_2 -frame satisfying this formula. One repeats this process, alternating between L_1 -formulas and L_2 -formulas until one obtains a rooted frame satisfying ϕ at the root.

We point the interested reader to [7] for more precise details about this construction; for clarity, we provide a simple example from [7], using the formula $\phi = p \wedge \Diamond_1(\neg p \wedge \Diamond_2 p) \wedge \Diamond_2(\neg p \wedge \Diamond_1(p \wedge \Diamond_2 p))$.

We rewrite ϕ as $p \wedge \Diamond_1(\neg p \wedge \mathbf{q}^2) \wedge \mathbf{r}^2$, where \mathbf{q}^2 is a ‘surrogate’ for $\Diamond_2 p$, \mathbf{r}^2 for $\Diamond_2(\neg p \wedge \mathbf{q}^1)$, \mathbf{q}^1 for $\Diamond_1(p \wedge \mathbf{s}^2)$, and \mathbf{s}^2 for $\Diamond_2 p$.

We construct a rooted L_1 -frame satisfying the rewritten formula (top left of Fig. 1); we make the node satisfying \mathbf{r}^2 into the root of an L_2 -frame satisfying its surrogate formula $\Diamond_2(\neg p \wedge \mathbf{q}^1)$ and the \mathbf{q}^2 node into frame satisfying $\Diamond_2 p$ (top right); we then proceed similarly with \mathbf{q}^1 (bottom left) and finally with \mathbf{s}^2 (bottom right) to find a $[L_1, L_2]$ -frame satisfying ϕ at its bottom point.

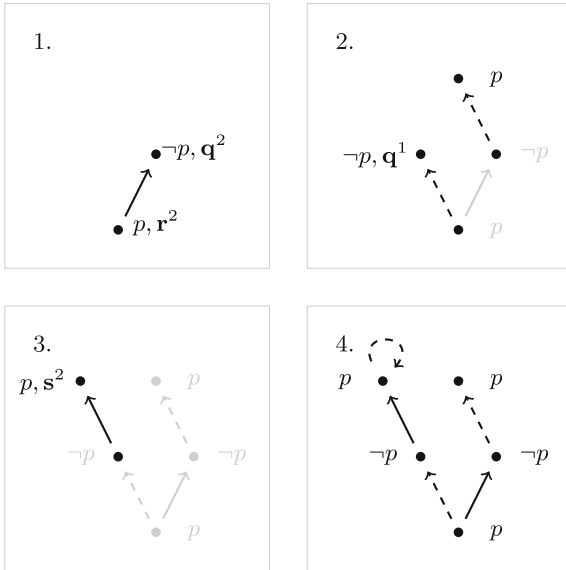


Fig. 1. ‘Dovetailed’ construction of a frame for $p \wedge \Diamond_1(\neg p \wedge \Diamond_2 p) \wedge \Diamond_2(\neg p \wedge \Diamond_1(p \wedge \Diamond_2 p))$.

For our current purposes it suffices to point out that the ‘dovetailed’ frames obtained by this method are always orthogonal, for this construction does not

allow for two distinct points x and y to be reachable from each other by both R_1 and R_2 . As an immediate consequence of Propositions 2 and 3:

Theorem 1. *The logic of $[L_1, L_2]$ -indexed frames is the fusion $L_1 \oplus L_2$.*

4 Orthogonal Structures

In the definition for full orthogonal frames (Definition 2) we demand the existence of equivalence relations which are supersets of the two given relations and satisfy the properties of full orthogonality. These relations are not made explicitly part of the structure and are not taken into account when defining the logic.

In this section we consider structures $(X, R_1, R_2, \equiv_1, \equiv_2)$ satisfying O1, O2 and O3, and we study the logic of these frames when we add modal operators to our language to explicitly account for the orthogonal equivalence relations.

Let us first note the following fact:

Lemma 2 (Generalized orthogonal frames). *If (W, R_1, R_2) is a Kripke frame such that there exist equivalence relations \equiv_1 and \equiv_2 on W satisfying*

- (O1) $R_i \subseteq \equiv_i$,
- (O2) $\equiv_1 \cap \equiv_2 = \text{Id}_W$, and
- (O3') $\equiv_1 \circ \equiv_2 = \equiv_2 \circ \equiv_1$,

then (W, R_1, R_2) is a disjoint union of full orthogonal frames. □

Definition 4. *An orthogonal structure is a tuple $(W, R_1, R_2, \equiv_1, \equiv_2)$, where (W, R_1, R_2) is a birelational Kripke frame and \equiv_1 and \equiv_2 are equivalence relations on W satisfying (O1), (O2), and (O3') above. A standard orthogonal structure satisfies moreover (O3) $\equiv_1 \circ \equiv_2 = W^2$.*

A tuple satisfying (O1) and (O3') is called a semistructure.

We define a semantics for (semi)structures $(W, R_{1,2}, \equiv_{1,2})$ with respect to a language containing operators \Box_i and \blacksquare_i for $i = 1, 2$ as follows:

- $w \models \Box_i \phi$ iff, for all v , wR_iv implies $v \models \phi$;
- $w \models \blacksquare_i \phi$ iff, for all v , $w \equiv_i v$ implies $v \models \phi$.

A very standard canonical model argument shows that:

Proposition 4. *The sound and complete logic of semistructures is*

$$K_{\Box_1} + K_{\Box_2} + S5_{\blacksquare_1} + S5_{\blacksquare_2} + \blacksquare_1 \blacksquare_2 \phi \leftrightarrow \blacksquare_2 \blacksquare_1 \phi + \blacksquare_i \phi \rightarrow \Box_i \phi.$$

Moreover, if L_1 and L_2 are canonical unimodal logics, the logic of semistructures $(W, R_{1,2}, \equiv_{1,2})$ such that $(W, R_i) \models L_i$ for $i = 1, 2$ is

$$L_1 + L_2 + S5_{\blacksquare_1} + S5_{\blacksquare_2} + \blacksquare_1 \blacksquare_2 \phi \leftrightarrow \blacksquare_2 \blacksquare_1 \phi + \blacksquare_i \phi \rightarrow \Box_i \phi.$$

□

Let us call these logics Log_{\perp} and $Log_{\perp}^{L_1 L_2}$ respectively. Let us now see that Log_{\perp} is also the logic of orthogonal structures (and, in turn, of “standard” structures).

Recall that a *bounded morphism* between Kripke frames $F = (W, R_1, \dots, R_n)$ and $F' = (W', R'_1, \dots, R'_n)$ is a map $f : W \rightarrow W'$ satisfying the *forth condition* ($wR_i v$ implies $f(w)R'_i f(v)$) and the *back condition* ($f(w)R'_i v$ implies there is an $w \in f^{-1}(w')$ such that $wR_i v$). If the bounded morphism is surjective, then every formula which is refutable in F' can be refuted in F . (See [4, Thm. 3.14] for details).

We shall show that a semistructure is always the image of a bounded morphism departing from an orthogonal structure, which in turn will let us prove that the logic of orthogonal structures is the above.

The proof below utilises the notion of a *matrix enumeration*. Given sets I and X , an I -matrix enumeration of X is a map $f : I \times I \rightarrow X$ such that, for any fixed $i_0 \in I$, both maps

$$j \in I \mapsto f(i_0, j) \in X \text{ and } j \in I \mapsto f(j, i_0) \in X$$

are surjective.

Lemma 3. *If $|I| \geq |X|$, there exists an I -matrix enumeration of X . □*

With this:

Proposition 5. *A semistructure is a bounded morphic image of an orthogonal structure.*

Proof. Let $(W, R_{1,2}, \equiv_{1,2})$ be a semistructure. Let I be a set of indices with the same cardinality as W .

Let us consider the quotient set $W/\equiv_{1 \cap 2}$. Let us fix a matrix enumeration $f_{[w]} : I \times I \rightarrow [w]$ of each equivalence class $[w] \in W/\equiv_{1 \cap 2}$. We use w_{ij} as a shorthand for $f_{[w]}(i, j)$. Note that it is always the case that $w \equiv_k w_{ij}$ for $k = 1, 2$.

We define the following relations on the set $W' = W/\equiv_{1 \cap 2} \times I^2$:

$$\begin{aligned} ([w], i_1, i_2) \equiv'_1 ([v], j_1, j_2) &\text{ iff } w \equiv_1 v \text{ and } i_2 = j_2; \\ ([w], i_1, i_2) \equiv'_2 ([v], j_1, j_2) &\text{ iff } w \equiv_2 v \text{ and } i_1 = j_1; \\ ([w], i_1, i_2) R'_1 ([v], j_1, j_2) &\text{ iff } w_{i_1 i_2} R_1 v_{j_1 j_2} \text{ and } i_2 = j_2; \\ ([w], i_1, i_2) R'_2 ([v], j_1, j_2) &\text{ iff } w_{i_1 i_2} R_2 v_{j_1 j_2} \text{ and } i_1 = j_1. \end{aligned}$$

Let us see that this is an orthogonal structure. Indeed,

(O1) $([w], i_1, i_2) R'_k ([v], j_1, j_2)$ implies $w_{i_1 i_2} R_k v_{j_1 j_2}$ and $i_l = j_l$ (for $k \neq l$), which in turn implies $w_{i_1 i_2} \equiv_k v_{j_1 j_2}$ and $i_l = j_l$. This means that $w \equiv_k v$ and $i_l = j_l$, and thus $([w], i_1, i_2) \equiv'_k ([v], j_1, j_2)$.

(O2) If $([w], i_1, i_2) \equiv'_k ([v], j_1, j_2)$ for $k = 1$ and 2 , then $i_1 = j_1$, and $i_2 = j_2$, and $(w, v) \in \equiv_1 \cap \equiv_2$, which implies $[w] = [v]$. Therefore, $\equiv'_1 \cap \equiv'_2 = Id_{W'}$.

(O3') If $([w], i_1, i_2) (\equiv'_1 \circ \equiv'_2) ([u], j_1, j_2)$, then $w (\equiv_1 \circ \equiv_2) u$. This, plus property (O3') of the semistructure, implies that there exists some v' such that $w \equiv_2 v' \equiv_1 u$. But then

$$([w], i_1, i_2) \equiv'_2 ([v'], i_1, j_2) \equiv'_1 ([u], j_1, j_2).$$

This shows that $(\equiv'_1 \circ \equiv'_2) \subseteq (\equiv'_2 \circ \equiv'_1)$; the converse inclusion is analogous.

Finally, the map

$$([w], i_1, i_2) \in W / \equiv_1 \cap \equiv_2 \times I^2 \mapsto w_{i_1 i_2} \in W$$

is a bounded morphism. For the forth condition, $([w], i_1, i_2) \equiv'_k ([v], j_1, j_2)$ implies $w_{i_1 i_2} \equiv_k v_{j_1 j_2}$ and $([w], i_1, i_2) R'_k ([v], j_1, j_2)$ implies $w_{i_1 i_2} R_k v_{j_1 j_2}$, by definition. For the back condition, if $w_{i_1 i_2} R_1 v$, then there exists an index $j \in I$ such that $f_{[v]}(j, i_2) = v$ and, by definition,

$$([w], i_1, i_2) R'_1 ([v], j, i_2).$$

An analogous argument can be made for R_2 , \equiv_1 and \equiv_2 .

As a consequence:

Theorem 2. *The sound and complete logic of standard orthogonal structures is Log_{\perp} ,*

$$K_{\square_1} + K_{\square_2} + S5_{\blacksquare_1} + S5_{\blacksquare_2} + \blacksquare_1 \blacksquare_2 \phi \leftrightarrow \blacksquare_2 \blacksquare_1 \phi + \blacksquare_i \phi \rightarrow \square_i \phi.$$

Proof. Consequence of Propositions 4 and 5.

Remark 1. The construction in the proof above respects many properties of the R_i relations: for instance, if R_i is reflexive, transitive, symmetric, Euclidean, etc., then so is R'_i . This means that this technique can be used to prove that $Log_{\perp}^{L_1 L_2}$ is the logic of indexed structures (W, R_i, \equiv_i) where $(W, R_i) \models L_i$ for a large family of logics that includes S4, S5, KD45, etc. We conjecture that the result is true for any pair L_1, L_2 of Kripke-complete unimodal logics.

Let us now define a semantics for this extended language directly on indexed frames $(W_1 \times W_2, R_1, R_2)$, taking advantage of the isomorphism between indexed frames and full orthogonal frames given in the proof of Proposition 1. The fact that the isomorphic image of the equivalence classes of the ‘orthogonal’ equivalence relations are sets of the form $W_1 \times \{w_2\}$ and $\{w_1\} \times W_2$ allows us to consider the \blacksquare modalities as coordinate-wise ‘universal modalities’; that is to say, if we interpret formulas of the extended language on indexed frames as follows:

$$\begin{aligned} (w_1, w_2) \models \blacksquare_1 \phi &\text{ iff } (v, w_2) \models \phi \text{ for all } v \in W_1, \text{ and} \\ (w_1, w_2) \models \blacksquare_2 \phi &\text{ iff } (w_1, v) \models \phi \text{ for all } v \in W_2, \end{aligned}$$

then we have that:

Proposition 6. *Log_{\perp} is the sound and complete logic of indexed frames for the language including \square_i and \blacksquare_i operators. \boxtimes*

We finish this Section by pointing out the fact that Log_{\perp} enjoys the Finite Model Property with respect to semistructures, orthogonal structures and indexed frames, in the following sense:

Proposition 7. *If $\phi \notin Log_{\perp}$, then there exists a finite indexed frame refuting ϕ . \boxtimes*

We conjecture that, if L_1 and L_2 have the Finite Model Property, then for all $\phi \notin Log_{\perp}^{L_1 L_2}$ there exists a finite $[L_1, L_2]$ -semistructure (perhaps a finite $[L_1, L_2]$ -indexed frame) refuting ϕ ; this problem, however, remains open.

5 Some Case Studies

In the present section we return to the Examples in Sect. 2 and, with the help of our orthogonality results above, we abstract from the “indexed frame” definition and give necessary and sufficient conditions on orthogonal frames to be isomorphic to these structures.

Products (Example 1). We have:

Proposition 8. *A frame (X, R_1, R_2) is isomorphic to a product of Kripke frames if and only if there exist two equivalence relations \equiv_1 and \equiv_2 such that:*

- (O1) $R_i \subseteq \equiv_i$, for $i = 1, 2$;
- (O2) $\equiv_1 \cap \equiv_2 = Id_X$;
- (O3) $\equiv_1 \circ \equiv_2 = X^2$, and (P1) $(R_i \circ \equiv_j) = (\equiv_j \circ R_i)$, for $i \neq j$.

Proof. That a product $(W_1, R_2) \times (W_2, R_2)$ satisfies these properties (with the equivalence relations $(w_1, w_2) \equiv_i (v_1, v_2)$ iff $w_j = v_j$) is trivial.

Now let us consider a frame (W, R_1, R_2) satisfying the properties above and let x_{wv} denote the unique element in $[w]_2 \cap [v]_1$ (as in the proof of Proposition 1). This frame satisfies, for all $w, w', v, v' \in W$: $x_{wv} R_1 x_{w'v}$ iff $x_{wv} R_1 x_{w'v'}$. Indeed, if $x_{wv} R_1 x_{w'v}$, since $x_{w'v} \equiv_2 x_{w'v'}$, then by (P1) there must exist some y such that $x_{wv'} \equiv_2 y R_1 x_{w'v'}$, and this y can be no other than $x_{wv'}$. We thus can define a relation on W/\equiv_2 as $[w]_2 R'_1 [w']_2$ iff $x_{wv} R_1 x_{w'v}$ for some (equiv.: for all) v . We proceed similarly to define a relation R'_2 on W/\equiv_1 : $[v]_1 R'_2 [v']_1$ iff $x_{wv} R_1 x_{wv'}$ for some (for all) w .

The product $(W/\equiv_2, R'_1) \times (W/\equiv_1, R'_2)$ is equal to $(W/\equiv_2 \times W/\equiv_1, R_1, R_2)$, isomorphic to (W, R_1, R_2) as per Proposition 1.

Subset spaces (Example 2). Recall the notion of a *subset space frame* from Example 2. We have:

Proposition 9. *A frame (W, R_K, R_\square) is isomorphic to a subset space frame if and only if*

- (SS1) R_K is an equivalence relation;
- (SS2) R_\square is a partial order (i.e. reflexive, transitive and antisymmetric);
- (SS3) $R_\square \circ R_K \subseteq R_K \circ R_\square$,

and there exists an equivalence relation \equiv_\square such that

- (O1) $R_\square \subseteq \equiv_\square$;
- (O2) $R_K \cap \equiv_\square = Id_W$,

and, moreover,

- (SS4) $([R_K \circ \equiv_\square](u) \supseteq [R_K \circ \equiv_\square](v) \text{ and } u \equiv_\square v) \text{ imply } u R_\square v$;
- (SS5) $[R_K \circ \equiv_\square](u) = [\equiv_\square \circ R_K](v) \text{ implies } u R_K v$.

□

Social Epistemic Logic (Example 3). Let us define a semantics for Social Epistemic Logic on full orthogonal structures $(X, R_K, R_F, \equiv_A, \equiv_W)$, where $R_K \subseteq \equiv_A$ and $R_F \subseteq \equiv_W$. The equivalence classes of these relations will represent agents and worlds respectively.

Recall that the only constraints on a SEL model (W, A, \sim, \succ) are that \sim must be an equivalence relation and \succ must be symmetric and irreflexive. Therefore, via the isomorphism in Proposition 1, one easily sees that:

Lemma 4. *Let $(X, R_K, R_F, \equiv_A, \equiv_W)$ be a full orthogonal structure. The full orthogonal frame (X, R_K, R_F) is isomorphic to a SEL frame if and only if, on top of (O1) – (O3), it satisfies:*

- (SEL1) R_K is an equivalence relation, and
- (SEL2) R_F is symmetric and irreflexive.

Recall (Proposition 1) that the corresponding isomorphic SEL model will be $(X/\equiv_W, X/\equiv_A, R_K, R_F)$, where R_K relates two pairs of equivalence classes if and only if the unique elements in the intersection of each pair are related by R_K (and likewise for R_F).

Now let us consider how a valuation must act upon this model. For a SEL model we demand that each $V(n)$ must be of the form $W \times \{a\}$ for some unique agent $a \in A$. Via the isomorphism outlined above, we can see, for the image of a valuation V defined on an orthogonal structure $(X, R_K, R_F, \equiv_A, \equiv_W)$ to be a valid valuation on a SEL model, we want the image of the set $V(n)$ to be $X/\equiv_W \times \{[y]_A\}$ for some $y \in X$. But the pre-image of this set is precisely $[y]_A$.

We thus demand the following property:

- (SEL3) $V(n) \in X/\equiv_A$ for all n .

For each nominal n and $x \in X$, we let n_x denote the unique element in $[x]_W \cap V(n)$.

Models of SEL must be *named*. A *named model* is a model wherein every agent has a name, i.e., for all $a \in A$, there exists a nominal n such that $a = \underline{n}$. In these isomorphic structures, the notion of *named model* translates to: for all $y \in X$, there exists n such that $V(n) = [y]_A$, or, equivalently,

- (SEL4) for all $x \in X$, there exists $n \in \text{Nom}$ such that $x \in V(n)$.

With all this we can define a semantics for Social Epistemic Logic on full orthogonal models $(X, R_K, R_F, \equiv_A, \equiv_W, V)$ where R_K , R_F and V satisfy the constraints (SEL1) – (SEL4) above as follows:

- $x \models F\phi$ iff $xR_F y$ implies $y \models \phi$;
- $x \models K\phi$ iff $xR_K y$ implies $y \models \phi$;
- $x \models n$ iff $x \in V(n)$ (iff $x = n_x$);
- $x \models @_n\phi$ iff $n_x \models \phi$.

The ‘non-rigid’ variant of SEL [18] assigns different names to agents in each possible world. This is imposed via the following, weaker, constraint of the valuation: for every nominal n and each world w , there exists a unique agent $a \in A$ such that $(w, a) \in V(n)$. In the isomorphic structures above, this translates to a constraint weaker than (SEL3), namely:

- (SEL3’) for each n and each $x \in X$,
the intersection $[x]_W \cap V(n)$ is a singleton.

A proof of completeness of (standard, rigid) SEL using ‘indexed’ canonical models was recently given in [2] (it had been proven in [14] with a different method). Completeness of ‘non-rigid’ SEL was proven in [18] by means of an

involved step-by-step construction, but a proof of this result using canonical models remains an open problem. We conjecture that the semantics above could assist in this endeavour.

STIT logic (Example 4). [5] compares three distinct semantics for STIT logic. One of them, in the form of ‘Kamp frames’, was briefly alluded to in Example 4. Another one, introduced in [10], interprets sentences on *T-STIT frames*: these are one-dimensional Kripke frames

$$(X, \equiv_{\square}, \{\equiv_i\}_{i \in \text{Agt}}, \prec_G)$$

wherein two different sorts of relations allow to reason, respectively, about time (\prec_G) and necessity/agency (the equivalence relations, with $\equiv_i \subseteq \equiv_{\square}$). These relations are defined to be orthogonal, for they satisfy ‘ $x \equiv_{\square} y$ implies $x \not\prec_G y$ ’.

In [5] it is shown that both T-STIT frames and Kamp frames satisfy the same formulas, via an argument which involves transforming one structure into the other in a truth-preserving manner. However, thanks to the isomorphism in Proposition 1 (and the (IF1) redefinition of a Kamp frame of Example 4) one can go beyond and show that a Kamp frame is always a T-STIT frame and that a T-STIT frame is isomorphic to a Kamp frame, wherein the set of ‘timelines’ W is defined by the connected components of \prec and the set of ‘moments’ T is given by the equivalence classes of \equiv_{\square} .

6 Discussion and Future Work

We have identified a structure that shows up with relative frequency in different areas of modal logic; we have argued that an independent study of this structure is warranted and have taken the first steps towards it.

We have shown that these structures are completely characterised by the ‘orthogonality’ of their relations. Proofs of completeness of frameworks based on indexed frames are not particularly easy to tackle; as an example, we point the reader to the completeness proof of SEL in [14]. We hope that the above observations about orthogonality will help facilitate some of these proofs.

Some work remains to be done and many questions are open. Among these are the following:

Is $\text{Log}_{\perp}^{L_1 L_2}$ the logic of orthogonal structures $(W, R_1, R_2, \equiv_1, \equiv_2)$ such that $(W, R_i) \models L_i$, for any pair of Kripke-complete logics L_1 and L_2 ? Can a formula $\phi \notin \text{Log}_{\perp}^{L_1 L_2}$ be refuted in a finite indexed frame whenever L_1 and L_2 have the FMP? We conjecture an affirmative answer to these questions, and we plan further research to resolve them.

Some variations on subset space logics consider families of subsets which are closed under intersection [12] or which are topologies [6, 8, for instance]. What further restrictions does one have to add to obtain a result analogous to Proposition 9 for these structures? In the latter case, is there a relation between these properties and the point-free topologies of [13]?

Perhaps the most obvious question: how does one generalise the definitions and results in this paper to the n -dimensional case? The reader may find that there are two reasonable generalisations of this framework to the n -th dimension:

- (A) $(W_1 \times \dots \times W_n, R_1, \dots, R_n)$ such that $(w_j)_{j=1}^n R_i (v_j)_{j=1}^n$ implies $w_j = v_j$ for all $j \neq i$;
- (B) $(W_1 \times \dots \times W_n, R_1, \dots, R_n)$ such that $(w_j)_{j=1}^n R_i (v_j)_{j=1}^n$ implies $w_i = v_i$.

Out of these two, we suggest (A) is more appropriate, for it does not make much sense to apply (B) to $n = 1$, and (A) is the only one which still generalises n -dimensional products. Many of the results of this paper may translate relatively easily to the n -dimensional case, whereas some may not. We plan to devote future work to this question.

Acknowledgements. We wish to thank Emiliano Lorini for a very interesting discussion about STIT logics, some of the fruits of which made it into this paper.

We also extend our gratefulness to the anonymous reviewers of this paper for their helpful comments and suggestions.

Appendix

Proof of Proposition 2. Given an orthogonal $[L_1, L_2]$ -frame (W, R_1, R_2) , we extend W to the set

$$W' = W \cup \{x_{wv} : w, v \in W, R_2^*(w) \cap R_1^*(v) = \emptyset\},$$

i.e., we add one element for each pair of connected components which have an empty intersection, and we extend the relations R_i as follows:

- if $F_\bullet \models L_i$, then $R'_i = R_i$;
- if $F_\circ \models L_i$, then $R'_i = R_i \cup \{(x, x) : x \in W' \setminus W\}$;

where F_\bullet is the irreflexive singleton frame $(\{*\}, \emptyset)$, and F_\circ is the reflexive singleton frame $(\{*\}, \{(*, *)\})$. (Recall that every logic is satisfied in either F_\bullet or F_\circ ; this is a consequence of a classical result by Makinson [11].)

Note that, in either case, no elements of W are related to any elements of $W' \setminus W$ and thus (W, R_1, R_2) is a generated subframe of (W', R'_1, R'_2) .

We define

$$\begin{aligned} \equiv'_1 &= (R_1 \cup \{(v, x_{wv}) : v \in W\})^*, \text{ and} \\ \equiv'_2 &= (R_2 \cup \{(w, x_{wv}) : w \in W\})^*. \end{aligned}$$

Note that \equiv'_1 and \equiv'_2 satisfy conditions O1 – O3 of Definition 2, and therefore (W', R'_1, R'_2) is a full orthogonal frame. Finally, for $i = 1, 2$, (W', R'_i) is the disjoint union of the L_i -frame (W, R_i) with a family of singleton L_i -frames, and thus it is an L_i -frame.

Proof of Lemma 2. We leave it to the reader to check that (O3') implies that $\equiv_1 \circ \equiv_2$ is an equivalence relation. Let W' be an equivalence class of $\equiv_1 \circ \equiv_2$. Let R'_i and \equiv'_i be the restrictions of R_i and \equiv_i to W' . It is routine to check that (O1) $R'_i \subseteq \equiv'_i$, (O2) $\equiv'_1 \cap \equiv'_2 = Id_{W'}$, and (O3) $\equiv'_1 \circ \equiv'_2 = (W')^2$. Each of these is therefore a full orthogonal frame and (W, R_1, R_2) is equal to the disjoint union $\bigcup_{W' \in W/\equiv_1 \circ \equiv_2} (W', R'_1, R'_2)$.

Proof (sketch) of Proposition 4. This uses the very standard technique of *canonical models*; we point the reader to [4, Chapter 4] for full details and we simply offer a sketch here:

Let X be the set of maximal consistent sets of formulas in the language. We define the relations xR_iy iff, for all ϕ , $\Box_i\phi \in x$ implies $\phi \in y$ and $x \equiv_i y$ iff, for all ϕ , $\blacksquare_i\phi \in x$ implies $\phi \in y$.

The *Truth Lemma* shows that, given the valuation $V(p) = \{x \in X : p \in x\}$, it is the case that $x \models \phi$ iff $\phi \in x$.

We note that the logic $Log_{\neg}^{L_1L_2}$ is canonical, for canonicity is preserved by fusions [9, Cor. 6] and the addition of Sahlqvist axioms [4, Chapter 4]. This canonicity ensures that $(X, R_i) \models L_i$; the S5 axioms for the \blacksquare_i 's ensure that \equiv_i is an equivalence relation; $\blacksquare_1\blacksquare_2\phi \leftrightarrow \blacksquare_2\blacksquare_1\phi$ ensures that (O3') is satisfied; finally, the axioms $\blacksquare_i\phi \rightarrow \Box_i\phi$ ensure (O1).

Therefore $(X, R_{1,2}, \equiv_{1,2})$ is a semistructure and any consistent formula ϕ can be satisfied in it.

Proof of Lemma 3. We simply show the existence of a matrix enumeration $f : X \times X \rightarrow X$ whenever X is infinite; we leave further details to the reader. Let $\{X_1, X_2\}$ be a partition of X into two sets which are equipotent to X itself (note that the existence of such partition requires the Axiom of Choice for uncountable cardinalities [16]). Let $f_1 : X_1 \rightarrow X$ and $f_2 : X_2 \rightarrow X$ be two surjections. The map

$$f(x, y) = \begin{cases} f_i(x) & \text{if } x, y \in X_i \\ f_j(y) & \text{if } x \in X_i, y \in X_j, i \neq j \end{cases}$$

is the desired enumeration.

Proof of Proposition 6. Soundness is routine. For completeness, given a formula $\phi \notin Log_{\neg}$, it suffices to use Theorem 2 to find a standard orthogonal structure $(W, R_{1,2}, \equiv_{1,2})$ that refutes ϕ , construct the indexed frame $(W/\equiv_2 \times W/\equiv_1, R_1, R_2)$ isomorphic to (W, R_1, R_2) via Proposition 1 and note that the equivalence relation $([w]_2, [v]_1) \cong_i ([w']_2, [v']_1)$ iff $x_{wv} \equiv_i x_{w'v'}$ relates two pairs if and only if their j -th coordinate coincides, for $j \neq i$.

Proof (sketch) of Proposition 7. This involves a rather standard filtration argument. (See [4, Chapter 2] for details on this technique).

Given a consistent formula ϕ , we let $(W, R_{1,2}, \equiv_{1,2})$ be a semistructure satisfying ϕ at a point w_0 , and Γ be a finite set of formulas closed under subformulas such that $\phi \in \Gamma$, and we define an equivalence relation $w \sim_{\Gamma} v$ iff for all $\psi \in \Gamma$, $(w \models \psi$ iff $v \models \psi)$. We define relations in the quotient set W/\sim_{Γ} as follows: for $i = 1, 2$,

$$[w]_{\Gamma} \equiv'_i [v]_{\Gamma} \text{ iff, for all } \blacksquare_i\psi \in \Gamma, (w \models \blacksquare_i\psi \text{ iff } v \models \blacksquare_i\psi), \text{ and}$$

$$[w]_{\Gamma} R'_i[v]_{\Gamma} \text{ iff } [w]_{\Gamma} \equiv'_i [v]_{\Gamma} \text{ and for all } \Box_i\psi \in \Gamma (w \models \Box_i\psi \text{ implies } v \models \psi).$$

We leave it to the reader to check that the resulting tuple is a semistructure and a filtration and therefore that $[w_0]_{\Gamma} \models \phi$. We can then use Proposition 5 and Lemma 2 to obtain an indexed frame satisfying ϕ .

Proof of Proposition 9. For the left-to-right direction, given a subset space frame we consider the relations $(x, U)R_K(y, V)$ iff $U = V$, $(x, U)R_{\square}(y, V)$ iff $x = y$ and $U \supseteq V$, and $(x, U) \equiv_{\square}(y, V)$ iff $x = y$. We note that $(R_K \circ \equiv_{\square})(x, U) = \{(x', U') \in \mathcal{O}_X : x' \in U\}$, and we leave it to the reader to check that this satisfies all the properties in Proposition 9.

Let us now consider a frame with these properties. We let $[\cdot]_{\square}$ and $[\cdot]_K$ denote the equivalence classes of \equiv_{\square} and R_K . Let us define the subset space

$$X = X / \equiv_{\square} = \{[w]_{\square} : w \in W\}$$

$$\mathcal{O} = \{U_v : v \in W\}, \text{ where } U_v = \{[w]_{\square} \in X : v[R_K \circ \equiv_{\square}]w\}.$$

Note that $[w]_{\square} \in U_v$ if and only if $[w]_{\square} \cap [v]_K \neq \emptyset$.

By (O2), an intersection $[w]_{\square} \cap [v]_K$ of two equivalence classes is at most a singleton. Let us map and element $([w]_{\square}, U_v)$ in the graph of (X, \mathcal{O}) to the unique element in $[w]_{\square} \cap [v]_K$. This is a bijection whose inverse maps each $w \in W$ to $([w]_{\square}, U_w)$. We define relations \equiv_K and \supseteq_{\square} on this graph as in Example 2 and, to show that this map is an isomorphism, it suffices to show that

$$wR_K v \text{ iff } ([w]_{\square}, U_w) \equiv_K ([v]_{\square}, U_v), \text{ and}$$

$$wR_{\square} v \text{ iff } ([w]_{\square}, U_w) \supseteq_{\square} ([v]_{\square}, U_v).$$

We start with the second item. From left to right, if $wR_{\square} v$, then $[w]_{\square} = [v]_{\square}$ by (O1), and let us see that $U_w \supseteq U_v$. If $[y]_{\square} \in U_v$, then there is a unique element $x \in [y]_{\square} \cap [v]_K$. But since $wR_{\square} vR_K x$, it follows by (SS3) that there must exist some x' such that $wR_K x'R_{\square} x$. Since $x' \equiv_{\square} x$, by (O1), and $x \equiv_{\square} y$, it follows that $x' \in [w]_K \cap [y]_{\square}$, and thus $[y]_{\square} \in U_w$. From right to left, it suffices to see that $U_w \supseteq U_v$ and $w \equiv_{\square} v$ implies $wR_{\square} v$. But this follows directly from (SS4), noting that $U_w \supseteq U_v$ implies $[R_K \circ \equiv_{\square}](w) \supseteq [R_K \circ \equiv_{\square}](v)$.

For the first item it suffices to show that $wR_K v$ iff $U_w = U_v$. The left-to-right direction is immediate from the definition of U_w , whereas the right-to-left direction follows from (SS5).

References

1. Areces, C., ten Cate, B.: Hybrid logics. In: Handbook of Modal Logic, pp. 821–868 (2006)
2. Balbiani, P., Fernández González, S.: Indexed frames and hybrid logics. In: Advances in Modal Logic (2020)
3. Belnap, N.D., Perloff, M., Xu, M., et al.: Facing the Future: Agents and Choices in Our Indeterminist World. Oxford University Press, Oxford (2001)
4. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2001). <https://doi.org/10.1017/CBO9781107050884>
5. Ciuni, R., Lorini, E.: Comparing semantics for temporal STIT logic. Logique et Anal. (N.S.) **61**(243), 299–339 (2017)
6. van Ditmarsch, H., Knight, S., Özgün, A.: Announcement as effort on topological spaces. Synthese **196**(7), 2927–2969 (2017). <https://doi.org/10.1007/s11229-017-1592-8>

7. Gabbay, D.M., Kurucz, A., Wolter, F., Zakharyashev, M.: *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier (2003)
8. Georgatos, K.: Knowledge theoretic properties of topological spaces. In: Masuch, M., Pólos, L. (eds.) *Logic at Work 1992*. LNCS, vol. 808, pp. 147–159. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58095-6_11
9. Kracht, M., Wolter, F.: Properties of independently axiomatizable bimodal logics. *J. Symb. Log.* **56**(4), 1469–1485 (1991)
10. Lorini, E.: Temporal logic and its application to normative reasoning. *J. Appl. Non-Classical Log.* **23**(4), 372–399 (2013)
11. Makinson, D.: Some embedding theorems for modal logic. *Notre Dame J. Formal Log.* **12**(2), 252–254 (1971)
12. Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. In: *TARK*, vol. 92, pp. 95–105 (1992)
13. Picado, J., Pultr, A.: *Frames and Locales: Topology Without Points*. Springer, Heidelberg (2011)
14. Sano, K.: Axiomatizing epistemic logic of friendship via tree sequent calculus. In: Baltag, A., Seligman, J., Yamada, T. (eds.) *LORI 2017*. LNCS, vol. 10455, pp. 224–239. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-55665-8_16
15. Seligman, J., Liu, F., Girard, P.: Logic in the community. In: Banerjee, M., Seth, A. (eds.) *ICLA 2011*. LNCS (LNAI), vol. 6521, pp. 178–188. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18026-2_15
16. Tarski, A.: Theorems on the existence of successors of cardinals, and the axiom of choice. In: *Indagationes Mathematicae (Proceedings)*, vol. 57, pp. 26–32. Elsevier (1954)
17. Thomason, R.H.: Combinations of tense and modality. In: Gabbay, D., Guenther, F. (eds.) *Handbook of Philosophical Logic*, pp. 135–165. Springer, Heidelberg (1984). https://doi.org/10.1007/978-94-009-6259-0_3
18. Zhen, L.: *Towards axiomatisation of social epistemic logic*. Ph.D. thesis, University of Auckland (2020)



Computable Execution Traces

Declan Thompson^(✉)

Stanford University, Stanford, CA 94305, USA
declan@stanford.edu

Abstract. This paper gives an execution trace set based account of computability by imposing restrictions on sets of arbitrary sequences of objects, based on a supplied stock of unary *tests* and binary *operations*. This account of *finite control computability* provides a highly general, top down perspective on computability. We prove equivalence with the Turing machine model, under appropriate assumptions, and show how finite control computability can be used to provide a unified account of computability across multiple levels of abstraction.

Keywords: Computability · Execution trace · Models of computation

1 Introduction

The classical accounts of computability due to Turing, Church and Kleene follow a somewhat standard form. A domain of primitive objects is identified, over which computations are taken to operate (tapes with symbols, strings over a finite alphabet, \mathbb{N} , ...). Primitive actions on this domain are chosen (moving a read/write head, rewrite rules, simple functions, ...), and assemblies of these operations are taken as the fundamental models of computation. It is well known that these accounts give rise to equivalent notions of *computable function*, up to encodings on denumerable domains, lending support to the claim that they capture the intuitive notion of computability, known as the Church-Turing Thesis.

The Church-Turing Thesis is posited for denumerable domains, but no equivalent for arbitrary domains has gained such widespread acceptance. For example, the BSS machine and Type 2 Turing machine models give rise to distinct sets of “computable” functions over \mathbb{R} [13]. Nonetheless, general accounts of computability on arbitrary domains (denumerable or otherwise) have been given by authors including Gandy [4], Gurevich [5] and Moschovakis [11]. Key among these *iterator accounts* is a set of states S equipped with a single *transition function* f . Computation proceeds by iteratively applying f to a state $s \in S$, a process which (if terminating) yields the desired output. The focus lies in finding appropriate restrictions on f to ensure the resulting process is computable.

These accounts highlight that computation involves a *process*: a sequence of stages moving from an input to a desired output. A computational model thus gives rise to a set of *execution traces* across all possible inputs. Classical and iterator accounts take a generative approach, where execution trace sets arise

from running a machine, or iterating a function. In this paper, we adopt a classificatory perspective, asking: what is required for a given set of sequences to be the execution trace set of some model of computation? This perspective frees us to focus on the abstract nature of processes, rather than the basic operations they use. It is a perspective orthogonal to the traditional focus of computability theory: whether particular sets or functions are effectively computable. We leave the notion of a computable function unanalysed, focussing instead on the sequences possibly generated when computing such functions. The paper forms part of a much broader research programme, and the observations discussed here are suggestive of a more general notion of computability, worthy of further investigation. The approach is partly inspired by work in randomness focussing on random sequences themselves, rather than how they are generated.

In this paper, we ask what a computable set of sequences is. Section 1.1 provides preliminary notation for sequences and introduces the Turing machine model [15]. Though the project at large extends beyond classical models, Turing machines undeniably generate computable execution traces, and thereby provide a useful model for developing a theory. The main results of the paper are:

1. An analysis of sets of sequences that could have been generated by some computable process;
2. A demonstration, over an appropriate domain, that sets of sequences constructed from arbitrary Turing-computable actions are able to be carried out by a Turing machine if, and only if, they satisfy our analysis.

Note that we assume the Church-Turing Thesis; the paper asks which sets of *sequences* are computable, not which operations or sets of arbitrary objects. The import of this distinction is highlighted in Sect. 2: simply composing computable actions is not enough. Our central contention is that sets of sequences are computable if they are *finite control computable*, a notion introduced in Sect. 2. Section 3 shows the compatibility of finite control computability with Turing computability and Sect. 4 concludes. A technical appendix provides further details of proofs.

1.1 Preliminaries

Sequences and Tasks. In studying execution traces, we will focus on sequences of objects over some a \mathfrak{D} . Each sequence represents a possible execution trace from some computation.

Formally, a sequence $\sigma = (\mathbf{a}_0, \mathbf{a}_1, \dots)$ is a function $\sigma : \alpha \rightarrow \mathfrak{D}$ for some ordinal $\alpha \leq \omega$. α is the length of σ , written $|\sigma| = \alpha$. For $\beta < |\sigma|$ we take

$$\sigma[0, \beta + 1) := (\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_\beta) \quad \text{and} \quad \sigma[\beta] := \mathbf{a}_\beta. \quad (1)$$

If $|\sigma| < \omega$ and $\alpha > 0$, we write $\sigma[-\alpha]$ for $\sigma[|\sigma| - \alpha]$ (so $\sigma[-1]$ is the final element of σ). Similarly, $\sigma[0, -\alpha) = \sigma[0, |\sigma| - \alpha)$ $= (\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{|\sigma| - \alpha - 1})$. If $\sigma[-1] = \tau[0]$ then

$$\sigma \circ \tau := (\sigma[0], \sigma[1], \dots, \sigma[-1], \tau[1], \tau[2], \dots) \quad (2)$$

is the composition of σ with τ . Intuitively, τ must continue where σ left off. We say σ is a *subsequence* of τ , written

$$\sigma \ll \tau, \quad (3)$$

if there is a strictly increasing $g_\sigma : |\sigma| \rightarrow |\tau|$ with $\sigma[\alpha] = \tau[g_\sigma(\alpha)]$ for all α .

A set of sequences is called a *task* F . Intuitively, a task could be an execution trace set for some computational process, across all inputs. We will sometimes discuss execution trace *points*, representing incomplete execution traces. These are given by the set of finite non-empty prefixes of sequences in a task,

$$\boxed{F} := \{\sigma[0, \alpha + 1] \mid \sigma \in F, \alpha < |\sigma|\}. \quad (4)$$

A task F is *fully deterministic* if $\sigma[0, 1) = \tau[0, 1)$ implies $\sigma = \tau$ for all $\sigma, \tau \in F$. Task composition is lifted from sequence composition by taking

$$F \circ G := \{\sigma \circ \tau \mid \sigma \in F, \tau \in G\}. \quad (5)$$

Task composition is similar to composition of binary relations, except that the entire sequence is preserved.¹ We will make use of two special types of task. If $|\sigma| = 1$ for all $\sigma \in Q$ then we call Q a *test*, by analogy with dynamic logic. For tests, $F \circ Q = \{\sigma \in F \mid \sigma[-1] \in Q\}$ is the set of $\sigma \in F$ ending with an element of Q . If $|\sigma| = 2$ for all $\sigma \in P$ then we call P an *operation*. In the case of an operation, $F \circ P = \{\sigma \mid \sigma[0, -1) \in F, (\sigma[-2], \sigma[-1]) \in P\}$ is the set of sequences starting in F and continuing in P . Intuitively, $F \circ Q \circ P$ applies a *guarded operation* to F , extending by one step every sequence $\sigma \in F$ that ends with an element of Q .

Turing Machines. Turing machines are defined formally in the appendix. Here we give intuition. A *Turing machine* is a tuple $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$, where S is a finite set of *states*; Γ is a finite *alphabet*, with a distinguished symbol B ; $\delta : S \times \Gamma \rightarrow S \times \Gamma \times \{L, R\}$ is a partial *transition function*; $s_0 \in S$ is the *start state*.

Turing machines operate on machine *tapes* $[u|v]$, with u, v strings over Γ and $|$ indicating the position of an imagined *read/write head*. A *configuration* of \mathfrak{M} is given by $[u\langle s \rangle v]$, with $s \in S$ and $[u|v]$ a tape contents. Configurations are iteratively updated by a partial function $\Delta_{\mathfrak{M}}$ defined as

$$\Delta_{\mathfrak{M}}([u\langle s \rangle bv]) = \begin{cases} [uac\langle t \rangle v] & \text{if } \delta(s, b) = \langle t, c, R \rangle \\ [u\langle t \rangle acv] & \text{if } \delta(s, b) = \langle t, c, L \rangle. \end{cases}$$

$[u\langle s \rangle bv]$ is a *halting configuration* if $\delta(s, b)$ is undefined. Iterating $\Delta_{\mathfrak{M}}$ yields

$$\text{CRUN}_{\mathfrak{M}} := \{([u\langle s_0 \rangle v], \Delta_{\mathfrak{M}}([u\langle s_0 \rangle v]), \Delta_{\mathfrak{M}}^2([u\langle s_0 \rangle v]), \dots) \mid u, v \in \Gamma^*\},$$

consisting of the completed execution traces of \mathfrak{M} . Turing machines are generally defined up to state labellings, so for \mathfrak{M} 's execution trace set we take

$$\text{RUN}_{\mathfrak{M}} := \{([u_0|v_0], [u_1|v_1], \dots) \mid ([u_0\langle s_0 \rangle v_0], [u_1\langle t_1 \rangle v_1], \dots) \in \text{CRUN}_{\mathfrak{M}}\},$$

¹ Sequence and task composition are sometimes called *fusion* or *coalesced product* in literature on trace-based semantics for dynamic logic and process logic [6, 8, 9].

i.e. the projection of $\text{CRUN}_{\mathfrak{M}}$ onto tape contents alone. Write $\mathfrak{M}([u|v]) = [w|x]$ if there is $\sigma \in \text{RUN}_{\mathfrak{M}}$ with $\sigma[0] = [u|v]$ and $\sigma[-1] = [w|x]$.

Turing Computable Tests and Operations. A test Q is *Turing computable* if there is a Turing machine \mathfrak{M}_Q such that $\mathfrak{M}_Q([u|v]) = [1]$ iff $[u|v] \in Q$. An operation P is *Turing computable* if there is a Turing machine \mathfrak{M}_P such that $\mathfrak{M}_P([u|v]) = [w|x]$ iff $([u|v], [w|x]) \in P$. In this paper, we will use *computable* and *Turing computable* interchangeably for tests and operations.

2 Finite Control Computability

We may now formulate our primary question: which tasks are possibly execution trace sets of computable processes? We start by recalling that in classical accounts computable execution traces are assembled from a stock of primitive actions. These involve conditionally modifying the current state, such as when a Turing machine writes and moves based on the symbol scanned. Actions can be modelled by *guarded operations* of the form $Q \circ P$, with Q a test and P an operation. We might say a task is computable if it is a composition of computable guarded operations, perhaps giving a co-inductive definition by taking F computable iff $F = Q \circ P \circ G$ for some computable test Q , operation P and other computable task G . The problem with this approach is that computable tasks can be combined in a non-computable manner. For $N \subseteq \mathbb{N}$, consider the task $\text{ENUM}_N = \{\sigma\}$, where σ enumerates the elements of N in increasing order. ENUM_N is a composition of computable tasks, since to move from $\sigma[\alpha]$ to $\sigma[\alpha+1]$ we need only apply addition. Yet N may not be computably enumerable.

Classical accounts avoid this problem by utilising a finitary control mechanism (e.g. state table or program text). This can be seen as providing an equivalence relation on stages of a computation: execution trace points are equivalent if they correspond to the same internal machine state, or program line, so the same actions are applied in each. We can define a similar *control equivalence* relation \simeq on \boxed{F} , without the need for specifying an underlying mechanism.

\simeq can be thought of like a bisimulation [2]. Each equivalence class $C \in \boxed{F}/\simeq$ is a set of execution trace points where the “same actions” are applied. Execution traces begin somewhere, so some $C \in \boxed{F}/\simeq_F$ contains all length 1 prefixes of F (c.f. a bisimulation has an initial pair). At any stage in a computation, one of two things may happen to an execution trace point σ : it may continue or it may end. If it continues, we extend σ by a guarded operation $Q \circ P$, conditioning on the current state $\sigma[-1]$. C is thus associated with a *construction set* \mathfrak{F}_C of guarded operations, containing all actions that could conditionally be performed. Each guarded operation must lead to the same next equivalence class, to ensure consistency in the computation (c.f. the *zig* and *zag* conditions of bisimulation). So for each $Q, P \in \mathfrak{F}_C$ there is $D \in \boxed{F}/\simeq$ with $C \circ Q \circ P \subseteq D$. If σ ends then $\sigma \in C \cap F$. In this case, we assume some test has been performed to confirm that computation is over: $\sigma \in C \circ Q$ for some Q .

Definition 1 (Control Equivalence). Let \mathbf{q} be a set of tests and \mathbf{p} of operations. Let F be a task. A control equivalence \simeq on F under \mathbf{q} and \mathbf{p} is an equivalence relation on \boxed{F} satisfying:

Starting State $\sigma[0, 1] \simeq \tau[0, 1]$ for all $\sigma, \tau \in F$.

Construction For each $C \in \boxed{F}/\simeq$ there is a construction set $\mathfrak{F}_C \subseteq \mathbf{q} \times \mathbf{p}$ such that both:

Composition $\{\sigma \in \boxed{F} \mid \sigma[0, -1] \in C\} = \bigcup_{\langle Q, P \rangle \in \mathfrak{F}_C} C \circ Q \circ P$; and

Consistency if $\langle Q, P \rangle \in \mathfrak{F}_C$ then $C \circ Q \circ P \subseteq D$ for some $D \in \boxed{F}/\simeq$.

Halting For each $C \in \boxed{F}/\simeq$ there is a halting set $\mathfrak{G}_C \subseteq \mathbf{q}$ such that $C \cap F = \bigcup_{Q \in \mathfrak{G}_C} C \circ Q$.

Definition 2 (Trace set). A trace set for \mathbf{q} and \mathbf{p} is a pair $A = (F, \simeq)$, where F is a task and \simeq is a control equivalence on F under \mathbf{q} and \mathbf{p} .

A trace set need not be fully deterministic; it may be understood as a description of possible behaviours. The tests contained within \mathbf{q} must cover all possible stages of the computation - there can be no sequence $\sigma \in A$ such that $\sigma[0, \alpha + 1] \circ Q = \emptyset$ for every $Q \in \mathbf{q}$.

Lemma 1. Let \simeq be a control equivalence on F under \mathbf{q} and \mathbf{p} . Let $\sigma \in C \in \boxed{F}/\simeq_F$ and \mathfrak{F}_C the construction set for C , and \mathfrak{G}_C the halting set. Then there is some $Q \in \{R \mid \exists P \langle R, P \rangle \in \mathfrak{F}_C\} \cup \mathfrak{G}_C$ with $\sigma[-1] \in Q$.

Proof. If $\sigma \in F$ then $\sigma \in C \cap F$ so by halting $\sigma \in C \circ Q$ for some $Q \in \mathfrak{G}_C$. It follows that $\sigma[-1] \in Q$. If $\sigma \notin F$ then $\sigma = \tau[0, -1]$ for some $\tau \in \boxed{F}$. From composition $\tau \in C \circ Q \circ P$ for some $\langle Q, P \rangle \in \mathfrak{F}_C$. Thus $\tau[-2] = \sigma[-1] \in Q$.

Not all trace sets are computable. Indeed, all tasks have trace sets.

Proposition 1. For every task F over \mathfrak{D} there is a test set \mathbf{e} , an operation set \mathbf{w} and a control equivalence \simeq such that $A = (F, \simeq)$ is a trace set for \mathbf{e} and \mathbf{w} .

Proof. Take $\mathbf{e} = \{\{\mathbf{a}\} \mid \mathbf{a} \in \mathfrak{D}\}$ and $\mathbf{w} = \{\{\langle \mathbf{a}, \mathbf{b} \rangle\} \mid \mathbf{a}, \mathbf{b} \in \mathfrak{D}\}$. Define \simeq by taking $\sigma \simeq \tau$ iff $|\sigma| = |\tau| = 1$ or $\sigma = \tau$. Starting state is satisfied since $|\sigma[0, 1]| = 1$. To check construction and halting take $C \in \boxed{F}/\simeq_F$ and let

$$\begin{aligned} \mathfrak{F}_C &= \{\{\sigma[-2]\}, \{\langle \sigma[-2], \sigma[-1] \rangle\} \mid \sigma \in \boxed{F} \text{ and } \sigma[0, -1] \in C\} \\ \mathfrak{G}_C &= \{\{\sigma[-1]\} \mid \sigma \in C \cap F\}. \end{aligned}$$

If $C = \{\tau\}$ for some τ then consistency is trivially satisfied since $C \circ Q \circ P$ is a singleton for all $\langle Q, P \rangle \in \mathfrak{F}_C$. For composition, note that

$$\{\sigma \in \boxed{F} \mid \sigma[0, -1] \in C\} = \bigcup_{\sigma \in \boxed{F}, \sigma[0, -1] = \tau} \{\sigma[0, -1]\} \circ \{\sigma[-2]\} \circ \{\langle \sigma[-2], \sigma[-1] \rangle\}.$$

$C \cap F = \{\tau\} \circ \{\tau[-1]\}$ if $\tau \in F$ and \emptyset otherwise, giving halting. If C is not a singleton then $C = \{\sigma[0, 1] \mid \sigma \in F\}$, the set of length 1 prefixes. In this case,

$$\{\sigma \in \boxed{F} \mid \sigma[0, -1] \in C\} = \{\langle \mathbf{a} \rangle \circ \langle \sigma[-2] \rangle \circ \langle \sigma[-2], \sigma[-1] \rangle \in \boxed{F} \mid \langle \mathbf{a} \rangle = \sigma[0, -1] \in C\},$$

from which construction quickly follows. Consistency follows as in the singleton case. For halting, note $C \cap F = \{\langle \mathbf{a} \rangle \circ \sigma \in F \mid \langle \mathbf{a} \rangle \in C, |\sigma| = 1\} = \bigcup_{Q \in \mathfrak{G}_C} C \circ Q$.

For a trace set A to be computable, we must impose restrictions on \equiv_A . Most trivially, every test in \mathfrak{q} and operation in \mathfrak{p} must be computable. Similarly, every \mathfrak{F}_C and \mathfrak{G}_C must be finite, since arbitrary unions of computable tests (i.e. computable sets) need not be computable. To avoid the problem of ENUM_N , we must make one further restriction: \equiv_A must have a finite number of equivalence classes, reflecting the finitary control mechanism of classical accounts. This substantial restriction is reminiscent of the Myhill-Nerode Theorem, which establishes that a language is regular if, and only if, a certain equivalence relation on that language has a finite number of equivalence classes [12]. Regular languages are less powerful than computable languages, so this similarity raises the concern that imposing a finite control equivalence over-reaches. We show next it does not, by establishing that $\text{RUN}_{\mathfrak{M}}$ is finite control computable for every Turing machine \mathfrak{M} .

Definition 3 (Finite control/finite control computable). *A trace set A for \mathfrak{q} and \mathfrak{p} is finite control if \boxed{A}/\equiv_A is finite. A is finite control computable if, in addition, every $Q \in \mathfrak{q}$ and $P \in \mathfrak{p}$ is computable, and both \mathfrak{q} and \mathfrak{p} are finite.*

Theorem 1. *If $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$ is a Turing machine then $\text{RUN}_{\mathfrak{M}}$ is finite control computable.*

Proof. For each $\sigma \in \text{RUN}_{\mathfrak{M}}$ there is $\sigma^C \in \text{CRUN}_{\mathfrak{M}}$ such that if $\sigma[\alpha] = [u_\alpha | v_\alpha]$ then $\sigma^C[\alpha] = [u_\alpha \langle t_\alpha \rangle v_\alpha]$ with $t_\alpha \in S$ for all $\alpha < |\sigma|$. Define \equiv by taking $\sigma[0, \alpha + 1] \equiv \tau[0, \beta + 1]$ iff $\sigma^C[\alpha] = [u \langle t \rangle v]$ and $\tau^C[\beta] = [w \langle t \rangle x]$ for some $t \in S$. $\sigma^C[0] = [u \langle s_0 \rangle v]$ for some u, v for all $\sigma \in \text{RUN}_{\mathfrak{M}}$, so starting state holds. For $\mathfrak{a} \in \Gamma$ define

$$\begin{aligned} \text{TT}_{\mathfrak{a}} &:= \{([u | av]) \mid u, v \in \Gamma^*\} \\ \text{TM}_{\mathfrak{a}, \text{R}} &:= \{([ub | cv], [uba | v]) \mid u, v \in \Gamma^*, b \in \Gamma\} \\ \text{TM}_{\mathfrak{a}, \text{L}} &:= \{([ub | cv], [u | bav]) \mid u, v \in \Gamma^*, b, c \in \Gamma\}. \end{aligned}$$

To verify Construction and Halting, let $C \in \boxed{\text{RUN}_{\mathfrak{M}}}/\equiv_{\text{RUN}_{\mathfrak{M}}}$ correspond to internal state $s \in S$. Take

$$\begin{aligned} \mathfrak{F}_C &= \{ \langle \text{TT}_{\mathfrak{a}}, \text{TM}_{\mathfrak{b}, \text{D}} \rangle \mid \delta(s, \mathfrak{a}) = \langle t, \mathfrak{b}, \text{D} \rangle \text{ for some } t \in S, \text{D} \in \{\text{L}, \text{R}\} \} \\ \mathfrak{G}_C &= \{ \text{TT}_{\mathfrak{a}} \mid \delta(s, \mathfrak{a}) \text{ is undefined} \}. \end{aligned}$$

Composition and halting are straightforward. For consistency, if $\langle \text{TT}_{\mathfrak{a}}, \text{TM}_{\mathfrak{b}, \text{D}} \rangle \in \mathfrak{F}_C$ and $\delta(s, \mathfrak{a}) = \langle t, \mathfrak{b}, \text{D} \rangle$ take the equivalence class corresponding to t as D . As Γ is finite, finitely many tests TT and operations TM are needed. As each equivalence class corresponds to some $s \in S$, and S is finite, we have that \boxed{A}/\equiv_A is finite.

The control equivalence defined in Theorem 1 is very natural for $\text{RUN}_{\mathfrak{M}}$. Henceforth, we will treat $\text{RUN}_{\mathfrak{M}}$ as a trace set with this control equivalence.

3 Carrying Out Trace Sets

Theorem 1 establishes that Turing computability implies finite control computability, at least over the domain of machine tapes. We now ask: is every finite control computable trace set the execution trace set of some Turing machine? The answer is no, as it should be for two reasons. First, while we have made no attempt to analyse computable operations and tests, we assume they need not be restricted to the sorts of fine-grained transitions in the Turing machine model. One of the benefits of the trace set framework is the generality afforded by using arbitrary computable tests and operations. By the Church-Turing Thesis, Turing machines can complete such operations, but may require multiple steps to do so. There is an intuitive sense that a trace set using arbitrary computable operations could be carried out by some \mathfrak{M} by using extra steps to compute the complex operations. In this section, we seek to make this notion of *carrying out* precise, and to show that every finite control computable trace set over machine tapes can be carried out by some Turing machine.

The second reason that finite control computable trace sets need not be Turing machine execution trace sets is that trace sets need not operate over the domain of machine tapes. A finite control computable trace set over \mathbb{N} cannot be the execution trace set for some Turing machine (at least, absent a suitable encoding of machine tapes). Addressing this requires an account of encoding beyond the scope of this paper. More generally, trace sets can be defined over non-denumerable domains.

3.1 Expansion Mapping

Intuitively, for a Turing machine \mathfrak{M} to carry out a task F , $\text{RUN}_{\mathfrak{M}}$ should consist of all sequences in F , but with each sequence *expanded* to allow \mathfrak{M} to carry out complex operations. In other words, every $\sigma \in F$ must be a subsequence of some $\tau \in \text{RUN}_{\mathfrak{M}}$. $\text{RUN}_{\mathfrak{M}}$ “fills in the gaps” in F using tests like TT_a and operations like $\text{TM}_{a,R}$. This idea has pedigree from process algebra and game semantics, where *silent*, *hidden* or *unimportant* moves are used to better highlight the major steps, or external behaviour, of a computation [1, 3, 10].

More generally, an expansion mapping from F to G requires a bijection matching each $\sigma \in F$ to an expansion $f(\sigma) \in G$. We must also map execution trace points of F to execution trace points of G to ensure that the stages of a computation are mapped consistently. This is important if F is not fully deterministic.

Definition 4 (Expansion mapping). *An expansion mapping from F to G is a pair (f, h) satisfying*

Pairing $f : F \rightarrow G$ is bijective, where $|\sigma| < \omega$ implies $|f(\sigma)| < \omega$; and

Expansion $\sigma \ll f(\sigma)$ via some g_σ for all $\sigma \in F$; and

Commutativity $h : \boxed{F} \rightarrow \boxed{G}$ is an injective function such that if $\sigma \in F$ and $\alpha < |\sigma|$ then $h(\sigma[0, \alpha + 1]) = f(\sigma)[0, g_\sigma(\alpha) + 1]$.

The bijection between sequences is provided by f , and h ensures consistency in the mapping of trace points. The following lemma details this interaction.

Lemma 2. *Let (f, h) be an expansion mapping from \mathbb{F} to \mathbb{G} and let $\sigma \in \boxed{\mathbb{F}}$ with $\sigma = \tau[0, |\sigma|]$ for some $\tau \in \mathbb{F}$.*

1. $\sigma \ll h(\sigma)$ via $g_\tau \upharpoonright |\sigma|$.
2. $\sigma[-1] = h(\sigma)[-1]$.
3. $|h(\sigma)| = g_\tau(|\sigma| - 1) + 1$.
4. $h(\sigma)[0, \alpha] = f(\tau)[0, \alpha]$ for $\alpha \leq |h(\sigma)|$.
5. If $\alpha < |\sigma|$ then $h(\sigma[0, \alpha + 1]) = h(\sigma)[0, g_\tau(\alpha) + 1]$.
6. Let $0 < \alpha, \beta \leq |\sigma|$. Then $\alpha < \beta$ iff $|h(\sigma[0, \alpha])| < |h(\sigma[0, \beta])|$.

Proof.

1. Follows from commutativity, and the fact that g_τ is strictly increasing.
2. $h(\sigma) = f(\tau)[0, g_\tau(|\sigma| - 1) + 1]$ and $\sigma[-1] = \tau[|\sigma| - 1] = f(\tau)[g_\tau(|\sigma| - 1)]$, so

$$\sigma[-1] = f(\tau)[g_\tau(|\sigma| - 1)] = f(\tau)[0, g_\tau(|\sigma| - 1) + 1][-1] = h(\sigma)[-1].$$

3. $|h(\sigma)| = |h(\tau[0, |\sigma|])| = |f(\tau)[0, g_\tau(|\sigma| - 1) + 1]| = g_\tau(|\sigma| - 1) + 1$.
4. $h(\sigma)[0, \alpha] = f(\tau)[0, g_\tau(|\sigma| - 1) + 1][0, \alpha] = f(\tau)[0, \alpha]$ since $\alpha \leq g_\tau(|\sigma| - 1) + 1 = |h(\sigma)|$.
5. Since $h(\sigma) = f(\tau)[0, g_\tau(|\sigma| - 1) + 1]$ and $\alpha \leq |\sigma| - 1$ we have

$$\begin{aligned} h(\sigma[0, \alpha + 1]) &= f(\tau)[0, g_\tau(\alpha) + 1] \\ &= f(\tau)[0, g_\tau(|\sigma| - 1) + 1][0, g_\tau(\alpha) + 1] \\ &= h(\sigma)[0, g_\tau(\alpha) + 1]. \end{aligned}$$

6. $|\sigma[0, \alpha]| = \alpha$ so applying Lemma 2.3 $|h(\sigma[0, \alpha])| = g_\tau(\alpha - 1) + 1$. Similarly $|h(\sigma[0, \beta])| = g_\tau(\beta - 1) + 1$. g_τ is strictly increasing, and so $\alpha < \beta$ iff $g_\tau(\alpha - 1) < g_\tau(\beta - 1)$ iff $g_\tau(\alpha - 1) + 1 < g_\tau(\beta - 1) + 1$ iff $|h(\sigma[0, \alpha])| < |h(\sigma[0, \beta])|$.

When \mathbb{F} and \mathbb{G} are fully deterministic, we need only provide a bijection from \mathbb{F} to \mathbb{G} satisfying *pairing* and *expansion*.

Lemma 3. *Let \mathbb{F} and \mathbb{G} be fully deterministic and $f : \mathbb{F} \rightarrow \mathbb{G}$ satisfy pairing and expansion. Then the mapping $h : \boxed{\mathbb{F}} \rightarrow \boxed{\mathbb{G}}$ defined by $h(\sigma[0, \alpha + 1]) = f(\sigma)[0, g_\sigma(\alpha) + 1]$ for all $\sigma \in \mathbb{F}$ and $\alpha < |\sigma|$ is an injective function, and (f, h) is an expansion mapping from \mathbb{F} to \mathbb{G} .*

Proof. If h is indeed an injective function then (f, h) is trivially an expansion mapping. To see h is well-defined, suppose $\sigma[0, \alpha + 1] = \tau[0, \alpha + 1]$ for $\sigma, \tau \in \mathbb{F}$. Since \mathbb{F} is fully deterministic, $\sigma = \tau$ and so

$$h(\sigma[0, \alpha + 1]) = f(\sigma)[0, g_\sigma(\alpha) + 1] = f(\tau)[0, g_\tau(\alpha) + 1] = h(\tau[0, \alpha + 1]).$$

For injectivity, suppose $h(\sigma[0, \alpha + 1]) = h(\tau[0, \beta + 1])$ where $\sigma, \tau \in \mathbb{F}$, $\alpha < |\sigma|$ and $\beta < |\tau|$. By commutativity, $f(\sigma)[0, g_\sigma(\alpha) + 1] = f(\tau)[0, g_\tau(\beta) + 1]$ and since \mathbb{G} is fully deterministic $f(\sigma) = f(\tau)$. By injectivity of f , $\sigma = \tau$.

Lemma 4. *Let (f, h) be an expansion mapping from F to G with G fully deterministic.*

1. F is fully deterministic.
2. If $\sigma, \tau \in F$ with $h(\sigma[0, \alpha]) = f(\tau)[0, \beta]$ then $\sigma = \tau$.
3. If $\sigma \in \boxed{F}$, $\tau \in F$ with $h(\sigma) = f(\tau)[0, \alpha]$ then $\sigma = \tau[0, |\sigma|]$.

Proof.

1. Let $\sigma, \tau \in F$, and suppose $\sigma[0, 1] = \tau[0, 1]$. Then $h(\sigma[0, 1]) = h(\tau[0, 1])$ and by commutativity $f(\sigma)[0, g_\sigma(0) + 1] = f(\tau)[0, g_\tau(0) + 1]$. $f(\sigma), f(\tau) \in G$ so since G is fully deterministic $f(\sigma) = f(\tau)$ and by injectivity $\sigma = \tau$.
2. We have $h(\sigma[0, \alpha]) = f(\tau)[0, \beta]$ implies $f(\sigma)[0, g_\sigma(\alpha - 1) + 1] = f(\tau)[0, \beta]$ via commutativity. Since $f(\sigma), f(\tau) \in G$ and G is fully deterministic, $f(\sigma) = f(\tau)$, so by injectivity $\sigma = \tau$.
3. Since $\sigma \in \boxed{F}$, there is $\sigma' \in F$ with $\sigma'[0, |\sigma|] = \sigma$. Using commutativity,

$$h(\sigma) = h(\sigma'[0, |\sigma|]) = f(\sigma')[0, g_{\sigma'}(|\sigma| - 1) + 1] = f(\tau)[0, \alpha].$$

Since G is fully deterministic and $f(\sigma'), f(\tau) \in G$, this means $f(\sigma') = f(\tau)$. By injectivity, $\sigma' = \tau$ so $\tau[0, |\sigma|] = \sigma$, as required.

Unfortunately, the mere existence of an expansion mapping is not sufficient to say one task carries out another, as the following example shows.

Example 1. Take $\text{PRIMES} := \{([11|B], [111|B], [11111|B], [1111111|B], \dots)\}$, the task of enumerating all prime numbers in increasing order as unary prefixes of a Turing machine tape. This is a singleton task, as there is only one possible execution trace. Consider a modified Turing machine $\mathfrak{M} = \langle \{s\}, \{1, B\}, \delta, s \rangle$, which always starts with a blank tape $[B]$ and has $\delta(s, 1) = \delta(s, B) = \langle 1, R, s \rangle$. \mathfrak{M} endlessly prints 1s, so $\text{RUN}_{\mathfrak{M}} = \{([|B], [1|B], [11|B], [111|B], [1111|B], \dots)\}$. There is a unique $f : \text{PRIMES} \rightarrow \text{RUN}_{\mathfrak{M}}$, and it satisfies *pairing* and *expansion*. By Lemma 3 there is an expansion mapping from PRIMES to $\text{RUN}_{\mathfrak{M}}$.

Example 1 is problematic because a list of all natural numbers is not a solution to the task of listing all primes; \mathfrak{M} does not carry out PRIMES . However, we might accept such an enumeration if the prime entries were highlighted in some manner. An expansion mapping from F to G does not allow for this sort of highlighting; absent the function g_σ there is no way to know what subsequence of $f(\sigma) \in G$ appears in F . For \mathfrak{M} to properly carry out A , we need some way of identifying which subsequence of each $\tau \in \text{RUN}_{\mathfrak{M}}$ corresponds to a sequence in A .

3.2 Carrying Out

A simple solution to the problem presented by Example 1 is possible if we change our focus from tasks to trace sets. The “highlighting” discussed above can be achieved with equivalence classes. If B carries out A then each $C \in \boxed{A}/\simeq_A$ must be mapped by an expansion mapping onto some $D \in \boxed{B}/\simeq_B$. That way, we can identify the subsequence of $\tau \in B$ corresponding to a sequence in A by taking the $\tau[\alpha]$ for which $\tau[0, \alpha + 1] \in h(C)$ for some $C \in \boxed{A}/\simeq_A$.

Definition 5 (Carrying out). We say B carries out A if there is an expansion mapping (f, h) from A to B such that $C \in \boxed{A}/\leftarrow_A$ iff $h(C) \in \boxed{B}/\leftarrow_B$.

We return to our main goal for this section, showing that every finite control computable trace set over machine tapes can be carried out by some Turing machine.

Theorem 2. Let A be a trace set for q and p such that A is finite control and fully deterministic, q and p are both finite and Turing computable, and there is a finite alphabet Γ such that

1. for every $\sigma \in A$, $\sigma[0] = [u|v]$ for some $u \in \Gamma^*$, $v \in \Gamma^+$;
2. for every $u \in \Gamma^*$, $v \in \Gamma^+$ there exists $\sigma \in A$ with $\sigma[0] = [u|v]$.

Then there is a Turing machine \mathfrak{M} such that $\text{RUN}_{\mathfrak{M}}$ carries out A .

Proof (idea). We directly construct a Turing machine \mathfrak{M} which carries out A . \mathfrak{M} contains an internal state s_C corresponding to each $C \in \boxed{A}/\leftarrow_A$. In these states, \mathfrak{M} simulates the Turing computable tests used in construction and halting. If a construction test succeeds, \mathfrak{M} simulates the corresponding operation and updates the tape contents with the result. If a halting test succeeds, \mathfrak{M} halts. An expansion mapping can be given by mapping each $C \in \boxed{A}/\leftarrow_A$ onto the $D \in \boxed{\text{RUN}_{\mathfrak{M}}}/\leftarrow_{\text{RUN}_{\mathfrak{M}}}$ matching s_C . For details, see the technical appendix.

On its own, this result is unremarkable, since we have not directly addressed the triviality concerns raised by Example 1. We must justify the definition of carrying out by showing Theorem 2’s converse: that if \mathfrak{M} carries out A then A is finite control computable. This provides a tight fit between Turing machines, carrying out and finite control computable trace sets over machine tapes.

$\text{RUN}_{\mathfrak{M}}$ is fully deterministic for any \mathfrak{M} , providing further properties.

Lemma 5. Let B carry out A via (f, h) , where B is fully deterministic. If $\sigma \in A$ and $f(\sigma)[0, \beta + 1] \in h(C)$ for some $C \in \boxed{A}/\leftarrow_A$ then there is $\alpha < |\sigma|$ with $g_\sigma(\alpha) = \beta$ and $f(\sigma)[0, \beta + 1] = h(\sigma[0, \alpha + 1])$.

Proof. If $f(\sigma)[0, \beta + 1] \in h(C)$ then $f(\sigma)[0, \beta + 1] = h(\tau)$ for some $\tau \in C$. Applying Lemma 4.3 $\sigma[0, |\tau|] = \tau$, with $|\tau| \leq |\sigma|$. Thus $f(\sigma)[0, \beta + 1] = h(\tau) = h(\sigma[0, |\tau|])$, as required.

Lemma 5 allows us to “reverse” expansion mappings. In particular, it ensures that the “highlighting” discussed above works as intended.

Corollary 1. Let B carry out A via (f, h) , where B is fully deterministic, and let $\sigma \in A$ and $\alpha + 1 < |\sigma|$. Then for all $C \in \boxed{A}/\leftarrow_A$ and β ,

$$g_\sigma(\alpha) < \beta < g_\sigma(\alpha + 1) \quad \Rightarrow \quad f(\sigma)[0, \beta + 1] \notin h(C).$$

Proof. By contradiction. Suppose $f(\sigma)[0, \beta + 1] \in h(C)$. As $g_\sigma(\alpha + 1) < |f(\sigma)|$, $|f(\sigma)[0, g_\sigma(\alpha + 1) + 1]| = g_\sigma(\alpha + 1) + 1$. Thus by assumption,

$$|f(\sigma)[0, g_\sigma(\alpha + 1)]| < |f(\sigma)[0, \beta + 1]| < |f(\sigma)[0, g_\sigma(\alpha + 1) + 1]|.$$

By Lemma 5, $f(\sigma)[0, \beta + 1] = h(\sigma[0, \gamma])$ for some γ , and by commutativity

$$|h(\sigma[0, \alpha + 1])| < |h(\sigma[0, \gamma])| < |h(\sigma[0, \alpha + 2])|.$$

Applying Lemma 2.6 we conclude $\alpha + 1 < \gamma < \alpha + 2$, which is impossible since γ is an ordinal. Thus, $f(\sigma)[0, \beta + 1] \notin h(C)$.

Lemma 6. *If \mathbb{B} carries out \mathbb{A} and \mathbb{B} is finite control then \mathbb{A} is finite control.*

Proof. Let (f, h) be the expansion mapping in question. Take $C, D \in \boxed{\mathbb{A}}/\simeq_{\mathbb{A}}$ with $C \neq D$. Then $h(C), h(D) \in \boxed{\mathbb{B}}/\simeq_{\mathbb{B}}$ and by injectivity of h , $h(C) \neq h(D)$. Thus each $C \in \boxed{\mathbb{A}}/\simeq_{\mathbb{A}}$ is mapped by h onto a distinct equivalence class in $\boxed{\mathbb{B}}/\simeq_{\mathbb{B}}$. By the pigeonhole principle, $|\boxed{\mathbb{A}}/\simeq_{\mathbb{A}}| \leq |\boxed{\mathbb{B}}/\simeq_{\mathbb{B}}|$, so $\boxed{\mathbb{A}}/\simeq_{\mathbb{A}}$ is finite.

Theorem 3. *Let \mathbb{A} be a trace set and $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$ be a Turing machine. If $\text{RUN}_{\mathfrak{M}}$ carries out \mathbb{A} then \mathbb{A} is finite control computable.*

Proof (Idea). That \mathbb{A} is finite control is an immediate consequence of Lemma 6. For computability (i.e., $\simeq_{\mathbb{A}}$ is a control equivalence under some finite, Turing computable \mathfrak{q} and \mathfrak{p}), note that if $C \in \boxed{\mathbb{A}}/\simeq_{\mathbb{A}}$ then $h(C) = \{\sigma \in \text{RUN}_{\mathfrak{M}} \mid \sigma^C[-1] = \langle s, \sigma[-1] \rangle\}$ for some $s \in S$. This gives a set $S_{\mathbb{A}} \subseteq S$ of states corresponding to equivalence classes of $\simeq_{\mathbb{A}}$. We define the elements of \mathfrak{F}_C and \mathfrak{G}_C by simulating \mathfrak{M} , starting from state s , until another state $t \in S_{\mathbb{A}}$ is reached. See the technical appendix for full details.

Note in passing that every operation \mathbb{P} has a trivial trace set using $\mathfrak{p} = \{\mathbb{P}\}$ and $\mathfrak{q} = \{\sigma[0] \mid \sigma \in \mathbb{P}\}$. Theorems 2 and 3 therefore show that for \mathbb{P} over some finite Γ , \mathbb{P} is Turing computable iff some Turing machine carries \mathbb{P} out. In other words, we can define Turing computability via carrying out, rather than an appeal to inputs and outputs.

4 Further Directions

Finite control computability provides a new, general perspective on computable processes. By moving away from computability theory’s traditional focus on single-step functions and operations, the trace set account highlights the importance of the finite control mechanisms used in classical accounts of computability. Since any domain and set of tests and operations can be used, finite control computability provides a highly flexible model, worthy of future investigation.

One direction for future work lies in incorporating an account of *encoding*, as alluded to in Sect. 3. In a similar manner to carrying out, encodings should preserve control equivalence classes. An account of encoding, matched with carrying

out, gives a candidate notion of *implementation*. Non-denumerable domains also demand investigation. The definition of control equivalence allows for results similar to those obtained here for BSS machines and Type 2 Turing machines. The flexibility of control equivalence in allowing for non-deterministic trace sets means they can provide a good model for non-deterministic models. Interactive computing might be captured by expanding the set \mathbf{p} of operations to allow actions by different agents. The restriction of \mathbf{q} and \mathbf{p} to Turing computable tests and operations here is also unnecessary; the trace set model can easily account for oracle machines. By taking a trace set as a sample space in some probability space, it may be possible to account for probabilistic computation. The many possible applications of trace sets show the flexibility of the model.

Arguably, trace sets also provide an insightful lens through which to study algorithms. Many extant accounts of algorithms share ideas with iterator accounts, or treat algorithms as programs specified in some formal programming language [5, 7, 14]. Frequently, these diminish the importance of underdeterminism in algorithms, and differences between algorithms computing the same function. A trace set based account of algorithms may avoid many of these problems, and is a focus for future work.

A Technical Appendix

A.1 Preliminaries

Fix potentially infinite sets \mathbb{G} and \mathbb{S} , and a distinguished element $\mathbf{B} \in \mathbb{G}$. Γ^* is the set of finite strings over any $\Gamma \subseteq \mathbb{G}$, with ϵ the empty string and $\Gamma^+ = \Gamma^* \setminus \{\epsilon\}$. We use $\mathbf{a}, \mathbf{b}, \dots$ for elements of Γ and $\mathbf{u}, \mathbf{v}, \dots$ for elements of Γ^* . Sequences of these symbols indicate concatenation.

Definition 6 (Turing machine). A Turing machine is a tuple $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$, where $S \subseteq \mathbb{S}$ is a finite set; $\Gamma \subseteq \mathbb{G}$ is a finite set with $\mathbf{B} \in \Gamma$; $\delta : S \times \Gamma \rightarrow S \times \Gamma \times \{\mathbf{L}, \mathbf{R}\}$ is a partial function; $s_0 \in S$ is the start state.

A Turing machine tape is a pair of strings over Γ , written $[\mathbf{u}|\mathbf{v}]$ with $|$ indicating the position of an imagined *read/write head*. A blank tape is denoted $[\mathbf{B}]$. The set of all possible Turing machine tapes is $\mathcal{T} := \{[\mathbf{u}|\mathbf{v}] \mid \mathbf{u} \in \mathbb{G}^*, \mathbf{v} \in \mathbb{G}^+\}$. A configuration of \mathfrak{M} is a pair $[\mathbf{u}\langle s \rangle \mathbf{a}\mathbf{v}]$ where $s \in S$ and $[\mathbf{u}|\mathbf{a}\mathbf{v}]$ is a machine tape. We also sometimes write $\langle s, [\mathbf{u}|\mathbf{a}\mathbf{v}] \rangle$ for configurations. \mathfrak{M} gives rise to a partial transition function $\Delta_{\mathfrak{M}}$ on its configurations satisfying

$$\Delta_{\mathfrak{M}}([\mathbf{u}\langle s \rangle \mathbf{a}\mathbf{v}]) = \begin{cases} [\mathbf{u}\mathbf{b}\langle u \rangle \mathbf{v}] & \text{if } \delta(s, \mathbf{a}) = \langle t, \mathbf{b}, \mathbf{R} \rangle \text{ and } \mathbf{v} \neq \epsilon \\ [\mathbf{u}\mathbf{b}\langle u \rangle \mathbf{B}] & \text{if } \delta(s, \mathbf{a}) = \langle t, \mathbf{b}, \mathbf{R} \rangle \text{ and } \mathbf{v} = \epsilon \\ [\mathbf{w}\langle u \rangle \mathbf{c}\mathbf{b}\mathbf{v}] & \text{if } \delta(s, \mathbf{a}) = \langle t, \mathbf{b}, \mathbf{L} \rangle \text{ and } \mathbf{w} = \mathbf{u}\mathbf{c} \\ [\langle u \rangle \mathbf{B}\mathbf{b}\mathbf{x}] & \text{if } \delta(s, \mathbf{a}) = \langle t, \mathbf{b}, \mathbf{L} \rangle \text{ and } \mathbf{w} = \epsilon; \end{cases}$$

$\Delta_{\mathfrak{M}}([\mathbf{u}\langle s \rangle \mathbf{a}\mathbf{v}])$ is undefined if $\delta(s, \mathbf{a})$ is. $\Delta_{\mathfrak{M}}$ generates a set of sequences of configurations labelled $\text{CRUN}_{\mathfrak{M}}$. If $\Delta_{\mathfrak{M}}(\sigma[\alpha])$ is undefined then $|\sigma| = \alpha + 1$.

$$\text{CRUN}_{\mathfrak{M}} := \{([\mathbf{u}\langle s_0 \rangle \mathbf{v}], \Delta_{\mathfrak{M}}([\mathbf{u}\langle s_0 \rangle \mathbf{v}]), \Delta_{\mathfrak{M}}^2([\mathbf{u}\langle s_0 \rangle \mathbf{v}]), \dots\} \mid \mathbf{u} \in \Gamma^*, \mathbf{v} \in \Gamma^+\}$$

$$\text{SRUN}_{\mathfrak{M}} := \{\sigma \mid \exists \tau \in \text{CRUN}_{\mathfrak{M}} \text{ with } \tau[\alpha] = \langle \sigma[\alpha], [\mathbf{u}\langle \mathbf{v} \rangle]_{\alpha} \rangle \text{ for } \alpha < |\sigma| = |\tau|\}$$

$$\text{RUN}_{\mathfrak{M}} := \{\sigma \mid \exists \tau \in \text{CRUN}_{\mathfrak{M}} \text{ with } \tau[\alpha] = \langle s_{\alpha}, \sigma[\alpha] \rangle \text{ for } \alpha < |\sigma| = |\tau|\}.$$

Each $\sigma \in \text{RUN}_{\mathfrak{M}}$ corresponds to sequences $\sigma^C \in \text{CRUN}_{\mathfrak{M}}$ and $\sigma^S \in \text{SRUN}_{\mathfrak{M}}$ satisfying $\sigma^C[\alpha] = \langle \sigma^S[\alpha], \tau[\alpha] \rangle$. Write $\mathfrak{M}([\mathbf{u}\langle \mathbf{v} \rangle]) = [\mathbf{w}\langle \mathbf{x} \rangle]$ if there is $\sigma \in \text{RUN}_{\mathfrak{M}}$ with $\sigma[0] = [\mathbf{u}\langle \mathbf{v} \rangle]$ and $\sigma[-1] = [\mathbf{w}\langle \mathbf{x} \rangle]$.

A.2 Proof of Theorem 2

Proof. Let $\overline{\mathbf{A}}/\equiv_{\mathbf{A}} = \{C_0, C_1, \dots, C_n\}$, with C_0 the starting state. We will construct a Turing machine \mathfrak{M} by considering the tests and operations performed at each C_i . Since \mathfrak{q} and \mathfrak{p} are Turing computable, \mathbf{A} is a trace set over \mathcal{T} .

Take $0 \leq i \leq n$. Let $\mathfrak{F}_i \subseteq \mathfrak{q} \times \mathfrak{p}$ be the construction set for C_i and $\mathfrak{G}_i \subseteq \mathfrak{q}$ the halting set. Since \mathfrak{q} and \mathfrak{p} are both finite, so are \mathfrak{F}_i and \mathfrak{G}_i . Suppose

$$\mathfrak{F}_i = \{\langle Q_{i,0}, P_{i,0} \rangle, \langle Q_{i,1}, P_{i,1} \rangle, \dots, \langle Q_{i,k_i-1}, P_{i,k_i-1} \rangle\}$$

$$\mathfrak{G}_i = \{Q_{i,k_i}, Q_{i,k_i+1}, \dots, Q_{i,\ell_i-1}\}$$

for some $0 \leq k_i \leq \ell_i$. Each test and operation is Turing computable, so $Q_{i,j}$ is associated with some Turing machine $\mathfrak{N}_{i,j}$ and $P_{i,j}$ some machine $\mathfrak{D}_{i,j}$. Since \mathbf{A} is fully deterministic, we may assume $Q_{i,j} \cap Q_{i,m} = \emptyset$ when $j \neq m$.² By *consistency*, each pair $\langle Q_{i,j}, P_{i,j} \rangle$ leads to a new equivalence class $C_{c(i,j)}$.

Define a new machine $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$ as follows:

$\mathfrak{M} =$ “On tape contents $[\mathbf{u}\langle \mathbf{v} \rangle]$ with $\mathbf{u} \in \Gamma^*$, $\mathbf{v} \in \Gamma^+$:

1. Assign $x \leftarrow 0$ and $y \leftarrow [\mathbf{u}\langle \mathbf{v} \rangle]$.
2. Replace the tape contents with y .
3. Run all $\mathfrak{N}_{x,j}$ with $j < \ell_x$ on copies of y in parallel until some $\mathfrak{N}_{x,z}$ halts with output $[[1]]$.
4. If $z < k_x$:
 - a. Run $\mathfrak{D}_{x,z}$ on y .
 - b. Assign $x \leftarrow c(x, z)$ and y to the tape content generated in stage 4a and go to stage 2.
5. If $z \geq k_x$: *halt.*”

There are two meta-variables: x for the currently considered equivalence class of $\overline{\mathbf{A}}/\equiv_{\mathbf{A}}$ and y for the currently considered tape contents. \mathfrak{M} 's operation proper starts by carrying out all the tests required in C_x (Stage 3). By Lemma 1, some test $Q_{x,z}$ will succeed (i.e. output $[[1]]$). If this $Q_{x,z}$ belongs to \mathfrak{F}_x (i.e. $z < k_x$) \mathfrak{M}

² If not, we can define new Turing computable $Q'_{i,j}$ and $Q'_{i,m}$ that do satisfy this requirement by setting $[\mathbf{u}\langle \mathbf{v} \rangle] \in Q'_{i,j}$ iff $\mathfrak{N}_{i,j}([\mathbf{u}\langle \mathbf{v} \rangle]) = [[1]] \neq \mathfrak{N}_{i,m}([\mathbf{u}\langle \mathbf{v} \rangle])$ or $\mathfrak{N}_{i,j}([\mathbf{u}\langle \mathbf{v} \rangle]) = [[1]] = \mathfrak{N}_{i,m}([\mathbf{u}\langle \mathbf{v} \rangle])$ and $\mathfrak{N}_{i,j}([\mathbf{u}\langle \mathbf{v} \rangle])$ halts before $\mathfrak{N}_{i,m}([\mathbf{u}\langle \mathbf{v} \rangle])$; and similarly for $Q'_{i,m}$.

runs the operation $P_{x,z}$ corresponding to $Q_{x,z}$, stores the result in y and updates x to the new equivalence class from *consistency* and the process is repeated (Stage 4). Otherwise, the sequence halts and so \mathfrak{M} halts (Stage 5). Since A is finite control, x takes finitely many values and can be encoded in the states of \mathfrak{M} . In implementing \mathfrak{M} , we require for each $i \leq n$ a unique state $s_i \in S$ corresponding to the end of Stage 2 with $x = i$. This preserves x 's value through Stage 2.

Let $S_A = \{s_0, \dots, s_n\}$. Intuitively, we will define an expansion mapping (f, h) associating each $C_i \in \boxed{A} / \simeq_A$ with s_i , satisfying $h(C_i) = \{\tau \in \boxed{\text{RUN}_{\mathfrak{M}}} \mid \tau^S[-1] = s_i\} \in \boxed{\text{RUN}_{\mathfrak{M}}} / \simeq_{\text{RUN}_{\mathfrak{M}}}$ for all $i \leq n$. Take $f : F \rightarrow \text{RUN}_{\mathfrak{M}}$ so that $f(\sigma)[0] = \sigma[0]$ for all $\sigma \in F$. Since F and $\text{RUN}_{\mathfrak{M}}$ are both fully deterministic, starting with the same domain, this is a well-defined bijection. If we can show $\sigma \ll f(\sigma)$ and $|\sigma| < \omega$ implies $|f(\sigma)| < \omega$ then we can extend f to an expansion mapping (f, h) using Lemma 3. To that end, take $\sigma \in F$ and let $\tau = f(\sigma)$. $\tau \in \text{RUN}_{\mathfrak{M}}$ so is associated with sequences $\tau^S \in \text{SRUN}_{\mathfrak{M}}$ and $\tau^C \in \text{CRUN}_{\mathfrak{M}}$. Define $g_\sigma : |\sigma| \rightarrow |\tau|$ so that:

1. $g_\sigma(0)$ is the least ordinal β for which $\tau^S[\beta] \in S_A$;
2. for all α with $\alpha + 1 < |\sigma|$, $g_\sigma(\alpha + 1)$ is the least ordinal β such that $g_\sigma(\alpha) < \beta$ and $\tau^S[\beta] \in S_A$;

Clearly, g_σ is strictly increasing. We can show by induction that (I) $\sigma[\alpha] = \tau[g_\sigma(\alpha)]$ and (II) $\sigma[0, \alpha + 1] \in C_i$ iff $\tau^S[g_\sigma(\alpha)] = s_i$ for all $\alpha < |\sigma|$.

Thus for each $\sigma \in A$ we can define a strictly increasing $g_\sigma : |\sigma| \rightarrow |f(\sigma)|$ such that $\sigma[\alpha] = f(\sigma)[g_\sigma(\alpha)]$ for all $\alpha < |\sigma|$, so $\sigma \ll f(\sigma)$ for all $\sigma \in A$. Next we show that $|\sigma| < \omega$ implies $|f(\sigma)| < \omega$. Suppose $\sigma \in A$ with $|\sigma| < \omega$. Then $\sigma \in C_i \cap A$ for some i and so by *halting* there is $Q_{i,j}$ with $k_i \leq j < \ell_i$ such that $\sigma[-1] \in Q_{i,j}$. Thus $\mathfrak{N}_{i,j}(\sigma[-1]) = [1]$. Let $\tau = f(\sigma)$. As established by Claims (I) and (II) above, $\tau[g_\sigma(|\sigma| - 1)] = \sigma[-1]$ and $\tau^S[g_\sigma(|\sigma| - 1)] = s_i$. So in $\tau^C[g_\sigma(|\sigma| - 1)]$ \mathfrak{M} has just finished Stage 2 with $x = i$. Since $\mathfrak{N}_{i,j}(\sigma[\alpha]) = [1]$, when \mathfrak{M} moves to Stage 3 it sets $z \leftarrow j$. Thus $z \geq k_x$ and so \mathfrak{M} next moves to Stage 5, halting. Thus, τ is finite and f meets the conditions of Lemma 3 and can be extended to an expansion mapping (f, h) from A to $\text{RUN}_{\mathfrak{M}}$.

Finally, we turn to verifying that $f(C_i) \in \boxed{\text{RUN}_{\mathfrak{M}}} / \simeq_{\text{RUN}_{\mathfrak{M}}}$ for all $C_i \in \boxed{A} / \simeq_A$. It is a quick consequence of Claim (II) that if $\sigma \in C_i$ then $h(\sigma)^S[-1] = s_i$. Conversely, suppose $\tau^S[-1] = s_i$ for some $\tau \in \boxed{\text{RUN}_{\mathfrak{M}}}$. Since f is surjective onto $\text{RUN}_{\mathfrak{M}}$ there is $\sigma \in A$ with $f(\sigma)[0, |\tau|] = \tau$ and $f(\sigma)^S[|\tau| - 1] = s_i$. Let $\alpha = \max\{\beta \mid g_\sigma(\beta) < |\tau| - 1\}$. Using Claim (II), we can see that $\sigma[0, \alpha + 2] \in C_i$. From this we see $f(C_i) = \{\tau \in \boxed{\text{RUN}_{\mathfrak{M}}} \mid \tau = f(\sigma) \text{ for some } \sigma \in C_i\} = \{\tau \in \boxed{\text{RUN}_{\mathfrak{M}}} \mid \tau^S[-1] = s_i\}$. Hence $f(C_i) \in \boxed{\text{RUN}_{\mathfrak{M}}} / \simeq_{\text{RUN}_{\mathfrak{M}}}$ by definition of $\simeq_{\text{RUN}_{\mathfrak{M}}}$.

A.3 Proof of Theorem 3

Definition 7. Let $\mathfrak{M} = \langle S, \Gamma, \delta, s_0 \rangle$ be a Turing machine and $T \subseteq S$. Write $\langle s, [u|v] \rangle \xrightarrow{\mathfrak{M}}_T \langle t, [w|x] \rangle$ if either $\Delta_{\mathfrak{M}}([u(s)v]) = [w(t)x]$ or $\Delta_{\mathfrak{M}}([u(s)v]) = [w'(u)x']$

for some $u \notin T$ with $\langle u, [w'x'] \rangle \xrightarrow[T]{\mathfrak{M}} \langle t, [wx] \rangle$. Write $\langle s, [uv] \rangle \xrightarrow[T]{\mathfrak{M}} \downarrow$ if $\Delta_{\mathfrak{M}}(\langle [u]s \rangle v)$ is undefined or $\Delta_{\mathfrak{M}}(\langle [u]s \rangle v) = [w\langle t \rangle x]$ for $t \notin T$ with $\langle t, [wx] \rangle \xrightarrow[T]{\mathfrak{M}} \downarrow$.

Proof (Theorem 3). Part 1 is immediate from Lemma 6. Suppose $\boxed{A}/\xrightarrow{A} = \{C_0, C_1, \dots, C_n\}$ and $\text{RUN}_{\mathfrak{M}}$ carries out A via (f, h) . For each $0 \leq i \leq n$ there is an $s_i \in S$ with $h(C_i) = \{\tau \in \boxed{\text{RUN}_{\mathfrak{M}}} \mid \tau^S[-1] = s_i\}$. Take $S_A = \{s_0, \dots, s_n\}$.

Let $0 \leq i \leq n$. Take the minimal set $\{D_{i,0}, D_{i,1}, \dots, D_{i,k_i}\} \subseteq \boxed{A}/\xrightarrow{A}$ where $\{\sigma \in \boxed{A} \mid \sigma[0, -1] \in C_i\} \subseteq \bigcup_{0 \leq j \leq k_i} D_{i,j}$. Let $0 \leq j \leq k_i$ and suppose $h(D_{i,j}) = \{\tau \in \boxed{\text{RUN}_{\mathfrak{M}}} \mid \tau^S[-1] = t_{i,j}\}$ (so $t_{i,j} = s_\ell$ for some ℓ). Define $\mathfrak{N}_{i,j}$ and $\mathfrak{D}_{i,j}$ as:

$\mathfrak{N}_{i,j}$ = “On tape contents $[uv]$:

1. Assign $x \leftarrow s_i$ and $y \leftarrow [uv]$.
2. If $\Delta_{\mathfrak{M}}(\langle x, y \rangle)$ is undefined: replace the tape with $[0]$ and *halt*.
3. If $\Delta_{\mathfrak{M}}(\langle x, y \rangle)$ is defined: update $\langle x, y \rangle \leftarrow \Delta_{\mathfrak{M}}(\langle x, y \rangle)$.
4. If $x \in S_A$:
 - a. If $s = t_{i,j}$: replace the tape with $[1]$ and *halt*.
 - b. If $s \neq t_{i,j}$: replace the tape with $[0]$ and *halt*.
5. If $x \notin S_A$: go to stage 2.”

$\mathfrak{D}_{i,j}$ = “On tape contents $[uv]$:

1. Assign $x \leftarrow s_i$ and $y \leftarrow [uv]$.
2. If $\Delta_{\mathfrak{M}}(\langle x, y \rangle)$ is undefined: enter an infinite loop.
3. If $\Delta_{\mathfrak{M}}(\langle x, y \rangle)$ is defined: update $\langle x, y \rangle \leftarrow \Delta_{\mathfrak{M}}(\langle x, y \rangle)$.
4. If $x \in S_A$:
 - a. If $s = t_{i,j}$: replace the tape with y and *halt*.
 - b. If $s \neq t_{i,j}$: enter an infinite loop.
5. If $x \notin S_A$: go to stage 2.”

We have that

$$\begin{aligned} \mathfrak{N}_{i,j}([uv]) = [1] & \quad \text{iff} \quad \langle s_i, [uv] \rangle \xrightarrow[S_A]{\mathfrak{M}} \langle s_j, [wx] \rangle \text{ for some } w, x \\ \mathfrak{D}_{i,j}([uv]) = [wx] & \quad \text{iff} \quad \langle s_i, [uv] \rangle \xrightarrow[S_A]{\mathfrak{M}} \langle s_j, [wx] \rangle. \end{aligned}$$

Let $Q_{i,j} := \{[uv] \mid \text{RUN}_{\mathfrak{N}_{i,j}}([uv]) = [1]\}$. and $P_{i,j} := \{([uv], [wx]) \mid \mathfrak{D}_{i,j}([uv]) = [wx]\}$. Clearly, $Q_{i,j}$ and $P_{i,j}$ are Turing computable. Repeating the above for each $0 \leq j \leq k_i$, take $\mathfrak{F}_i := \{\langle Q_{i,j}, P_{i,j} \rangle \mid 0 \leq j \leq k_i\}$. The halting set \mathfrak{G}_i for C_i is defined similarly. Repeating the same process for each C_i provides \mathfrak{q} and \mathfrak{p} .

In Stage (II), we verify that *construction* holds using \mathfrak{q} and \mathfrak{p} . To do this, let $0 \leq i \leq n$. We start with *composition*. Let $\sigma \in \{\tau \in \boxed{A} \mid \tau[0, -1] \in C_i\}$. Then $\sigma[0, -1] \in C_i$ and $\sigma \in D_{i,j}$ for some j . Take $\tau = h(\sigma)$. We have $h(\sigma[0, -1]) = \tau[0, g_\sigma(|\sigma| - 2) + 1] \in h(C_i)$ and $\tau \in h(D_{i,j})$. Thus we know $\tau^S[g_\sigma(|\sigma| - 2)] = s_i$ and $\tau^S[-1] = \tau[g_\sigma(|\sigma| - 1)] = t_{i,j}$. We claim that

$$\langle \tau^S[g_\sigma(|\sigma| - 2)], \tau[g_\sigma(|\sigma| - 2)] \rangle \xrightarrow[S_A]{\mathfrak{M}} \langle \tau^S[g_\sigma(|\sigma| - 1)], \tau[g_\sigma(|\sigma| - 1)] \rangle.$$

If not, then there must be some $g_\sigma(|\sigma| - 2) < \beta < g_\sigma(|\sigma| - 1)$ with

$$\langle \tau[g_\sigma(|\sigma| - 2)], \tau[g_\sigma(|\sigma| - 2)] \rangle \overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} \langle \tau^S[\beta], \tau[\beta] \rangle$$

and $\tau^S[\beta] \in S_A$. But then $\tau[0, \beta + 1] \in h(C_k)$ for some k , impossible by Corollary 1. Since $\tau[g_\sigma(|\sigma| - 2)] = \sigma[-2]$ and $\tau[g_\sigma(|\sigma| - 1)] = \sigma[-1]$, we conclude that $\langle s_i, \sigma[-2] \rangle \overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} \langle t_{i,j}, \sigma[-1] \rangle$. From this we have $\mathfrak{N}_{i,j}(\sigma[-2]) = [1]$, so $\sigma[-2] \in Q_{i,j}$, and that $\mathfrak{D}_{i,j}(\sigma[-2]) = \sigma[-1]$, so $(\sigma[-2], \sigma[-1]) \in P_{i,j}$. We have $\sigma \in C_i \circ Q_{i,j} \circ P_{i,j}$.

In the other direction, suppose for some j that $\sigma \in C_i \circ Q_{i,j} \circ P_{i,j}$, so as above $\sigma[0, -1] \in C_i$, $\mathfrak{N}_{i,j}(\sigma[-2]) = [1]$ and $\mathfrak{D}_{i,j}(\sigma[-2]) = \sigma[-1]$. We wish to show that $\sigma \in \boxed{A}$. Since $\sigma[0, -1] \in C_i \subseteq \boxed{A}$, there is some $\tau \in A$ with $\tau[0, |\sigma| - 1] = \sigma[0, -1]$. If we can establish that $|\sigma| \leq |\tau|$ and $\tau[|\sigma| - 1] = \sigma[-1]$ we will have that $\tau[0, |\sigma|] = \sigma$, so $\sigma \in \boxed{A}$. Since $\mathfrak{D}_{i,j}(\sigma[-2]) = \sigma[-1]$, we have $\langle s_i, \sigma[-2] \rangle \overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} \langle t_{i,j}, \sigma[-1] \rangle$. By *expansion*, $\sigma[-2] = \tau[|\sigma| - 2] = f(\tau)[g_\sigma(|\sigma| - 2)]$. Since $\tau[0, |\sigma| - 1] = \sigma[0, -1] \in C_i$ we know $f(\tau)^S[g_\sigma(|\sigma| - 2)] = h(\tau[0, |\sigma| - 1])^S[-1] = s_i$. Hence we have $f(\tau)^C[g_\sigma(|\sigma| - 2)] = \langle s_i, \sigma[-2] \rangle$ so $f(\tau)^C[g_\sigma(|\sigma| - 2)] \overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} \langle t_{i,j}, \sigma[-1] \rangle$.

It follows there must be $\beta > g_\sigma(|\sigma| - 2)$ with $f(\tau)^C[\beta] = \langle t_{i,j}, \sigma[-1] \rangle$. We then have $f(\tau)[0, \beta + 1] \in h(D_{i,j})$ so by Lemma 5 $f(\tau)[0, \beta + 1] = h(\tau[0, \alpha + 1])$ for some $\alpha < |\tau|$, where $g_\tau(\alpha) = \beta$. Since $g_\tau(|\sigma| - 2) < \beta$, $|\sigma| - 2 < \alpha$. Thus $|\sigma| \leq |\tau|$ and so $g_\tau(|\sigma| - 1)$ is defined. $\beta \geq g_\tau(|\sigma| - 1)$ otherwise $f(\tau)[0, \beta + 1] \notin h(D_{i,j})$ by Corollary 1. $\tau[0, |\sigma|] \in C_k$ for some k , so $f(\tau)^S[g_\tau(|\sigma| - 1)] \in S_A$. Thus $\beta \not\prec g_\tau(|\sigma| - 1)$ or else $f(\tau)^C[g_\sigma(|\sigma| - 2)] \not\overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} f(\tau)^C[\beta]$, contradicting $f(\tau)^C[\beta] = \langle t_{i,j}, \sigma[-1] \rangle$. Thus $g_\tau(|\sigma| - 1) = \beta$. We have $\tau[|\sigma| - 1] = f(\tau)[g_\tau(|\sigma| - 1)] = f(\tau)[\beta] = \sigma[-1]$. Since $\tau[0, |\sigma| - 1] = \sigma[0, -1]$ we conclude $\tau[0, |\sigma|] = \sigma$. Thus $\sigma \in \{v \mid v[0, -1] \in C_i\}$. This establishes *composition*.

Consistency is easy to verify, since if $\sigma \in C_i \circ Q_{i,j} \circ P_{i,j}$ then following the reasoning above where $\tau \in A$ has $\tau[0, |\sigma|] = \sigma$, $h(\sigma) = f(\tau)[0, g_\tau(|\sigma| - 1) + 1] \in h(D_{i,j})$ and since h is injective $\sigma \in D_{i,j}$. Thus, *construction* holds with \mathfrak{q} and \mathfrak{p} . *Halting* follows similar reasoning to *construction*, except that we must establish

$$f(\sigma)^C[g_\sigma(|\sigma| - 1)] \overset{\mathfrak{M}}{\underset{S_A}{\rightsquigarrow}} \downarrow.$$

The details are omitted for reasons of space.

References

1. Abramsky, S.: Information, processes and games. In: Adriaans, P., Benthem, J. (eds.) *Philosophy of Information*. No. 8 in *Handbook of the Philosophy of Science*, pp. 483–550. Elsevier (2008)
2. Blackburn, P., Rijke, M., Venema, Y.: *Modal Logic*. No. 53 in *Cambridge Tracts in Theoretical Computer Science*, 4th edn. Cambridge University Press (2002)
3. Dowek, G.: Execution traces and reduction sequences. lsv.fr/~dowek/Publi/traces.pdf

4. Gandy, R.: Church's thesis and principles for mechanisms. In: Barwise, J., Keisler, H.J., Kunen, K. (eds.) *Studies in Logic and the Foundations of Mathematics, The Kleene Symposium*, vol. 101, pp. 123–148. Elsevier (1980). <http://www.sciencedirect.com/science/article/pii/S0049237X08712576>
5. Gurevich, Y.: Sequential abstract-state machines capture sequential algorithms. **1**(1), 77–111 (2000). <http://portal.acm.org/citation.cfm?doid=343369.343384>
6. Harel, D., Kozen, D., Parikh, R.: Process logic: Expressiveness, decidability, completeness. **25**(2), 144–170 (1982). <http://www.sciencedirect.com/science/article/pii/0022000082900034>
7. Huber, H.G.M.: Algorithm and formula. *Commun. ACM* **9**(9), 653–654 (1966)
8. Ju, F., Cui, N., Li, S.: Trace semantics for IPDL. In: van der Hoek, W., Holliday, W.H., Wang, W. (eds.) *LORI 2015. LNCS*, vol. 9394, pp. 169–181. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48561-3_14
9. Kozen, D., Smith, F.: Kleene algebra with tests: completeness and decidability. In: van Dalen, D., Bezem, M. (eds.) *CSL 1996. LNCS*, vol. 1258, pp. 244–259. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63172-0_43
10. Milner, R.: An algebraic definition of simulation between programs. In: *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pp. 481–489. Morgan Kaufmann Publishers Inc. (1971)
11. Moschovakis, Y.N.: The logic of functional recursion. In: Dalla Chiara, M.L., Doets, K., Mundici, D., Bentham, J. (eds.) *Logic and Scientific Methods. Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science)*, vol. 259, pp. 179–207. Springer, Dordrecht (1997). https://doi.org/10.1007/978-94-017-0487-8_10
12. Nerode, A.: Linear automaton transformations. *Proc. Am. Math. Soc.* **9**(4), 541–544 (1958)
13. Pégny, M.: How to make a meaningful comparison of models: the church-turing thesis over the reals. *Minds Mach.* **26**(4), 359–388 (2016). <https://doi.org/10.1007/s11023-016-9407-0>
14. Rapaport, W.J.: What is an algorithm? In: *Philosophy of Computer Science (Draft)*, pp. 229–291. <https://cse.buffalo.edu/~rapaport/Papers/phics.pdf>
15. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. **2**(1), 230–265 (1937). <http://onlinelibrary.wiley.com/doi/10.1112/plms/s2-42.1.230/full>



Axiomatic Reals and Certified Efficient Exact Real Computation

Michal Konečný¹, Sewon Park², and Holger Thies³(✉)

¹ Aston University, Birmingham, UK
m.konecny@aston.ac.uk

² KAIST, Daejeon, Korea
swelite@kaist.ac.kr

³ Kyoto University, Kyoto, Japan
thies.holger.5c@kyoto-u.ac.jp

Abstract. We introduce a new axiomatization of the constructive real numbers in a dependent type theory. Our main motivation is to provide a sound and simple to use backend for verifying algorithms for exact real number computation and the extraction of efficient certified programs from our proofs. We prove the soundness of our formalization with regards to the standard realizability interpretation from computable analysis. We further show how to relate our theory to a classical formalization of the reals to allow certain non-computational parts of correctness proofs to be non-constructive. We demonstrate the feasibility of our theory by implementing it in the Coq proof assistant and present several natural examples. From the examples we can automatically extract Haskell programs that use the exact real computation framework AERN for efficiently performing exact operations on real numbers. In experiments, the extracted programs behave similarly to hand-written implementations in AERN in terms of running time.

Keywords: Constructive real numbers · Formal proofs · Exact real number computation · Program extraction

1 Introduction

Verifying the correctness of software is becoming increasingly important, in particular in safety critical application domains. Often, such programs need to interact in some way with the outside, physical world requiring numerical calculations over the real numbers and other uncountable mathematical entities. While

Holger Thies is supported by JSPS KAKENHI Grant Number JP20K19744. Sewon Park is supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (No. NRF-2016K1A3A7A03950702, NRF-2017R1E1A1A03071032 (MSIT) & No. NRF-2017R1D1A1B05031658 (MOE)).

🇪🇺 This project has received funding from the EU's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 731143.

© Springer Nature Switzerland AG 2021

A. Silva et al. (Eds.): WoLLIC 2021, LNCS 13038, pp. 252–268, 2021.

https://doi.org/10.1007/978-3-030-88853-4_16

proof assistants and formal methods are becoming more mature and are increasingly used in practical applications, verification of numerical programs remains extremely challenging [2]. One difficulty arises from the fact that in practice real numbers are commonly replaced by floating-point approximations, introducing rounding errors and uncertainties that pose additional problems for verification.

While there is active ongoing work on the verification of floating-point arithmetic [4, 15], we here consider a different approach known as *exact real computation*. In exact real computation, real numbers are basic entities that allow exact manipulation without rounding errors. Programs can output finite approximations up to any desired absolute precision. This is often realized by adding a datatype for reals and arithmetic operations on them as primitives in programming languages. Several implementations exist demonstrating the feasibility of the approach [1, 10, 16]. Although less efficient than optimized hardware-based floating-point calculations, implementations in exact real computation are by design more reliable than the former and are thus well-suited for situations where correctness is of high importance. Further, for efficient implementations there is often only a small overhead. However, subtleties of the semantics such as multivaluedness can still make writing correct programs difficult and stronger guarantees of correctness are highly desirable. One of the strongest such guarantees is a computer verified correctness proof e.g. in a proof assistant which however requires a sound model of the semantics. This poses some theoretical challenges as operations such as partial comparisons and multivalued branching are common in exact real computation and need to be computable [18].

Software packages for exact real computation often build on the theoretical framework of computable analysis and the theory of representations [13, 25]. In previous work [11] two of the authors of the present paper worked on verified exact real computation using the Incone library [23], which aims to directly formulate the model of computable analysis in Coq. Incone requires to define computational *realizers*, i.e. functions that work on low-level encodings of the reals e.g. by sequences of rational numbers. Working directly with such encodings facilitates high control over the algorithm and allows fine-grained optimizations. However, algorithms and their correctness proofs depend on the concrete encoding and the approach is therefore less elegant and more labour-intensive than working with a high-level abstract implementation of a real number type. While this issue has also been addressed in Incone by providing an abstract specification of some important real number operations, in this work we chose an even higher level of abstraction. That is, instead of reimplementing and verifying basic real number operations, we trust the implementation of a core of simple real number operations and to verify programs using those operations under the assumption that they are correctly implemented. The basic idea is to axiomatically model sophisticated implementations of exact real computation which exist for many modern programming languages, e.g. AERN [10] for Haskell or iRRAM for C++. This approach also provides a certain amount of independence of the concrete implementation of real numbers and thus allows to easily switch the underlying framework.

More concretely, we define a new constructive axiomatization that models the real numbers in a conceptually similar way as some mature implementations of exact real computation. We formally define our theory on top of a simple type theory inspired by the one used in Coq and prove its soundness with respect to the realizability interpretation used in computable analysis. We also give a theoretical foundation of relating proofs written over a classical theory of real numbers with our real numbers.

There are already several formalizations of real numbers and real analysis in most proof assistants (see e.g. [3] for an overview), including the C-CoRN library [6], a large constructive framework based on Coq setoids. Our axiomatization is different in that it very closely models classical reasoning used in computable analysis and concepts used in practical implementations of exact real computation, such as multivalued operations. We therefore think that it can be appealing to people working in this area.

Our approach further allows to easily map the constructive real type, and its axiomatically defined basic operations such as arithmetic or limits, to corresponding types and operations in an exact real computation framework. Concretely, utilising this mapping and program extraction techniques, we obtain certified programs over an implementation of exact real computation from correctness proofs.

We implemented the theory in the Coq proof assistant and extracted Haskell programs from our proofs using Coq code extraction. In the extracted programs, primitive operations on the reals are mapped to operations in the exact real computation framework AERN [10] which is written and maintained by one of the authors. Our first examples show that the extracted programs perform efficiently, having only a small overhead compared to hand-written implementations.

2 Computable Analysis and Exact Real Computation

In this section, we recap some essential concepts and limitations of computable analysis and exact real computation in order to justify our choice of axioms.

To compute over uncountable mathematical structures such as real numbers exactly, computable analysis takes *assemblies* over Kleene's second algebra (assemblies for short) as the basic data type [8, 22].¹ An assembly is a pair of a set A and a relation $\Vdash \subseteq \mathbb{N}^{\mathbb{N}} \times A$, which is surjective in that $\forall x \in A. \exists \varphi. \varphi \Vdash x$. We call $\varphi \in \mathbb{N}^{\mathbb{N}}$ a realizer of an abstract entity $x \in A$ if $\varphi \Vdash x$ holds. Given two assemblies of A and B , a function $f : A \rightarrow B$ is said to be computable if there is a computable partial function $\tau : \subseteq \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$ that tracks f , i.e. for any $x \in A$ and its realizer φ , $\tau(\varphi)$ is a realizer of $f(x)$.

For real numbers, there is a unique assembly (up to isomorphism in the category of assemblies $\mathbf{Asm}(\mathcal{K}_2)$) that makes the model-theoretic structure [7] of real numbers computable: (1) $0, 1 \in \mathbb{R}$ are computable, (2) field arithmetic

¹ Assemblies are generalizations of represented sets [13, 25] which are exactly the assemblies where the surjective relations are required to be partial surjective functions. The terminology *multi-representation* [20] may be more familiar to some readers.

is computable, (3) the order relation $<$ that is undefined at $\{(x, x) \mid x \in \mathbb{R}\}$ is computable, and (4) the limit operation defined at rapidly converging sequences is computable. An example is the Cauchy reals where φ is a realizer of $x \in \mathbb{R}$ if and only if φ encodes a sequence of rationals converging rapidly towards x . An assembly of reals satisfying the above computability conditions is called *effective*.

An inevitable side-effect of this approach is partiality. Whichever realizability relation for reals we take, comparisons of real numbers are only partially computable [25, Theorem 4.1.16]. Let *Kleenean* \mathbf{K} be the assembly of $\{\mathit{ff}, \mathit{tt}, \perp\}$ where an infinite sequence of zeros realizes \perp , an infinite sequence that starts with 1 after a finite prefix of zeros realizes ff , and an infinite sequence that starts with 2 after a finite prefix of zeros realizes tt (see e.g. [11, Example 3]). The assembly \mathbf{K} can be seen as a generalization of the Booleans by adding an explicit state of divergence \perp . Comparison in any effective assembly of reals \mathbf{R} is computable as a function $x <_k y = \mathit{tt}$ if $x < y$, ff if $y < x$, and \perp if $x = y$.

As the usual comparisons are partial, multivaluedness becomes essential in exact real computation [14]. For two assemblies of A and B , a multivalued function $f : A \rightrightarrows B$, which is basically a nonempty set-valued function, is computable if there is a computable function that takes a realizer φ of $x \in A$ and computes a realizer of any $y \in f(x)$. An example is the multivalued soft comparison [5]:

$$x <_k y = \{\mathit{tt} \mid x < y + 2^k\} \cup \{\mathit{ff} \mid y < x + 2^k\}.$$

The above total multivalued function approximates the order relation. It is tracked by evaluating two partial comparisons $x < y + 2^k$ and $y < x + 2^k$ in parallel, returning tt if $x < y + 2^k = \mathit{tt}$, and ff if $y < x + 2^k = \mathit{tt}$. It is nondeterministic in the sense that for the same x and y but with different realizers, which of the tests terminates first may vary. Exact real number computation software such as [10, 16] further offer operators like **select** $:\subseteq \mathbf{K} \times \mathbf{K} \rightrightarrows \mathbf{K}$ such that **select** $(k_1, k_2) \ni \mathit{tt}$ iff $k_1 = \mathit{tt}$ and **select** $(k_1, k_2) \ni \mathit{ff}$ iff $k_2 = \mathit{tt}$ as a primitive operation for generating multivaluedness.

3 Axiomatization

In this section we give an overview of the formalization and the axioms we introduce. For space reasons we omit most axioms in the main part but provide a complete list in Appendix A. For those axioms that we do introduce here, we also reference the corresponding entry from the appendix.

Our theory is formalized in a type theory similar to the one of Coq. More precisely, we work with a dependent type theory with basic types $0, 1, 2, \mathbf{N}, \mathbf{Z}$, and a universe of classical propositions. That is, we have an impredicative à la Russel universe **Prop**, closed under $\rightarrow, \wedge, \vee, \exists, \Pi$, where the law of excluded middle $\Pi(P : \mathbf{Prop}). P \vee \neg P$ holds (Axiom **TT1**) [17]. We assume that the identity types belong to **Prop**. Opposed to **Prop**, **Type** is an à la Russel universe of types (with an implicit type level) with type constructors $\rightarrow, \times, +, \Sigma, \Pi$. We further suppose propositional extensionality in **Prop** (Axiom **TT2**) and function extensionality (Axiom **TT3**). Based on this setting, we propose an axiomatization for the assemblies \mathbf{K}, \mathbf{R} and computable multivalued functions from Sect. 2.

3.1 Kleenean and Multivalued Lifting

First, we assume that there is a type $K : \text{Type}$ of Kleeneans (Axiom [K1](#)) and that there are two *distinct* elements $\text{true} : K$ and $\text{false} : K$ (Axioms [K2](#), [K3](#) and [K4](#)). Let us define the abbreviation $\lceil t \rceil : \text{Prop} \equiv t = \text{true}$. In many cases, we do not work directly with Kleeneans. Instead, we call a proposition $P : \text{Prop}$ semi-decidable (in its free variables) if there is a Kleenean t that identifies P :

$$\text{semiDec}(P) \equiv \Sigma(t : K). P = \lceil t \rceil$$

Multivalued computations are axiomatized by a monad M (Axioms [M1–M9](#)) such that a mapping $f : A \rightarrow B$ expresses a singlevalued function and $f : A \rightarrow M B$ expresses a multivalued function. We assume the monad structure: (1) there is a type constructor $M : \text{Type} \rightarrow \text{Type}$, (2) there is a unit $\text{unitM} : \Pi(A : \text{Type}). A \rightarrow M A$, (3) a multiplication $\text{multM} : \Pi(A : \text{Type}). M(M A) \rightarrow M A$, (4) a function lifting $\text{liftM} : \Pi(A, B : \text{Type}). (A \rightarrow B) \rightarrow M A \rightarrow M B$, (5) and the corresponding coherence conditions.

Intuitively, the monad can be understood as the nonempty power-set monad. In this sense, we assume that there is a mapping

$$\text{elimM} : \Pi(A : \text{Type}). (\Pi(x, y : A). x = y) \rightarrow (M A) \rightarrow A$$

which is an inverse of unitM (Axioms [M10–M11](#)).

For any sequence of types $P : \mathbb{N} \rightarrow \text{Type}$, we assume that the map

$$\lambda(X : M(\Pi(x : \mathbb{N}). P x)). \lambda(n : \mathbb{N}). \text{liftM}(\lambda(f : \Pi(x : \mathbb{N}). P x). f n) X$$

which is of type $M(\Pi(x : \mathbb{N}). P x) \rightarrow \Pi(x : \mathbb{N}). M(P x)$ admits a section (Axioms [M12–M13](#)):

$$\omega\text{lift } P : (\Pi(x : \mathbb{N}). M(P x)) \rightarrow M(\Pi(x : \mathbb{N}). P x).$$

Intuitively, given a set of sequences S , the first map transforms it to a sequence of sets $(n \mapsto \bigcup_{f \in S} \{f(n)\})$. And, ωlift is its section which transforms a sequence of sets f to a set of sequences $\{g \mid \forall n. g(n) \in f(n)\}$. This operation enables, for example, to interchange multivalued sequences of real numbers with sequences of multivalued real numbers.

The most important axiom we assume is multivalued branching (Axiom [M14](#)):

$$\text{select} : \Pi(x, y : K). (\lceil x \rceil \vee \lceil y \rceil) \rightarrow M(\lceil x \rceil + \lceil y \rceil).$$

The above axiom yields the following, which we use more frequently:

$$\text{choose} : \Pi(P, Q : \text{Prop}). P \vee Q \rightarrow \text{semiDec}(P) \rightarrow \text{semiDec}(Q) \rightarrow M(P + Q).$$

Namely, given two semi-decidable propositions and at least one of them holds classically, we can nondeterministically decide if P holds or Q holds.

For any two types A, B , we write $f : A \rightrightarrows B$ to denote $f : A \rightarrow \mathbb{M} B$ and $\overline{\Sigma}(x : A). P(x)$ for $\mathbb{M} \Sigma(x : A). P(x)$ (multivalued functions and existences).

Example 1. For any proposition P , suppose both $\text{semiDec}(P)$ and $\text{semiDec}(\neg P)$ hold. As $P \vee \neg P$ holds by the classical law of excluded middle, we have $\mathbb{M}(P + \neg P)$ by applying choose . As it is provable that $P + \neg P$ is subsingleton, using elimM , we have $P + \neg P$, the decidability of the proposition P .

3.2 Real Numbers

We assume real numbers by declaring that there is a type $\mathbb{R} : \text{Type}$ for real numbers (Axiom R1) and axiomatizing its model-theoretic structure. There are distinct constants $0 : \mathbb{R}$ and $1 : \mathbb{R}$, (infix) binary operators $+, \times : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$, a unary operator $- : \mathbb{R} \rightarrow \mathbb{R}$, a term $/ : \Pi(x : \mathbb{R}). x \neq 0 \rightarrow \mathbb{R}$, and a (infix) binary predicate $< : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \text{Prop}$ (Axioms R2–R8). We assume the properties of the structure classically in a safe way that does not damage constructivity (Axioms R11–R27). For example, trichotomy (Axiom R22) is assumed as a term of type

$$\Pi(x, y : \mathbb{R}). x < y \vee x = y \vee y < x.$$

However, an inhabitant of the type $\Pi(x, y : \mathbb{R}). (x < y) + (x = y) + (y < x)$ is not posed anywhere.

In addition to the axioms in Prop , we assume $\Pi(x, y : \mathbb{R}). \text{semiDec}(x < y)$ (Axiom R9). Namely, for any two real numbers its order, as a classical proposition, is semi-decidable.

Example 2. Using the classical trichotomy, we can construct a term of type

$$\Pi(x, y, \epsilon : \mathbb{R}). 0 < \epsilon \rightarrow x < y + \epsilon \vee y < x + \epsilon.$$

Since the inequalities are semi-decidable, using choose , the multivalued version of the approximate splitting lemma [21, Lemma 1.23]

$$\text{mSplit} : \Pi(x, y, \epsilon : \mathbb{R}). 0 < \epsilon \rightrightarrows ((x < y + \epsilon) + (y < x + \epsilon))$$

is obtainable, which roughly says, for any real numbers x, y, ϵ , when ϵ is positive, we can nondeterministically decide if $x < y + \epsilon$ or $y < x + \epsilon$.

The set of classical axioms living in Prop includes the completeness of the set of real numbers (Axiom R27). For its constructive counterpart (Axiom R10), for any predicate $P : \mathbb{R} \rightarrow \text{Prop}$ such that $p : \exists!(z : \mathbb{R}). P z$ holds, we assume

$$\lim P p : (\Pi(n : \mathbb{N}). \Sigma(e : \mathbb{R}). \exists(a : \mathbb{R}). P a \wedge -2^{-n} < e - a < 2^{-n}) \rightarrow \Sigma(a : \mathbb{R}). P a.$$

Here, for any $n : \mathbb{N}$, $2^{-n} : \mathbb{R}$ is constructed by recursive division of $1 + 1$ on 1 and $\exists!(a : A). P a$ stands for $\exists(a : A). P a \wedge \Pi(b : A). P b \rightarrow a = b$. Note that P can be seen as a data that classically defines a real number. The axiom says

that when we have a procedure that computes a 2^{-n} approximation to the real number for each n , we have the real number constructively.

Example 3. In many cases, we compute an approximation of a real number using multivalued computation. Using `elimM` and `ωlift`, we can define

$$\overline{\lim} P p : (\Pi(n : \mathbb{N}). \overline{\Sigma}(e : \mathbb{R}). \exists!(a : \mathbb{R}). P a \wedge -2^{-n} < e - a < 2^{-n}) \rightarrow \Sigma(a : \mathbb{R}). P a.$$

where $P : \mathbb{R} \rightarrow \mathbf{Prop}$ and $p : \exists!(z : \mathbb{R}). P z$. Namely, when we have a procedure that computes a *multivalued* approximation to a real number, the procedure itself gets converted to the real number.

3.3 Soundness by Realizability

To prove soundness of the set of axioms, we extend the standard realizability interpretation of extensional dependent type theories to the category of assemblies over Kleene’s second algebra with computable morphisms $\mathbf{Asm}(\mathcal{K}_2)$ [19, § 4 and § 5]. That is, to each type constant $A : \mathbf{Type}$ we axiomatize, we designate an assembly $\llbracket A : \mathbf{Type} \rrbracket$ and to each axiomatic term constant $c : A$, we assign a morphism $\llbracket c : A \rrbracket : \mathbf{1} \rightarrow \llbracket A : \mathbf{Type} \rrbracket$ in $\mathbf{Asm}(\mathcal{K}_2)$ where $\mathbf{1}$ is a terminal object.

In consequence, by extending the interpretation, we not only prove soundness of the axiomatization but also argue that a closed term in our type theory automatically gives a construction of a computable function in the sense of computable analysis. For example, suppose we have a proof of the statement

$$\Pi(x : \mathbb{R}). P x \Rightarrow \Sigma(y : \mathbb{R}). Q x y$$

where $P : \mathbb{R} \rightarrow \mathbf{Prop}$ and $Q : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbf{Prop}$. The interpretation of the proof is a computable partial multifunction $f : \subseteq \mathbb{R} \Rightarrow \mathbb{R}$ where for any $x \in \mathbb{R}$ such that $\llbracket P \rrbracket(x) = \mathbf{1}$, $f(x)$ is well-defined and for any $y \in f(x)$, $\llbracket Q \rrbracket(x, y) = \mathbf{1}$.

For our axioms, we interpret \mathbf{K} to the Kleenean assembly \mathbf{K} and \mathbf{R} to any effective assembly of real numbers \mathbf{R} . Mapping the axiomatic constants properly, e.g., `true` to tt and `false` to ff , validates most of the axioms.

In order to interpret the multivaluedness, we specify the endofunctor $\mathbf{M} : \mathbf{Asm}(\mathcal{K}_2) \rightarrow \mathbf{Asm}(\mathcal{K}_2)$ such that for an assembly \mathbf{A} , $\mathbf{M} \mathbf{A}$ is an assembly of the set of nonempty subsets of \mathbf{A} whose realization relation \Vdash is defined by

$$\varphi \Vdash_{\mathbf{M} \mathbf{A}} S \quad :\Leftrightarrow \quad \exists x. x \in S \wedge \varphi \Vdash_{\mathbf{A}} x.$$

In words, φ realizes a nonempty subset S of \mathbf{A} if φ realized an element x of S in the original \mathbf{A} . Note that for any assemblies \mathbf{A}, \mathbf{B} , a multifunction $f : \mathbf{A} \Rightarrow \mathbf{B}$ is computable if and only if it appears as a morphism $f : \mathbf{A} \rightarrow \mathbf{M} \mathbf{B}$.

The endofunctor \mathbf{M} is a monad whose unit is $\eta_{\mathbf{A}} : x \mapsto \{x\}$, multiplication is $\mu_{\mathbf{A}} : S \mapsto \bigcup_{T \in S} T$, and its action on morphisms is $\mathbf{M}(f) : S \mapsto \bigcup_{x \in S} \{f(x)\}$.

When \mathbf{A} is sub-singleton, $\mathbf{M} \mathbf{A}$ is isomorphic to \mathbf{A} . And, for any sequence of assemblies $(\mathbf{A}_i)_{i \in \mathbb{N}}$, there is a mapping $\Pi_{i \in \mathbb{N}} \mathbf{M}(\mathbf{A}_i) \rightarrow \mathbf{M}(\Pi_{i \in \mathbb{N}} \mathbf{A}_i)$ that collects all sections of $f \in \Pi_{i \in \mathbb{N}} \mathbf{M}(\mathbf{A}_i)$. The axioms of multivalued types are validated by

mapping the monad structure of \mathbf{M} to the monad structure of \mathbf{M} and mapping `select` to `select`.

Discussions thus far conclude the soundness of our axioms:

Lemma 1. *The axiomatization is sound admitting a realizability interpretation.*

4 Relating Classical Analysis

Although our axiomatization is constructive, in some cases we allow a certain amount of classical reasoning to prove non-computational properties. For example, in terms of program extraction (cf. Sect. 5) we often want to prove a statement of the form $\Pi(x : \mathbb{R}). \Sigma(y : \mathbb{R}). P x y$ where $P : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbf{Prop}$. To do this, we assume any $x : \mathbb{R}$, provide an explicit $y : \mathbb{R}$ and prove that $P x y$ holds. $P x y : \mathbf{Prop}$ is a classical statement and thus admits nonconstructive proofs.

As mentioned in the introduction, most proof assistants already provide formalizations of classical reals and some theory upon them. Instead of rebuilding all this theory on top of our axiomatization, in the above situation it would be more practical to have a way to carefully apply classical results to our type without breaking constructivity.

More concretely, let us assume a Coq-like dependent type theory that already provides a rich theory of classical analysis through a type $\tilde{\mathbb{R}}$. Here, by classical analysis, we mean that classical statements such as $\Pi(x : \tilde{\mathbb{R}}). x > 0 + \neg(x > 0)$ hold in the type theory. We want to embed our axiomatization and apply theorems proven over the classical theory to our formalization while separating the constructive part and the classical part of the type theory correctly so that realizability results like those from Sect. 3.3 still hold.

Even though the type theory provides classical types and terms, it stays fully constructive for the terms that do not access the classical axioms. That means, a term in the type theory can be formally interpreted into two different models. We have two type judgements $\vDash t : A$ saying that t of type A may rely on classical axioms and $\vdash t' : A'$ saying that t' of type A' is free from any classical axioms. When $\vDash t : A$, we interpret it in the category of sets \mathbf{Set} and when $\vdash t : A$, we interpret it in $\mathbf{Asm}(\mathcal{K}_2)$. For example, $\vDash t : \Pi(x : \tilde{\mathbb{R}}). x > 0 + \neg(x > 0)$ is derivable for some t , but $\vdash t : \Pi(x : \tilde{\mathbb{R}}). x > 0 + \neg(x > 0)$ is not for the same t .

The goal is to correctly relate the two type judgements. One way is obvious: when $\vdash t : A$ is derivable, then so is $\vDash t : A$.² However, we are more interested in the other direction, i.e. how we can get a constructively well-typed term from classical well-typedness.

Recall that \mathbf{Set} is a reflective subcategory of $\mathbf{Asm}(\mathcal{K}_2)$ by the forgetful functor $\Gamma : \mathbf{Asm}(\mathcal{K}_2) \rightarrow \mathbf{Set}$ and its right adjoint $\nabla : \mathbf{Set} \rightarrow \mathbf{Asm}(\mathcal{K}_2)$ where for any set A , ∇A is the assembly of A with the trivial realization relation [24, Theorem 1.5.2].

For each type A , define

$$\nabla A \equiv \Sigma(P : A \rightarrow \mathbf{Prop}). \exists!(x : A). P x.$$

² However, this is no longer true if we assumed counter-classical axioms such as the continuity principle.

See that for any type A , $\llbracket \vDash \nabla A : \text{Type} \rrbracket$ is isomorphic to $\llbracket \vDash A : \text{Type} \rrbracket$ in Set and $\llbracket \vdash \nabla A : \text{Type} \rrbracket$ is isomorphic to $\nabla \Gamma \llbracket \vdash A : \text{Type} \rrbracket$ in $\text{Asm}(\mathcal{K}_2)$. It can be understood as a functor that erases all the computational structure of A while keeping its set-theoretic structure.

It is provable in the type theory using the assumptions of Prop being the type of classical propositions admitting propositional extensionality that ∇ is an idempotent monad where its unit $\text{unit}_\nabla : \Pi(A : \text{Type}). A \rightarrow \nabla A$ on ∇A , is an equivalence with the inverse being the multiplication. Moreover, it holds that $\text{unit}_\nabla \text{Prop} : \text{Prop} \rightarrow \nabla \text{Prop}$ is an equivalence. That means, given a mapping $f : A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_d$, there is a naturally defined lifting $f^{\dagger\nabla} : \nabla A_1 \rightarrow \nabla A_2 \rightarrow \dots \rightarrow \nabla A_d$ and given a predicate $P : A_1 \rightarrow A_2 \rightarrow \dots \rightarrow \text{Prop}$, there is $P^{\dagger\nabla} : \nabla A_1 \rightarrow \nabla A_2 \rightarrow \dots \rightarrow \text{Prop}$.

We add the type judgement rule:

$$\frac{\vDash t : A}{\vdash t : A} \text{ A is transferable (RELATE)}$$

saying that when t is a classically constructed term of a transferable type A , we have a constructive term t of type A . A type is transferable if it is of the form ∇c for a constant type, a Type , or a Prop variable c ; $A \rightarrow B$, $A \times B$, $A \wedge B$, $A \vee B$, or $\nabla(A + B)$ for transferable types A and B ; $\Pi(x : A). P(x)$, $\Sigma(x : A). P(x)$, or $\exists(x : A). P(x)$ for transferable types A and $P(x)$; $x = y$, $x < y$, or $x <^\dagger y$; or is Type or Prop . Roughly speaking, a type is transferable if its subexpressions in the construction of the type are guarded by ∇ .

This judgement rule is validated in our interpretation. First, note that $\nabla\{*\} \simeq \mathbf{1}$. And, when a type A is transferable, $\llbracket \vdash A : \text{Type} \rrbracket \simeq \nabla \llbracket \vDash A : \text{Type} \rrbracket$ holds. When $\vDash t : A$, we have a function $\llbracket \vDash t : A \rrbracket : \{*\} \rightarrow \llbracket \vDash A : \text{Type} \rrbracket$ in Set . Hence, we define $\llbracket \vdash t : A \rrbracket : \mathbf{1} \rightarrow \llbracket \vdash A : \text{Type} \rrbracket$ by pre and postcomposing the above isomorphisms to $\nabla \llbracket \vDash t : A \rrbracket : \nabla\{*\} \rightarrow \nabla \llbracket \vDash A : \text{Type} \rrbracket$.

We assume the map $\text{relator} : \mathbb{R} \rightarrow \nabla \tilde{\mathbb{R}}$ to relate our axiomatic real numbers with classical analysis (Axiom $\nabla 1$). Its interpretation in Set is the identity map $\llbracket \vDash \text{relator} : \mathbb{R} \rightarrow \nabla \tilde{\mathbb{R}} \rrbracket : \mathbb{R} \ni x \mapsto x \in \mathbb{R}$. We assume enough axioms that characterize the mapping (Axiom $\nabla 1$ – $\nabla 10$). For example, $\text{relator } 0 = \text{unit}_\nabla \tilde{\mathbb{R}} 0$ (Axiom $\nabla 4$), $\Pi(x, y : \mathbb{R}). \text{relator}(x + y) = (\text{relator } x) +^{\dagger\nabla} (\text{relator } y)$ (Axiom $\nabla 6$), $\Pi(x, y : \mathbb{R}). (x < y) = (\text{relator } x) <^{\dagger\nabla} (\text{relator } y)$ (Axiom $\nabla 10$), and so on.

Example 4. Suppose from the theory of classical analysis, we have a term f saying that for any positive real number, there is a square root:

$$\vDash f : \Pi(x : \tilde{\mathbb{R}}). 0 < x \rightarrow \Sigma(y : \tilde{\mathbb{R}}). x = y \times y$$

As $\text{unit}_\nabla A : A \rightarrow \nabla A$ is an equivalence in the classical type theory, we can derive

$$\vDash f' : \Pi(x : \nabla \tilde{\mathbb{R}}). (\text{unit}_\nabla 0) <^{\dagger\nabla} x \rightarrow \Sigma(y : \nabla \tilde{\mathbb{R}}). x = y \times^{\dagger\nabla} y$$

As the type of the above judgement is transferable, we have

$$\vdash f' : \Pi(x : \nabla \tilde{\mathbb{R}}). (\text{unit}_\nabla 0) <^{\dagger\nabla} x \rightarrow \Sigma(y : \nabla \tilde{\mathbb{R}}). x = y \times^{\dagger\nabla} y$$

Using the axioms of the relator, we can obtain a term of type

$$\vdash \Pi(x : \mathbb{R}). 0 < x \rightarrow \exists(y : \mathbb{R}). x = y \times y : \text{Prop.}$$

It illustrates how we can transport a classical proof of the existence of square root based on $\tilde{\mathbb{R}}$ to a constructive proof of the classical existence of square root based on \mathbb{R} . This existence result is used when verifying our implementation of the square root function as described in Sect. 5.3.

Note that in Coq we can not formally deal with having two independent type theories simultaneously and therefore a complete correctness proof when applying the relator notion can only be proven on the meta-level. We plan to address this issue in future work e.g. by writing a Coq plugin that would allow this distinction.

5 Implementation and Examples

We implemented the above theory in the Coq proof assistant.³ From a correctness proof in our implementation, we can extract Haskell code that uses the AERN library to perform basic real number arithmetic operations. For this, we introduce several extraction rules replacing operations on the constructive reals with the corresponding AERN function. The extracted code requires only minor mechanical editing, namely adding import statements (cf. Appendix B for details of the extraction process).

Let us present the main features of our implementation by giving some examples of operations on real numbers.

5.1 Maximization

A simple example of an operation that requires multivaluedness in its definition is the maximization operator that takes to real numbers x and y and returns their maximum. We can define it by the following Coq statement.⁴

```
forall x y, {z | (x > y -> z = x) /\ (x = y -> z = x) /\ (x < y -> z = y)}.
```

The statement can be proven by applying the limit operator defined in Example 3. That is, we have to show that there exists exactly one $z : \mathbf{Real}$ for which the condition in the above statement holds and that for each $n : \mathbf{nat}$ we can construct a $e : \mathbf{Real}$ multivaluedly that approximates z up to error 2^{-n} . The first part can be easily concluded from the axioms over the \mathbf{Real} type. The approximation can be constructed by concurrently testing whether $x > y - 2^{-n}$ or $x < y + 2^{-n}$, i.e. by multivalued branching from Example 2. In the first case, x can be used as the desired approximation and in the second case y .

Extracting code from this proof yields a maximization operator in AERN. Figure 1 shows parts of the Coq proof and the extracted Haskell code.

³ The source code is on <https://github.com/holgerthies/coq-aern/tree/release>.

⁴ For sake of presentation, we applied some slight, non-essential simplifications to the Coq statements in this section compared to the original source code.

```

Lemma Realmax : forall x y, {z | (x > y -> z = x) /\ ...}.
Proof.
  intros.
  apply mslimit.
  + (* max is single valued predicate *) ...
  + (* construct limit *)
    intros.
    apply (mjoin (x>y - prec n)
              (y > x - prec n)).
  ++ intros [c1|c2].
     +++ (* when x>y-2^n *)
     exists x. ...
     +++ (* when x<y+2^n *)
     exists y. ...
  ++ apply M_split.
     apply prec_pos.
Defined.

```

```

realmax ::
  AERN2.CReal ->
  AERN2.CReal ->
  AERN2.CReal
realmax x y =
  mslimit (\n ->
    Prelude.id
    (\h ->
      case h of {
        Prelude.True -> x;
        Prelude.False -> y})
    (m_split x y ((0.5 Prelude.~) n)))

```

Fig. 1. Outline of a Coq proof and corresponding extracted Haskell code

5.2 Intermediate Value Theorem (IVT)

A classical example from computable analysis (see e.g. [25, Chapter 6.3]) is finding the zero of a continuous, real valued function $f : [0, 1] \rightarrow \mathbb{R}$ with $f(0) < 0$ and $f(1) > 0$ under the assumption that there is exactly one zero in the interval (i.e. a constructive version of the intermediate value theorem from analysis).

More precisely, we prove the following statement in Coq.

```

forall (f : Real -> Real),
  continuous f -> uniq f 0 1 -> {z | 0 < z < 1 /\ f z = 0}.

```

Here, `continuous` is defined using the usual ϵ - δ -criterion and `uniq f a b` is the statement that f has exactly one zero in the interval $[a, b]$. The statement can be proven using the trisection method which is similar to the classical bisection method but avoids uncomputable comparison to 0. That is we inductively define sequences a_i, b_i with $f(a_i) * f(b_i) < 0$ and $b_i - a_i \leq (2/3)^i$. In each step we let $a'_i := (2a_i + b_i)/3$, $b'_i := (a_i + 2b_i)/3$ and in parallel check if $f(a'_i) * f(b_i) < 0$ or $f(a_i) * f(b'_i) < 0$. In the first case we set $a_{i+1} := a'_i$, $b_{i+1} := b_i$, in the second case $a_{i+1} := a_i$, $b_{i+1} := b'_i$. As at least one of the inequalities is true by the assumptions, this selection can be done using the multivalued `choose` operator from Sect. 3.1. The zero can then be defined using the limit operator. Again, we can extract an AERN program from the proof. The extracted program is an implementation of root finding using the above algorithm.

5.3 Classical Proofs and a Fast Square Root Algorithm

As a final example let us look at how to use the relator operation defined in Sect. 4 to prove facts about our constructive real type using classical results from the

Coq standard library. We follow an example from [11] that implements the Heron method to compute the square root of a real number in the Incone library. The proof is interesting as it is mostly classical and makes use of some of the theory and external libraries for classical analysis that are already available for Coq. Making use of this huge repertoire on theory already formalized in Coq vastly simplifies the proof. We repeated the example using our new implementation and compared it to the implementation in Incone.

The Heron method is an approximation scheme for the square root of a real $x \in \mathbb{R}$ by the sequence inductively defined by $x_0 := 1$ and $x_{i+1} := \frac{1}{2} \left(x_i + \frac{x}{x_i} \right)$. In this work we only consider a restricted version where $\frac{1}{4} \leq x \leq 2$. In this interval, the sequence converges quadratically to \sqrt{x} , i.e. $|x_i - \sqrt{x}| \leq 2^{-2^i}$. This restricted version can be expanded to all non-negative reals (see the aforementioned work on Incone).

We prove the following statement in Coq.

```
forall x, ( / 4 ) <= x -> ( x <= 2 ) -> {y | 0 <= y /\ y * y = x}.
```

The Coq standard library already contains a (non-constructive) definition of a function `sqrt` and proves many of its properties. To prove our statement, we construct a real number y by applying the limit operator to the sequence defined by the Heron iteration scheme. We then relate it to the classical real number `sqrt(x)` and use the characteristics of `sqrt` to show the condition. All necessary properties to show that the relation holds are again proven purely classical using tools from the standard library and other libraries building upon it.

The proof is very similar to the one in Incone and we could reuse large parts of it without major adaptations. It should be noted though, that Incone additionally requires to prove the existence of a realizer in the sense of computable analysis which adds an additional layer of complexity that is not required with our axiomatic approach and the new proof therefore becomes significantly simpler.

5.4 Performance Measurements

Since our axiomatization of constructive reals is built on a datatype similar to that used by AERN, we expect the performance of the extracted programs to be similar to that of hand-written AERN code. The measurements summarized below are consistent with our hypothesis.⁵ iRRAM is known to be one of the most efficient implementations of exact real computation and thus we also included hand-written iRRAM versions for calibration. The last three rows are examples of root finding by trisection. The iRRAM trisection code benefits from in-place update.

⁵ Benchmarks were run 10 times on a Lenovo T440p laptop with Intel i7-4710MQ CPU and 16 GB RAM, OS Ubuntu 18.04, compiled using Haskell Stackage LTS 17.2.

Benchmark		Average execution time (s)		
Formula	Accuracy	Extracted	Hand-written AERN	iRRAM
$\max(0, \pi - \pi)$	10^6 bits	16.8	16.2	1.59
$\sqrt{2}$	10^6 bits	0.72	0.72	0.62
$\sqrt{\sqrt{2}}$	10^6 bits	1.51	1.54	1.15
$x - 0.5 = 0$	10^3 bits	3.57	2.3	0.03
$x(2 - x) - 0.5 = 0$	10^3 bits	4.30	3.08	0.04
$\sqrt{x + 0.5} - 1 = 0$	10^3 bits	19.4	17.8	0.29

6 Conclusion and Future Work

We presented a new axiomatization of constructive reals in a type theory and proved its soundness with respect to the standard realizability interpretation from computable analysis. We implemented our theory in Coq and used Coq’s code extraction features to generate efficient Haskell programs for exact real computation based on the AERN library.

We think our new axiomatization is particularly well-suited for verifying exact real computation programs built on top of the theory of computable analysis. Nevertheless, we plan to more thoroughly compare our implementation with other implementations of constructive reals in Coq and other proof assistants in the future. In particular, we plan to take a deeper look at the C-CoRN library and how it differs from our implementation. Relating to other constructive formalization would also allow execution directly in the proof assistant.

From a more practical point of view, we plan to extend our implementation by other important operations on real numbers such as trigonometric and exponential functions and mathematical constants such as π and e . Such extensions should be straight-forward and we do not expect any major difficulties in their implementation. Maybe more interestingly, we also plan to extend to more complicated operations such as solution operators for ordinary or partial differential equations by applying recent ideas from real complexity theory [9, 12].

A Full List of Axioms

Here we list all our axioms, grouped by the files in our implementation.

`Base.v` defines our base type theory, making it extensional and `Prop` classical:

TT1 $\Pi(P : \text{Prop}). P \vee \neg P$

TT2 $\Pi(P, Q : \text{Prop}). (P \rightarrow Q) \rightarrow (Q \rightarrow P) \rightarrow P = Q$

TT3 $\Pi(A : \text{Type}). \Pi(P : A \rightarrow \text{Type}). \Pi(f, g : \Pi(x : A). P(x)). (\Pi(x : A). f x = g x) \rightarrow f = g$

`Kleene.v` axiomatizes the type of Kleeneans and the multivalued monad.

K1 $K : \text{Type}$

K2 $\text{true} : K$

- K3 $\text{false} : \mathbb{K}$
 K4 $\text{true} \neq \text{false}$
 K5 $\hat{\cdot} : \mathbb{K} \rightarrow \mathbb{K}$
 K6 $\hat{\vee} : \mathbb{K} \rightarrow \mathbb{K} \rightarrow \mathbb{K}$
 K7 $\hat{\wedge} : \mathbb{K} \rightarrow \mathbb{K} \rightarrow \mathbb{K}$
 Define $\lceil k : \mathbb{K} \rceil := k = \text{true}$, $\lfloor k : \mathbb{K} \rfloor := k = \text{false}$, and $(k : \mathbb{K}) \downarrow := \lceil k \rceil \vee \lfloor k \rfloor$.
 K8 $x \downarrow \rightarrow \lceil x \rceil + \lfloor x \rfloor$
 Kleene logic operations:
 K9 $\lceil \hat{x} \rceil = \lfloor x \rfloor$ and $\lfloor \hat{x} \rfloor = \lceil x \rceil$
 K10 $\lceil x \hat{\wedge} y \rceil = (\lceil x \rceil \wedge \lceil y \rceil)$ and $\lfloor x \hat{\wedge} y \rfloor = (\lfloor x \rfloor \vee \lfloor y \rfloor)$
 K11 $\lceil x \hat{\vee} y \rceil = (\lceil x \rceil \vee \lceil y \rceil)$ and $\lfloor x \hat{\vee} y \rfloor = (\lfloor x \rfloor \wedge \lfloor y \rfloor)$

The monad structure:

- M1 $M : \text{Type} \rightarrow \text{Type}$
 M2 $\text{unitM} : \Pi(A : \text{Type}). A \rightarrow M A$
 M3 $\text{multM} : \Pi(A : \text{Type}). M (M A) \rightarrow M A$
 M4 $\text{liftM} : \Pi(A, B : \text{Type}). (A \rightarrow B) \rightarrow (M A \rightarrow M B)$
 unitM and multM are natural transformations:
 M5 $\Pi(A, B : \text{Type}). \Pi(f : A \rightarrow B). \Pi(x : A). \text{liftM } A B f(\text{unitM } A x) = \text{unitM } B (f x)$
 M6 $\Pi(A, B : \text{Type}). \Pi(f : A \rightarrow B). \Pi(x : M (M A)).$
 $\text{multM } B((\text{liftM } (M A) (M B) (\text{liftM } A B f)) x) = (\text{liftM } A B f) (\text{multM } A x)$

The coherence conditions:

- M7 $\Pi(A : \text{Type}). \Pi(x : M A). \text{multM } A (\text{unitM } (M A) x) = x$
 M8 $\Pi(A : \text{Type}). \Pi(x : M A). \text{multM } A (\text{liftM } A (M A) (\text{unitM } A) x) = x$
 M9 $\Pi(A : \text{Type}). \Pi(x : M (M (M A))). \text{multM } A (\text{multM } (M A) x) = \text{multM } A (\text{liftM } (M (M A))(M A)(\text{multM } A) x)$

Further characterization of the monad:

- M10 $\text{elimM} : \Pi(A : \text{Type}). (\Pi(x, y : A). x = y) \rightarrow M A \rightarrow A$
 M11 $\Pi(A : \text{Type}). \Pi(p : (\Pi(x, y : A). x = y)). \Pi(a : M A). \text{unitM } A (\text{elimM } A p a) = a$
 M12 $\omega\text{lift} : \Pi(P : \mathbb{N} \rightarrow \text{Type}). (\Pi(x : \mathbb{N}). M P(x)) \rightarrow M(\Pi(x : \mathbb{N}). P(x))$
 M13 $\Pi(P : \mathbb{N} \rightarrow \text{Type}). \Pi(f : (\Pi(x : \mathbb{N}). M P(x))). f n = \lambda(n : \mathbb{N}). \text{liftM}(\lambda(f : (\Pi(x : \mathbb{N}). P(x))). f n) (\omega\text{lift } P f)$
 M14 $\text{select} : \Pi(x, y : \mathbb{K}). (\lceil x \rceil \vee \lceil y \rceil) \rightarrow M (\lceil x \rceil + \lceil y \rceil)$

`RealAxioms.v` axiomatizes the real numbers:

The structure of real numbers:

- R1 $\mathbb{R} : \text{Type}$
 R2 $0 : \mathbb{R}$
 R3 $1 : \mathbb{R}$
 R4 $+$: $\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$
 R5 \times : $\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$

R6 $- : \mathbb{R} \rightarrow \mathbb{R}$

R7 $/ : \Pi(x : \mathbb{R}). x \neq 0 \rightarrow \mathbb{R}$

R8 $< : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbf{Prop}$

Semi-decidability of comparison tests:

R9 $\Pi(x, y : \mathbb{R}). \text{semiDec}(x < y)$

Constructive completeness:

R10 $\Pi(P : \mathbb{R} \rightarrow \mathbf{Prop}). (\exists!(x : \mathbb{R}). P x) \rightarrow (\Pi(n : \mathbb{N}). \Sigma(x : \mathbb{R}). \exists(\tilde{x} : \mathbb{R}). P x \wedge -2^{-n} < x - \tilde{x} < 2^{-n}) \rightarrow \Sigma(x : \mathbb{R}). P x$

Classical axioms in \mathbf{Prop} :

R11 $\Pi(x, y : \mathbb{R}). x + y = y + x$

R12 $\Pi(x, y, z : \mathbb{R}). (x + y) + z = x + (y + z)$

R13 $\Pi(x : \mathbb{R}). x + -x = 0$

R14 $\Pi(x : \mathbb{R}). 0 + x = x$

R15 $\Pi(x, y : \mathbb{R}). x \times y = y \times x$

R16 $\Pi(x, y, z : \mathbb{R}). (x \times y) \times z = x \times (y \times z)$

R17 $\Pi(x : \mathbb{R}). \Pi(p : x \neq 0). (/ x p) \times x = 1$

R18 $\Pi(x : \mathbb{R}). 1 \times x = x$

R19 $\Pi(x, y, z : \mathbb{R}). x \times (y + z) = x \times y + x \times z$

R20 $1 \neq 0$

R21 $1 > 0$

R22 $\Pi(x, y : \mathbb{R}). x < y \vee x = y \vee x > y$

R23 $\Pi(x, y : \mathbb{R}). x < y \rightarrow \neg(y < x)$

R24 $\Pi(x, z, y : \mathbb{R}). x < y \rightarrow y < z \rightarrow x < z$

R25 $\Pi(x, y, z : \mathbb{R}). y < z \rightarrow x + y < x + z$

R26 $\Pi(x, y, z : \mathbb{R}). 0 < x \rightarrow y < z \rightarrow x \times y < x \times z$

Define $x \leq y \equiv x < y \vee x = y$. For each $P : \mathbb{R} \rightarrow \mathbf{Prop}$ and $x : \mathbb{R}$, define $P < x \equiv \Pi(y : \mathbb{R}). P y \rightarrow y \leq x$.

R27 $\Pi(P : \mathbb{R} \rightarrow \mathbf{Prop}). (\exists(x : \mathbb{R}). P x) \rightarrow (\exists(x : \mathbb{R}). P < x) \rightarrow \exists(x : \mathbb{R}). P \leq x \wedge \Pi(y : \mathbb{R}). P \leq y \rightarrow x \leq y$.

`RealCoqReal.v` defines the idempotent monad ∇ and `RealCoqReal.v` axiomatizes the relator:

$\nabla 1$ $\text{relator} : \mathbb{R} \rightarrow \nabla \tilde{\mathbb{R}}$

$\nabla 2$ $\Pi(x, y : \mathbb{R}). \text{relator } x = \text{relator } y \rightarrow x = y$

$\nabla 3$ $\Pi(y : \nabla \tilde{\mathbb{R}}). \exists(x : \mathbb{R}). y = \text{relator } x$

$\nabla 4$ $\text{relator } 0 = \text{unit}_{\nabla} \tilde{\mathbb{R}} 0$

$\nabla 5$ $\text{relator } 1 = \text{unit}_{\nabla} \tilde{\mathbb{R}} 1$

$\nabla 6$ $\Pi(x, y : \mathbb{R}). \text{relator } (x + y) = (\text{relator } x) +^{\dagger \nabla} (\text{relator } y)$

$\nabla 7$ $\Pi(x, y : \mathbb{R}). \text{relator } (x \times y) = (\text{relator } x) \times^{\dagger \nabla} (\text{relator } y)$

$\nabla 8$ $\Pi(x : \mathbb{R}). \text{relator } (-x) = -^{\dagger \nabla} (\text{relator } x)$

$\nabla 9$ $\Pi(x : \mathbb{R}). \Pi(p : x \neq 0). \text{relator } (/ x p) = /^{\dagger \nabla} (\text{relator } x)$

$\nabla 10$ $\Pi(x, y : \mathbb{R}). (x < y) = (\text{relator } x) <^{\dagger \nabla} (\text{relator } y)$

B Code Extraction

Extraction is defined in file `Extract.v`, including the following key mappings:

Coq	Haskell
<code>Real</code>	<code>AERN2.CReal</code>
<code>Real0</code>	<code>0</code>
<code>Realplus</code>	<code>(Prelude.+)</code>
<code>limit</code>	<code>AERN2.limit</code>
<code>choose</code>	<code>AERN2.select</code>
<code>Realltb</code>	<code>(OGB.<)</code>
<code>K</code>	<code>AERN2.Kleenean</code>
<code>sumbool</code>	<code>Prelude.Bool</code>
<code>M</code>	<code>type identity</code>
<code>unitM</code>	<code>Prelude.id</code>
<code>Nat.log2</code>	<code>(integer . integerLog2)</code>

Note that the monad `M` does not appear in the extracted programs. Multi-valuedness is intrinsic thanks to redundancy in the underlying representations.

The AERN comparison `OGB.<` returns a (lazy) Kleenean for real numbers.

Running the extracted code requires adding a few import statements, which are specified in file `Extract.v`.



References

1. Balluchi, A., Casagrande, A., Collins, P., Ferrari, A., Villa, T., Sangiovanni-Vincentelli, A.L.: Ariadne: a framework for reachability analysis of hybrid automata. In: Proceedings of the International Symposium on Mathematical Theory of Networks and Systems (2006)
2. Boldo, S., Filiâtre, J.-C., Melquiond, G.: Combining Coq and Gappa for certifying floating-point programs. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) CICM 2009. LNCS (LNAI), vol. 5625, pp. 59–74. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02614-0_10
3. Boldo, S., Lelay, C., Melquiond, G.: Formalization of real analysis: a survey of proof assistants and libraries. *Math. Struct. Comput. Sci.* **26**(7), 1196–1233 (2016). <http://hal.inria.fr/hal-00806920>
4. Boldo, S., Melquiond, G.: Flocq: a unified library for proving floating-point algorithms in Coq. In: 2011 IEEE 20th Symposium on Computer Arithmetic, pp. 243–252. IEEE (2011)
5. Brattka, V., Hertling, P.: Feasible real random access machines. *J. Complex.* **14**(4), 490–526 (1998). <https://doi.org/10.1006/jcom.1998.0488>. <https://www.sciencedirect.com/science/article/pii/S0885064X98904885>
6. Cruz-Filipe, L., Geuvers, H., Wiedijk, F.: C-CoRN, the constructive Coq repository at Nijmegen. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 88–103. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27818-4_7
7. Hertling, P.: A real number structure that is effectively categorical. *Math. Log. Q.* **45**, 147–182 (1999). <https://doi.org/10.1002/malq.19990450202>

8. Hofmann, M.: On the interpretation of type theory in locally cartesian closed categories. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 427–441. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0022273>
9. Kawamura, A., Steinberg, F., Thies, H.: Parameterized complexity for uniform operators on multidimensional analytic functions and ODE solving. In: Moss, L.S., de Queiroz, R., Martinez, M. (eds.) WoLLIC 2018. LNCS, vol. 10944, pp. 223–236. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-57669-4_13
10. Konečný, M.: aern2-real: A Haskell library for exact real number computation (2021). <https://hackage.haskell.org/package/aern2-real>
11. Konečný, M., Steinberg, F., Thies, H.: Computable analysis for verified exact real computation. In: 40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
12. Koswara, I., Selivanova, S., Ziegler, M.: Computational complexity of real powering and improved solving linear differential equations. In: van Bevern, R., Kucherov, G. (eds.) CSR 2019. LNCS, vol. 11532, pp. 215–227. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19955-5_19
13. Kreitz, C., Weihrauch, K.: Theory of representations. *Theoret. Comput. Sci.* **38**, 35–53 (1985)
14. Luckhardt, H.: A fundamental effect in computations on real numbers. *Theoret. Comput. Sci.* **5**(3), 321 – 324 (1977). [https://doi.org/10.1016/0304-3975\(77\)90048-2](https://doi.org/10.1016/0304-3975(77)90048-2). <http://www.sciencedirect.com/science/article/pii/0304397577900482>
15. Melquiond, G.: Proving bounds on real-valued functions with computations. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 2–17. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_2
16. Müller, N.T.: The iRRAM: exact arithmetic in C++. In: Blanck, J., Brattka, V., Hertling, P. (eds.) CCA 2000. LNCS, vol. 2064, pp. 222–252. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45335-0_14
17. Palmgren, E.: On universes in type theory. In: *Twenty Five Years of Constructive Type Theory*, pp. 191–204 (1998)
18. Park, S., et al.: Foundation of computer (algebra) analysis systems: semantics, logic, programming, verification. arXiv e-prints [arXiv:1608.05787](https://arxiv.org/abs/1608.05787) (2016)
19. Reus, B.: Realizability models for type theories. *Electron. Notes Theoret. Comput. Sci.* **23**(1), 128–158 (1999)
20. Schröder, M.: Effectivity in spaces with admissible multirepresentations. *Math. Logic Q.* **48**(S1), 78–90 (2002)
21. Schwichtenberg, H.: Constructive analysis with witnesses. In: *Proof Technology and Computation*. Natio Science Series, pp. 323–354 (2006)
22. Seely, R.A.G.: Locally cartesian closed categories and type theory. *Math. Proc. Cambridge Philos. Soc.* **95**(1), 33–48 (1984). <https://doi.org/10.1017/S0305004100061284>
23. Steinberg, F., They, L., Thies, H.: Computable analysis and notions of continuity in Coq. *Log. Methods Comput. Sci.* **17**(2), May 2021. <https://lmcs.episciences.org/7478>
24. Van Oosten, J.: *Realizability: an introduction to its categorical side*. Elsevier (2008)
25. Weihrauch, K.: *Computable Analysis*. Springer, Berlin (2000). <https://doi.org/10.1007/978-3-642-56999-9>



Lorenzen Won the Game, Lorenz Did Too: Dialogical Logic for Ellipsis and Anaphora Resolution

Davide Catta^{1,2}  and Symon Jory Stevens-Guille³ 

¹ LIRMM Université de Montpellier, Montpellier, France
davide.catta@lirmm.fr

² CNRS, Paris, France

³ Department of Linguistics, The Ohio State University, Columbus, USA
stevensguille.1@osu.edu

Abstract. We propose a novel solution to anaphora and ellipsis resolution using multi-sorted first order logic. Our theory is proof-theoretic, employing methods from the study of dialogical logic. The first order propositions are extracted from reduced lambda terms, which are themselves derived from Lambek Categorical Grammar proofs.

Keywords: Natural language semantics · Inferential semantic · Dialogical logic · Ellipsis · Anaphora

1 Introduction

We propose a novel solution to anaphora and ellipsis resolution. A meaning η is anaphoric just in case it cannot occur without the co-occurrence of another meaning η' from which it is indistinguishable, dubbed its antecedent. One obvious example of anaphora in English is ‘he’ in ‘John believes he proved the theorem’, where ‘he’ is understood to be John. But ellipsis—the absence of some subcategorized for expression—is a variety of anaphora too [41]. Anaphora resolution is the process by which the meaning of anaphoric expressions is identified with the meaning of their antecedents. Since the space of possible antecedents is in principle wide, but people find the correct (=intended) referent quickly and often without error, anaphora resolution poses an interesting puzzle for any theory of linguistic semantics.

The present paper proposes that anaphora can be accounted for by means of proof-theoretic methods in (multi-sorted) first order dialogical logic [14, 15, 32, 33]. We introduce a new quantifier \mathcal{E} , which is intuitively understood to correspond to quantification over non-fresh terms. Given a context p which includes a proposition with \mathcal{E} binding an occurrence x , there is a proof of some proposition q which is the result of substituting some term t in the context of p for the

Authors contributed equally and are listed in alphabetic order.

© Springer Nature Switzerland AG 2021

A. Silva et al. (Eds.): WoLLIC 2021, LNCS 13038, pp. 269–286, 2021.

https://doi.org/10.1007/978-3-030-88853-4_17

x bound by \mathcal{E} . This proof-theoretic approach is extended to resolution of post-auxiliary ellipsis (PAE) [36] by the introduction of events into the inventory of sorts. In short, PAE under the present theory is event anaphora.

The formulae over which deductions are performed are produced in an entirely compositional manner using a lambek categorial grammar (LCG). The semantic component of this grammar implements a synthesis of recent type-theoretic approaches to events in linguistic semantics [5, 20, 46].

In the next section, we introduce the phenomena under study—pronouns and PAE—and our conception of their resolution. In the subsequent Sect. 3 we discuss the methods used for implementing our theory: events and categorial grammar. A grammar is introduced and formulae for some examples from the preceding section are derived with it. In Sect. 4 dialogical logic is introduced and the proof-theoretic approach to resolution is exemplified. Section 5 is a brief overview of competing and complementary theories of resolution in linguistic semantics. Section 6 concludes.

2 Data

2.1 Pronouns

Since the pioneering work of [22, 26], pronouns in natural language semantics have received much attention. The conception of pronouns in these theories is undergirded by the view that some expressions introduce semantic objects dubbed discourse referents (drefs) and others refer back to these drefs. Much of the work in this tradition is concerned with when an antecedent dref is ‘accessible’ for a pronoun to be resolved to it. Consider the following:

- (1) Pedro didn’t buy a donkey. *It is grey. cf. [18]
- (2) Bill bought a donkey. It is grey.
- (3) Bill didn’t visit Sue. She is out.

According to DRT [27], negation blocks the introduction of (some) drefs in its scope, which are otherwise introduced by indefinites. Names, unlike indefinites, project the dref they introduce outside the scope of negation. Consequently, while ‘she’ can refer to Sue in (3), and ‘it’ can refer to Bill’s donkey in (2), ‘it’ cannot be understood to refer to the donkey Pedro didn’t buy in (1), which needn’t even exist. However, since the scope of indefinites is not restricted to the complement of negation, there is a second reading of (1) on which the indefinite outscopes the negation. The two scopes of the indefinite are represented in first order logic below, where p is ‘Pedro’:

- (4) $\neg\exists x.\textit{donkey}(x) \wedge \textit{buy}(p, x)$
- (5) $\exists x.\textit{donkey}(x) \wedge \neg\textit{buy}(p, x)$

The second reading is sometimes said to be ‘specific’ in that it is felicitous (=judged coherent) in the context in which Bill bought some grey donkey which Pedro didn’t buy. Since there is a grey donkey in this context, one could felicitously utter the discourse in (1). A variety of other contexts where drefs don’t seem to project have exercised semanticists. Some drefs don’t project outside the scope of ‘if, then’ expressions; others don’t project outside the scope of ‘every’. Modelling the contexts in which drefs do and don’t project is one of the primary projects of those theories of semantics dubbed ‘dynamic’, which depart from the static Montagovian [37] picture of meaning in viewing the meaning of an utterance first and foremost in terms of how it can change the context of a discourse.

We argue that (1), (2), and (3) can be modelled in a multi-sorted first order logic. Following [42] we employ a rich set of sorts, including *male human* (=m) versus *female human* (=f) versus *nonhuman* (=nh), which, due to the model of the lexicon, will produce multiple different properties, differing only in the sorts of the complements they select. Thus (1), (2), and (3) will correspond to the following (non)theorems, the specifics of which will be spelled out in the sequel:

- $$(6) \quad \neg \exists x^{nh}.donkey(x) \wedge buy(p^m, x) \wedge \mathcal{E}y^{nh}.grey(y) \\ \not\Rightarrow \exists x^{nh}.donkey(x) \wedge grey(x)$$
- $$(7) \quad \exists x^{nh}.donkey(x) \wedge buy(b^m, x) \wedge \mathcal{E}y^{nh}.grey(y) \\ \Rightarrow \exists x^nh.donkey(x) \wedge buy(b^m, x) \wedge grey(x)$$
- $$(8) \quad \neg visit(b^m, s^f) \wedge \mathcal{E}y^f.out(y) \\ \Rightarrow out(s)$$

2.2 Ellipsis

We consider the problem of resolving Post-Auxiliary Ellipsis (PAE) [21], more commonly referred to by the term verb-phrase ellipsis (VPE), using the technology used to resolve pronouns.¹ The following discourse exemplifies the phenomenon:

- (9) a. John slept.
b. Bill did not.

If the only context for the second sentence of (9) is the first sentence, one would correctly infer that what Bill didn’t do is sleep. The terseness of the antecedent might give the impression that ellipsis resolution is resolution to some preceding property—here the property of sleeping. This picture is complicated by the possibility of resolving the ellipsis to a modified property, without thereby including the modifier in the resolution. The following discourse from [7] exemplifies the phenomenon:

¹ VPE doesn’t describe the whole distribution of PAE, since, unless one extends the notion of VP well beyond its descriptive use, the antecedents for PAE need not be VPs. See [24] for excellent descriptive discussion of the distribution of ellipsis in English.

- (10) a. John spoke to Mary at four o'clock.
 b. And Bill did at five o'clock.

PAE is a highly studied topic in both theoretical syntax and linguistic semantics [35]. Linguistic theories are often split by whether they presume there is hidden syntactic structure in the ellipse, i.e. whether ‘and Bill did at five o'clock’ is underlyingly ‘and Bill spoke to Mary at five o'clock’, and whether the resolution of the antecedent is in terms of (more or less) syntactic or semantic representation. Among the theories proposed, few of them enjoy the rigour of a logic [2, 8, 12, 16] and fewer directly employ the proof theory of the logic [25]. This list is not exhaustive, but it suffices to show the privilege of denotation over deduction in the use of logic for linguistic semantics.

We contend that ellipsis can be resolved by means of deduction in the logic we use to resolve pronouns. We propose to use the notion of an event, common to philosophy since [9] but widely employed in subsequent linguistic theory [43]. Events are denoted by verbs, and therefore provide objects which can be subsequently referred to. Using the terminology of events, we consider the object of ‘did’ in (10) to be the event introduced by the preceding verb ‘spoke’. The object of ‘did’ in (9) is the event introduced by the preceding verb ‘slept’. The theorems corresponding to (10) and (9) are the following:²

- (11) $\exists w^v, n^i. speak(w) \wedge agent(j^m, w, n) \wedge theme(m^f, w, n) \wedge time(n, 4)$
 $\wedge \mathcal{E}u^v \exists o^i. agent(b, u, o) \wedge (\exists k^i, x^f. theme(x, u, k) \Rightarrow theme(x, u, o)) \wedge time(o, 5)$
 $\Rightarrow \exists w^v, o^i. speak(w) \wedge agent(b^m, w, o) \wedge theme(m^f, w, o) \wedge time(o, 5)$
- (12) $\exists w^v, n^i. sleep(w) \wedge agent(j^m, w, n) \wedge \mathcal{E}u^v \neg \exists o^i. (agent(b^m, u, o))$
 $\Rightarrow \exists w^v \neg \exists o^i. sleep(w) \wedge agent(b^m, w, o)$

Here w and u are of the sort of event (v) or—to use a term we introduce shortly—the sort of event kind. Both n and o are of the sort of time (i).³ Times correspond to when the participants are involved in the event. The participants are determined by the meaning of the verb, with the set of roles the participants may occupy roughly corresponding to those found in orthodox Neo-Davidsonian theories of event structure.

² Note we write $Qx_i, \dots, x_n.P$ for $Qx_i \dots Qx_n.P$.

³ We do not employ equality or functions in the logic. Both of these devices could be added to the logic, but we prefer to use the simplest system—the one without these devices—for expository purposes. There might be independent need for functions and even second order quantification over functions, depending on how one conceives of ‘strict’ versus ‘sloppy’ anaphoric reference, but we will not discuss this topic further here.

3 Methods

3.1 Events

We propose every verb introduces an event kind, which may subsequently be referred to. While events are put to a wide variety of uses in philosophy and linguistics, they remain a somewhat vexed notion.⁴

There is precedent for the notion of event kind in linguistic theory [17], though we commit to no pre-existing theory of whether or how such kinds could be (re)constructed from the Davidsonian [9] notion of event. Subsequently we will use the term ‘event kind’ to refer to our theoretic conception of events and the term ‘event proper’ to refer to the tokening of an event kind with respect to some time, extension, and participant(s). When no confusion is possible, we will simply use the term ‘event’. We will subsequently use the Neo-Davidsonian idiom of ‘thematic roles’ when discussing the participants of an event token, which will be modelled in terms of properties of triples of entities, times, and event kinds. We argue that referring back to an event kind needn’t involve referring to those involved in the event proper or the time of the event proper—under our theory these further references correspond to further anaphors. To see how the time and extension of an event proper are resolved independently of event kinds consider the following:

- (13) John reviewed the paper on Tuesday at his home. Bill did (it) the same day and at John’s house too.

The expression ‘same’ is anaphoric to the time of the preceding event proper, while the additive expression ‘too’ is anaphoric to the extension of the previous event proper. Without these explicit devices for referring to parts of the preceding event proper, it would be unjustified to conclude the event kind corresponding to reviewing includes reference to the time or extension of John reviewing the paper. Note that the theme ‘the paper’ isn’t part of the event kind. Reference to the theme depends on whether the event proper corresponding to the event kind involved a theme.⁵

3.2 Categorical Grammar

We employ a Lambek Categorical Grammar (LCG) [30,31,38,39,41] to derive proofs of the well-formedness of sentences. Given a set \mathcal{B} of basic categories, including the categories $NP, N, Pred, S$ where NP is the category of noun phrases, N the category of nouns, $Pred$ the category of predicative phrases

⁴ See [43] for an overview of some of the uses of events in linguistics and philosophy. [44] divided events into various subtypes: achievements, activities, accomplishments, and states. We will not be concerned with subtyping events in the present paper.

⁵ This dependence is explicitly modelled in (12), where the theme is recovered iff the event proper of the antecedent involved a theme.

and S the category of sentences, Lambek categories are defined by the following grammar:

$$L ::= p \mid L/L \mid L \setminus L$$

where p is a basic category. We use $A, B, C \dots$ for arbitrary categories. Sequents have the form $\Gamma ; \phi \vdash A ; M^\rho$, where Γ is a list of lambek categories, ϕ a list of typed variables, A a lambek category and M^ρ a simply typed λ -term with type ρ . We use Γ and Δ for lists of lambek categories, Φ and Ψ for lists of typed variables, M and N for λ -terms. Finally ρ and τ denotes arbitrary types.

Let \mathcal{W} be a set of words over an alphabet. A lexicon is a function that associates each $w \in \mathcal{W}$ with a pair of finite sets (C, L) where C is a set of lambek categories and L is a set of lambda-terms. We provide two lexicons, one event-free in Fig. 2 and the other one using events in Fig. 3. We represent a lexicon as a table with three columns. In the first column we write down the word. In the second column we write the first component of the pair associated to the word and in the third column we write the second component of the pair. If one of the two components of the pair is a singleton $\{x\}$ we will simply write x .

$$\begin{array}{c}
 \frac{}{A ; \vdash A ; M^\rho} \text{Lex} \qquad \frac{}{A ; x : \rho \vdash A ; x : \rho} \text{Id} \\
 \\
 \frac{\Gamma ; \Phi \vdash B/A ; M^{\rho \rightarrow \tau} \quad \Delta ; \Psi \vdash A ; N^\rho}{\Gamma, \Delta ; \Phi, \Psi \vdash B ; \llbracket MN \rrbracket^\tau} /E \qquad \frac{\Delta ; \Psi \vdash A ; N^\rho \quad \Gamma ; \Phi \vdash A \setminus B ; M^{\rho \rightarrow \tau}}{\Delta, \Gamma ; \Psi, \Phi \vdash B ; \llbracket MN \rrbracket^\tau} \setminus E \\
 \\
 \frac{\Gamma, A ; \Phi, x : \rho \vdash B ; M^\tau}{\Gamma ; \Phi \vdash B/A ; \lambda x^\rho. M} /I \qquad \frac{A, \Gamma ; x : \rho, \Phi \vdash B ; M^\tau}{\Gamma ; \Phi \vdash A \setminus B ; \lambda x^\rho. M} \setminus I
 \end{array}$$

Fig. 1. Rules of LCG with term correspondences.

It is known through the Curry-Howard (CH) isomorphism one can extract proofs in the lambda calculus from natural deduction proofs in LCG [1]. The result of substituting the semantic content of the lexemes for the hypotheses of the proof is a semantic recipe, the reduction of which will produce a term of the type corresponding to the proposition deduced by the grammar.

The inference rules of LCG given in Fig. 1 suffice to produce the hypotheses for formula (7) using the lexicon in Fig. 2. Figure 4 shows the construction of the proof for the formula (12) using the lexicon in Fig. 3. For perspicuity we show strings on the left side of the turnstile instead of their corresponding lambek categories and typed variables whenever the sequence of lambda variables is empty i.e., whenever the sequent is introduced by a Lex-rule.

In the $/E$ and $\setminus E$ rules the expression $\llbracket MN \rrbracket$ denotes the β -normal form of the application of the λ -term $M^{\rho \rightarrow \tau}$ to the lambda term N^ρ . There are two initial rules: the Id-rule and the Lex-rule. In the Lex-rule the sequence of typed variables on the left-hand of the turnstile is empty, A is the lambek category of some word \mathbf{w} and and the λ -term M^ρ on the right-hand side of the turnstile is some term associated by the lexicon to the word \mathbf{w} .

Word	L-Categories	Lambda terms
bill	NP	b^h
bought	$(NP \setminus S) / NP$	$\lambda x^{nh} \lambda y^h [(bought(y, x))^t]$
a	$((S / NP) \setminus S) / N$	$\lambda P^{nh \rightarrow t}, Q^{nh \rightarrow t} [\exists x^{nh}. P(x) \wedge Q(x)]^t$
donkey	N	$\lambda z^{nh}. [donkey(z)]^t$
it	$S / (NP \setminus S)$	$\lambda P^{nh \rightarrow t} [\mathcal{E} x^{nh}. P(x)]^t$
is	$(NP \setminus S) / (NP \setminus Pred)$	$\lambda R^{nh \rightarrow t}. R$
grey	$NP \setminus Pred$	$\lambda x^{nh}. [grey(x)]^t$

Fig. 2. Event-free Lexicon.

Word	L-Categories	Lambda terms
john	NP	j^h
slept	$NP \setminus V$	$\lambda x^h \lambda R^{(v \rightarrow i) \rightarrow t} \lambda w^v. [\exists n^i. slept(w) \wedge agent(x, w, n) \wedge R(w, n)]$
ϵ	$\{S / V, S / V_{\mathcal{E}}\}$	$\{\lambda P^{((v \rightarrow (i \rightarrow t)) \rightarrow v \rightarrow t)}. [\exists w^v. P(\lambda q^v, k^i. \top, w)]^t,$ $\lambda P^{((v \rightarrow i \rightarrow t) \rightarrow t)}. [\mathcal{E} w^v. P(\lambda u^v, k^i. \top, w)]^t\}$
bill	NP	b^h
did	$NP \setminus V_{\mathcal{E}}$	$\lambda x^h \lambda R^{v \rightarrow i \rightarrow t} \lambda w^v. [\exists n^i. agent(x, w, n) \wedge R(w, n)]^t$
not	$V_{\mathcal{E}} \setminus V_{\mathcal{E}}$	$\lambda P^{(v \rightarrow i \rightarrow t) \rightarrow v \rightarrow t} \lambda R^{v \rightarrow i \rightarrow t} \lambda w^v. [\neg(P(R, w))]^t$

Fig. 3. Lexicon with events.

Our lexicon for the event fragment synthesizes the approaches of [5, 20, 46], who propose to combine event semantics with the rigour of Montague Grammar-esque theories of the syntax to semantics mapping. While [5] is mostly agnostic with respect to the underlying syntactic theory, [20, 46] employ abstract categorial grammars (ACG), which enjoy a transparent mapping between both syntax and semantics and syntax and word order.⁶ For present purposes it suffices to use just LCG, since we don't here consider the more complex syntactic phenomena which require the move to ACG.

We follow [20, 46] in including a closure operator over events, the purpose of which is to form a sentence of syntactic type S from expressions of syntactic type V —though, given the possibility of adjuncts, this closure includes the truth preserving function $\lambda u^v, k^i. \top$, which prevent further modifying the event.⁷ Our notion of V differs from [20, 46] by the subdivision of V into anaphoric $V_{\mathcal{E}}$ and nonanaphoric V , which is needed to ensure the consistent mapping of types to LCG categories—the semantic type of $V_{\mathcal{E}}$ differs from the semantic type of V . These differences notwithstanding, neither our lexicon nor our combinatorics differ in spirit from either [20, 46] or [5].

⁶ One could extend our theory to ACG or any of the other Curry-esque Categorical Grammars [30, 34, 40] which can express syntactic dependencies which cannot be expressed in LCG without modifying the underlying type-logic.

⁷ John's sleeping could be modified by the expression 'well', which would correspond to $V \setminus V$. Since multiple adjuncts are possible for most verbs, the term for 'well' would be $\lambda P^{(v \rightarrow i \rightarrow t) \rightarrow v \rightarrow t}, R^{v \rightarrow i \rightarrow t}, w^v. [P(\lambda u^v, k^i. well(q, k) \wedge R(q, k), w)]^t$.

$$\begin{array}{c}
\frac{}{\epsilon \vdash S/V; \lambda P^{(v \rightarrow i \rightarrow t) \rightarrow v \rightarrow t}. [\exists w^v. [P(\lambda w^v. k^i. \top, w)]]^t} \text{Lex} \quad \frac{}{john \vdash NP; j; h} \text{Lex} \quad \frac{}{slept \vdash NP \setminus S; \lambda x^h. R^{v \rightarrow i \rightarrow t}. w^v. [\exists n^t. slept(w) \wedge agent(x, w, n) \wedge R(w, n)]^t} \text{Lex} \\
\frac{}{\epsilon, john, slept \vdash [\exists w, n. slept(w) \wedge agent(j, w, n) \wedge \top]^t} \wedge E \\
\frac{}{bill \vdash NP; j^h} \text{Lex} \quad \frac{}{did \vdash NP \setminus V_{\mathcal{E}}; \lambda x^h. R^{v \rightarrow i \rightarrow t}. w^v. [\exists n^t. agent(x, w, n) \wedge R(w, j)]^t} \text{Lex} \\
\frac{}{bill, did \vdash V_{\mathcal{E}}; \lambda R^{v \rightarrow i \rightarrow t}. w^v. [\exists n^t. agent(b, w, n) \wedge R(w, j)]^t} \wedge E \quad \frac{}{not \vdash V_{\mathcal{E}} \setminus V_{\mathcal{E}}; \lambda P^{(v \rightarrow i \rightarrow t) \rightarrow v \rightarrow t}. R^{v \rightarrow i \rightarrow t}. w^v. [\neg(P(R, w))]^t} \text{Lex} \\
\frac{}{bill, did, not \vdash \lambda R^{v \rightarrow i \rightarrow t}. w^v. [\exists n^t. agent(b, v, n) \wedge R(w, n)]^t} \text{Premise 1} \\
\vdots \\
\frac{}{\epsilon \vdash S/V_{\mathcal{E}}; \lambda P^{(v \rightarrow i \rightarrow t) \rightarrow v \rightarrow t}. [\mathcal{E} w^v. P(\lambda u^v. k^i. \top, w)]^t} \text{Premise 1} \\
\frac{}{\epsilon, bill, did, not \vdash S; [\mathcal{E} w. \neg \exists n. agent(b, w, n) \wedge \top]^t} /E
\end{array}$$

Fig. 4. John slept. Bill did not.

4 Dialogical Logic

4.1 First Order Language

We consider a standard first order multi-sorted language \mathcal{L} over a signature $\Sigma = (\mathcal{P}, \mathcal{C}, S)$. Where $S = \{\alpha_1, \alpha_2, \dots\}$ is a set of base sorts, \mathcal{P} is a set of predicate variables and $\mathcal{C} = \{a, b, c, d, \dots\}$ is a set of individual constants. There is a function f that assign a type of the form $\alpha_1 \rightarrow \alpha_2 \rightarrow \dots \alpha_n \rightarrow t$ where the α_i are base sorts and t is the type of truth values to each predicate variable, and a base sort α_i to each individual constant. We will denote an individual constant c such that $f(c) = \alpha$ for $\alpha \in S$ by c^α . Terms or our language will be either individual variables (denoted by $x^\alpha, y^\alpha, z^\alpha$, etc. where α is an arbitrary sort) or individual constants.

Let $\wedge, \vee, \Rightarrow, \forall, \exists$ be the symbols for the usual connective and quantifiers of first order logic, \mathcal{E} the symbol for our new quantifier, \perp and \top be predicate constants. Formulas are specified by the following grammar:

$$F := P(t_1, \dots, t_n) \mid \perp \mid \top \mid F \wedge F \mid F \vee F \mid F \Rightarrow F \mid \forall x^\alpha F \mid \exists x^\alpha F \mid \mathcal{E} x^\alpha F$$

where P is a predicate variables with type $\alpha_1 \rightarrow \dots \alpha_n \rightarrow t$ and for all $i \in \{1, \dots, n\}$ t_i is a term with sort α_i . The predicate constants \perp and \top and formulas of the form $P(t_1, \dots, t_n)$, where P is a predicate variable, will be called atomic formulas. The negation of a formula is defined as $\neg F \equiv F \Rightarrow \perp$. In the present work we will often talk about sequences. The length of a sequence is the number of elements in it. Given two sequences s and t , s is a prefix (resp. suffix) of t iff there is a sequence r such that $t = sr$ (resp $t = rs$). If r (resp. s) is not the empty sequence ϵ then s is said to be a proper prefix (resp. proper suffix) of t . Given a set of sequences S , a sequence $t \in S$ is said to be maximal in S whenever there is no proper suffix of t in S .

4.2 Dialogical Games: Informal Overview

Before entering into the formal matter of dialogical logic let us give an informal example of game about a formula. Let A, B and C stand for three arbitrary formulas.

0. **P**: I affirm that $A \wedge B \Rightarrow B \vee C$
1. **O**: Let me assume, for the sake of the proof, that $A \wedge B$ holds, can you show that $B \vee C$ holds?
2. **P**: You admitted that $A \wedge B$ holds, can you admit that B holds?
3. **O**: Indeed, I must admit that B holds.
4. **P**: I thus affirm that $B \vee C$ holds.
5. **O**: Can you show that one of B or C holds?
6. **P**: Of course. You have admitted that B holds if $A \wedge B$ holds. Thus I can safely affirm that B holds.

We can see that the Proponent (**P**) and the Opponent (**O**) alternate in the game. The game is a sequence of interventions. Each intervention but the first consist in either an attack against a preceding intervention of the other player or a defence against an attack of the other player. In intervention 1 **O** attacks intervention 0 by asking **P** for evidence that $B \vee C$ holds provided that $A \wedge B$ holds. **P**'s defence against 1 is the intervention 4. What counts as a question against an asserted formula A , and what counts as an answer to such a question depends upon the logical form of A . For example, in 2 **P** attacks the formula asserted in 1 by asking **O** to assert B . This is because if one concedes that a conjunction holds he must be ready to concede that both members of the conjunction holds.

In the sequel a game will be a sequence of alternated interventions made by the Proponent and the Opponent. Each intervention in the game is an attack or a defence against a preceding intervention, the game ends whenever the Opponent cannot produce a new intervention without contradicting what he already conceded. The content of the next subsection will be devoted to giving formal content to this intuitive discussion. In Subsect. 4.3 we define what a question on a formula is and what counts as an answer to such a question. In Subsect. 4.4 we formally define what it means for an intervention in a dialogue to refer to another preceding intervention in the same game (Definitions 1 and 2). Finally in Subsect. 4.5 we define (Definition 3) the class of games and when **P** wins a game.

4.3 Argumentation Forms

The set of auxiliary symbols Aux is the smallest set containing the symbols $\wedge_1, \wedge_2, \vee, \exists, \mathcal{E}$ and the expressions $\forall[c^\alpha/x]$ for all constants $c \in \mathcal{C}$. An argumentation form Arg is a function assigning to each non atomic formula F a set of pairs, where each pair consists of one *question* (also called *attacks* in the literature) and one *answer* (also called *defense* in the literature). Questions are either

formulas or symbols in Aux and answers are formulas.

$$\begin{aligned}
 Arg(A \Rightarrow B) &= \{(A, B)\} \\
 Arg(A \wedge B) &= \{(\wedge_1, A), (\wedge_2, B)\} \\
 Arg(A \vee B) &= \{(\vee, A), (\vee, B)\} \\
 Arg(\forall x^\alpha A) &= \{(\forall[c^\alpha/x], A[c^\alpha/x]) \mid c^\alpha \in \mathcal{C}\} \\
 Arg(\exists x^\alpha A) &= \{(\exists, A[c^\alpha/x]) \mid c^\alpha \in \mathcal{C}\} \\
 Arg(\mathcal{E}x^\alpha A) &= \{(\mathcal{E}, A[c^\alpha/x]) \mid c^\alpha \in \mathcal{C}\}
 \end{aligned}$$

Given a formula A , a question q that belongs to a pair $(q, a) \in Arg(A)$ is called a *question on A* . Given a formula A and a question q on A , a formula B is called an *answer to the question q on the formula A* whenever the pair (q, B) is an element of $Arg(A)$. So, for example, if A is $B \wedge C$, both \wedge_1 and \wedge_2 are question on A but only B is an answer to \wedge_1 and only C is an answer to \wedge_2 . If A is $B \vee C$, the symbol \vee is a question on A , and both B, C are answers to \vee .

4.4 Moves, and Augmented Sequences

A *defense move* is a pair $(!, A)$ where A is a formula. An *attack move* is a pair $(?, s)$ where s is either a formula or an auxiliary symbol. A *move* is either an attack move or a defense move. A move (\star, A) in which A is a formula and $\star \in \{?, !\}$ is called *assertion*. We will also say that the move asserts the formula A or that A is the asserted formula. Let $\rho = m_0 m_1 \dots m_n \dots$ be a sequence of moves and let $m_i \in \rho$. The parity of m_i is the parity of i . An assertion move $m_j \in \rho$ and $m_j = (\star, A)$ is called a *reprise* iff there is move $m_k \in \rho$ with $k < j$ such that $m_k = (\star', A)$ and m_k, m_j have different parities; we will also say that m_j is a reprise of m_k .

Definition 1. An *augmented sequence* is a pair (ρ, ϕ) where $\rho = m_0 \dots m_n \dots$ is a sequence of moves and ϕ is a function that is defined on each m_i with $i > 0$. For each $i > 0$ the function ϕ maps an $m_i \in \rho$ to an $m_j \in \rho$ such that $j < i$.

Definition 2. Let (ρ, ϕ) be an augmented sequence where $\rho = m_0, \dots, m_n, \dots$ and let m_i, m_j be moves in ρ .

- An attack move $m_i = (?, s)$ is justified whenever $\phi(m_i)$ is of the form (\star, A) and s is a question on A .
- A defense move $m_i = (!, B)$ is justified whenever $\phi(m_i) = m_j$, $m_j = (?, s)$ is a justified attack move, $\phi(m_j) = (\star, A)$ and B is an answer to the question s on A .

Example 1. We give an example of an augmented sequence (ρ, ϕ) ; we represent the augmented sequence by a table with two columns and as many rows as there are moves in the sequence of moves. In the first column we write down the moves of the sequence. In the second column the value of the function ϕ for the corresponding entry in the first column:

σ	value of ϕ
$m_0 = (? , P \wedge Q)$	
$m_1 = (! , P)$	m_0
$m_2 = (? , \wedge_1)$	m_1
$m_3 = (? , \wedge_1)$	m_0
$m_4 = (! , P)$	m_3
$m_5 = (! , R \vee Q)$	m_2
$m_6 = (? , \vee)$	m_5

The moves m_3 and m_6 (colored in blue) are both justified attack-moves. The move m_3 is a justified attack move because \wedge_1 is a question on the formulas $P \wedge Q$ asserted by the move m_0 and m_0 is the enabler of m_3 . The move m_4 (colored in red) is both the unique reprise of the augmented sequence, and the unique justified defense move: it is a reprise because it is an assertion move and there is a move with a smaller index of opposite parity i.e., m_1 that asserts the same formula. It is a justified defense because its enabler (the move m_3) is a justified attack move and the asserted formula P of m_4 is an answer to the question \wedge_1 on the formula that is asserted by the move m_0 i.e., $P \wedge Q$.

4.5 Games

Let (ρ, ϕ) be an augmented sequence, we say that a formula A appears in the augmented sequence iff there is a move $m \in \rho$ that asserts A . We say that a constant c appears in ρ whenever c occurs in some asserted formula or there is a move $m = (? , \forall[c/x])$ in ρ . Fix an enumeration $(c_i)_{i \in I}$ of constants of \mathcal{C}

Definition 3 (Game). A game \mathcal{G} for a formula A is an augmented sequence (ρ, ϕ) where $\rho = m_0, \dots, m_n, \dots$ is non empty and such that

1. $m_0 = (! , A)$ and for all $i > 0$ the move m_i is justified;
2. $\phi(m_i) = m_{i-1}$ if i is odd, $\phi(m_i) = m_j$ with j odd if i is even;
3. if $m_i = (\star , B)$ with $B \neq \top$ atomic formula and i even then m_i is a reprise and $B \neq \perp$;
4. if m_i is an attack move of the form $(? , \forall[c^\alpha/x])$ and i is odd then c^α is the first constant in the enumeration $(c_i)_{i \in I}$ that does not appear in the prefix of ρ ending in m_{i-1} ;
5. if $m_i = (! , B[c^\alpha/x])$ is a defense move, i is odd and m_{i-1} is of the form $(? , \exists)$ then c^α is the first constant in the enumeration $(c_i)_{i \in I}$ that does not appear in the prefix of ρ ending in m_{i-1} ;
6. if $m_i = (! , B[c^\alpha/x])$ is a defense move and $\phi(\rho_i)$ is of the form $(? , \mathcal{E})$ then there is an assertion move $m_j = (\star , C)$ with $j < i$ and j odd such that c^α occurs in C .

In a game \mathcal{G} moves m_i with i even are called **P**-moves. They are called **O**-moves otherwise. If $\mathcal{G}m$ is a game and m is **P**-move we will write $\mathcal{G}m^P$. We

will write $\mathcal{G}m^{\mathbf{O}}$ otherwise. Let us make some comments about the definition of game. Conditions 1 and 2 ensure that every move of the game but the first is justified by a previous move and that moves of one player are justified by moves of the other player. Moreover the player \mathbf{O} always reacts to the last move of \mathbf{P} . Condition 3 ensures that \mathbf{P} can assert an atomic proposition only if \mathbf{O} already asserted it in the game. Condition 4 ensures that \mathbf{P} must instantiate a universal quantifier with a fresh constant. Condition 5 ensures that \mathbf{O} must do the same thing with existential quantifiers. Condition 6 determines the behaviour of the quantifier \mathcal{E} in a game. This quantifier must be instantiated with a constant that appears in the *common ground* of the game where the common ground is the set of formulas asserted by \mathbf{O} through the game.

Let $\mathcal{G} = (\rho, \phi)$ be a finite game and m be a move. The move m is legal for \mathcal{G} iff the augmented sequence $(\rho m, \sigma)$ is a game, where $\sigma|_{\rho} = \phi$ and $\sigma(m) \in \rho$

Definition 4. A game \mathcal{G} is won by \mathbf{P} iff it is finite and either

- the game is of the form $\mathcal{G}'m^{\mathbf{P}}$ and there is no move n legal for \mathcal{G}
- the game is of the form $\mathcal{G}'m^{\mathbf{O}}$ and m asserts \perp

Intuitively speaking a strategy for a game \mathcal{G} is a function that specifies which move a player must play according to the moves previously played. A strategy is *winning* when the player that follows the strategy wins irrespective of the history of the game. We informally describe how a strategy should operate and then formalize this notion.

Imagine being engaged in a game \mathcal{G} , that the last move of \mathcal{G} was played according to the strategy, and that it is now your opponent's turn to play. Your opponent could extend the game in different ways: if you are playing chess, you are white and you just made your first move by moving a pawn to a certain position of the chessboard, black can in turn move a pawn or move a horse. If you are playing according to the strategy, the strategy should tell you how to react against either type of move. If black moves a pawn to C6 and you just moved your pawn to C3 then move the horse to H3. If black moves a horse to H6 and you just moved your pawn to C3 then move your pawn in B4. Therefore, a strategy can be viewed as a tree in which each vertex is a move in the game, the moves of my opponent have at most one daughter, and my moves have as many daughters as there are available moves for my opponent. A tree can be seen as a prefix-closed set of sequences over an alphabet [28]. Since our games are sequences over the alphabet of moves we can define strategies in the following manner:

Definition 5 (Strategy). A strategy \mathcal{S} for a formula A is a non empty prefix closed set of games for A such that

- if $\mathcal{G}m^{\mathbf{P}}$ and $\mathcal{G}n^{\mathbf{P}}$ belongs to \mathcal{S} then $m = n$ and m, n are enabled by the same move
- if $\mathcal{G} = \mathcal{G}'m^{\mathbf{P}} \in \mathcal{S}$ then $\mathcal{G}n^{\mathbf{O}} \in \mathcal{S}$ for all moves n legal for \mathcal{G}

A strategy \mathcal{S} is winning if and only if every maximal prefix of the strategy is a game won by \mathbf{P} .

A game (ρ, ϕ) will be represented by a table with three columns and as many rows as there are moves in the game. The left hand-column specifies whether it's \mathbf{P} 's turn or \mathbf{O} 's turn and we draw it only for the sake of clarity. The central column represents the sequence ρ . Each row of the right-hand column represent the value of the function ϕ for the entry in the center column. Figure 5 shows a winning strategy for the formula (7); we use abbreviations of predicates for typographic reasons e.g. we write *do* instead of *donkey*, *bo* instead of *bought* etc.

In the strategy we have colored—with the same color—a move and its reprise. In move m_0 \mathbf{P} asserts that *if bill bought a grey donkey and it is grey then there is a grey donkey*. We have underlined the anaphoric expression and coloured the logical connective of the sentence. In the second move m_1 \mathbf{O} supposes that the proposition ‘Bill bought a donkey and it is grey’ holds and asks, implicitly, for evidence that the proposition ‘there is a grey donkey’ holds.

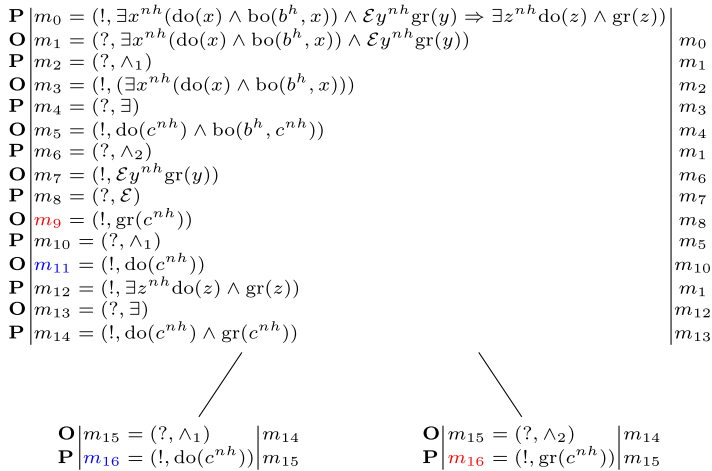


Fig. 5. Winning strategy for formula (7)

In moves m_2 through m_5 \mathbf{P} forces \mathbf{O} to instantiate the existential quantifier with a fresh constant c . In moves m_6 through m_9 \mathbf{P} forces \mathbf{O} to instantiate the quantifier \mathcal{E} that translates the pronoun ‘it’ with the constant c —the only term in the proof of the right sort.⁸ By m_9 \mathbf{O} concurs that if he concedes that Bill bought a donkey and it is grey, then the pronoun ‘it’ in ‘it is grey’ and the expression ‘a donkey’ must be instantiated with the same individual. The player \mathbf{P} can then conclude by affirming that ‘there is a grey donkey’ in move m_{12} . This conclusion

⁸ If there were more possible antecedents, more axioms would be needed to pick among those antecedents or else the strongest proposition to be proved would be a disjunction, with each disjunct corresponding to the proposition proved here modulo the choice of antecedent.

is the result of instantiating the existential quantifier with the constant c , and asserting, in move m_{14} , that c is a donkey and c is grey. According to Definition 5 of strategy the move m_{14} has two daughters. Under both daughters \mathbf{P} asserts an atomic proposition and wins the game.

Figure 6 shows a winning strategy for the formula 12 i.e., the formula that translates the sentence *if John slept and Bill did not then Bill did not sleep*. The predicate sl has type $v \rightarrow t$ where v is the sort of events and t the sort of truth values. The predicate ag has type $h \rightarrow v \rightarrow i \rightarrow t$ where h is the sort of humans and i is the sort of time-moments. b and j are two constants with sort h .

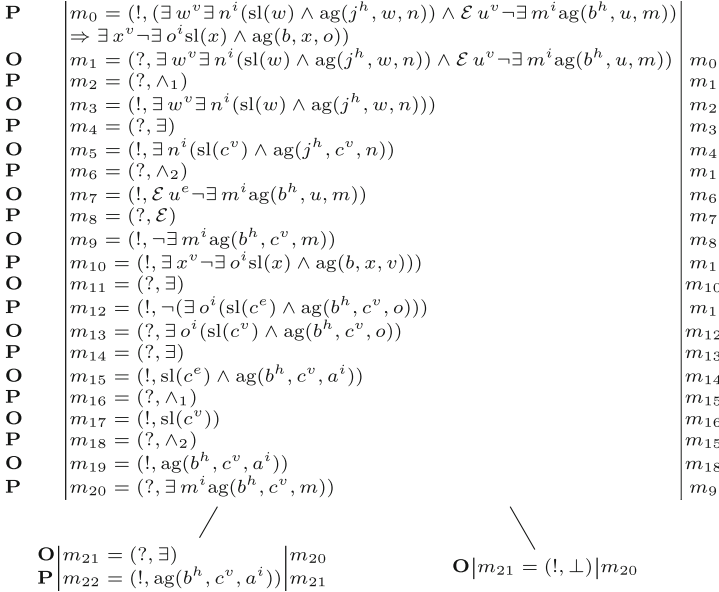


Fig. 6. Winning strategy for formula (12).

5 Discussion

The most obvious theories to compare the present work to are Discourse Representation Theory (DRT) [26, 27], Dynamic Predicate Logic (DPL) [19], Dekker’s Predicate Logic with Anaphora [11] and Predicate Logic with Indices [10]. We discuss these theories in turn. We note here that we can derive the famous ‘donkey sentences’ e.g.,

$$\begin{aligned} \forall x^h [\text{farmer}(x) \wedge (\exists y^{nh} \text{donkey}(y) \wedge \text{owns}(x, y)) \Rightarrow \mathcal{E} z^{nh} \text{cuddles}(x, z)] \\ \Rightarrow \\ \forall x^h \forall y^{nh} [\text{farmer}(x) \wedge \text{donkey}(y) \wedge \text{owns}(x, y) \Rightarrow \text{cuddles}(x, y)] \end{aligned}$$

though the proof of this formula will need to be deferred to a longer follow-up to this paper.

DRT is concerned with the construction of Discourse Representation Structures (DRS), which serve to track objects invoked by speakers in the course of producing some discourse. It includes rules for introducing, blocking, and linking (sub)DRSs to one another. The semantics of the DRS language, in its standard version, is given by an embedding function into first order logic. Some attempts have been made to do inference directly over DRSs; [29] provides a brief but non-exhaustive survey of existing inference systems for DRT and DPL circa 2000. Subsequently he proposes a tableaux theorem proving method for DRT that includes a connective quite close to our \mathcal{E} connective.

DPL, unlike DRT, just is first order logic. However it effectively redefines the semantics of FOL in order to extend the scope of \exists . It is, in principle, a reconstruction of DRT. While much work develops DPL's novel semantics for FOL [45], the proof theory of the logic receives less study. DPL's proof theory only fully developed some years subsequent to its discovery. The most successful study of DPL's proof theory is [13]. However, DPL relies on the linguist to select the correct variable for a pronoun to be resolved by its antecedent and is therefore modelling anaphoric dependence but not resolution.

Dekker develops two logics for anaphora, intending to show the minimum needed to modify FOL to account for pronouns. Dekker's approach is more proof-theoretic in adding pronouns to the inventory of terms, which, in [10], are de Bruijn (pre-)indexed identity functions on (sequences of) the i th antecedent. Such terms are subject to inference rules. Nonetheless, the preindexing corresponds to the choice of variable made by the linguist in choosing the antecedent of a pronoun in DPL.

The price of modelling resolution in the logic is the multiplicity of possible antecedents for an anaphor. If there is a disjunction of possible antecedents, the selection of just one could be enforced by means of further axioms. In fact, the winnowing down of possible antecedents is present in common symbolic approaches to anaphora resolution. We are presently considering how to implement Centering Theory [3] and/or Coherence Theory [23] in our logic, which we intend to present in future work.

6 Conclusion

This paper proposes a novel proof-theoretic method of resolving anaphora in first order multi-sorted logic. This method is the introduction of a new quantifier, the substitutions of which are restricted to constants introduced in the course of proof. The proof-system for the logic without the new quantifier is known to be sound and complete for first order classical logic [4, 6]. The proof-system extended with \mathcal{E} does not derive contradictions since there is no strategy for atomic formulae distinct from \top . In future work we intend to account for 'strict' versus 'sloppy' readings of both pronouns and PAE. Study concerning the proof-theoretic and semantic properties of the new quantifier we employ is under development. Furthermore, we intend to implement the proof system for automated theorem proving.

References

1. van Benthem, J.: The semantics of variety in categorial grammar. *Categ. Gramm.* **25**, 37–55 (1988)
2. Bos, J.: Vp ellipsis in a drt-implementation. In: Sixth Conference of the European Chapter of the Association for Computational Linguistics (1993)
3. Brennan, S.E., Friedman, M.W., Pollard, C.: A centering approach to pronouns. In: 25th Annual Meeting of the Association for Computational Linguistics, pp. 155–162 (1987)
4. Catta, D., Moot, R., Retoré, C.: Dialogical argumentation and textual entailment. In: Loukanova, R. (ed.) *Natural Language Processing in Artificial Intelligence—NLPinAI 2020*. *SCI*, vol. 939, pp. 191–226. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-63787-3_7
5. Champollion, L.: The interaction of compositional semantics and event semantics. *Linguist. Philos.* **38**(1), 31–66 (2014). <https://doi.org/10.1007/s10988-014-9162-8>
6. Clerbout, N.: First-order dialogical games and tableaux. *J. Philos. Log.* **43**(4), 785–801 (2014)
7. Cooper, R., et al.: Using the framework. Tech. rep., Technical report LRE 62–051 D-16, The FraCaS Consortium (1996)
8. Dalrymple, M., Shieber, S.M., Pereira, F.C.: Ellipsis and higher-order unification. *Linguist. Philos.* **14**(4), 399–452 (1991)
9. Davidson, D.: The logical form of action sentences. In: Rescher, N. (ed.) *The Logic of Decision and Action*, pp. 81–95. University of Pittsburgh Press (1967)
10. Dekker, P., et al.: Exclusively indexical deduction. *Rev. Symb. Log.* **9**(3), 603–637 (2016)
11. Dekker, P.J.: Predicate logic with anaphora. In: *Dynamic Semantics*, pp. 7–47. *Studies in Linguistics and Philosophy*, vol. 91. Springer, Dordrecht (2012). https://doi.org/10.1007/978-94-007-4869-9_2
12. van Eijck, J., Francez, N.: Verb-phrase ellipsis in dynamic semantics. In: Pólos L., Masuch, M. (eds.) *Applied Logic: How, What and Why*. *Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science)*, vol. 247, pp. 29–59. Springer, Dordrecht (1995). https://doi.org/10.1007/978-94-015-8533-0_2
13. van Eijck, J., Heguiabehe, J., Ó Nualláin, B.: Tableau reasoning and programming with dynamic first order logic. *Log. J. IGPL* **9**(3), 411–445 (2001)
14. Felscher, W.: Dialogues, strategies, and intuitionistic provability. *Ann. Pure Appl. Log.* **28**(3), 217–254 (1985)
15. Felscher, W.: Dialogues as a foundation for intuitionistic logic. In: Gabbay, D.M., Guenther, F. (eds.) *Handbook of Philosophical Logic*, pp. 115–145. Springer, Netherlands, Dordrecht (2002)
16. Gardent, C.: Sloopy identity. In: Retoré, C. (ed.) *LACL 1996*. *LNCS*, vol. 1328, pp. 188–207. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052158>
17. Gehrke, B.: Event kinds. *Oxf. Handb. Event Struct.* **205**, 233 (2019)
18. Geurts, B., Beaver, D.I., Maier, E.: Discourse representation theory. In: Zalta, E.N. (ed.) *The Stanford Encyclopedia of Philosophy*, Spring 2020 edn. *Metaphysics Research Lab, Stanford University* (2020)
19. Groenendijk, J., Stokhof, M.: Dynamic predicate logic. *Linguistics and Philosophy*, pp. 39–100 (1991)
20. de Groote, P., Winter, Y.: A type-logical account of quantification in event semantics. In: Murata, T., Mineshima, K., Bekki, D. (eds.) *JSAI-isAI 2014*. *LNCS (LNAI)*, vol. 9067, pp. 53–65. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48119-6_5

21. Hankamer, J.: On the nontransformational derivation of some null vp anaphors. *Linguist. Inq.* **9**(1), 66–74 (1978)
22. Heim, I.: The semantics of definite and indefinite noun phrases. Ph.D. thesis, University of Massachusetts Amherst (1982)
23. Hobbs, J.R.: Coherence and coreference. *Cogn. Sci.* **3**(1), 67–90 (1979)
24. Huddleston, R., Pullum, G.K., et al.: *The Cambridge grammar of English. Language*, vol. 1, p. 23. Cambridge University Press, Cambridge (2002)
25. Jäger, G.: *Anaphora and Type Logical Grammar*, vol. 24. Springer Science & Business Media, Heidelberg (2006)
26. Kamp, H.: A theory of truth and semantic representation. In: *Truth, interpretation and information*, pp. 1–41. Foris Dordrecht (1984)
27. Kamp, H., Reyle, U.: From discourse to logic: introduction to model theoretic semantics of natural language, formal logic and discourse representation theory. Part 1. Kluwer Academic (1993)
28. Kechris, A.: *Classical Descriptive Set Theory*. Springer, New York, NY, USA (December 2012)
29. Kohlhase, M.: Model generation for discourse representation theory. In: *ECAI*, pp. 441–445 (2000)
30. Kubota, Y., Levine, R.D.: *Type-Logical Syntax*. MIT Press, Cambridge (2020)
31. Lambek, J.: The mathematics of sentence structure. *Am. Math. Mon.* **65**(3), 154–170 (1958)
32. Lorenz, K.: 1 arithmetic and logic as games. excerpts (1978 [1961]). In: *From dialogical logic to dialogical constructivism*, pp. 1–74. De Gruyter (2021)
33. Lorenzen, P., Lorenz, K.: *Dialogische Logik*. Wissenschaftliche Buchgesellschaft, [Abt. Verlag] (1978)
34. Martin, S., Pollard, C.: A dynamic categorial grammar. In: Morrill, G., Muskens, R., Osswald, R., Richter, F. (eds.) *Formal Grammar 2014. LNCS*, vol. 8612, pp. 138–154. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44121-3_9
35. Merchant, J.: Ellipsis: a survey of analytical approaches. *The Oxford Handbook of Ellipsis*, pp. 18–46 (2019)
36. Miller, P., Pullum, G.K.: Exophoric vp ellipsis. *Core Periphery: Data-Driven Perspectives Syntax Inspired Ivan A. Sag* **5**, 32 (2013)
37. 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. Synthese Library (Monographs on Epistemology, Logic, Methodology, Philosophy of Science, Sociology of Science and of Knowledge, and on the Mathematical Methods of Social and Behavioral Sciences)*, vol. 49, pp. 221–242. Springer, Dordrecht (1973). https://doi.org/10.1007/978-94-010-2506-5_10
38. Moortgat, M.: Categorial type logics. In: *Handbook of logic and language*, pp. 93–177. Elsevier (1997)
39. Moot, R., Retoré, C.: The Logic of Categorial Grammars. LNCS, vol. 6850. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-3-642-31555-8>
40. Moot, R., Stevens-Guille, S.J.: Proof-theoretic aspects of hybrid type-logical grammars. In: Bernardi, R., Kobele, G., Pogodalla, S. (eds.) *FG 2019. LNCS*, vol. 11668, pp. 84–100. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-59648-7_6
41. Morrill, G.V.: *Type Logical Grammar: Categorial Logic of Signs*. Springer Science & Business Media, Heidelberg (2012)

42. Retoré, C.: The montagovian generative lexicon lambda tyn: a type theoretical framework for natural language semantics. In: 19th International Conference on Types for Proofs and Programs (TYPES 2013), vol. 26, pp. 202–229 (2014)
43. Truswell, R.: The Oxford Handbook of Event Structure. Oxford University Press, Oxford (2019)
44. Vendler, Z.: *Linguistics in Philosophy*. Cornell University Press, Ithaca (2019)
45. Vermeulen, C.F.M.: Sequence semantics for dynamic predicate logic. *J. Log. Lang. Inform.* **2**(3), 217–254 (1993)
46. Winter, Y., Zwarts, J.: Event semantics and abstract categorial grammar. In: Kanazawa, M., Kornai, A., Kracht, M., Seki, H. (eds.) *MOL 2011. LNCS (LNAI)*, vol. 6878, pp. 174–191. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23211-4_11



Uniform Lyndon Interpolation for Basic Non-normal Modal Logics

Amirhossein Akbar Tabatabai^(✉), Rosalie Iemhoff, and Raheleh Jalali

Utrecht University, Janskerkhof 13, 3512 BL Utrecht, The Netherlands
{s.akbartabatabai,r.iemhoff,r.jalalikesavarz}@uu.nl

Abstract. In this paper, a proof-theoretic method to prove uniform Lyndon interpolation for non-normal modal logics is introduced and applied to show that the logics E, M, MC, EN, MN have that property. In particular, these logics have uniform interpolation. Although for some of them the latter is known, the fact that they have uniform Lyndon interpolation is new. Also, the proof-theoretic proofs of these facts are new, as well as the constructive way to explicitly compute the interpolants that they provide. It is also shown that the non-normal modal logics EC and ECN do not have Craig interpolation, and whence no uniform (Lyndon) interpolation.

Keywords: Non-normal modal logics · Uniform interpolation · Uniform Lyndon interpolation · Craig interpolation

1 Introduction

Uniform interpolation (UIP), a strengthening of interpolation in which the interpolant only depends on the premise or the conclusion of an implication, is an intriguing logical property. One of the reasons is that it is hard to predict which logic does have the property and which does not. Well-behaved logics like K and KD have it, but then, other well-known modal logics, such as K4, do not. Early results on the subject were by Shavrukov [17], who proved UIP for the modal logic GL, and by Ghilardi [4] and Visser [19], who independently proved the same for K, followed later by Bílková, who showed that KT has the property as well [2]. Surprisingly, K4 and S4 do not have UIP, although they do have interpolation [2, 4]. Pitts provided the first proof-theoretic proof of UIP, for intuitionistic propositional logic, IPC, the smallest intermediate logic [13]. Results from [5, 9] imply that there are exactly seven intermediate logics with interpolation and that they are exactly the intermediate logics with UIP. Pitts' result is especially important to us, as also in our paper the approach is proof-theoretic.

The study of UIP in the context of non-normal modal logics has a more recent history. The area is less explored than its normal counterpart, but for

Support by the Netherlands Organisation for Scientific Research under grant 639.073.807 is gratefully acknowledged.

several well-known non-normal logics UIP has been established, for example, for the monotone logic M [14], a result later extended in [12, 15] to other non-normal modal and conditional logics, such as E and basic conditional logic CK.

Non-normal modal logics are modal logics in which the K -axiom, i.e. the axiom $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$, does not hold but a weaker version that is given by the following E -rule does:

$$\frac{\varphi \leftrightarrow \psi}{\Box\varphi \leftrightarrow \Box\psi}$$

Thus the minimal non-normal modal logic, E, is propositional logic plus the E -rule above. Over the last decades non-normal modal logics have emerged in various fields, such as game theory and epistemic and deontic logic [3]. Two well-known non-normal modal logics that are investigated in this paper are natural weakenings of the principle $\Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi)$ that implies K over E. Namely, the two principles:

$$(M) \quad \Box(\varphi \wedge \psi) \rightarrow (\Box\varphi \wedge \Box\psi) \quad (C) \quad (\Box\varphi \wedge \Box\psi) \rightarrow \Box(\varphi \wedge \psi).$$

Because the K -axiom holds in the traditional relational semantics for modal logic, non-normal modal logics require different semantics, of which the most well-known is neighborhood semantics. As we do not need semantics in this paper, we refer the interested reader to the textbook [11].

Our interest in the property of UIP for non-normal modal logics lies in the fact that it can be used as a tool in what we would like to call *universal proof theory*, the area where one is concerned with the general behavior of proof systems, investigating problems such as the existence problem (when does a theory have a certain type of proof system?) and the equivalence problem (when are two proof systems equivalent?). The value of UIP for the existence problem has been addressed in a series of recent papers in which a method is developed to prove UIP that applies to many intermediate, (intuitionistic) modal, and substructural (modal) logics [1, 6, 7]. The proof-theoretic method makes use of sequent calculi, and shows that general conditions on the calculi imply UIP for the corresponding logic. Thus implying that any logic without UIP cannot have a sequent calculus satisfying these conditions. The generality of the conditions, such as closure under weakening, makes this into a powerful tool, especially for those classes of logics in which UIP is rare, such as intermediate logics. Note that in principle other regular properties than UIP could be used in this method, as long as the property is sufficiently rare to be of use.

In this paper we do not focus on the connection with the existence problem as just described, but rather aim to show the flexibility and utility of our method to prove UIP by showing that it can be extended to (yet) another class of logics, namely the class of non-normal modal logics, that it is constructive and can be easily adapted to prove not only UIP but even uniform Lyndon interpolation. Uniform Lyndon interpolation (ULIP) is a strengthening of UIP in which the interpolant respects the polarity of propositional variables (a definition follows in the next section). It first occurred in [8], where it was shown that several normal

modal logics, including K and KD , have that property. In this paper we show that the non-normal modal logics E , M , MC , EN , MN have uniform Lyndon interpolation and the interpolant can be constructed explicitly from the proof. In the last part of this paper we show that the non-normal modal logics EC and ECN do not have interpolation, and whence no uniform (Lyndon) interpolation either. This surprising fact makes EC and ECN potential candidates for our approach to the existence problem discussed above, but that we have to leave for another paper.

Our proof-theoretic method to prove UIP makes use of sequent calculi for non-normal modal logics that are equivalent or equal to calculi introduced in [10]. In this paper it is also shown that E , M , MC , EN , MN have Craig interpolation. The proof that these logics have UIP is not a mere extension of the proof that they have interpolation but requires a very different approach. That the logics E and M have UIP has already been established in [12, 14], but that they have uniform Lyndon interpolation is, as far as we know, a new insight. Interestingly, for logics with LIP, that fact does not always follow easily from the proof that they have IP, as is for example the case for GL [16]. But for our method this indeed is the case: the proof of UIP easily implies ULIP. Thus the hard work lies in proving the former, in a way that turns out to imply the latter. Therefore we consider the proof-theoretic method to prove uniform interpolation for non-normal modal logics the main contribution of this paper, as until now such proofs have always been semantical in nature. In [15] the search for proof-theoretic techniques to prove uniform interpolation in the setting of non-normal modal logics is explicitly mentioned in the conclusion of that paper.

2 Preliminaries

Set $\mathcal{L} = \{\wedge, \vee, \rightarrow, \perp, \Box\}$ as the language of modal logics. We use \top and $\neg A$ as abbreviations for $\perp \rightarrow \perp$ and $A \rightarrow \perp$, respectively, and write $\varphi \in \mathcal{L}$ to indicate that φ is a formula in the language \mathcal{L} . The *weight of a formula* is defined inductively by: $w(\perp) = w(p) = 0$, for any atomic p and $w(A \odot B) = w(A) + w(B) + 1$, for any $\odot \in \{\wedge, \vee, \rightarrow\}$, and $w(\Box A) = w(A) + 1$.

Definition 1. *The sets of positive and negative variables of a formula $\varphi \in \mathcal{L}$, denoted by $V^+(\varphi)$ and $V^-(\varphi)$, respectively, are defined recursively by:*

- $V^+(p) = \{p\}$, $V^-(p) = V^+(\top) = V^-(\top) = V^+(\perp) = V^-(\perp) = \emptyset$, for atom p ,
- $V^+(\varphi \odot \psi) = V^+(\varphi) \cup V^+(\psi)$ and $V^-(\varphi \odot \psi) = V^-(\varphi) \cup V^-(\psi)$, for $\odot \in \{\wedge, \vee\}$,
- $V^+(\varphi \rightarrow \psi) = V^-(\varphi) \cup V^+(\psi)$ and $V^-(\varphi \rightarrow \psi) = V^+(\varphi) \cup V^-(\psi)$,
- $V^+(\Box \varphi) = V^+(\varphi)$ and $V^-(\Box \varphi) = V^-(\varphi)$.

Define $V(\varphi)$ as $V^+(\varphi) \cup V^-(\varphi)$. For an atomic formula p , a formula φ is called p^+ -free (p^- -free), if $p \notin V^+(\varphi)$ ($p \notin V^-(\varphi)$). It is called p -free if $p \notin V(\varphi)$. Note that a formula is p -free iff p occurs nowhere in it.

For the sake of brevity, when we want to refer to both $V^+(\varphi)$ and $V^-(\varphi)$, we use the notation $V^\dagger(\varphi)$ with the condition “for any $\dagger \in \{+, -\}$ ”. If we want to refer to one of $V^+(\varphi)$ and $V^-(\varphi)$ and its dual, we write $V^\circ(\varphi)$ for the one we intend and $V^\diamond(\varphi)$ ¹ for the other one. For instance, if we state that for any atomic formula p , any $\circ \in \{+, -\}$ and any p° -free formula φ , there is a p° -free formula ψ such that $\varphi \vee \psi \in L$, we are actually stating that if φ is p^+ -free, there is a p^- -free ψ such that $\varphi \vee \psi \in L$ and if φ is p^- -free, there is a p^+ -free ψ such that $\varphi \vee \psi \in L$.

Definition 2. A logic L is a set of formulas in \mathcal{L} extending the set of classical tautologies, CPC, and closed under substitution and modus ponens $\varphi, \varphi \rightarrow \psi \vdash \psi$.

Definition 3. A logic L has Lyndon interpolation property (LIP) if for any formulas $\varphi, \psi \in \mathcal{L}$ such that $L \vdash \varphi \rightarrow \psi$, there is a formula $\theta \in \mathcal{L}$ such that $V^\dagger(\theta) \subseteq V^\dagger(\varphi) \cap V^\dagger(\psi)$, for any $\dagger \in \{+, -\}$ and $L \vdash \varphi \rightarrow \theta$ and $L \vdash \theta \rightarrow \psi$. A logic has Craig interpolation (CIP) if it has the above properties, omitting all the superscripts $\dagger \in \{+, -\}$.

Definition 4. A logic L has uniform Lyndon interpolation property (ULIP) if for any formula $\varphi \in \mathcal{L}$, atom p , and $\circ \in \{+, -\}$, there are p° -free formulas, $\forall^\circ p\varphi$ and $\exists^\circ p\varphi$, such that $V^\dagger(\exists^\circ p\varphi) \subseteq V^\dagger(\varphi)$ and $V^\dagger(\forall^\circ p\varphi) \subseteq V^\dagger(\varphi)$, for any $\dagger \in \{+, -\}$ and

- (i) $L \vdash \forall^\circ p\varphi \rightarrow \varphi$,
- (ii) for any p° -free formula ψ if $L \vdash \psi \rightarrow \varphi$ then $L \vdash \psi \rightarrow \forall^\circ p\varphi$,
- (iii) $L \vdash \varphi \rightarrow \exists^\circ p\varphi$, and
- (iv) for any p° -free formula ψ if $L \vdash \varphi \rightarrow \psi$ then $L \vdash \exists^\circ p\varphi \rightarrow \psi$.

A logic has uniform interpolation property (UIP) if it has all the above properties, omitting the superscripts $\circ, \dagger \in \{+, -\}$, everywhere.

Remark 1. As the formulas $\forall^\circ p\varphi$ and $\exists^\circ p\varphi$ are provably unique, using the functional notation of writing $\forall^\circ p\varphi$ and $\exists^\circ p\varphi$ as the functions with the arguments $\circ \in \{+, -\}$, p and φ is allowed.

Theorem 1. If a logic L has ULIP, then it has both LIP and UIP.

Proof. For UIP, set $\forall p\varphi = \forall^+ p\forall^- p\varphi$ and $\exists p\varphi = \exists^+ p\exists^- p\varphi$. We only prove the claim for $\forall p\varphi$, as the case for $\exists p\varphi$ is similar. First, it is clear that $V^\dagger(\forall p\varphi) \subseteq V^\dagger(\varphi)$, for any $\dagger \in \{+, -\}$. Hence, we have $V(\forall p\varphi) \subseteq V(\varphi)$. Moreover, $\forall p\varphi$ is p -free. Because $\forall^- p\varphi$ is p^- -free by definition and as $V^-(\forall p\varphi) \subseteq V^-(\forall^- p\varphi)$, the formula $\forall^+ p\forall^- p\varphi$ is also p^- -free. As $\forall^+ p\forall^- p\varphi$ is p^+ -free by definition, we have $p \notin V(\forall p\varphi) = V^+(\forall p\varphi) \cup V^-(\forall p\varphi)$. For condition (i) in Definition 4, as $L \vdash \forall^+ p\forall^- p\varphi \rightarrow \forall^- p\varphi$ and $L \vdash \forall^- p\varphi \rightarrow \varphi$, we have $L \vdash \forall p\varphi \rightarrow \varphi$. For condition (ii), if $L \vdash \psi \rightarrow \varphi$, for a p -free ψ , then ψ is also p^- -free and hence $L \vdash \psi \rightarrow \forall^- p\varphi$. As ψ is also p^+ -free, we have $L \vdash \psi \rightarrow \forall^+ p\forall^- p\varphi$.

¹ The label \diamond has nothing to do with the modal operator $\diamond = \neg\Box\neg$.

For LIP, assume $L \vdash \varphi \rightarrow \psi$. For any $\dagger \in \{+, -\}$, set $P^\dagger = V^\dagger(\varphi) - [V^\dagger(\varphi) \cap V^\dagger(\psi)]$. Define $\theta = \exists^+ P^+ \exists^- P^- \varphi$, where by $\exists^\dagger \{p_1, \dots, p_n\}^\dagger$ we mean $\exists p_1^\dagger \dots \exists p_n^\dagger$. Since θ is p^\dagger -free for any $p \in P^\dagger$ and any $\dagger \in \{+, -\}$, we have $V^\dagger(\theta) \subseteq V^\dagger(\varphi) - P^\dagger \subseteq V^\dagger(\varphi) \cap V^\dagger(\psi)$. For the provability condition, it is clear that $L \vdash \varphi \rightarrow \theta$ and as ψ is p^\dagger -free for any $p \in P^\dagger$, we have $L \vdash \theta \rightarrow \psi$.

2.1 Sequent Calculi

We use capital Greek letters and the bar notation in $\bar{\varphi}$ and \bar{C} to denote multisets. A *sequent* is an expression in the form $\Gamma \Rightarrow \Delta$, where Γ (the *antecedent*) and Δ (the *succedent*) are multisets of formulas. It is interpreted as $\bigwedge \Gamma \rightarrow \bigvee \Delta$. For sequents $S = (\Gamma \Rightarrow \Delta)$ and $T = (\Pi \Rightarrow \Lambda)$ we denote the sequent $\Gamma, \Pi \Rightarrow \Delta, \Lambda$ by $S \cdot T$, and the multisets Γ and Δ by S^a and S^s , respectively. Define $V^+(S) = V^-(S^a) \cup V^+(S^s)$ and $V^-(S) = V^+(S^a) \cup V^-(S^s)$ and the *weight of a sequent* as the sum of the weights of the formulas occurring in that sequent. A sequent S is *lower than* a sequent T , if the weight of S is less than the weight of T .

$$\begin{array}{c}
 \frac{}{\Gamma, p \Rightarrow p, \Delta} \\
 \\
 \frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta} L\wedge \qquad \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta} R\wedge \\
 \\
 \frac{\Gamma, \varphi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta} L\vee \qquad \frac{\Gamma \Rightarrow \varphi, \psi, \Delta}{\Gamma \Rightarrow \varphi \vee \psi, \Delta} R\vee \\
 \\
 \frac{\Gamma \Rightarrow \varphi, \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \rightarrow \psi \Rightarrow \Delta} L\rightarrow \qquad \frac{\Gamma, \varphi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \rightarrow \psi, \Delta} R\rightarrow
 \end{array}$$

Fig. 1. The sequent calculus **G3cp**. In the axiom, p must be an atomic formula.

In this paper we are interested in modal extensions of the well-known sequent calculus **G3cp** from [18] (Fig. 1) for classical logic CPC and its extension by the following two weakening rules, denoted by **G3W**:

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma, \varphi \Rightarrow \Delta} Lw \qquad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \varphi, \Delta} Rw$$

In each rule in **G3W**, the multiset Γ (res. Δ) is called the left (res. right) *context*, the formulas outside $\Gamma \cup \Delta$ are called the *active formulas* of the rule and the only formula in the conclusion outside $\Gamma \cup \Delta$ is called the *main formula*. If S is the conclusion of an instance of a rule R , we say that R is *backwards applicable* to S . The modal rules by which we extend **G3cp** or **G3W** are given in Fig. 2. For any such rule (X) except (EC), (N) and (NW), if we add it to **G3W** we denote the resulting system by **GX**, and if we add (N) to that system we get **GXN**. Note that **GMCN** is the usual system for the logic K. If we add (EC) to **G3cp**, we get **GEC** and if we also add the rule (NW), we get **GECN**. Note that **GEC** and **GECN** have no explicit weakening rules.

The systems **GEC** and **GECN** are introduced in [10]. The others are equivalent to the systems introduced in [10]. The only difference is that in our representation, the weakening rules are explicitly present, while the extra context in the conclusion of the modal rules are omitted. We will present the systems as such for convenience in our later proofs. As the systems **GE**, **GM**, **GMC**, **GEN** and **GMN** are equivalent to the systems introduced in [10], they all admit the cut rule and the contraction rules. Moreover, the logics of these systems, i.e., the sets of formulas φ for which the systems prove $(\Rightarrow \varphi)$ are the well-known basic non-normal modal logics E, M, MC, EN and MN, respectively. The logics of the systems **GEC** and **GECN** are the logics EC and ECN, respectively [10].

$$\frac{\varphi \Rightarrow \psi \quad \psi \Rightarrow \varphi}{\Box \varphi \Rightarrow \Box \psi} E \quad \frac{\varphi_1, \dots, \varphi_n \Rightarrow \psi \quad \psi \Rightarrow \varphi_1 \quad \dots \quad \psi \Rightarrow \varphi_n}{\Sigma, \Box \varphi_1, \dots, \Box \varphi_n \Rightarrow \Box \psi, \Lambda} EC \ (n \geq 1)$$

$$\frac{\varphi \Rightarrow \psi}{\Box \varphi \Rightarrow \Box \psi} M \quad \frac{\Rightarrow \psi}{\Rightarrow \Box \psi} N \quad \frac{\Rightarrow \psi}{\Sigma \Rightarrow \Box \psi, \Lambda} NW \quad \frac{\varphi_1, \dots, \varphi_n \Rightarrow \psi}{\Box \varphi_1, \dots, \Box \varphi_n \Rightarrow \Box \psi} MC$$

Fig. 2. The modal rules

Here are some remarks about the rules introduced above. First, for any rule the weight of each premise is less than the weight of its conclusion. Specifically, the weight of $\Gamma, \Sigma \Rightarrow \Delta, \Lambda$ is less than the weight of $\Box \Gamma, \Sigma \Rightarrow \Box \Delta, \Lambda$, as long as $\Gamma \cup \Delta$ is non-empty. Second, in any rule in **G3W**, if we add a multiset, both to the antecedent (succedent) of the premises and to the antecedent (succedent) of the conclusion, the result remains an instance of the rule. We call this property the *context extension property*. Conversely, if a multiset is a sub-multiset of the left (right) context of the rule, then if we eliminate this multiset both from the premises and the conclusion, the result remains an instance of the rule. We call this property the *context restriction property*. Third, for any rule in **G3W** and any $\circ \in \{+, -\}$, if the main formula φ is in the antecedent, then for any active formula α in the antecedent of a premise and any active formula β in the succedent of a premise, we have $V^\circ(\alpha) \cup V^\circ(\beta) \subseteq V^\circ(\varphi)$, and if φ is in the succedent, we have $V^\circ(\alpha) \cup V^\circ(\beta) \subseteq V^\circ(\varphi)$ (note the use of \circ and \diamond). We call this property, the *variable preserving property*. As a consequence of this property for the rule $\frac{S_1 \dots S_n}{S}$ in **G3W**, we have $\bigcup_{i=1}^n V^\circ(S_i) \subseteq V^\circ(S)$.

3 Uniform Lyndon Interpolation

In this section, we prove ULIP for the logics E, M, MC, EN, and MN. To this end, we need to first extend the notion to the sequent calculi of these logics. Since all these logics are classical, we only define the universal quantifier, as the existential quantifier is constructed by the universal quantifier and negation.

Definition 5. Let G be one of the sequent calculi introduced in Preliminaries. G has uniform Lyndon interpolation property (ULIP) if for any sequent S , any atom p and any $\circ \in \{+, -\}$, there exists a formula $\forall^\circ pS$ such that:

- (var) $\forall^\circ pS$ is p° -free and $V^\dagger(\forall^\circ pS) \subseteq V^\dagger(S)$, for any $\dagger \in \{+, -\}$,
- (i) $S \cdot (\forall^\circ pS \Rightarrow)$ is derivable in G ,
- (ii) for any sequent $\Gamma \Rightarrow \Delta$ such that $p \notin V^\circ(\Gamma \Rightarrow \Delta)$, if $S \cdot (\Gamma \Rightarrow \Delta)$ is derivable in G then $(\Gamma \Rightarrow \forall^\circ pS, \Delta)$ is derivable in G .

$\forall^\circ pS$ is called a uniform \forall°_p -interpolant of S in G . For any set of rules \mathcal{R} of G , a formula $\forall^\circ_{\mathcal{R}} pS$ is called a uniform $\forall^\circ_{\mathcal{R}}$ -interpolant of S with respect to \mathcal{R} , if it satisfies the conditions (var) and (i), when $\forall^\circ pS$ is replaced by $\forall^\circ_{\mathcal{R}} pS$, and:

- (ii') for any sequent $\Gamma \Rightarrow \Delta$ such that $p \notin V^\circ(\Gamma \Rightarrow \Delta)$, if there is a derivation of $S \cdot (\Gamma \Rightarrow \Delta)$ in G whose last inference rule is an instance of a rule in \mathcal{R} , then $(\Gamma \Rightarrow \forall^\circ_{\mathcal{R}} pS, \Delta)$ is derivable in G .

Remark 2. As the formula $\forall^\circ pS$ is provably unique, using the functional notation of writing $\forall^\circ pS$ as a function with the arguments $\circ \in \{+, -\}$, p and S is allowed. The same does not hold for $\forall^\circ_{\mathcal{R}} pS$. However, as there is no risk of confusion and we will be specific about the construction of the formula $\forall^\circ_{\mathcal{R}} pS$, we will also use the functional notation in this case.

The following theorem connects ULIP for sequent calculi to the original version.

Theorem 2. Let G be one of the sequent calculi introduced in Preliminaries and L be its logic. Then, G has ULIP iff L has ULIP.

Proof. If G has ULIP, set $\forall^\circ pA = \forall^\circ p(\Rightarrow A)$ and $\exists^\circ pA = \neg \forall^\circ p \neg A$. Conversely, if L has ULIP, set $\forall^\circ p(\Gamma \Rightarrow \Delta) = \forall^\circ p(\bigwedge \Gamma \rightarrow \bigvee \Delta)$.

Our strategy to prove ULIP for the logics E, M, MC, EN, and MN is to prove ULIP for their sequent calculi. From now on, up to Subsect. 3.1, we assume that G is one of **GE**, **GM**, **GMC**, **GEN**, and **GMN**. As stated previously, backward applications of the rules decreases the weight of the sequent. Using this property and recursion on the weight of the sequents, for any given sequent $S = (\Gamma \Rightarrow \Delta)$, any atom p and any $\circ \in \{+, -\}$, we first define a p° -free formula $\forall^\circ pS$ and then by induction on the weight of S , we prove that $\forall^\circ pS$ meets the conditions in Definition 5. Towards that end, both in the definition of $\forall^\circ pS$ and in the proof of its properties, we must address all the rules of the system G , one by one. To make the presentation uniform, modular, and more clear, we divide the rules of G into two families: the rules of **G3W** and the modal rules specific for G . The rules in the first class has one of the following forms:

$$\frac{\{ \Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \Delta \}_i}{\Gamma, \varphi \Rightarrow \Delta} \qquad \frac{\{ \Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \Delta \}_i}{\Gamma \Rightarrow \varphi, \Delta}$$

where Γ and Δ are free for all multiset substitutions, and $\bar{\varphi}_i$'s and $\bar{\psi}_i$'s are multisets of formulas (possibly empty). The rules have the variable preserving condition, i.e., given $\circ \in \{+, -\}$, for the left rule $\bigcup_i \bigcup_{\theta \in \bar{\varphi}_i} V^\circ(\theta) \cup \bigcup_i \bigcup_{\theta \in \bar{\psi}_i} V^\circ(\theta) \subseteq V^\circ(\varphi)$, and for the right one $\bigcup_i \bigcup_{\theta \in \bar{\varphi}_i} V^\circ(\theta) \cup \bigcup_i \bigcup_{\theta \in \bar{\psi}_i} V^\circ(\theta) \subseteq V^\circ(\varphi)$. Rather than addressing each rule in **G3W**, we simply address these two forms.

Lemma 1. *For any sequent S , atom p and $\circ \in \{+, -\}$, a uniform \forall_p° -interpolant of S with respect to the set of all axioms of G exists.*

Proof. Let us define a formula $\forall_{ax}^\circ pS$: if S is provable, define it as \top , otherwise, define it as the disjunction of all p° -free formulas in S^s and the negation of all p° -free formulas in S^a . We show that $\forall_{ax}^\circ pS$ is the uniform \forall_p° -interpolant of S with respect to the set of axioms of G . It is easy to see that $\forall_{ax}^\circ pS$ is p° -free, $V^\dagger(\forall_{ax}^\circ pS) \subseteq V^\dagger(S)$, for $\dagger \in \{+, -\}$ and $S \cdot (\forall_{ax}^\circ pS \Rightarrow)$ is provable in G . To prove the condition (i') in Definition 5, if S is provable, then as $\forall_{ax}^\circ pS = \top$, we have $\bar{C} \Rightarrow \forall_{ax}^\circ pS, \bar{D}$. If S is not provable, then let $S \cdot (\bar{C} \Rightarrow \bar{D})$ be an axiom. There are two cases to consider. First, if $S \cdot (\bar{C} \Rightarrow \bar{D})$ is in the form $\Gamma, q \Rightarrow q, \Delta$, where q is an atomic formula. Then, if $q \notin \bar{C}$ and $q \notin \bar{D}$, we have $q \in \Gamma \cap \Delta$ and hence the sequent S is provable which contradicts our assumption. Therefore, either $q \in \bar{C}$ or $q \in \bar{D}$. If $q \in \bar{C} \cap \bar{D}$, then $\bar{C} \Rightarrow \forall_{ax}^\circ pS, \bar{D}$ is provable. Hence, we assume either $q \in \bar{C}$ and $q \notin \bar{D}$ or $q \notin \bar{C}$ and $q \in \bar{D}$. In the first case, if $q \in \bar{C}$, it is p° -free and since it occurs in Δ , it is a disjunct in $\forall_{ax}^\circ pS$. Hence, $\bar{C} \Rightarrow \forall_{ax}^\circ pS, \bar{D}$ is provable. In the second case, if $q \in \bar{D}$, it is p° -free and as $q \in \Gamma$, its negation occurs in $\forall_{ax}^\circ pS$. Therefore $\bar{C} \Rightarrow \forall_{ax}^\circ pS, \bar{D}$ is provable. If $S \cdot (\bar{C} \Rightarrow \bar{D})$ is in the form $\Gamma, \perp \Rightarrow \Delta$, then $\perp \in \bar{C}$, because otherwise, $\perp \in \Gamma$ and hence S will be provable. Now, since $\perp \in \bar{C}$, we have $\bar{C} \Rightarrow \forall_{ax}^\circ pS, \bar{D}$.

Definition 6. *Let $\mathcal{U}_p^\circ(S)$ be the statement that “all sequents lower than S have uniform \forall_p° -interpolants”. A calculus G has MUIP if for any sequent S , atom p , and $\circ \in \{+, -\}$, there exists a formula $\forall_m^\circ pS$ such that if $\mathcal{U}_p^\circ(S)$, then $\forall_m^\circ pS$ is a uniform \forall_p° -interpolant for S with respect to the set of modal rules of G .*

Theorem 3. *If a sequent calculus G has MUIP, then it has ULIP.*

Proof. Define a formula $\forall^\circ pS$ by recursion on the weight of S : if S is provable define it as \top , otherwise, define it as:

$$\bigvee_R \left(\bigwedge_i \forall^\circ pS_i \right) \vee (\forall_{ax}^\circ pS) \vee (\forall_m^\circ pS)$$

where the first disjunction is over all rules R in **G3W** backward applicable to S , where S is the consequence and S_i 's are the premises. $\forall_{ax}^\circ pS$ is a uniform \forall_p° -interpolant of S with respect to the set of axioms of G that Lemma 1 provides. $\forall_m^\circ pS$ is the formula that MUIP provides. To prove that $\forall^\circ pS$ is a \forall_p° -interpolant for S , we use induction on the weight of S to prove:

- (var) $\forall^\circ pS$ is p° -free and $V^\dagger(\forall^\circ pS) \subseteq V^\dagger(S)$, for any $\dagger \in \{+, -\}$,
- (i) $S \cdot (\forall^\circ pS \Rightarrow)$ is provable in G ,

(ii) for any p° -free sequent $\bar{C} \Rightarrow \bar{D}$, if $S \cdot (\bar{C} \Rightarrow \bar{D})$ is derivable in G then $\bar{C} \Rightarrow \forall^\circ pS, \bar{D}$ is derivable in G .

By induction hypothesis, (var), (i), and (ii) hold for all sequents T lower than S . Now, (var) also holds for $\forall^\circ pS$, because both $\forall_{ax}^\circ pS$ and $\forall_m^\circ pS$ satisfy (var) and all rules in **G3W** have the variable preserving property.

To prove (i), it is enough to show that the following are provable in G :

$$S \cdot \left(\bigwedge_i \forall^\circ pS_i \Rightarrow \right) \quad (1), \quad S \cdot (\forall_{ax}^\circ pS \Rightarrow) \quad (2), \quad S \cdot (\forall_m^\circ pS \Rightarrow) \quad (3).$$

Sequent (3) is provable by induction hypothesis and the assumption that G has MUIP. Sequent (2) is proved in Lemma 1. For the sequent (1), assume that the rule R of **G3W** is backward applicable to S , i.e., the premises of R are S_i 's and its conclusion S . As S_i 's are lower than S , by induction hypothesis we have $S_i \cdot (\forall^\circ pS_i \Rightarrow)$. Therefore, by weakening, we have $S_i \cdot (\{\forall^\circ pS_i\}_i \Rightarrow)$. Since any rule in **G3W** has the context extension property, we can add $\{\forall^\circ pS_i\}_i$ to the antecedent of both premises and conclusion and by the rule itself, we have $S \cdot (\{\forall^\circ pS_i\}_i \Rightarrow)$ and hence $S \cdot (\bigwedge_i \forall^\circ pS_i \Rightarrow)$.

For (ii), we use induction on the length of the proof of $S \cdot (\bar{C} \Rightarrow \bar{D})$. Let $S \cdot (\bar{C} \Rightarrow \bar{D})$ be derivable in G . If it is an axiom, we have $\bar{C} \Rightarrow \bar{D}, \forall_{ax}^\circ pS$ by Lemma 1, and hence $\bar{C} \Rightarrow \bar{D}, \forall^\circ pS$. If the last rule is a rule in **G3W** of the form:

$$\frac{\{\Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \Delta\}_i}{\Gamma, \varphi \Rightarrow \Delta},$$

then there are two cases to consider, i.e., either $\varphi \in \bar{C}$ or $\varphi \in S^a$. If $\varphi \in \bar{C}$, then set $\bar{C}' = \bar{C} - \{\varphi\}$. Since $\varphi \in \bar{C}$, it is p° -free by the assumption and φ_i 's are all p° -free and ψ_i 's are all p° -free by the variable preserving property. By induction hypothesis, as $(\bar{C}', \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \bar{D})$ is p° -free and $S \cdot (\bar{C}', \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \bar{D})$ has a shorter proof, we have $\bar{C}', \bar{\varphi}_i \Rightarrow \forall^\circ pS, \bar{\psi}_i, \bar{D}$. By using the rule itself, we have

$$\frac{\{\bar{C}', \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \forall^\circ pS, \bar{D}\}_i}{\bar{C}', \varphi \Rightarrow \forall^\circ pS, \bar{D}}$$

which implies $\bar{C} \Rightarrow \forall^\circ pS, \bar{D}$.

If $\varphi \notin \bar{C}$, then both \bar{C} and \bar{D} do not contain any active formula of the rule and hence the last rule is in form

$$\frac{\{\bar{C}, \Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \bar{D}, \Delta\}_i}{\bar{C}, \Gamma, \varphi \Rightarrow \bar{D}, \Delta}.$$

By context restriction property, if we erase \bar{C} and \bar{D} both on the premises and the consequence of the last rule, the rule remains valid and it changes to:

$$\frac{\{\Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \Delta\}_i}{\Gamma, \varphi \Rightarrow \Delta}.$$

Therefore, the rule is backward applicable to $S = (\Gamma, \varphi \Rightarrow \Delta)$. Set $S_i = (\Gamma, \bar{\varphi}_i \Rightarrow \bar{\psi}_i, \Delta)$. As the weight of S_i 's are less than the weight of S and $S_i \cdot (\bar{C} \Rightarrow \bar{D})$

are provable, by induction hypothesis, we have $\bar{C} \Rightarrow \forall^\circ pS_i, \bar{D}$. Hence, $\bar{C} \Rightarrow \bigwedge_i \forall^\circ pS_i, \bar{D}$ and as $\bigwedge_i \forall^\circ pS_i$ is a disjunct in $\forall^\circ pS$, we have $\bar{C} \Rightarrow \forall^\circ pS, \bar{D}$. The case where the last rule is in **GW3** with its main formula in the antecedent is similar. For the modal rules, by induction hypothesis $\mathcal{U}_p^\circ(S)$ and the assumption that G has MUIP, we get that $\forall_m^\circ pS$ is a uniform \forall_p° -interpolant for S with respect to the set of modal rules of G . By (ii') in Definition 5, this gives $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$ and hence $\bar{C} \Rightarrow \forall^\circ pS, \bar{D}$.

In the upcoming subsections, for the following choices of the system G , we show that it has MUIP. Therefore, by Theorem 3 and Theorem 2, we will have:

Theorem 4. *Logics E, M, MC, EN and MN have ULIP, hence UIP and LIP.*

3.1 Modal Logics M and MN

Let G be either **GM** or **GMN**. We will show that G has MUIP. To define $\forall_m^\circ pS$, if $\neg \mathcal{U}_p^\circ(S)$, define $\forall_m^\circ pS$ as \perp . If $\mathcal{U}_p^\circ(S)$, (i.e., for any sequent T lower than S a uniform \forall_p° -interpolant, denoted by $\forall^\circ pT$, exists), define $\forall_m^\circ pS$ in the following way: if S is provable, define it as \top , otherwise, if it is of the form $(\Box\varphi \Rightarrow)$, define $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$, where $S' = (\varphi \Rightarrow)$, if S is of the form $(\Rightarrow \Box\psi)$, define $\forall_m^\circ pS = \Box\forall^\circ pS''$, where $S'' = (\Rightarrow \psi)$, and otherwise, define $\forall_m^\circ pS = \perp$. Note that $\forall_m^\circ pS$ is well-defined as we have $\mathcal{U}_p^\circ(S)$ and S' and S'' are lower than S .

To show that G has MUIP, we assume $\mathcal{U}_p^\circ(S)$ to prove the three conditions (var), (i) and (ii') in Definition 5 for $\forall_m^\circ pS$. First, note that using $\mathcal{U}_p^\circ(S)$ on $(\varphi \Rightarrow)$ and $(\Rightarrow \psi)$ that are lower than $(\Box\varphi \Rightarrow)$ and $(\Rightarrow \Box\psi)$, respectively, the variable conditions are implied from (var) for S' and S'' , respectively.

For (i), if S is provable, there is nothing to prove. Otherwise, if $S = (\Box\varphi \Rightarrow)$ then $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$. As S' is lower than S , we have $(\varphi, \forall^\circ pS' \Rightarrow)$ by $\mathcal{U}_p^\circ(S)$, which implies $(\varphi \Rightarrow \neg\forall^\circ pS')$. Using the rule (M), we get $(\Box\varphi \Rightarrow \Box\neg\forall^\circ pS')$, which is equivalent to $(\Box\varphi, \neg\Box\neg\forall^\circ pS' \Rightarrow)$. Hence, $S \cdot (\forall_m^\circ pS \Rightarrow)$ is provable.

If S is not provable and $S = (\Rightarrow \Box\psi)$, we have $\forall_m^\circ pS = \Box\forall^\circ pS''$. Using $\mathcal{U}_p^\circ(S)$ on S'' and the fact that S'' is lower than S , we have $(\forall^\circ pS'' \Rightarrow \psi)$ and by the rule (M), we can show that $S \cdot (\Box\forall^\circ pS'' \Rightarrow)$ is provable in G . If S is not provable and has none of the mentioned forms, as $\forall_m^\circ pS = \perp$, there is nothing to prove.

For (ii'), let $S \cdot (\bar{C} \Rightarrow \bar{D})$ be derivable in G for a p° -free sequent $\bar{C} \Rightarrow \bar{D}$ and the last rule is a modal rule. We want to show that $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$ is derivable in G . If the last rule used in the proof of $S \cdot (\bar{C} \Rightarrow \bar{D})$ is (M), the sequent must have the form $(\Box\varphi \Rightarrow \Box\psi)$ and the rule must be in form:

$$\frac{\varphi \Rightarrow \psi}{\Box\varphi \Rightarrow \Box\psi} M$$

If S is provable, as $\forall_m^\circ pS = \top$, we clearly have $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$. Assume S is not provable and hence $\bar{C} \cup \bar{D}$ cannot be empty. Therefore, there are three cases to consider, either \bar{C} is $\Box\varphi$ or \bar{D} is $\Box\psi$ or both. First, if $\bar{C} = \Box\varphi$ and $\bar{D} = \emptyset$, then, $S = (\Rightarrow \Box\psi)$ and φ is p° -free. Set $S'' = (\Rightarrow \psi)$. Then $\forall_m^\circ pS = \Box\forall^\circ pS''$. As S''

is lower than S , by $\mathcal{U}_p^\circ(S)$ we have $(\varphi \Rightarrow \forall^\circ pS'')$. Using the modal rule (M) , we have $(\Box\varphi \Rightarrow \Box\forall^\circ pS'')$ and hence $(\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D})$.

In the second case, assume $\bar{C} = \emptyset$ and $\bar{D} = \Box\psi$. Hence, $S = (\Box\varphi \Rightarrow)$ and ψ is p° -free. Set $S' = (\varphi \Rightarrow)$. Hence, $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$. Since $(\varphi \Rightarrow \psi)$ is provable in G and S' is lower than S , by \mathcal{U}_p° we have $(\Rightarrow \forall^\circ pS', \psi)$, or equivalently $(\neg\forall^\circ pS' \Rightarrow \psi)$. Using the rule (M) , we get $(\Box\neg\forall^\circ pS' \Rightarrow \Box\psi)$ or equivalently $(\Rightarrow \neg\Box\neg\forall^\circ pS', \Box\psi)$. Therefore, we have $(\Rightarrow \forall_m^\circ pS, \Box\psi)$ or $(\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D})$.

In the third case, if $\bar{C} = \Box\varphi$ and $\bar{D} = \Box\psi$, then S is the empty sequent and $\bar{C} \Rightarrow \bar{D}$ is provable. Hence, $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$ is also provable.

For the case $G = \mathbf{GMN}$, if $S \cdot (\bar{C} \Rightarrow \bar{D}) = (\Rightarrow \Box\psi)$ is proved by the rule (N) , it must have the following form:

$$\frac{\Rightarrow \psi}{\Rightarrow \Box\psi} N$$

Then $\bar{C} = \emptyset$ and there are two cases to consider. The first case is when $S = (\Rightarrow \Box\psi)$ and $\bar{D} = \emptyset$. Then, it means that S is provable which contradicts our assumption. The second case is when $S = (\Rightarrow)$ and $\bar{D} = \Box\psi$. Hence, $\bar{C} \Rightarrow \bar{D}$ is provable and we have the provability of $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$ in G .

3.2 Modal Logic MC

Similar to the argument of the previous subsection, to define $\forall_m^\circ pS$, if $\neg\mathcal{U}_p^\circ(S)$, define $\forall_m^\circ pS$ as \perp . If $\mathcal{U}_p^\circ(S)$, (i.e., for any sequent T lower than S the uniform \forall_p° -interpolant, denoted by $\forall^\circ pT$, exists), define $\forall_m^\circ pS$ as the following: if S is provable, define $\forall_m^\circ pS = \top$. Otherwise, if S is of the form $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow)$, for some $i \geq 1$, define $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$, where $S' = (\varphi_1, \dots, \varphi_i \Rightarrow)$. If S is of the form $(\Rightarrow \Box\psi)$, define $\forall_m^\circ pS = \Box\forall^\circ pS''$, where $S'' = (\Rightarrow \psi)$. If S is of the form $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow \Box\psi)$, for some $i \geq 1$, define $\forall_m^\circ pS = \Box\forall^\circ pS''$, where $S'' = (\varphi_1, \dots, \varphi_i \Rightarrow \psi)$. Otherwise, define $\forall_m^\circ pS = \perp$. Note that $\forall_m^\circ pS$ is well-defined as we assumed $\mathcal{U}_p^\circ(S)$ and in each case S' or S'' are lower than S .

To show that \mathbf{GMC} has MUIP, we assume $\mathcal{U}_p^\circ(S)$ to prove the three conditions (var) , (i) and (ii') in Definition 5 for $\forall_m^\circ pS$. The condition (var) is an immediate consequence of $\mathcal{U}_p^\circ(S)$ and the fact that S' or S'' are lower than S . For (i) , if S is provable, there is nothing to prove. If S is of the form $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow)$ and $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$, where $S' = (\varphi_1, \dots, \varphi_i \Rightarrow)$, as S' is lower than S , by $\mathcal{U}_p^\circ(S)$ we have $(\varphi_1, \dots, \varphi_i, \forall^\circ pS' \Rightarrow)$ or equivalently $(\varphi_1, \dots, \varphi_i \Rightarrow \neg\forall^\circ pS')$. Using the rule (MC) , we get $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow \Box\neg\forall^\circ pS')$, which is equivalent to $(\Box\varphi_1, \dots, \Box\varphi_i, \neg\Box\neg\forall^\circ pS' \Rightarrow)$ and hence $S \cdot (\forall_m^\circ pS \Rightarrow)$.

If S is of the form $(\Rightarrow \Box\psi)$ and $S'' = (\Rightarrow \psi)$, or S is of the form $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow \Box\psi)$, for some $i \geq 1$ and S'' is of the form $(\varphi_1, \dots, \varphi_i \Rightarrow \psi)$, we have $\forall_m^\circ pS = \Box\forall^\circ pS''$. In both cases, using $\mathcal{U}_p^\circ(S)$ on S'' , we have either $\forall^\circ pS'' \Rightarrow \psi$ or $\varphi_1, \dots, \varphi_i, \forall^\circ pS'' \Rightarrow \psi$, respectively. In both cases, using the rule (MC) , we can show that $S \cdot (\Box\forall^\circ pS'' \Rightarrow)$ is provable and hence $S \cdot (\forall_m^\circ pS \Rightarrow)$.

For (ii') , let $S \cdot (\bar{C} \Rightarrow \bar{D})$ be derivable in \mathbf{GMC} and the last rule is the modal rule (MC) , for a p° -free sequent $\bar{C} \Rightarrow \bar{D}$. We want to show that $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$

is derivable in **GMC**. If S is provable, as $\forall_m pS = \top$, we have $\bar{C} \Rightarrow \forall_m pS, \bar{D}$. Therefore, we assume that S is not provable. As the last rule used in the proof of $S \cdot (\bar{C} \Rightarrow \bar{D})$ is (MC) , the sequent must have the form $(\Box\varphi_1, \dots, \Box\varphi_n \Rightarrow \Box\psi)$ and the rule is:

$$\frac{\varphi_1, \dots, \varphi_n \Rightarrow \psi}{\Box\varphi_1, \dots, \Box\varphi_n \Rightarrow \Box\psi} MC$$

Then, there are two cases to consider, either $\bar{D} = \Box\psi$ or $\bar{D} = \emptyset$. First, assume S is of the form $(\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow)$, for $i \leq n$, then $\bar{C} = \Box\varphi_{i+1}, \dots, \Box\varphi_n$ and $\bar{D} = \Box\psi$ and hence $\varphi_{i+1}, \dots, \varphi_n \Rightarrow \psi$ is p° -free. Set $S' = (\varphi_1, \dots, \varphi_i \Rightarrow)$. By the form of S , we have $\forall_m pS = \neg\Box\neg\forall^\circ pS'$. As S' is lower than S , by $\mathcal{U}_p^\circ(S)$, we have $(\varphi_{i+1}, \dots, \varphi_n \Rightarrow \forall^\circ pS', \psi)$. Hence, by moving $\forall^\circ pS'$ to the left, applying the rule (MC) and moving back, we have $(\Box\varphi_{i+1}, \dots, \Box\varphi_n \Rightarrow \neg\Box\neg\forall^\circ pS', \Box\psi)$ or equivalently $(\bar{C} \Rightarrow \forall_m pS, \bar{D})$.

If S is of the form $\Box\varphi_1, \dots, \Box\varphi_i \Rightarrow \Box\psi$, for some $i \leq n$, we must have $\bar{C} = \Box\varphi_{i+1}, \dots, \Box\varphi_n$ and $\bar{D} = \emptyset$. Hence, $\varphi_{i+1}, \dots, \varphi_n$ are p° -free. Note that $i < n$, because if $i = n$, then S will be provable that contradicts our assumption. Set $S'' = (\varphi_1, \dots, \varphi_i \Rightarrow \psi)$. As S'' is lower than S , by $\mathcal{U}_p^\circ(S)$ we have $\varphi_{i+1}, \dots, \varphi_n \Rightarrow \forall^\circ pS''$. By the fact that $i < n$, we can apply the rule (MC) to prove $\Box\varphi_{i+1}, \dots, \Box\varphi_n \Rightarrow \Box\forall^\circ pS''$ and hence $(\bar{C} \Rightarrow \forall_m pS, \bar{D})$.

3.3 Modal Logics E and EN

Let G be **GE** or **GEN**. Similar to the argument of the previous subsection, if $\neg\mathcal{U}_p^\circ(S)$, define $\forall_m pS$ as \perp . If $\mathcal{U}_p^\circ(S)$, then: if S is provable in G , define $\forall_m pS = \top$. Otherwise, if $S = (\Box\varphi \Rightarrow)$ and both $(\neg\forall^\circ pS' \Rightarrow \varphi)$ and $(\varphi \Rightarrow \neg\forall^\circ pS')$ are provable in G , define $\forall_m pS = \neg\Box\neg\forall^\circ pS'$ for $S' = (\varphi \Rightarrow)$. If S has the form $(\Rightarrow \Box\psi)$ and both $(\forall^\circ pS'' \Rightarrow \psi)$ and $(\psi \Rightarrow \forall^\circ pS'')$ are provable in G , define $\forall_m pS = \Box\forall^\circ pS''$ for $S'' = (\Rightarrow \psi)$. Otherwise, define $\forall_m pS = \perp$. Note that $\forall^\circ pS$ is well-defined as S' and S'' are lower than S and we assumed $\mathcal{U}_p^\circ(S)$.

To show that G has MUIP we assume $\mathcal{U}_p^\circ(S)$ to prove (var) , (i) and (ii') in Definition 5 for $\forall_m pS$. Condition (var) is a consequence of $\mathcal{U}_p^\circ(S)$ and that S' or S'' are lower than S . For (i) , if S is provable, there is nothing to prove. If $S = (\Box\varphi \Rightarrow)$ and $S' = (\varphi \Rightarrow)$ and both $(\neg\forall^\circ pS' \Rightarrow \varphi)$ and $(\varphi \Rightarrow \neg\forall^\circ pS')$ are provable in G , then using the rule (E) , we have $(\Box\varphi \Rightarrow \Box\neg\forall^\circ pS')$ which implies $(\Box\varphi, \neg\Box\neg\forall^\circ pS' \Rightarrow)$ and hence $S \cdot (\forall_m pS \Rightarrow)$ is provable in G .

If $S = (\Rightarrow \Box\psi)$ and $S'' = (\Rightarrow \psi)$ and both $(\forall^\circ pS'' \Rightarrow \psi)$ and $(\psi \Rightarrow \forall^\circ pS'')$ are provable in G , then using the rule (E) , we have $(\Box\forall^\circ pS'' \Rightarrow \Box\psi)$ and hence $S \cdot (\forall_m pS \Rightarrow)$ is provable in G . If $\forall_m pS = \perp$, there is nothing to prove.

For (ii') , if S is provable, then $\forall_m pS = \top$ and hence $\bar{C} \Rightarrow \forall_m pS, \bar{D}$. Therefore, assume that S is not provable. If the last rule used in the proof of $S \cdot (\bar{C} \Rightarrow \bar{D})$ is the rule (E) , the sequent $S \cdot (\bar{C} \Rightarrow \bar{D})$ is of the form $\Box\varphi \Rightarrow \Box\psi$. There are four cases to consider based on if \bar{C} or \bar{D} are empty or not. First, if $\bar{C} = \bar{D} = \emptyset$, then S is provable which contradicts our assumption. If S is the empty sequent (\Rightarrow) , then $\bar{C} \Rightarrow \bar{D}$ is provable and hence $\bar{C} \Rightarrow \forall_m pS, \bar{D}$ is provable.

If $S = (\Box\varphi \Rightarrow)$, then $\bar{C} = \emptyset$ and $\bar{D} = \Box\psi$ and hence ψ is p° -free. Set $S' = (\varphi \Rightarrow)$ and as the last rule is (E) , both $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$ are provable. By $\mathcal{U}_p^\circ(S)$ and the fact that S' is lower than S , we have $(\varphi, \forall^\circ pS' \Rightarrow)$ or equivalently, $(\varphi \Rightarrow \neg\forall^\circ pS')$. Again by $\mathcal{U}_p^\circ(S)$ for S' , the provability of $S' \cdot (\Rightarrow \bar{D}) = (\varphi \Rightarrow \psi)$ and the fact that $(\Rightarrow \psi)$ is p° -free, we have $(\Rightarrow \forall^\circ pS', \psi)$ or equivalently, $(\neg\forall^\circ pS' \Rightarrow \psi)$. Since $(\varphi \Rightarrow \psi)$ and $(\psi \Rightarrow \varphi)$ are provable, by cut we can prove the equivalence between φ, ψ and $\neg\forall^\circ pS'$. Using this fact, we have:

$$\frac{\psi \Rightarrow \neg\forall^\circ pS' \quad \neg\forall^\circ pS' \Rightarrow \psi}{\Box\neg\forall^\circ pS' \Rightarrow \Box\psi} E$$

Hence, $(\Rightarrow \neg\Box\neg\forall^\circ pS', \Box\psi)$. Then, as $S = (\Box\varphi \Rightarrow)$ and both $(\neg\forall^\circ pS' \Rightarrow \varphi)$ and $(\varphi \Rightarrow \neg\forall^\circ pS')$ are provable in G , by definition we have $\forall_m^\circ pS = \neg\Box\neg\forall^\circ pS'$ and hence $(\Rightarrow \neg\Box\neg\forall^\circ pS', \Box\psi) = (\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D})$ is provable in G . The last case where $S = (\Rightarrow \Box\psi)$ and $\bar{C} = \Box\varphi$ and $\bar{D} = \emptyset$ is similar.

For the case $G = \mathbf{GEN}$, if $S \cdot (\bar{C} \Rightarrow \bar{D}) = (\Rightarrow \Box\psi)$ is proved by the rule (N) , it must have the form $\frac{\Rightarrow \psi}{\Rightarrow \Box\psi} N$. Then $\bar{C} = \emptyset$ and there are two cases. First, $S = (\Rightarrow \Box\psi)$ and $\bar{D} = \emptyset$, which means that S is provable which contradicts our assumption. Second, if $S = (\Rightarrow)$ and $\bar{D} = \Box\psi$, and hence $\bar{C} \Rightarrow \bar{D}$ is provable, we have the provability of $\bar{C} \Rightarrow \forall_m^\circ pS, \bar{D}$ in G .

4 Modal Logics EC and ECN

In this section we prove that the logics EC and ECN do not enjoy the Craig interpolation property. To this end, we set $\varphi = \Box(\neg q \wedge r)$ and $\psi = \Box(p \wedge q) \rightarrow \Box\perp$, where p, q , and r are three distinct atomic formulas and show that if L is either EC or ECN, the formula $\varphi \rightarrow \psi$ is provable in L , while there is no formula θ such that $V(\theta) \subseteq \{q\}$ and both formulas $\varphi \rightarrow \theta$ and $\theta \rightarrow \psi$ are provable in L .

To show that $\varphi \rightarrow \psi$ is in EC and hence ECN, we use the following proof tree in **GE**C:

$$\frac{p \wedge q, \neg q \wedge r \Rightarrow \perp \quad \perp \Rightarrow p \wedge q \quad \perp \Rightarrow \neg q \wedge r}{\Box(p \wedge q), \Box(\neg q \wedge r) \Rightarrow \Box\perp} EC$$

Now, for the sake of contradiction, assume that the interpolant θ for $\varphi \rightarrow \psi$ exists. Let G be either **GE**C or **GE**CN. Hence, both $\Box(\neg q \wedge r) \Rightarrow \theta$ and $\Box(p \wedge q), \theta \Rightarrow \Box\perp$ are provable in G . We first analyse the general form of θ .

First, note that by a simple induction on the structure of the formulas in the language \mathcal{L} , it is possible to show that any formula A is **G3cp**-equivalent to a CNF-style formula $\bigwedge_{i \in I} \bigvee_{j \in J_i} L_{ij}$, where I and J_i 's are (possibly empty) finite sets, $V(L_{ij}) \subseteq V(A)$, and each L_{ij} is either an atomic formula, the negation of an atomic formula, $\Box C$ or $\neg\Box C$, for a formula C . In particular, the formula θ is **G3cp**-equivalent to a CNF-style formula in the form $\bigwedge_{i \in I} \bigvee_{j \in J_i} L_{ij}$. W.l.o.g., assume that for any $i \in I$, it is impossible to have both an atomic formula and its negation in $\{L_{ij}\}_{j \in J_i}$, and that none of sequents $(\Rightarrow L_{ij})$ or $(L_{ij} \Rightarrow)$ are provable in G .

Back to the main argument, as $\varphi \Rightarrow \theta$ is provable in G , we have $\varphi \Rightarrow \bigwedge_{i \in I} \bigvee_{j \in J_i} L_{ij}$ which means that for every $i \in I$, we have $\varphi \Rightarrow \bigvee_{j \in J_i} L_{ij}$. Based on the form of each L_{ij} , we can transform the sequent to a provable sequent of the form $\varphi, P, \Box \Gamma \Rightarrow Q, \Box \Delta$, where P and Q are multisets of atomic formulas and Γ and Δ are multisets of formulas. We claim that for any $i \in I$, the corresponding Γ is non-empty. Suppose $\Gamma = \emptyset$. Then, we have $\varphi, P \Rightarrow Q, \Box \Delta$. This sequent must have been the conclusion of the rule (EC) , because for $G = \mathbf{GEC}$, the other possible case is being an axiom which implies either $\perp \in P$ or the existence of an atomic s in $P \cap Q$. Both contradict the structure of $\bigvee_{j \in J_i} L_{ij}$. For $G = \mathbf{GECN}$, the same holds. Moreover, if the last rule is (NW) , then for an element $\delta \in \Delta$, the sequent $(\Rightarrow \delta)$ and hence $(\Rightarrow \Box \delta)$ must be provable in G which contradicts the structure of L_{ij} 's again. Therefore, $T = (\varphi, P \Rightarrow Q, \Box \Delta)$ is the consequence of (EC) and hence, it has the form $(\Sigma, \Box \alpha_1, \dots, \Box \alpha_n \Rightarrow \Box \beta, \Lambda)$ and the last rule is:

$$\frac{\alpha_1, \dots, \alpha_n \Rightarrow \beta \quad \beta \Rightarrow \alpha_1 \quad \dots \quad \beta \Rightarrow \alpha_n}{\Sigma, \Box \alpha_1, \dots, \Box \alpha_n \Rightarrow \Box \beta, \Lambda} EC$$

Now there are two cases, either $\varphi \in \Sigma$ or $\varphi \notin \Sigma$. In the first case, as the formulas outside Σ are either atomic or boxed, we must have no boxed formula outside Σ . This is impossible, as the form of the rule (EC) dictates that we must have at least one boxed formula in the antecedent of the conclusion. Hence, $\varphi \notin \Sigma$.

As all formulas in T^a (except φ) are atomic, we must have only one boxed formula in T^a , which is φ . Therefore, in the premises of the rule, we have $\neg q \wedge r \Rightarrow \beta$ and $\beta \Rightarrow \neg q \wedge r$. Since $V(\beta) \subseteq V(\theta) \subseteq \{q\}$, then β is r -free. If we once substitute \perp for r and then $\neg q$ for r , as β remains intact, we will have $\beta \Leftrightarrow \perp$ and $\beta \Leftrightarrow \neg q$, which implies the contradictory $\perp \Leftrightarrow \neg q$. Hence, Γ cannot be empty.

So far, we have proved that Γ is non-empty, for any $i \in I$. Let D_i be a formula in Γ and note that $\neg \Box D_i$ occurs as one of the L_{ij} 's. Now, as $\Box(p \wedge q), \theta \Rightarrow \Box \perp$ or equivalently $\Box(p \wedge q), \bigwedge_{i \in I} \bigvee_{j \in J_i} L_{ij} \Rightarrow \Box \perp$ is provable in G , we have $\Box(p \wedge q), \{\neg \Box D_i\}_{i \in I} \Rightarrow \Box \perp$ is provable in G . Define $\mathcal{D} = \{D_i\}_{i \in I}$. Thus $S = (\Box(p \wedge q) \Rightarrow \Box \mathcal{D}, \Box \perp)$ is provable. As all the formulas are boxed, this must have been the conclusion of the rule (EC) . The reason is that G has no weakening rules, and for $G = \mathbf{GEC}$, the only modal rule is (EC) and for $G = \mathbf{GECN}$, the last rule cannot be the rule (NW) as it implies that for one $D \in \mathcal{D}$ the sequent $(\Rightarrow D)$ is provable in G which means that $(\Rightarrow \Box D)$ and hence $(\neg \Box D \Rightarrow)$ is provable. The last contradicts with the structure of L_{ij} 's. This implies that the last inference is of the form:

$$\frac{\alpha_1, \dots, \alpha_n \Rightarrow \beta \quad \beta \Rightarrow \alpha_1 \quad \dots \quad \beta \Rightarrow \alpha_n}{\Sigma, \Box \alpha_1, \dots, \Box \alpha_n \Rightarrow \Box \beta, \Lambda} EC$$

Similar as before, there are two cases, either $\beta = \perp$ or $\beta \in \mathcal{D}$. If $\beta = \perp$, in the premises we must have $p \wedge q \Leftrightarrow \perp$ which is impossible. If $\beta \in \mathcal{D}$, it means that in the premises we had $p \wedge q \Leftrightarrow \beta$. Note that as $\beta \in \mathcal{D}$ we have $V(\beta) \subseteq V(\theta) \subseteq \{q\}$. Hence β is p -free. Substituting once \perp and then q for p , leave β intact and hence we get $\perp \Leftrightarrow \beta$ and $q \Leftrightarrow \beta$ which implies $q \Leftrightarrow \perp$, which is impossible.

Theorem 5. *Logics EC and ECN do not have CIP, hence not UIP or ULIP either.*

Acknowledgements. We thank Iris van der Giessen for fruitful discussions on the topic of this paper and three referees for comments that helped improving the paper.

References

1. Akbar Tabatabai, A., Jalali, R.: Universal proof theory: semi-analytic rules and uniform interpolation. arXiv preprint [arXiv:1808.06258](https://arxiv.org/abs/1808.06258) (2018)
2. Bílková, M.: Interpolation in modal logics. Ph.D. thesis, Univerzita Karlova, Filozofická fakulta (2006)
3. Chellas, B.F.: *Modal Logic: An Introduction*. Cambridge University Press, Cambridge (1980)
4. Ghilardi, S., Zawadowski, M.: Undefinability of propositional quantifiers in the modal system S4. *Stud. Log.* **55**(2), 259–271 (1995)
5. Ghilardi, S., Zawadowski, M.: Sheaves, Games, and Model Completions. *A Categorical Approach to Nonclassical Propositional Logics. Trends in Logic*, vol. 14. Springer, Netherlands (2002). <https://doi.org/10.1007/978-94-015-9936-8>
6. Iemhoff, R.: Uniform interpolation and sequent calculi in modal logic. *Arch. Math. Log.* **58**(1), 155–181 (2019)
7. Iemhoff, R.: Uniform interpolation and the existence of sequent calculi. *Ann. Pure Appl. Log.* **170**(11), 102711 (2019)
8. Kurahashi, T.: Uniform Lyndon interpolation property in propositional modal logics. *Arch. Math. Log.* **59**, 659–678 (2020)
9. Maksimova, L.L.: Craig’s theorem in superintuitionistic logics and amalgamable varieties. *Algebra Log.* **16**(6), 643–681 (1977)
10. Orlandelli, E.: Sequent calculi and interpolation for non-normal logics (2019). arXiv preprint [arXiv:1903.11342](https://arxiv.org/abs/1903.11342)
11. Pacuit, E.: *Neighborhood Semantics for Modal Logic*. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-319-67149-9>
12. Pattinson, D.: The logic of exact covers: completeness and uniform interpolation. In: 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 418–427. IEEE (2013)
13. Pitts, A.M.: On an interpretation of second order quantification in first order intuitionistic propositional logic. *J. Symb. Log.* **59**(1), 33–52 (1992)
14. Santocanale, L., Venema, Y., et al.: Uniform interpolation for monotone modal logic. *Adv. Modal Log.* **8**, 350–370 (2010)
15. Seifan, F., Schröder, L., Pattinson, D.: Uniform interpolation in coalgebraic modal logic. In: 7th Conference on Algebra and Coalgebra in Computer Science (CALCO 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
16. Shamkanov, D.S.: Interpolation properties for provability logics GL and GLP. *Proc. Steklov Inst. Math.* **274**(1), 303–316 (2011)
17. Shavrukov, V.Y.: *Subalgebras of diagonalizable algebras of theories containing arithmetic*. Polska Akademia Nauk, Instytut Matematyczny Warsaw (1993)
18. Troelstra, A.S., Schwichtenberg, H.: *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science, vol. 43. Cambridge University Press, Cambridge (2000)
19. Visser, A., et al.: Uniform interpolation and layered bisimulation. In: Hajek, P. (ed.) *Gödel’96: Logical Foundations of Mathematics, Computer Science and Physics—Kurt Gödel’s Legacy*. Lecture Notes in Logic, pp. 139–164. Cambridge University Press (1996)



On the Expressive Power of TeamLTL and First-Order Team Logic over Hyperproperties

Juha Kontinen  and Max Sandström  

Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland
{juha.kontinen,max.sandstrom}@helsinki.fi

Abstract. In this article we study linear temporal logics with team semantics (TeamLTL) that are novel logics for defining hyperproperties. We define Kamp-type translations of these logics into fragments of first-order team logic and second-order logic. We also characterize the expressive power and the complexity of model-checking and satisfiability of team logic and second-order logic by relating them to second- and third-order arithmetic. Our results set in a larger context the recent results of Lück showing that the extension of TeamLTL by the Boolean negation is highly undecidable under the so-called synchronous semantics. We also study stutter-invariant fragments of extensions of TeamLTL.

Keywords: Hyperproperties · Linear temporal logic · Team semantics

1 Introduction

Linear temporal logic (LTL) is a simple logic for formalising concepts of time. It has become important in theoretical computer science, when Amir Pnueli connected it to system verification in 1977, and within that context the logic has been studied extensively [15]. With regards to expressive power, a classic result by Hans Kamp from 1968 shows that LTL is expressively equivalent to $FO^2(<)$.

LTL has found applications in the field of formal verification, where it is used to check whether a system fulfils its specifications. However, the logic cannot capture all of the interesting specifications a system may have, since it cannot express dependencies between its executions, known as traces. These properties, coined hyperproperties by Clarkson and Schneider in 2010, include properties important for cybersecurity such as noninterference and secure information flow [3]. Due to this background, extensions of LTL have recently been the focus of research. In order to specify hyperproperties, temporal logics like LTL and QPTL have been extended with explicit trace and path quantification to define HyperLTL [2] and HyperQPTL [4, 16], respectively.

HyperLTL is one of the most extensively studied of these extensions. Its formulas are interpreted over sets of traces and the syntax extends LTL with quantification of traces. A serious limitation of HyperLTL is that only a fixed

number of traces can be named via the quantifiers when evaluating a formula and hence global hyperproperties cannot be expressed in HyperLTL.

Analogously to Kamp's theorem, HyperLTL can be also related to first-order logic $\text{FO}(<, \mathbf{E})$ by encoding a set of traces T in a natural way by a first-order structure $T \times \mathbb{N}$ that, in addition to linear-orders for the traces, has an equal level predicate \mathbf{E} to synchronize time between different traces [5].

On the other hand, there are alternative approaches to extending LTL to capture hyperproperties. Team semantics is a framework in which one moves on from considering truth through single assignments to regarding teams of assignments as the linchpin for the satisfaction of a formula [10, 17]. Clearly, this framework, when applied to LTL, provides an approach on the hyperproperties. Krebs et al. in 2018 introduced two semantics for LTL under team semantics: the synchronous semantics and the asynchronous variant that differ on the interpretation of the temporal operators [11]. In team semantics the temporal operators advance time on all traces of the current team and with the disjunction \vee , a team can be split into two parts during the evaluation of a formula, hence the nickname splitjunction.

While HyperLTL and other hyperlogics have been studied extensively, many of the basic properties of TeamLTL are still not understood. Already in [11] it was shown that asynchronous TeamLTL collapses to LTL and that the synchronous version and HyperLTL are incomparable in expressivity [11]. Furthermore, the model checking problem of synchronous TeamLTL without splitjunctions \vee is in PSPACE [11]. Recently it was shown by Lück that the complexity of satisfiability and model checking of synchronous TeamLTL with Boolean negation \sim is equivalent to the decision problem of third-order arithmetic [14] and hence highly undecidable. Furthermore, a more fine-grained analysis of the complexity of synchronous TeamLTL has been obtained in [18] where a decidable fragment of TeamLTL was identified and new undecidable extensions that allow restricted uses of the Boolean negation. The paper also showed that TeamLTL and its extensions can be translated to HyperQPTL⁺, which is an extension of HyperLTL by (non-uniform) quantification of propositions [18].

In this article we analyse the expressivity and complexity of TeamLTL via a different route compared to that of [18]. We define several translations between extensions of TeamLTL and first-order team logic interpreted over the structures $T \times \mathbb{N}$. The benefit of translating TeamLTL into first-order team logic is that, e.g., finding a translation for \vee , which is the main source of complexity of TeamLTL, becomes trivial. Furthermore such translations allow us to utilize the better understanding of the properties of logics in first-order team semantics in the study of TeamLTL.

We consider both the synchronous and asynchronous TeamLTL extended by the Boolean negation. In Sect. 3 we show that the asynchronous version can be translated into $\text{FO}^3(=, \dots, \sim)$, whereas for the synchronous variant the so-called equal-level predicate is also needed. In Sect. 4 we define a version of stutter-equivalence suitable for asynchronous semantics and show that X -free TeamLTL(\sim)-formulas are stutter invariant. In Sect. 5 we consider the full first-

order team logic, equivalently second-order logic, as a language for hyperproperties. We show that any second-order definable hyperproperty can be expressed in third-order arithmetic and, over countable teams, in second-order arithmetic. We also show that SO captures the trace-order invariant fragment of second-order arithmetic over countable teams. Finally in Sect. 6 we combine our translations with the results of Lück to show that any logic between the synchronous TeamLTL(\sim) and SO inherits the complexity properties of TeamLTL(\sim).

2 Preliminaries

In this section we briefly discuss the basics of linear temporal logic, its team semantic extension, and first-order team semantics. We begin with describing the classical semantics for LTL.

Let AP be a set of atomic propositions. A *trace* t over AP is an infinite sequence $t \in (2^{\text{AP}})^\omega$. We denote a trace as $t = (t(i))_{i=0}^\infty$, and given $j \geq 0$ we denote the suffix of t starting at the j th element $t[j, \infty) := (t(i))_{i=j}^\infty$.

Now formulas of LTL are defined by the grammar (where $p \in \text{AP}$)

$$\varphi := p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi U\varphi \mid \varphi R\varphi.$$

The truth definition for this language is usually defined as follows.

Definition 1 (Classical Semantics for LTL). *Given a trace t , proposition $p \in \text{AP}$, and LTL formulas φ and ψ , the semantics of linear temporal logic are as follows.*

$$\begin{array}{ll} t \models p \Leftrightarrow p \in t(0) & t \models F\varphi \Leftrightarrow \exists k \geq 0 : t[k, \infty) \models \varphi \\ t \models \neg p \Leftrightarrow p \notin t(0) & t \models G\varphi \Leftrightarrow \forall k \geq 0 : t[k, \infty) \models \varphi \\ t \models \varphi \wedge \psi \Leftrightarrow t \models \varphi \text{ and } t \models \psi & t \models \varphi U\psi \Leftrightarrow \exists k \geq 0 : t[k, \infty) \models \psi \text{ and} \\ & \quad \forall k' < k : t[k', \infty) \models \varphi \\ t \models \varphi \vee \psi \Leftrightarrow t \models \varphi \text{ or } t \models \psi & t \models \varphi R\psi \Leftrightarrow \forall k \geq 0 : t[k, \infty) \models \psi \text{ or} \\ t \models X\varphi \Leftrightarrow t[1, \infty) \models \varphi & \quad \exists k' < k : t[k', \infty) \models \varphi \end{array}$$

Next we consider the team semantics of LTL. As is usual for team semantics, we extended classical semantics by evaluating truth through a set of classical evaluations, which in this case means truth is defined by a set of traces.

A team of TeamLTL is a set T of traces. We denote $T[i, \infty) := \{t[i, \infty) \mid t \in T\}$ and, for $f : T \rightarrow \mathbb{N}$, $T[f, \infty) := \{t[f(t), \infty) \mid t \in T\}$. Below, for functions as above, we write $f' < f$ if for all $t \in T$ it holds that $f'(t) < f(t)$.

Definition 2 (Team Semantics for LTL). *Suppose T is a team, $p \in \text{AP}$, and φ and ψ are LTL formulas. Then the semantics of LTL are defined as follows. The cases marked with * are the same in both the asynchronous and synchronous semantics.*

$$\begin{array}{ll}
 T \models^* p \Leftrightarrow p \in t(0) \text{ for all } t \in T & T \models^a G\varphi \Leftrightarrow \forall f: T \rightarrow \mathbb{N} \\
 T \models^* \neg p \Leftrightarrow p \notin t(0) \text{ for all } t \in T & T[f, \infty] \models \varphi \\
 T \models^* \varphi \wedge \psi \Leftrightarrow T \models \varphi \text{ and } T \models \psi & T \models^s \varphi U \psi \Leftrightarrow \exists k \geq 0: T[k, \infty] \models \psi \\
 T \models^* \varphi \vee \psi \Leftrightarrow \exists T_1, T_2 \subseteq T: & \text{and } \forall k' < k: T[k', \infty] \models \psi \\
 T_1 \cup T_2 = T \text{ and} & T \models^a \varphi U \psi \Leftrightarrow \exists f: T \rightarrow \mathbb{N}: \\
 T_1 \models \varphi \text{ and } T_2 \models \psi & T[f, \infty] \models \psi \text{ and} \\
 T \models^* X\varphi \Leftrightarrow T[1, \infty] \models \varphi & \forall f' < f: T[f', \infty] \models \varphi \\
 T \models^s F\varphi \Leftrightarrow \exists k \geq 0: & T \models^s \varphi R\psi \Leftrightarrow \forall k \geq 0: \\
 T[k, \infty] \models \varphi & T[k, \infty] \models \psi \text{ or } \exists k' < k: \\
 T \models^a F\varphi \Leftrightarrow \exists f: T \rightarrow \mathbb{N}: & T[k', \infty] \models \varphi \\
 T[f, \infty] \models \varphi & T \models^a \varphi R\psi \Leftrightarrow \forall f: T \rightarrow \mathbb{N} \\
 T \models^s G\varphi \Leftrightarrow \forall k \geq 0: T[k, \infty] \models \varphi & T[f, \infty] \models \psi \text{ or} \\
 & \exists f' < f: T[f', \infty] \models \varphi
 \end{array}$$

We denote the asynchronous and the synchronous versions of LTL by TeamLTL^a and TeamLTL^s, respectively. In the following we consider extensions of these logics by the Boolean negation \sim with the usual interpretation:

$$T \models \sim \varphi \Leftrightarrow T \not\models \varphi.$$

We note that in the following it suffices to consider the temporal operators X and U as the rest can be easily defined using \sim and the shorthand $\top = p \vee \neg p$ under both synchronous and asynchronous semantics: $G\varphi \equiv \sim F \sim \varphi$, $F\varphi \equiv \top U \varphi$, $\varphi R\psi \equiv \sim (\sim \varphi U \sim \psi)$.

Definition 3 (FO under team semantics and FO(=(...), ~)). *Formulas of FO are defined by the grammar*

$$\varphi := x = y \mid R(x_1, \dots, x_n) \mid \neg x = y \mid \neg R(x_1, \dots, x_k) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x\varphi \mid \forall x\varphi,$$

where x , y and x_1, \dots, x_n are variables, and R is a relation symbol of arity n . Formulas of FO(=(...), ~) extend the grammar by dependence atoms $=(x_1, \dots, x_n, y)$ and the Boolean negation $\sim \varphi$.

First-order team semantics is defined using sets S of assignments $s: X \rightarrow M$ with a common finite domain X and an arbitrary set M as co-domain. For an assignment s , $s[a/x]$ denotes the modified assignment that acts otherwise as s except that it maps x into a . For a so-called supplementation function $F: S \rightarrow \mathcal{P}(M) \setminus \emptyset$, we define

$$S[F/x] := \{s[a/x] \mid s \in S \text{ and } a \in F(s)\}.$$

The duplication of a team with respect to variable x is defined by $S[M/x] := \{s[m/x] \mid m \in M, s \in S\}$. Supplementation and duplication are used to generalise existential and universal quantification, respectively, into team semantics.

Definition 4 (Team Semantics for FO). *Suppose \mathcal{M} is a first-order model with domain M , and let S be a team of \mathcal{M} . Suppose $n \geq 1$, and φ and ψ are FO formulas. Then the team semantics of FO are defined by the following.*

$$\begin{array}{ll}
 \mathcal{M} \models_S x = y \Leftrightarrow \forall s \in S: s(x) = s(y) & \mathcal{M} \models_S \varphi \wedge \psi \Leftrightarrow \mathcal{M} \models_S \varphi \text{ and} \\
 \mathcal{M} \models_S R(x_1, \dots, x_n) \Leftrightarrow \forall s \in S: & \mathcal{M} \models_S \psi \\
 (s(x_1), \dots, s(x_n)) \in R^{\mathcal{M}} & \mathcal{M} \models_S \varphi \vee \psi \Leftrightarrow \exists S_1, S_2 \subseteq S \\
 & \text{such that } S_1 \cup S_2 = S \text{ and} \\
 \mathcal{M} \models_S \neg x = y \Leftrightarrow \forall s \in S: s(x) \neq s(y) & \mathcal{M} \models_{S_1} \varphi \text{ and } \mathcal{M} \models_{S_2} \psi \\
 \mathcal{M} \models_S \neg R(x_1, \dots, x_n) \Leftrightarrow \forall s \in S: & \mathcal{M} \models_S \exists x \varphi \Leftrightarrow \exists F: S \rightarrow \mathcal{P}(M) \setminus \emptyset \\
 (s(x_1), \dots, s(x_n)) \notin R^{\mathcal{M}} & \text{such that } \mathcal{M} \models_{S[F/x]} \varphi \\
 & \mathcal{M} \models_S \forall x \varphi \Leftrightarrow \mathcal{M} \models_{S[M/x]} \varphi
 \end{array}$$

First-order team logic $\text{FO}(=(\dots), \sim)$ extends FO with \sim and dependence atoms:

$$\mathcal{M} \models_S =(x_1, \dots, x_n, y) \Leftrightarrow \forall s_1, s_2 \in S: \text{ if } s_1(x_i) = s_2(x_i) \text{ for all } i \in \{1, \dots, n\}, \\
 \text{ then } s_1(y) = s_2(y)$$

For further details see for example [17]. Note that for the so-called constancy atoms $=(y)$ the truth definition above amounts to requiring that y has a constant value in the team. Dependence logic $\text{FO}(=(\dots))$ is known to be equi-expressive with existential second-order logic while $\text{FO}(=(\dots), \sim)$ is equi-expressive full second-order logic (SO) [9, 17]. On the other hand, the extensions of FO by mere constancy atoms or \sim alone collapse to FO for sentences [7, 13]. We now define some properties of team logics relevant to the results presented in this paper.

The formulas of TeamLTL^a satisfy the following flatness property: $T \models \varphi$ if and only if $\forall t \in T: t \models \varphi$ while the same does not hold for the formulas of TeamLTL^s [11]. This means that TeamLTL^a without any extensions cannot define non-trivial hyperproperties. Similarly, FO-formulas (by themselves) are also flat in the sense that for all \mathcal{M} and $T: \mathcal{M} \models_T \varphi$ if and only if $\forall s \in T: \mathcal{M} \models_s \varphi$, where \models_s refers to the standard Tarskian semantics of FO [17]. On the other hand, e.g., the formula $=(y)$ does not have the flatness property. A logic L with team semantics has the locality property if for all formulas $\varphi \in L$, \mathcal{M} and T it holds that $\mathcal{M} \models_T \varphi$ if and only if $\mathcal{M} \models_{T \upharpoonright \text{Fr}(\varphi)} \varphi$, where $\text{Fr}(\varphi)$ denotes the free variables of φ . All of the logics in the so-called lax team semantics, including $\text{FO}(=(\dots), \sim)$, satisfy the locality property [6].

3 Embedding TeamLTL and Its Extensions into First-Order Team Logic

In this section we define translations of TeamLTL into first-order team logic. We begin by considering the translation under the asynchronous semantics.

3.1 Asynchronous Semantics

Let $T = \{t_j \mid j \in J\}$ be a set of traces. In order to simulate TeamLTL^a and its extensions in first-order team semantics we encode T by a first-order structure \mathcal{M}_T of vocabulary $\{\leq\} \cup \{P_i \mid p_i \in \text{AP}\}$ such that

$$\begin{aligned} \text{Dom}(\mathcal{M}_T) &= T \times \mathbb{N} \\ \leq^{\mathcal{M}_T} &= \{((t_i, n), (t_j, m)) \mid i = j \text{ and } n \leq m\} \\ P_i^{\mathcal{M}_T} &= \{(t_k, j) \mid p_i \in t_k(j)\}. \end{aligned}$$

The first positions of each of the traces of the temporal team T are encoded as the values of variable x by the set of assignments $S_T^x = \{s_i \mid s_i(x) = (t_i, 0) \text{ for all } t_i \in T\}$, which we will use as the first-order team in our translation. The first-order encoding of a set of traces goes back to [5].

We define a translation of TeamLTL^a formulas into the three-variable fragment $\text{FO}^3(=(\dots), \sim)$ of team logic. We use the variables x, y and z in the first-order side. In fact, since $\text{FO}^3(=(\dots), \sim)$ is closed under the Boolean negation, we may assume \sim also available in TeamLTL^a. Furthermore, it now suffices to consider the operators X and U as the rest can be easily defined using the Boolean negation. For readability we abbreviate the following formula

$$x < y \wedge \forall z(z \leq x \vee y \leq z \vee (\neg x \leq z \wedge \neg z \leq x))$$

defining y as the successor of x by $S(x, y)$.

Define the translation ST_u , for $u \in \{x, y, z\}$ via simultaneous induction as follows (we only list the formulas for ST_x):

$$\begin{aligned} ST_x(p_i) &= P_i(x) & ST_x(\sim \varphi) &= \sim ST_x(\varphi) \\ ST_x(\neg p_i) &= \neg P_i(x) & ST_x(X\varphi) &= \exists y(S(x, y) \wedge ST_y(\varphi)) \\ ST_x(\varphi \wedge \psi) &= ST_x(\varphi) \wedge ST_x(\psi) & ST_x(\varphi U \psi) &= \exists y(x \leq y \wedge =(x, y) \wedge ST_y(\psi) \wedge \\ ST_x(\varphi \vee \psi) &= ST_x(\varphi) \vee ST_x(\psi) & \sim \exists z(x \leq z \wedge z \leq y \wedge =(x, z) \wedge \sim ST_z(\varphi)) \end{aligned}$$

The next theorem can now be proved using induction on the formula φ .

Theorem 1. *Let φ be a TeamLTL^a(\sim)- formula. Then for all non-empty T :*

$$T \models \varphi \Leftrightarrow \mathcal{M}_T \models_{S_T^x} ST_x(\varphi).$$

Proof. See the Appendix.

As a corollary (see Appendix for the proof) we obtain that any TeamLTL^a(\sim)-definable trace property can be defined by a sentence of the logic $\text{FO}^3(=(\dots), \sim)$.

Corollary 1. *Let φ be a TeamLTL^a(\sim)- formula. Then there exists a sentence ψ of $\text{FO}^3(=(\dots), \sim)$ such that for all non-empty T :*

$$T \models \varphi \Leftrightarrow \mathcal{M}_T \models \psi.$$

Synchronous Semantics

Under the synchronous team semantics TeamLTL does not have the flatness property [11] and it is incomparable to HyperLTL. Armed with the translation from the previous section, we need to modify it to capture the synchronicity of the semantics on the first-order side. For this end we assume that the structure \mathcal{M}_T is equipped with the equal-level predicate E that is interpreted as follows:

$$E^{\mathcal{M}_T} = \{((t, n), (t', m)) \mid n = m\}.$$

Next we define a translation from $\text{TeamLTL}^s(\sim)$ into $\text{FO}^3(=(\dots), \sim)$ as follows: The translation is analogous to the previous translation for the atomic propositions, \wedge , \vee , and X . For U we define the translation as follows:

$$\begin{aligned} ST_x^*(\varphi U \psi) &= \exists z \exists y (=(z) \wedge x \leq y \wedge E(z, y) \wedge ST_y^*(\psi)) \wedge \\ &\sim \exists z (\exists x (=(x) \wedge E(z, x)) \wedge x \leq z < y \wedge \sim ST_z^*(\varphi)). \end{aligned}$$

The proof of the following theorem is now analogous to that of Theorem 1 and Corollary 1.

Theorem 2. *Let φ be a $\text{TeamLTL}^s(\sim)$ - formula. Then $T \models \varphi \Leftrightarrow \mathcal{M}_T \models_{S_T^x} ST_x^*(\varphi)$ for all non-empty T and there exists a $\text{FO}^3(=(\dots), \sim)$ -sentence ψ such that for all non-empty T*

$$T \models \varphi \Leftrightarrow \mathcal{M}_T \models \psi.$$

It is interesting to note that the role of the dependence atom in the translation of U in the asynchronous semantics can be replaced by the equal level predicate and the constancy atom in the synchronous case.

4 Asynchronicity and Stutter Equivalence

In [14] Lück generalised the notions of stutter-equivalence and stutter-invariance to TeamLTL^s [14]. In this section we define an analogous concept for the asynchronous team semantics, and show how these two conceptualizations relate to each other.

We begin by defining the classical stuttering function.

Definition 5 (Stuttering function). *A stuttering function of a trace t is a strictly increasing function $f: \mathbb{N} \rightarrow \mathbb{N}$, such that $f(0) = 0$ and $t(f(k)) = \dots = t(f(k + 1) - 1)$ for all $k \in \mathbb{N}$.*

For functions $f: \mathbb{N} \rightarrow \mathbb{N}$ we denote the trace $t(f(0))t(f(1))t(f(2))\dots$ by $t[f]$ and similarly for teams $T[f] = \{t[f] \mid t \in T\}$. Lück defined the synchronous stuttering function of a team T to be a function that is a stuttering function for all traces $t \in T$ simultaneously.

The asynchronous variants of these definitions are similar, with the generalization that the stuttering functions are independent for each trace. We restrict our attention to finite teams T below.

Definition 6 (Asynchronous stuttering function). *An asynchronous stuttering function of a team $T = \{t_1, \dots, t_k\}$ is a function $F: \mathbb{N} \rightarrow \mathbb{N}^k$, such that $F(n) = (f_{t_1}(n), \dots, f_{t_k}(n))$ for stuttering functions f_{t_1}, \dots, f_{t_k} of traces $t_1, \dots, t_k \in T$.*

For stuttering functions $F: \mathbb{N} \rightarrow \mathbb{N}^k$ we denote by $t[F]$ the trace

$$t(f_t(0))t(f_t(1))t(f_t(2))\dots,$$

and furthermore for teams $T[F] = \{t[F] \mid t \in T\}$.

Definition 7 (Asynchronous stutter-equivalence). *Teams T, T' are asynchronously stutter-equivalent, in symbols $T \equiv_{st}^a T'$, if there are asynchronous stuttering functions F of T and F' of T' such that $T[F] = T'[F']$.*

Now it is clear that every synchronous stuttering function is also an asynchronous stuttering function, but the converse does not necessarily hold.

A formula φ is stutter-invariant if $T \models \varphi$ if and only if $T' \models \varphi$, for all stutter equivalent teams T and T' . Note that an asynchronous stuttering function F of team T induces a stuttering function $F|S$ for any subteam $S \subseteq T$. Next we prove analogous claims to the ones shown by Lück.

We call a function $i: T \rightarrow \mathbb{N}$ a configuration, and we define an component-wise order among configurations, i.e. for configurations i and j $i < j$ if and only if $i(t) < j(t)$ for all $t \in T$. The definition of \leq is analogous. We consider the component-wise order because not all configurations of a team are attainable from each other with regards to the until-operator. For instance, let $T = \{t_1, t_2\}$ be a team. Now the configuration $(t_1(0), t_2(1))$ is not attainable from the configuration $(t_1(1), t_2(0))$, or vice versa. Hence we say that a configuration is attainable from another if for each trace of the former configuration the time-point is equal or later to the time-point of the same trace in the latter configuration.

Lemma 1. *Let T and T' be teams. Then $T \equiv_{st}^a T'$ if and only if, if $T = T_1 \cup T_2$, then there are subteams T'_1, T'_2 such that $T' = T'_1 \cup T'_2$ and $T_i \equiv_{st}^a T'_i$ for $i \in \{1, 2\}$.*

Proof. See the Appendix.

Lemma 2. *Let T and T' be teams such that $T \equiv_{st}^a T'$, as witnessed by the stuttering functions F and G , and let j be a configuration of T . Then there is a configuration $i \leq j$ of T such that for all $t \in T$ there is a $a_t \in \mathbb{N}$ such that $f_t(a_t) = i(t)$ and $T[i, \infty) \equiv_{st}^a T[j, \infty)$. Furthermore, there is a configuration k of T' such that $T'[i, \infty) \equiv_{st}^a T'[k, \infty)$.*

Proof. See the Appendix.

Theorem 3. *Every X -free TeamLTL(\sim)-formula is stutter invariant.*

Proof. We consider formulas $\varphi \in \text{TeamLTL}(\sim, U)$. Let T and T' be teams such that $T \equiv_{st}^a T'$, as witnessed by the functions F and G respectively. In this proof we say that a configuration c is in accordance with a stuttering function F to mean that for all t there is an n such that $c(t) = f_t(n)$. We prove the claim through induction on the structure of φ .

- For all propositional formulas the claim holds, since $F(0) = (0, \dots, 0) = G(0)$.
- For the Boolean connectives \wedge and \sim the claim follows immediately from the induction hypothesis.
- The case for the splitjunction \vee is an immediate consequence of the induction hypothesis and Lemma 1.
- For the case of the until operator U , we only show \Rightarrow , since the other direction is symmetric. Assume $T \models \psi U \theta$, i.e. there is a configuration i of T such that $T[i, \infty) \models \theta$ and $T[j, \infty) \models \psi$ for all configurations $j < i$. We aim to show that $T' \models \psi U \theta$. By Lemma 2 we know that there is a configuration c in accordance with the stuttering function F such that $T[c, \infty) \models \theta$ and $c \leq i$, and there is a configuration k of T' such that $T[i, \infty) \equiv_{st}^a T'[k, \infty)$, which is in accordance with the stuttering function G . Thus by the induction hypothesis $T'[k, \infty) \models \theta$. We still need to show that $T'[l, \infty) \models \psi$ for all $l < k$. If $k(t) = 0$ for all $t \in T'$, then we are done. If not, choose an arbitrary configuration $l < k$. Now by Lemma 2 there is a configuration c' in accordance with the stuttering function G such that $c' \leq l$ and $T'[c', \infty) \equiv_{st}^a T'[l, \infty)$, and a configuration p of T such that $T'[c', \infty) \equiv_{st}^a T[p, \infty)$. Now $c' \leq l < k$, and thus $p < c \leq i$. Therefore $T[p, \infty) \models \psi$, and by the induction hypothesis, also $T'[l, \infty) \models \psi$.

Example 1. We show that asynchronous stutter-equivalence does not imply synchronous stutter-equivalence. Consider the teams $T = \{\{\emptyset\}\{p\}^\omega, \{\emptyset\}\{\emptyset\}\{p\}^\omega\}$ and $T' = \{\{\emptyset\}\{p\}^\omega\}$. These two teams are asynchronously stutter-equivalent, as witnessed by the asynchronous stuttering functions $F(n) = (f_1, f_2)$ and $G(n) = n$, where $f_1(n) = n$ and

$$f_2(n) = \begin{cases} n & \text{if } n = 0 \\ n + 1 & \text{otherwise.} \end{cases}$$

On the other hand, synchronously stutter-equivalent teams necessarily have the same cardinality hence T and T' are not synchronously stutter-equivalent [14].

Now for all X -free $\text{TeamLTL}^a(\sim)$ formulas φ , $T \models \varphi$ if and only if $T' \models \varphi$, however $T \not\models^s Fp$ and $T' \models^s Fp$, which indicates that the TeamLTL^s formula Fp cannot be expressed by any X -free formula of $\text{TeamLTL}^a(\sim)$.

5 SO Versus Arithmetic Definability

The translations given in the previous section show that both $\text{TeamLTL}^s(\sim)$ and $\text{TeamLTL}^a(\sim)$ can be embedded into three-variable fragments of first-order team logic (which is known to be equi-expressive with second-order logic). In this section we compare second-order logic and arithmetic as formalisms for defining hyperproperties.

We begin by showing that any SO-definable trace property can be also defined in third-order arithmetic. The analogous result relating $\text{TeamLTL}^s(\sim)$ properties to third-order arithmetic was shown in [14]. In arithmetic, a set T can be represented by a third-order relation. Before going to the results, we briefly discuss third-order arithmetic (see, e.g., [12]). The standard model of arithmetic

is denoted by $(\mathbb{N}, +, \times, \leq 0, 1)$. A third-order type is a tuple $\tau = (n_1, \dots, n_k)$, for natural numbers $k, n_1, \dots, n_k \geq 1$. For each type τ , we adopt a countable set of τ -variables $\mathcal{V}_\tau := \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots\}$, which are interpreted by third-order objects whose type is determined by τ . Syntactically, third-order logic extends the (more familiar) language of second-order logic by

- new atomic formulas of the form $\mathbf{a}(A_1, \dots, A_k)$, where for \mathbf{a} of type $\tau = (n_1, \dots, n_k)$, A_i is a relation symbol of arity n_i for $1 \leq i \leq k$,
- existential and universal quantification over third-order variables \mathbf{a} .

For a type $\tau = (n_1, \dots, n_k)$, the third-order objects \mathbf{a} vary over elements \mathcal{A} of the set

$$\mathcal{P}(\mathcal{P}(\mathbb{N}^{n_1}) \times \dots \times \mathcal{P}(\mathbb{N}^{n_k}))$$

The set of all formulas of third-order arithmetic, i.e., third-order formulas over the vocabulary $\{+, \times, 0, 1, =, \leq\}$ is denoted by Δ_0^3 (and second-order arithmetic by Δ_0^2). The subset of Δ_0^3 -sentences true in $(\mathbb{N}, +, \times, \leq 0, 1)$ is denoted by a boldface letter $\mathbf{\Delta}_0^3$ (analogously $\mathbf{\Delta}_0^2$).

We are now ready to define the encoding of a team T using suitable arithmetical relations. We identify the proposition p_i with number i and encode a trace t by a binary relation S that $(j, k) \in S$ iff $p_k \in t(j)$. Now clearly a team T can be encoded by a third-order object $\mathcal{A}_T \subseteq \mathcal{P}(\mathbb{N}^2)$ of type $((2))$.

Theorem 4. *Let $\varphi \in SO$ be a sentence. Then there exists a formula $\text{Tr}(\varphi)(\mathbf{a})$ of Δ_0^3 such that for all trace sets T :*

$$\mathcal{M}_T \models \varphi \iff (\mathbb{N}, +, \times, \leq 0, 1) \models \text{Tr}(\varphi)(\mathcal{A}_T/\mathbf{a}),$$

where \mathcal{M}_T is defined as in Sect. 3.1.

Proof. See the Appendix.

Next we consider the special case of countable teams T . In this case we can precisely characterize SO -definable hyperproperties arithmetically. Assume that T is countable, i.e., $T = \{t_i \mid i \in \mathbb{N}\}$ or $T = \{t_i \mid 0 \leq i \leq n\}$ for some n . Now T can be encoded by a single ternary relation such that

$$(i, j, k) \in A_T \iff p_k \in t_i(j). \quad (1)$$

In order to encode also the cardinality of T , we let $A_T \subseteq \mathbb{N}^4$ to consist of the tuples

$$(0, i, j, k) \in A_T \iff p_k \in t_i(j)$$

together with $(1, n, n, n)$ if $n = |T|$. It is now straightforward to modify the translation Tr in such a way that $\text{Tr}'(\varphi)$ becomes a Δ_0^2 -formula as now an element of the domain $T \times \mathbb{N}$ can be encoded by a pair (i, j) (recall (1)) and a k -ary relation $X \subseteq (T \times \mathbb{N})^k$ directly by a $2k$ -ary relation $R_X \subseteq \mathbb{N}^{2k}$. Define a translation Tr' inductively as follows. Below i is a definable constant.

$$\begin{aligned} \text{Tr}'(x = y) &:= x_1 = y_1 \wedge x_2 = y_2 & \text{Tr}'(X(x_1, \dots, x_k)) &:= R_X(x_1^1, x_2^1, \dots, x_1^k, x_2^k) \\ \text{Tr}'(x \leq y) &:= x_1 = y_1 \wedge x_2 \leq y_2 & \text{Tr}'(\exists x \varphi) &:= \exists x_1 \exists x_2 (\theta_1 \wedge \text{Tr}'(\varphi)) \\ \text{Tr}'(P_i(x)) &:= R_T(0, x_1, x_2, i) & \text{Tr}'(\exists X \varphi) &:= \exists R_X (\theta_2 \wedge \text{Tr}'(\varphi)) \end{aligned}$$

Above the formula θ_1 restricts the values of x_1, x_2 (θ_2 analogously restricts R_X) to range either over \mathbb{N}^2 or $\{0, 1, \dots, n\} \times \mathbb{N}$ depending on whether T is infinite or finite (which can be detected by the existence of a tuple $(1, n, n, n) \in T$). The following theorem can be now proved analogously to Theorem 4.

Theorem 5. *Let $\varphi \in SO$ be a sentence. Then there exists a formula $\text{Tr}'(\varphi)(R_T)$ of Δ_0^2 such that for all countable trace sets T :*

$$\mathcal{M}_T \models \varphi \iff (\mathbb{N}, +, \times, \leq, 0, 1) \models \text{Tr}'(\varphi)(A_T/R_T).$$

The above translation is quite simple due to the fact that only the representation of T changes but the logic remains the same. It is worth noting though that in arithmetic we can, e.g., express a property

$$T = \{t_i \mid t_i(i) = \{p_i\} \text{ and } t_i(j) = \emptyset \text{ for } j \neq i \text{ and } i, j \in \mathbb{N}\}$$

that addresses infinitely many propositions p_i whereas over \mathcal{M}_T only finitely many of the propositions can be mentioned in a formula via the relations P_i . Another difference between the representations is that in arithmetic the team T is always naturally ordered whereas \mathcal{M}_T carries no ordering for the traces. It turns out that these properties are the only obstacles for proving a converse of Theorem 5. Below I is either \mathbb{N} or $\{0, 1, \dots, n\}$ for some n .

Definition 8. *Let $\varphi(R) \in \Delta_0^2$ be a formula. The formula $\varphi(R)$ is called trace-order invariant if for all countable teams $T = (t_i)_{i \in I}$ and all permutations $f: I \rightarrow I$:*

$$(\mathbb{N}, +, \times, \leq, 0, 1) \models \varphi(A_T/R) \leftrightarrow \varphi(A_{T^f}/R)$$

where $T^f = (t_{f(i)})_{i \in I}$.

Theorem 6. *Let $\varphi(R) \in \Delta_0^2$ be trace-order invariant and let AP be finite. Then there exists a SO-sentence ψ such that for all countable trace sets T over AP:*

$$\mathcal{M}_T \models \psi \iff (\mathbb{N}, +, \times, \leq, 0, 1) \models \varphi(A_T/R).$$

Proof. Let $T = (t_i)_{i \in I}$ where $I = \{0, \dots, n\}$ for some n or $I = \mathbb{N}$. Let us assume first that the structure \mathcal{M}_T is also equipped with some linear ordering \leq_t of the elements $\{(t_i, 0) \mid i \in I\}$. Now the set $\{(t_0, i) \mid i \in \mathbb{N}\}$ (which is linearly ordered by \leq) can be treated as an isomorphic copy of \mathbb{N} over which we can interpret the structure $(\mathbb{N}, +, \times, \leq, 0, 1)$ (the predicates $+$ and \times can be defined using second-order existential quantification.) Now the following formulas can be constructed utilizing the “dual” of this interpretation (see e.g., [8]):

- A formula $\theta(x, y, u, v)$ that defines (an isomorphic copy of) the relation A_T from the information in the structure \mathcal{M}_T . For this an order preserving bijection f between (an initial segment of) $\{(t_0, i) \mid i \in \mathbb{N}\}$ and the set $\{(t_i, 0) \mid i \in I\}$ and an equal-level predicate E can be first existentially quantified. The formula then asserts that

$$\bigvee_{p_k \in \text{AP}} (x = 0 \wedge v = k \wedge \exists z (P_k(z) \wedge f(y) \leq z \wedge E(u, z))).$$

Note that the finiteness of AP is crucial for this to work and that $f(y) \leq z$ guarantees that the index of the trace of z equals y .

- A sentence ψ that expresses that

$$(\mathbb{N}, +, \times, \leq 0, 1) \models \varphi(A_T/R)$$

using the formula for A_T .

Finally we can get rid of the ordering \leq_t using second-order existential quantification as the set $\{(t, 0) \mid t \in T\}$ is a definable subset of $T \times \mathbb{N}$. Now the sentence $\exists \leq_t \psi$ will satisfy the claim of the theorem.

6 Complexity of Model Checking and Satisfiability

In this section we apply our results to characterize the complexity of model checking and satisfiability for first-order team logic and some of its variable fragments for hyperproperties (i.e., over structures \mathcal{M}_T).

For the model checking problem it is asked whether a team of traces generated by a given finite Kripke structure satisfies a given formula. We consider Kripke structures of the form $K = (W, R, \eta, w_0)$, where W is a finite set of states, $R \subseteq W^2$ a left-total transition relation, $\eta: W \rightarrow 2^{\text{AP}}$ a labelling function, and $w_0 \in W$ an initial state of W . A path σ through K is an infinite sequence $\sigma \in W^\omega$ such that $\sigma[0] = w_0$ and $(\sigma[i], \sigma[i + 1]) \in R$ for every $i \geq 0$. The trace of σ is defined as $t(\sigma) := \eta(\sigma[0])\eta(\sigma[1]) \cdots \in (2^{\text{AP}})^\omega$. A Kripke structure K then induces a trace set $\text{Traces}(K) = \{t(\sigma) \mid \sigma \text{ is a path through } K\}$.

- Definition 9.** 1. *The model checking problem of a logic \mathcal{L} is the following decision problem: Given a formula $\varphi \in \mathcal{L}$ and a Kripke structure K over AP, determine whether $\text{Traces}(K) \models \varphi$,*
2. *The (countable) satisfiability problem of a logic \mathcal{L} is the following decision problem: Given a formula $\varphi \in \mathcal{L}$, determine whether $T \models \varphi$ for some (countable) $T \neq \emptyset$.*

In [14] Lück gave a complete picture of the complexity properties of synchronous TeamLTL(\sim). By combining Lück’s results with ours, we are able to show the following. Below $\text{FO}^3(=(\dots), \sim)$ is assumed to be equipped with the equal-level predicate E .

- Theorem 7.** 1. *The model checking and satisfiability problems of first-order team logic $\text{FO}(=(\dots), \sim)$, its three-variable fragment $\text{FO}^3(=(\dots), \sim)$, and SO are equivalent to Δ_0^3 under logspace-reductions.*
2. *The countable satisfiability problem of $\text{FO}(=(\dots), \sim)$, its three-variable fragment $\text{FO}^3(=(\dots), \sim)$, and SO is equivalent to Δ_0^2 under logspace-reductions.*
3. *The model checking and (countable) satisfiability problems of $\text{TeamLTL}^a(\sim)$ are logspace-reducible to Δ_0^3 (Δ_0^2).*

Proof. In each of the logics the results follow by utilizing the results of [14] and the translations given in the previous sections. Let us consider the claim for satisfiability in 1. Note first that by Theorem 2 the satisfiability problem of $\text{TeamLTL}^s(\sim)$ can be easily reduced to that of $\text{FO}^3(=(\dots), \sim)$ implying a logspace-reduction from Δ_0^3 to the satisfiability problem of $\text{FO}^3(=(\dots), \sim)$. On the other hand, any sentence φ of $\text{FO}^3(=(\dots), \sim)$ can be first translated to an equivalent SO-sentence φ^* [9], and then, using Theorem 4, we see that φ is satisfiable iff

$$(\mathbb{N}, +, \times, \leq 0, 1) \models \exists \mathbf{a}(\mathbf{a} \neq \emptyset \wedge \text{Tr}(\varphi^*)).$$

For model checking we note that a logspace-reduction from Δ_0^3 to the model checking problem of $\text{FO}^3(=(\dots), \sim)$ can be obtained just like with satisfiability above. On the other hand, by Theorem 4.3 in [14] it is possible to construct (in logspace) a Δ_0^3 -formula $\psi_K(\mathbf{a})$ defining the set $\text{Traces}(K)$ for any finite $K = (W, R, \eta, w_0)$. Hence now for any given sentence $\varphi \in \text{SO}$ and K it holds that

$$\mathcal{M}_{\text{Traces}(K)} \models \varphi \iff (\mathbb{N}, +, \times, \leq 0, 1) \models \exists \mathbf{a}(\psi_K(\mathbf{a}) \wedge \text{Tr}(\varphi)).$$

Hence the model checking problem of SO, $\text{FO}(=(\dots), \sim)$, and $\text{FO}^3(=(\dots), \sim)$ reduces to Δ_0^3 .

Finally Claim 3 follows by 1 and 2 and the fact that $\text{TeamLTL}^a(\sim)$ can be translated into SO by Theorem 1.

It is worth noting that the previous theorem can be extended to any logic \mathcal{L} effectively residing between:

$$\text{TeamLTL}^s(\sim) \leq \mathcal{L} \leq \text{SO}.$$

This is a potent result, as the upper bound for many hyperlogics have not as of yet been studied.

7 Conclusion and Future Work

We have studied $\text{TeamLTL}(\sim)$ under both synchronous and asynchronous semantics, showing through compositional translations embeddings into first-order team logic and second-order logic. Furthermore, using these translations we were able to transfer the known complexity properties of $\text{TeamLTL}^s(\sim)$ to various logics that reside between $\text{TeamLTL}^s(\sim)$ and second-order logic. Many questions remain such as:

- How does $\text{TeamLTL}^a(\sim)$ relate to the recently defined asynchronous variant of HyperLTL [1]?
- Is it possible to find a non-trivial extension of TeamLTL^a that translates to $\text{FO}(\sim)$ or to the extension of FO by constancy atoms? This would be interesting as both of the logics collapse to FO for sentences.

- What is the relationship of the logics $\text{TeamLTL}^s(\sim)$ and $\text{TeamLTL}^a(\sim)$; by our result the synchronous Fp cannot be expressed in $\text{TeamLTL}^a(\sim)$ by any X -free formula and, on the other hand, asynchronous Fp cannot be expressed by any TeamLTL^s -formula even if the Boolean *disjunction* is allowed in the formulas (see Proposition 8 in [18]).

Acknowledgements. This research was supported by the Finnish Academy (grants 308712 and 322795).

Appendix

Proofs

Proof (Sketch of the proof of Theorem 1). Let $T' \subseteq T$ and $i: T' \rightarrow \mathbb{N}$. We use T' and i as a means to refer to any team that might be relevant for the evaluation of $\text{TeamLTL}^a(\sim)$ formulas when starting the evaluation with T . On the first-order side the corresponding team will be

$$S_{T',i}^x := \{s \mid s(x) = (t, i(t)) \text{ and } t \in T'\}.$$

We can now show using simultaneous induction on φ that for all $T' \subseteq T$, $i: T' \rightarrow \mathbb{N}$, and $u \in \{x, y, z\}$

$$\{t[i(t), \infty] \mid t \in T'\} \models \varphi \Leftrightarrow \mathcal{M}_T \models_{S_{T',i}^u} ST_u(\varphi).$$

- Assume $\varphi = p_i$ and $T' \subseteq T$, $i: T' \rightarrow \mathbb{N}$ are arbitrary. Now

$$\begin{aligned} \{t[i(t), \infty] \mid t \in T'\} \models \varphi &\Leftrightarrow p_i \in t(i(t)) \text{ for all } t \in T' \\ &\Leftrightarrow (t, i(t)) \in P_i^{\mathcal{M}_T} \text{ for all } t \in T' \\ &\Leftrightarrow \mathcal{M}_T \models_{S_{T',i}^x} P_i(x) \\ &\Leftrightarrow \mathcal{M}_T \models_{S_{T',i}^x} ST_x(\varphi) \end{aligned}$$

Note that the second equivalence holds by the definition of the structure \mathcal{M}_T and the third equivalence by first-order team semantics of atomic formulas.

- Assume $\varphi = X\psi$ and $T' \subseteq T$ and $i: T' \rightarrow \mathbb{N}$ are arbitrary. Let i^+ be defined by $i^+(t) := i(t) + 1$ for all t . Now

$$\begin{aligned} \{t[i(t), \infty] \mid t \in T'\} \models \varphi &\Leftrightarrow \{t[i^+(t), \infty] \mid t \in T'\} \models \psi \\ &\Leftrightarrow \mathcal{M}_T \models_{S_{T',i^+}^y} ST_y(\psi) \\ &\Leftrightarrow \mathcal{M}_T \models_{S_{T',i}^x} \exists y(S(x, y) \wedge ST_y(\psi)) \\ &\Leftrightarrow \mathcal{M}_T \models_{S_{T',i}^x} ST_x(X\varphi) \end{aligned}$$

The second equivalence above holds by the induction assumption for $ST_y(\psi)$. For the the third equivalence we use the facts that the supplementation function F for y is uniquely determined by the formula and x is not free in $ST_y(\psi)$. Note that by locality it holds that

$$\mathcal{M}_T \models_{S_{T',i}^{x[F/y]}} ST_y(\psi) \Leftrightarrow \mathcal{M}_T \models_{S_{T',i^+}^y} ST_y(\psi),$$

since $S_{T',i+}^y$ is the reduct of $S_{T',i}^x[F/y]$ to the team with domain $\{y\}$.

The proof for the connectives is straightforward and for the temporal operator U it is similar to the case of X .

Proof (Proof of Corollary 1). We show that $T \models \varphi$ if and only if $\mathcal{M}_T \models \psi$, where ψ is the sentence:

$$\forall x(\exists y(y < x) \vee (\forall y(\neg y < x) \wedge ST_x(\varphi))).$$

Note that

$$\begin{aligned} \mathcal{M}_T \models \psi &\Leftrightarrow \mathcal{M}_T \models_{\{\emptyset\}[\text{dom}(\mathcal{M}_T)/x]} \exists y(y < x) \vee (\forall y\neg(y < x) \wedge ST_x(\varphi)). \\ &\Leftrightarrow \mathcal{M}_T \models_{S_T^x} ST_x(\varphi). \end{aligned}$$

In the second line the team $\{\emptyset\}[\text{dom}(\mathcal{M}_T)/x]$ (i.e., $\text{dom}(\mathcal{M}_T)$) has to be split into two disjoint parts: the subset of elements having a predecessor and to those not having a predecessor ($= S_T^x$). The first disjunct is then trivially satisfied (by flatness it behaves classically) hence we arrive at the case which is equivalent to $T \models \varphi$ by Theorem 1.

Proof (Proof of Lemma 1). Suppose 2. Now $T = T \cup \emptyset$, however the empty set is only stutter equivalent to itself. Thus $T \equiv_{st}^a T'$.

Suppose 1 and suppose that $T = T_1 \cup T_2$, hence we have asynchronous stuttering functions F of T and G of T' , such that $T[F] = U = T'[G]$. We consider the subteams induced by the stuttering function, i.e. $T_i[F|T_i]$. Since $T[F] = T'[G]$, there exist subteams T'_1, T'_2 such that $T'_i[G|T'_i] = T_i[F|T_i]$. Thus $T_i \equiv_{st}^a T'_i$.

It remains to show that the subteams T'_1 and T'_2 constitute the entirety of the team T' . It is clear that $T'_1 \cup T'_2 \subset T'$, so it remains to show the converse. Let $t' \in T'$. Now, since $T \equiv_{st}^a T'$, there exists a $t \in T$ such that $t[F|\{t\}] = t'[G|\{t'\}]$. By our assumption, the trace t belongs to either T_1 or T_2 . Without loss of generality we may assume that $t \in T_1$, but then $t'[G|\{t'\}] \in T'_1[G|T'_1]$. Since the team T' is a set, i.e. it does not contain duplicates, we may conclude that $t' \in T'$.

Proof (Proof of Lemma 2). By the definition of the asynchronous stuttering function, for each coordinate of $j(t)$ there exists a constant a_t such that $f_t(a_t) \leq j(t)$ and $t(f_t(a_t)) = t(j(t))$. Let i be the configuration defined by $i(t) = f_t(a_t)$. Now we can use the stuttering function F to construct stuttering functions F' and F'' for $T[i, \infty)$ and $T[j, \infty)$ respectively. First of we define F' via $f'_t(n) = f_t(n + a_t)$ for all $t \in T$, which clearly is a stuttering function of $T[i, \infty)$. Next we define

$$F''(n) = \begin{cases} (j(t))_{t \in T} & \text{if } n = 0 \\ F'(n) & \text{otherwise.} \end{cases}$$

Since $t(f_t(a_t)) = t(i(t)) = t(j(t))$ for all $t \in T$, it follows that $T[i, \infty)[F'] = T[j, \infty)[F'']$. Thus $T[i, \infty) \equiv_{st}^a T[j, \infty)$.

For the second claim we use the assumption that $T \equiv_{st}^a T'$. We let the configuration i be as above. Now for all $n, t \in T$ and $t' \in T'$ it holds that $t(f_t(n)) = t'(g_{t'}(n))$. Thus there exists some configuration $k: T' \rightarrow \mathbb{N}$ such that $t'(g_{t'}(k(t'))) = t(f_t(i(t)))$, which allows us to define the stuttering function G' of $T'[k, \infty)$ as $g'_{t'}(n) = g_{t'}(n + k(t'))$ for all $t' \in T'$. Clearly now $T'[k, \infty)[G'] = T[i, \infty)[F']$, and hence $T[i, \infty) \equiv_{st}^a T'[k, \infty)$.

Proof (Proof of Theorem 4). We define an inductive translation Tr from second-order logic to third order arithmetic as follows. The key ideas in the translation are:

- an element of the domain $T \times \mathbb{N}$ of the structure \mathcal{M}_T can be uniquely identified by specifying a trace t and $i \in \mathbb{N}$. Hence, syntactically, a first-order variable x can be encoded by a pair of variables (R_x, z_x) where R_x is a binary relation and z_x is a first-order variable;
- a subset of $T \times \mathbb{N}$ is a set of pairs (t, i) and hence in the translation a unary relation X is encoded by a third-order variable \mathbf{b}_X of type $((2), (1))$, where the unary relation encodes i by the singleton $\{i\}$.

Define now a formula translation Tr as follows. We omit below the obvious cases of the Boolean connectives and, for clarity, we consider only unary relations X on the side of SO . It is straightforward to write the corresponding translations also for relations of arbitrary arities.

$$\begin{array}{ll}
 \text{Tr}(x = y) := \forall u \forall v (R_x(u, v) \leftrightarrow & \text{Tr}(X(x)) := \exists Y (\mathbf{b}_X(R_x, Y) \wedge Y = \{z_x\}) \\
 R_y(u, v) \wedge z_x = z_y & \\
 \text{Tr}(x \leq y) := \forall u \forall v (R_x(u, v) \leftrightarrow & \text{Tr}(\exists x \varphi) := \exists R_x \exists z_x (\mathbf{a}(R_x) \wedge \text{Tr}(\varphi)) \\
 R_y(u, v) \wedge z_x \leq z_y & \text{Tr}(\exists X \varphi) := \exists \mathbf{b}_X (\forall R \forall Y (\mathbf{b}_X(R, Y) \rightarrow \mathbf{a}(R) \wedge \\
 \text{Tr}(P_i(x)) := \mathbf{a}(R_x) \wedge R_x(z_x, i) & |Y| = 1) \wedge \text{Tr}(\varphi))
 \end{array}$$

In the above formulas, i denotes a (definable) constant. It is now straightforward to show using induction on φ that for all s and s^* :

$$\mathcal{M}_T \models_s \varphi \iff (\mathbb{N}, +, \times, \leq, 0, 1) \models_{s^*} \text{Tr}(\varphi)(\mathcal{A}_T/\mathbf{a}),$$

where the interpretations s and s^* relate to each other as described above.

References

1. Baumeister, J., Coenen, N., Bonakdarpour, B., Finkbeiner, B., Sánchez, C.: A temporal logic for asynchronous hyperproperties. CoRR abs/2104.14025 (2021)
2. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: Abadi, M., Kremer, S. (eds.) POST 2014. LNCS, vol. 8414, pp. 265–284. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54792-8_15
3. Clarkson, M.R., Schneider, F.B.: Hyperproperties. J. Comput. Secur. **18**(6), 1157–1210 (2010)

4. Coenen, N., Finkbeiner, B., Hahn, C., Hofmann, J.: The hierarchy of hyperlogics. In: LICS 2019, pp. 1–13. IEEE (2019)
5. Finkbeiner, B., Zimmermann, M.: The first-order logic of hyperproperties. In: Vollmer, H., Vallée, B. (eds.) STACS 2017, LIPIcs, vol. 66, pp. 30:1–30:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
6. Galliani, P.: Inclusion and exclusion dependencies in team semantics: on some logics of imperfect information. *Ann. Pure Appl. Log.* **163**(1), 68–84 (2012)
7. Galliani, P.: Epistemic operators in dependence logic. *Stud. Log.* **101**(2), 367–397 (2013). <https://doi.org/10.1007/s11225-013-9478-3>
8. Immerman, N.: *Descriptive Complexity*. Graduate Texts in Computer Science, Springer, Heidelberg (1999)
9. Kontinen, J., Nurmi, V.: Team logic and second-order logic. *Fundam. Inform.* **106**(2–4), 259–272 (2011)
10. Krebs, A., Meier, A., Virtema, J.: A team based variant of CTL. In: TIME 2015, pp. 140–149 (2015). <https://doi.org/10.1109/TIME.2015.11>
11. Krebs, A., Meier, A., Virtema, J., Zimmermann, M.: Team semantics for the specification and verification of hyperproperties. In: Potapov, I., Spirakis, P., Worrell, J. (eds.) MFCS 2018, vol. 117, pp. 10:1–10:16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018)
12. Leivant, D.: Higher order logic. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A., Siekmann, J.H. (eds.) *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 2, pp. 229–322. Oxford University Press, Oxford (1994)
13. Lück, M.: Axiomatizations of team logics. *Ann. Pure Appl. Log.* **169**(9), 928–969 (2018). <https://doi.org/10.1016/j.apal.2018.04.010>
14. Lück, M.: On the complexity of linear temporal logic with team semantics. *Theor. Comput. Sci.* **837**, 1–25 (2020)
15. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, pp. 46–57. IEEE Computer Society (1977)
16. Rabe, M.N.: A temporal logic approach to information-flow control. Ph.D. thesis, Saarland University (2016)
17. Väänänen, J.: *Dependence Logic*. Cambridge University Press, Cambridge (2007)
18. Virtema, J., Hofmann, J., Finkbeiner, B., Kontinen, J., Yang, F.: Linear-time temporal logic with team semantics: Expressivity and complexity. *CoRR abs/2010.03311* (2020)



Characterizations for XPath $_{\mathcal{R}}(\downarrow)$

Nicolás González¹ and Sergio Abriola^{1,2}(✉)

¹ University of Buenos Aires, Buenos Aires, Argentina
sabriola@dc.uba.ar

² ICC-CONICET, Buenos Aires, Argentina

Abstract. Over the semantic universe of trees augmented with arbitrary sets of relations between nodes, we study model-theoretic properties of the extension XPath $_{\mathcal{R}}(\downarrow)$ of the downward fragment of XPath, equipped with a finite set \mathcal{R} of relation symbols. We introduce an adequate notion of bisimulation, dependant on the set of relations \mathcal{R} in consideration, and show a characterization result in the style of Hennessy-Milner's, relating bisimulation and logical equivalence and showing that both coincide over finitely branching \mathcal{R} -trees. Furthermore, we also give a van Benthem-like theorem characterizing each XPath $_{\mathcal{R}}(\downarrow)$ as the fragment of first-order logic (over an adequate signature) with one free variable that is \mathcal{R} -bisimulation-invariant. Finally, we show that our results are also valid when applied to universes of trees with some fixed semantics for the symbols of \mathcal{R} . This contains in particular the case of XPath $_{=}$ (\downarrow) over data trees.

Keywords: XPath · Bisimulation · Characterization · Data logics

1 Introduction

XPath is the most widely used query language for XML documents; it is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [1]. XPath has syntactic operators to navigate the tree using accessibility relations such as ‘child’, ‘parent’, ‘sibling’, et cetera, and can make tests on intermediate nodes. Core-XPath [2] is the fragment of XPath 1.0 containing only the navigational behavior of XPath. Core-XPath can express properties on nodes with respect to the underlying tree structure of the XML document, such as ‘nodes with label B’, or ‘nodes that have both a child with label A and a grandchild with label B’. It can also express properties on paths along the tree such as ‘the ending node is the grandchild of the starting node’, or ‘the initial node has label A and has a child with A, and the ending node is the grandparent of the starting node’. The first type of formulas are evaluated on individual nodes and are called *node expressions*, while the formulas of the second type are evaluated on pairs of nodes and are called *path expressions*. However, Core-XPath cannot express conditions on the actual data contained in the attributes, such as with a node expression saying ‘this node has two children with different data values’, or

‘the value of this node coincides with the value of some descendant’. In contrast, Core-Data-XPath [3] (which we here call simply XPath₌) can perform these data comparisons. Indeed, XPath₌ is the extension of Core-XPath with the addition of (in)equality tests between attributes of elements in an XML document.

In the paper [4], the expressive power of fragments of XPath₌ was studied, from a logical and model-theoretical point of view, when the set of navigational axes was taken among \downarrow , \uparrow , and the reflexive-transitive closure of those axes. In that work, the semantic universe of study was that of *data trees*, whose nodes have a single label taken from a finite alphabet and a single data value from an infinite domain. A focus of study in that work was that of bisimulation, which is a classic tool of modal logics, used to determine equivalence between relational models. A node x of a data tree T and a node x' of a data tree T' are said to be *bisimilar* if they satisfy some special (depending of the studied fragment) back-and-forth conditions over the structure of the data tree. In [4], suitable notions of bisimulation were devised for the XPath₌ fragments under considerations. Then, showing a characterization result in the style of the Hennessy-Milner’s theorem for Basic Modal Logic [6], it was proven that if x and x' are bisimilar then they satisfy exactly the same node expressions, and that the converse is also true for trees whose every node only has a finite number of children. Hence, bisimulation coincides with logical equivalence, i.e., with *indistinguishability by means of node expressions*. The paper [4] also stated and proved theorems in the style of van Benthem’s for Basic Modal Logic [7], but in the context of XPath₌. One of these theorems states that the downward fragment of XPath₌ coincides with the bisimulation-invariant fragment of first-order logic with one free variable (over the adequate signature). For the case of the vertical fragment of XPath₌, this characterization fails, but a weaker result is proved instead.

In [5], the study of bisimulation for XPath₌ was expanded in order to encompass bisimulation notions over two-pointed data trees (i.e. a data tree and two specified nodes), giving a bisimulation notion for the downward and the vertical fragment of XPath₌ over two-pointed data trees, and proving the corresponding characterization results between logical equivalence and bisimulation. In this way, the paper expanded the results of [4] from the domain of node expressions to that of path expressions.

In the current work we focus our study in a family of generalizations of XPath₌(\downarrow), which includes not only the capability of comparing the end nodes of two paths by data (in)equality, but also checking for other types of arbitrary n -ary predicates over nodes at the end nodes of n paths. Given a fixed set \mathcal{R} of relation symbols with their arity, we generalize the concept of data trees to encompass arbitrary relations between nodes, and study the logic XPath _{\mathcal{R}} (\downarrow) over this universe. We give a general suitable bisimulation notion for these \mathcal{R} fragments, and show a Hennessy-Milner-like characterization result connecting this notion with that of logical equivalence. Furthermore, we provide a theorem in the style of van Benthem’s result for Basic Modal Logic, thus characterizing these logics as fragments of first-order logic whose formulas are invariant under this new notion of bisimulation. While we initially state these results for the

case of the full universe with no semantic restrictions, in the end we show that restricting ourselves to some universes of data trees preserves our results.

2 Preliminaries

A *data tree* is a directed tree whose nodes have a single label from a finite alphabet and a single data value from a (possibly infinite) data domain. XPath $_{=}$ (\downarrow) is a logic that can express properties about these structures, for instance, we can say if a node has or not certain label or if a node is a child of another via the child axis \downarrow . The most important capabilities of this logic lie in its *data tests* $\langle \alpha = \beta \rangle$ or $\langle \alpha \neq \beta \rangle$, which can compare the data values of two nodes. More precisely, a node satisfying $\langle \alpha = \beta \rangle$ (resp. $\langle \alpha \neq \beta \rangle$) means that there are paths from it such that the first one satisfies α , the second one satisfies β and their final points have equal (resp. non-equal) data value. Having the same data value can be thought of as a binary equivalence relation between nodes, so a natural question that arises is the possibility of extending the types of comparisons that can be made at the end of paths.

Fixed \mathcal{R} , XPath $_{\mathcal{R}}$ (\downarrow) is an extension of XPath $_{=}$ (\downarrow) in the sense that now it can allow relations with arbitrary arity (not necessarily binary) between final points of paths from a certain node of a tree. Here, formulas of the type $\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_r$ express this type of operation, where r represents a particular relation symbol and \mathcal{R} is the set of such symbols. In this context, the data test $\varphi := \langle \alpha = \beta \rangle$ can be re-expressed as $\langle \alpha, \beta \rangle_{=_d}$ where the sub index $=_d$ can be interpreted over a data tree as the equivalence relation of having same data value. Similarly, a label test $\varphi := a$ can be re-expressed as $\langle \varepsilon \rangle_a$, where a is a unary relation symbol.

Initially, we will consider that the universe of the models for XPath $_{\mathcal{R}}$ (\downarrow) is still that of trees, but extended with an *arbitrary* relation over nodes for each $r \in \mathcal{R}$ (with its respective arity). We call these models \mathcal{R} -trees.

Definition 1. *Let \mathcal{R} be a finite and non-empty set of relational symbols with arities given by the function $\mathcal{A} : \mathcal{R} \rightarrow \mathbb{N}_{\geq 1}$. The formulas of XPath $_{\mathcal{R}}$ (\downarrow) are defined by the grammar below:*

$$\begin{aligned} \varphi, \psi = \varphi \wedge \psi \mid \neg \varphi \mid \langle \alpha_1, \alpha_2, \dots, \alpha_{\mathcal{A}(r)} \rangle_r \mid \langle \alpha_1, \alpha_2, \dots, \alpha_{\mathcal{A}(r)} \rangle_{\bar{r}} \quad r \in \mathcal{R} \\ \alpha, \beta = \varepsilon \mid \downarrow \mid \alpha\beta \mid \alpha \cup \beta \mid [\varphi] \end{aligned}$$

The first row generates the **node expressions**, and the second one, the **path expressions**: $\langle \alpha_1, \alpha_2, \dots, \alpha_{\mathcal{A}(r)} \rangle_r$ and $\langle \alpha_1, \alpha_2, \dots, \alpha_{\mathcal{A}(r)} \rangle_{\bar{r}}$ are called path tests where α_i is a path expression for all $i \in \{1, 2, \dots, \mathcal{A}(r)\}$; $\alpha\beta$ and $\alpha \cup \beta$ are respectively the concatenation and union between α and β ; $[\varphi]$ are node tests (as part of a path expression) where φ is a node expression; and the symbols ε and \downarrow are the self and child axes, respectively.

As usual we use $\varphi \vee \psi$ as a shorthand for $\neg(\neg\varphi \wedge \neg\psi)$. By $\overline{\mathcal{R}}$ we indicate the set of the complement symbols \bar{r} with $r \in \mathcal{R}$ and extend \mathcal{A} to $\mathcal{R} \cup \overline{\mathcal{R}}$ as $\mathcal{A}(\bar{r}) := \mathcal{A}(r)$ for $r \in \mathcal{R}$.

From now on, we consider a fixed $\mathcal{R} \neq \emptyset$, a family of predicate symbols as in Definition 1.

Definition 2. An \mathcal{R} -tree $\mathcal{T} = \langle T, E_{\downarrow}, \{R_r\}_{r \in \mathcal{R}} \rangle$ is a set T with a binary relation $E_{\downarrow} \subseteq T^2$ such that $\langle T, E_{\downarrow} \rangle$ is a rooted tree, and with a family of relations $\{R_r\}_{r \in \mathcal{R}}$, where $R_r \subseteq T^{\mathcal{A}(r)}$. We use $R_{\bar{r}}$ to abbreviate the complement of R_r , that is, $R_{\bar{r}} := T^{\mathcal{A}(r)} \setminus R_r$. If $x, y \in T$, the pair (\mathcal{T}, x) is called a **pointed model**, and (\mathcal{T}, x, y) is called a **two-pointed model**.

Note 1. To simplify the notation when there is no risk of confusion, we will often simply use r to refer to the semantics R_r corresponding to the \mathcal{R} -tree currently in discussion.

In some cases when a is an unary symbol, we might simply write a instead of the node expression $\langle \varepsilon \rangle_a$.

Example 1. Let us consider $\mathcal{T} = \langle T, E_{\downarrow}, \{a, b, c, =_n, =_w\} \rangle$ a representation of a very simple and brief bibliographical database as in Fig. 1. Here a, b, c are unary predicates (representing in \mathcal{T} that a node is an author, book, or chapter, respectively). $=_n, =_w$ are binary predicates, which express in \mathcal{T} when two nodes have the same numerical value for $=_n$, or the same word for $=_w$.

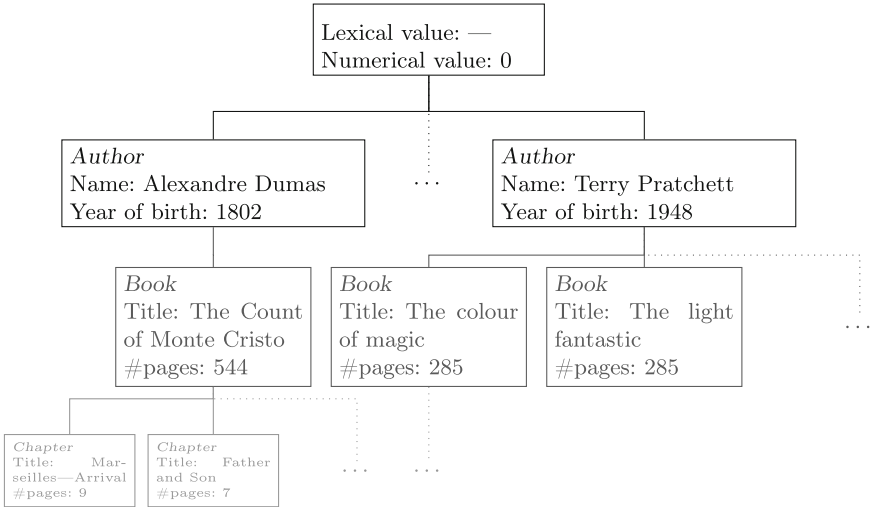


Fig. 1. A representation of a bibliographical database as in Example 1, via a tree whose nodes contain both a numeric and lexical value and where nodes can belong to any of three types of unary relations ((a)uthor), (b)ook, (c)hapter).

Definition 3. We now give the interpretation of the symbols from Definition 1 over an \mathcal{R} -tree $\mathcal{T} = \langle T, E_\downarrow, \{R_r\}_{r \in \mathcal{R}} \rangle$.

$$\begin{aligned} \llbracket \varepsilon \rrbracket_{\mathcal{T}} &:= \{(x, y) \in T^2 \mid x = y\} & \llbracket \downarrow \rrbracket_{\mathcal{T}} &:= E_\downarrow & \llbracket [\varphi] \rrbracket_{\mathcal{T}} &:= \{(x, x) \in T^2 \mid x \in \llbracket \varphi \rrbracket_{\mathcal{T}}\} \\ \llbracket \alpha\beta \rrbracket_{\mathcal{T}} &:= \llbracket \alpha \rrbracket_{\mathcal{T}} \circ \llbracket \beta \rrbracket_{\mathcal{T}} & \llbracket \alpha \cup \beta \rrbracket_{\mathcal{T}} &:= \llbracket \alpha \rrbracket_{\mathcal{T}} \cup \llbracket \beta \rrbracket_{\mathcal{T}} \\ \llbracket \varphi \wedge \psi \rrbracket_{\mathcal{T}} &:= \llbracket \varphi \rrbracket_{\mathcal{T}} \cap \llbracket \psi \rrbracket_{\mathcal{T}} & \llbracket \neg \varphi \rrbracket_{\mathcal{T}} &:= T \setminus \llbracket \varphi \rrbracket_{\mathcal{T}} \end{aligned}$$

$$\begin{aligned} \llbracket \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_r \rrbracket_{\mathcal{T}} &:= \{x \in T \mid \exists y_1, y_2, \dots, y_n \in T \forall i \in \{1, 2, \dots, n\} \\ &\quad (x, y_i) \in \llbracket \alpha_i \rrbracket_{\mathcal{T}} \wedge R_r(y_1, y_2, \dots, y_n)\} \\ \llbracket \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_{\bar{r}} \rrbracket_{\mathcal{T}} &:= \{x \in T \mid \exists y_1, y_2, \dots, y_n \in T \forall i \in \{1, 2, \dots, n\} \\ &\quad (x, y_i) \in \llbracket \alpha_i \rrbracket_{\mathcal{T}} \wedge R_{\bar{r}}(y_1, y_2, \dots, y_n)\} \end{aligned}$$

If φ is a node expression and α a path expression, then we write $(\mathcal{T}, x) \models \varphi$ iff $x \in \llbracket \varphi \rrbracket_{\mathcal{T}}$ and $(\mathcal{T}, x, y) \models \alpha$ iff $(x, y) \in \llbracket \alpha \rrbracket_{\mathcal{T}}$.

Remark 1. Note that we can express the node property $\langle \alpha \rangle$, which indicates that from a node it is possible to descend to some node via the path expression α . Indeed, we can see that for any $r \in \mathcal{R}$, the following formula expresses the desired property: $\langle \alpha, \dots, \alpha \rangle_r \vee \langle \alpha, \dots, \alpha \rangle_{\bar{r}}$.

We can also define a node expression that is true on any node, irrespective of the semantics of the tree: take any $r \in \mathcal{R}$ and define $\top := \langle \varepsilon, \dots, \varepsilon \rangle_r \vee \langle \varepsilon, \dots, \varepsilon \rangle_{\bar{r}}$.

Example 2. From the root t_0 of the database represented in Fig. 1, we could ask whether there is an author having the same name as some book title. The answer to this depends on whether $(\mathcal{T}, t_0) \models \langle \downarrow [\langle \varepsilon \rangle_a], \downarrow \downarrow [\langle \varepsilon \rangle_b] \rangle_{=w}$. From what we can see in the graphical representation, this does not happen, as we can see that $\llbracket \langle \downarrow [\langle \varepsilon \rangle_a], \downarrow \downarrow [\langle \varepsilon \rangle_b] \rangle_{=w} \rrbracket_{\mathcal{T}} = \emptyset$.

Note that here (as in some types of data logics) asking whether there are two different nodes with the same data value is in general not possible. For example, $\llbracket a \wedge \langle \downarrow, \downarrow \rangle_{=n} \rrbracket_{\mathcal{T}}$ (see Note 1) will contain *any* node that corresponds to an author that has written a book, since there is no guarantee that the two witnesses of \downarrow are different. Indeed, in the figure $\llbracket \langle \varepsilon \rangle_a \wedge \langle \downarrow, \downarrow \rangle_{=n} \rrbracket_{\mathcal{T}}$ would contain the node of Alexandre Dumas, even though it has only one book child.

Example 3. Consider an extension of \mathcal{T} and \mathcal{R} including the binary predicate $=_n^d$, such that over \mathcal{T} , $x =_n^d y$ iff $(x =_n y$ and $x \neq y)$. Now we can express in \mathcal{T} properties such as ‘this author has two different books with the same number of pages’, via $\varphi : a \wedge \langle \downarrow, \downarrow \rangle_{=n}^d$.

Definition 4. Two expressions η_1 and η_2 of the same type (node or path expression) are said to be **semantically equivalent**, written $\eta_1 \equiv \eta_2$ if for all \mathcal{T} we have that $\llbracket \eta_1 \rrbracket_{\mathcal{T}} = \llbracket \eta_2 \rrbracket_{\mathcal{T}}$.

Remark 2. It easy to see that the semantic equivalence is preserved by any syntactic construction. That is, negating two equivalent node expression results in equivalent node expressions, concatenating a path expression to two different but equivalent path expressions results in two equivalent path expressions, etc.

Remark 3. Let $\bar{\gamma} = \gamma_1, \dots, \gamma_k$ and $\bar{\gamma}' = \gamma_{k+1}, \dots, \gamma_n$ be finite (and potentially empty) sequences of path expressions. For all α and β path expressions and for all $* \in \mathcal{R} \cup \bar{\mathcal{R}}$ with $\mathcal{A}(\alpha) = n + 1$ we have that

$$\langle \bar{\gamma}, \alpha \cup \beta, \bar{\gamma}' \rangle_* \equiv \langle \bar{\gamma}, \alpha, \bar{\gamma}' \rangle_* \vee \langle \bar{\gamma}, \beta, \bar{\gamma}' \rangle_*$$

Guided by this fact, we will re-define $\mathbf{XPath}_{\mathcal{R}}(\downarrow)$ as the fragment of the original one (Definition 1) where we do not include the rule $\alpha \cup \beta$ in the grammar. That is, the fragment whose path expressions do not have the symbol \cup in their syntax. Even though this **union-free fragment** is less expressive when considering both path and node expressions, the expressive power remains the same as the full fragment when considering only pointed models and node expressions, because of the semantic equivalence given in Remark 3.

We now give the definition of direct path expressions, which are path expressions without unnecessary concatenations of symbols. This definition is used later to define normal expressions, whose purpose is to simplify the proofs by induction.

Definition 5. A **direct path expression** α is a path expression of the form $\alpha = \varepsilon$ or $\alpha = \downarrow \xi_1 \downarrow \dots \downarrow \xi_n$ where each ξ_i is an empty string or a node test.

A normal expression is one with all the path expressions in its path tests being direct. We will formally define this idea by means the operator $\text{sub}(-)$.

Definition 6. For a formula η we denote by $\text{sub}(\eta)$ the set defined recursively as follows:

$$\begin{aligned} \text{sub}(\varepsilon) &= \text{sub}(\downarrow) := \emptyset & \text{sub}(\alpha\beta) &:= \text{sub}(\alpha) \cup \text{sub}(\beta) \\ \text{sub}(\neg\varphi) &= \text{sub}([\varphi]) := \text{sub}(\varphi) & \text{sub}(\varphi \wedge \psi) &:= \text{sub}(\varphi) \cup \text{sub}(\psi) \\ \text{sub}(\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*) &:= \{ \alpha_1, \alpha_2, \dots, \alpha_n \} \cup \bigcup_{i=1}^n \text{sub}(\alpha_i) & \text{for } * \in \mathcal{R} \cup \bar{\mathcal{R}} \end{aligned}$$

Definition 7. A formula η is a **normal expression** if all the path expressions in $\text{sub}(\eta) \cup \{\eta\}$ are direct.

The downward depth of an expression measures the maximum depth from the current point of evaluation that the formula could potentially ‘see’. The idea is that, when analysing such an expression over a particular point or pair of points, nodes that are further down than this depth have no effect on the resulting truth value of the expression.

Definition 8. The **downward depth** of η denoted by $\text{dd}(\eta)$ is the number defined as follows:

$$\begin{aligned} \text{dd}(\neg\varphi) &:= \text{dd}(\varphi) & \text{dd}(\varphi \wedge \psi) &:= \max\{\text{dd}(\varphi), \text{dd}(\psi)\} \\ \text{dd}(\lambda) &:= 0 & \text{where } \lambda &\text{ represents the empty string} \\ \text{dd}(\varepsilon\beta) &:= \text{dd}(\beta) & \text{dd}([\varphi]\beta) &:= \max\{\text{dd}(\varphi), \text{dd}(\beta)\} & \text{dd}(\downarrow\beta) &:= 1 + \text{dd}(\beta) \\ \text{dd}(\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*) &:= \max\{\text{dd}(\alpha_1), \text{dd}(\alpha_2), \dots, \text{dd}(\alpha_n)\} & \text{for } * \in \mathcal{R} \cup \bar{\mathcal{R}} \end{aligned}$$

The set of all formulas with downward depth less than or equal to $\ell \geq 0$ is written as $\mathbf{XPath}_{\mathcal{R}}(\downarrow)$.

Note that this definition encompasses all (union-free) formulas in $\mathbf{XPath}_{\mathcal{R}}(\downarrow)$ and the function $\text{dd}(-)$ is well-defined.

Remark 4. Let $\bar{\gamma} = \gamma_1, \dots, \gamma_k$ and $\bar{\gamma}' = \gamma_{k+1}, \dots, \gamma_n$ be finite (and potentially empty) sequences of path expressions. The following semantic equivalences also preserve the downward depth.

1. $\langle \bar{\gamma}, [\varphi]\alpha, \bar{\gamma}' \rangle_* \equiv \varphi \wedge \langle \bar{\gamma}, \alpha, \bar{\gamma}' \rangle_*$ $* \in \mathcal{R} \cup \overline{\mathcal{R}}$ with $\mathcal{A}(*) = n + 1$
2. $\varepsilon\alpha \equiv \alpha$
3. $[\varphi][\psi] \equiv [\varphi \wedge \psi]$

Proposition 1. *For every formula η we have that*

- i) *If η is a node expression then there exists a normal node expression η' with $\text{dd}(\eta) = \text{dd}(\eta')$ and $\eta \equiv \eta'$.*
- ii) *If η is a path expression then there exists a normal path expression η' with $\text{dd}(\eta) = \text{dd}(\eta')$ and $\eta \equiv \eta'$.*

Proof. One can easily prove the statement by syntactic induction over η and making use of the semantic equivalences from Remark 4.

3 Bisimulation and Equivalence

The classic Hennessy-Milner's characterization theorem [6] for Basic Modal Logic establishes the relation between two notions: logical equivalence and bisimilarity. In our case, the former notion indicates when a pair of pointed models are indistinguishable by means of node expressions. The latter intuitively ensures that for each selection of paths in one of the models, there are copies in the other, preserving the possible relational properties between their respective final points.

Definition 9. *Let $\ell \geq 0$. Given (\mathcal{T}, x) and (\mathcal{T}', x') we say that they are \mathcal{R}_ℓ -logically equivalent and denote it by $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$ if for all node expression $\varphi \in \mathbf{XPath}_{\mathcal{R}}^\ell(\downarrow)$ we have that $(\mathcal{T}, x) \models \varphi$, if and only if, $(\mathcal{T}', x') \models \varphi$.*

(\mathcal{T}, x) and (\mathcal{T}', x') are \mathcal{R} -logically equivalent, written $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$, if $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$ for all $\ell \geq 0$. In other words, if for any node expression φ , $(\mathcal{T}, x) \models \varphi$, if and only if, $(\mathcal{T}', x') \models \varphi$.

Definition 10. *A path μ in a tree \mathcal{T} is a sequence $\mu = \mu_0 \downarrow \mu_1 \downarrow \dots \downarrow \mu_n$ where $n \geq 0$, $\mu_i \in T$ for all $i \in \{0, \dots, n\}$ and $E_\downarrow(\mu_i, \mu_{i+1})$ for all $i \in \{0, 1, \dots, n-1\}$. The length $\text{len}(\mu) := n$ is the number of symbols \downarrow in μ . The i -th node of μ is $[\mu]_i := \mu_i$ and $\text{end}(\mu) := [\mu]_{\text{len}(\mu)}$ is the final node of μ .*

We denote by $\text{Path}(\mathcal{T})$ the set of all paths μ in \mathcal{T} . For a node $x \in \mathcal{T}$, $\text{Path}(\mathcal{T}, x)$ are the paths μ starting from the node x , i.e., $[\mu]_0 = x$ and by $\text{Path}_k(\mathcal{T}, x)$ we refer to the subset of paths in $\text{Path}(\mathcal{T}, x)$ with length at most k .

The concatenation $\mu \odot \nu$ of two paths $\mu, \nu \in \text{Path}(\mathcal{T})$ such that $\nu_0 = \text{end}(\mu)$ is defined as $[\mu \odot \nu]_i := [\mu]_i$ for all $i \in \{0, \dots, \text{len}(\mu)\}$ and $[\mu \odot \nu]_{\text{len}(\mu)+i} := [\nu]_i$ for all $i \in \{0, \dots, \text{len}(\nu)\}$.

Definition 11. Given a \mathcal{R} -tree \mathcal{T} and a node $x \in T$, $\mathcal{T}|_\ell^x$ is the \mathcal{R} -tree whose underlying tree is the set of nodes $y \in T$ for which there exists a path $\mu \in \text{Path}_\ell(\mathcal{T}, x)$ with $\text{end}(\mu) = y$ (note that x is the root of such tree).

Remark 5. For all $\ell \geq 0$ and \mathcal{R} -tree \mathcal{T} , we have that $(\mathcal{T}|_\ell^x, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}, x)$.

Definition 12. Let \mathcal{T} and \mathcal{T}' be \mathcal{R} -trees. An \mathcal{R}_ℓ -bisimulation $\mathcal{Z} = \{Z_k\}_{0 \leq k \leq \ell}$ is a family of relations $Z_k \subseteq T \times T'$ such that for all k , for all $(x, x') \in Z_k$, and for all $n \in \text{Im}(\mathcal{A})$, the clauses below hold.

Zig For every selection of paths $\mu_1, \mu_2, \dots, \mu_n \in \text{Path}_k(\mathcal{T}, x)$, there exist paths $\mu'_1, \mu'_2, \dots, \mu'_n \in \text{Path}_k(\mathcal{T}', x')$ such that for all $j \in \{1, \dots, n\}$ and for all $r \in \mathcal{R}$ we have that:

- i) $\text{len}(\mu_j) = \text{len}(\mu'_j)$
- ii) $([\mu_j]_i, [\mu'_j]_i) \in Z_{k-i} \ \forall i \in \{0, \dots, \text{len}(\mu_j)\}$
- iii) $R_r(\text{end}(\mu_1), \text{end}(\mu_2), \dots, \text{end}(\mu_n)) \Leftrightarrow R'_r(\text{end}(\mu'_1), \text{end}(\mu'_2), \dots, \text{end}(\mu'_n))$

Zag For every selection of paths $\mu'_1, \mu'_2, \dots, \mu'_n \in \text{Path}_k(\mathcal{T}', x')$ there exist paths $\mu_1, \mu_2, \dots, \mu_n \in \text{Path}_k(\mathcal{T}, x)$ such that for all $j \in \{1, \dots, n\}$ and for all $r \in \mathcal{R}$ we have that:

- i) $\text{len}(\mu_j) = \text{len}(\mu'_j)$
- ii) $([\mu_j]_i, [\mu'_j]_i) \in Z_{k-i} \ \forall i \in \{0, \dots, \text{len}(\mu_j)\}$
- iii) $R_r(\text{end}(\mu_1), \text{end}(\mu_2), \dots, \text{end}(\mu_n)) \Leftrightarrow R'_r(\text{end}(\mu'_1), \text{end}(\mu'_2), \dots, \text{end}(\mu'_n))$

Two pointed models (\mathcal{T}, x) and (\mathcal{T}', x') are said to be \mathcal{R}_ℓ -bisimilar, denoted $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$, if there exists an \mathcal{R}_ℓ -bisimulation $\mathcal{Z} = \{Z_i\}_{0 \leq i \leq \ell}$ such that $(x, x') \in Z_\ell$.

If $Z \subseteq T \times T'$ is a relation such that for all $\ell \geq 0$ the family $\{Z_i \mid Z_i = Z\}_{0 \leq i \leq \ell}$ is a \mathcal{R}_ℓ -bisimulation then we call Z an \mathcal{R} -bisimulation. If there exists a \mathcal{R} -bisimulation Z with $(x, x') \in Z$ then (\mathcal{T}, x) and (\mathcal{T}', x') are \mathcal{R} -bisimilar, written as $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$.

Remark 6. It is useful to observe that, when there are predicates of arity 2 or greater, the Zig (and Zag) conditions can be replaced in the case of unary predicates with a simpler ‘Harmony’ condition in the style of bisimulation for modal logics and $\text{XPath}_=$: for any unary predicate u it is enough to check that whenever xZx' , then $u(x)$ iff $u(x')$. Note, however, that this replacement cannot be done if we *only* have unary predicates in \mathcal{R} , since doing so would remove all Zig and Zag conditions, and thus we would not be comparing any topological information about the models.

Remark 7. Using the Remark 6, we can see that our Definition 12 for the concept of \mathcal{R} -bisimulation generalizes the definition of bisimulation for $\text{XPath}_=(\downarrow)$ over the universe of data trees from [4]. It does so by taking $\mathcal{R} = \{=_d\} \cup \mathbb{A}$, where $=_d$ is a binary symbol (interpreted as data equality over data trees) and the finite symbols in the label set \mathbb{A} are unary predicates. See also Theorem 4 and the discussion preceding it.

Remark 8. The notion of \mathcal{R} -bisimilarity given in Definition 12 does not coincide with bisimilarity for multi-relational Kripke models (where \downarrow and each relation from \mathcal{R} get their own modal operator, and where we translate from \mathcal{R} -trees into Kripke models). Indeed, consider the case where \mathcal{R} consists solely of a binary relation $=_d$ (we will represent the semantics of $=_d$ with numeric data values). For an example of two pointed models that are modally bisimilar but not \mathcal{R} -bisimilar, consider: on one hand the infinite linear tree (\mathcal{T}, x) , where the root x has data value 1, the sole next child has data value 2, the next one has data value 1, and so on alternating between these two values (i.e. 1, 2, 1, 2, ...); on the other hand, take the infinite linear tree (\mathcal{T}', x') that has data value 1 in all nodes (i.e. 1, 1, 1, 1, ...).

Intuitively, while modal bisimulation appears to have greater navigational freedom by being able to move with any modality from \mathcal{R} , when doing that it cannot keep track of the actual topology of the model (given by \downarrow) nor can it ask whether the endpoints of paths are related via an $r \in \mathcal{R}$.

Example 4. Let $\mathcal{R} = \{b, f, S\}$, where b, f are unary predicate symbols and S is a ternary predicate symbol. We consider the \mathcal{R} -trees $\mathcal{T}, \mathcal{T}'$ from Fig. 2, where a node in b is represented as having a red border, and a node satisfying f is represented as being filled with the color black. In both trees, S has an interpretation related to the numbers: $S(x, y, z)$ iff $d(x) = d(y) + d(z)$, where $d(w)$ represents the number drawn on the node w . If we call t_0, t'_0 the respective roots of both trees, it is easy to verify that (\mathcal{T}, t_0) and (\mathcal{T}', t'_0) are \mathcal{R} -bisimilar via the represented Z relation. It is important to note that it does not matter that \mathcal{T} represents the values of nodes in integers and \mathcal{T}' does so with rational numbers: the only thing that matters is the semantics of each predicate in \mathcal{R} , as the logic itself has no way of ‘seeing’ these particular values (and indeed some possible semantics of S cannot be expressed in this form).

As we have for modal logics, the bounded notions \mathcal{R}_ℓ -logical equivalence and \mathcal{R}_ℓ -bisimulation coincide. That is, two pointed models are \mathcal{R}_ℓ -logically equivalent, if and only if, they are \mathcal{R}_ℓ -bisimilar.

The proof of each one of the following statements can be found in the Appendix.

Lemma 1. *For every $\ell \geq 0$, there are finitely many equivalence classes (modulo semantic equivalence) of node expressions with downward depth at most ℓ .*

Corollary 1. *For each $\ell \geq 0$ and a pointed model (\mathcal{T}, x) there is a node expression $\chi_{(\mathcal{T}, x)}^\ell \in \text{XPath}_{\mathcal{R}}^\ell(\downarrow)$ satisfying that for any pointed model (\mathcal{T}', x') , $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$ iff $(\mathcal{T}', x') \models \chi_{(\mathcal{T}, x)}^\ell$.*

Proposition 2. *Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that if $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$ then $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$.*

Proposition 3. *Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that if $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$ then $(\mathcal{T}, x) \equiv_\ell^{\mathcal{R}} (\mathcal{T}', x')$.*

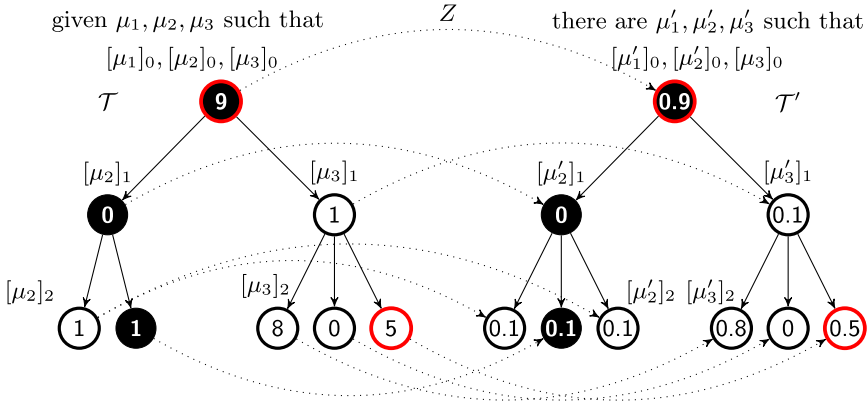


Fig. 2. A representation of two \mathcal{R} -trees \mathcal{T} and \mathcal{T}' for $\mathcal{R} = \{b, f, S\}$, as in Example 4. Also represented in the figure is a Zig step and, via a dotted line connecting nodes, a $\{b, f, S\}$ -bisimulation Z between (\mathcal{T}, t_0) and (\mathcal{T}', t'_0) .

We can unify Proposition 2 and Proposition 3 in the following statement:

Theorem 1. *Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that $(\mathcal{T}, x) \rightleftharpoons_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$, if and only if, $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$.*

4 Characterizations

In this section we state and prove characterization results for $\text{XPath}_{\mathcal{R}}(\downarrow)$, in the style of Hennessy-Milner’s theorem [6] for Basic Modal Logic. Our result states that \mathcal{R} -logical equivalence and \mathcal{R} -bisimulation agree over models whose underlying tree is finitely branching. After giving this result, we will treat $\text{XPath}_{\mathcal{R}}(\downarrow)$ as a fragment of the first order logic in order to show a characterization result in the style of van Benthem’s theorem [7] for Basic Modal Logic, concluding that the formulas of $\text{XPath}_{\mathcal{R}}(\downarrow)$ are exactly those of the first order logic (with certain signature) which are preserved by \mathcal{R} -bisimulation.

Proposition 4. *Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that if $(\mathcal{T}, x) \rightleftharpoons^{\mathcal{R}} (\mathcal{T}', x')$ then $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$.*

Proof. Suppose $(\mathcal{T}, x) \rightleftharpoons^{\mathcal{R}} (\mathcal{T}', x')$ via $(x, x') \in Z \subseteq T \times T'$. For every $\ell \geq 0$, the family $\mathcal{Z}_{\ell} := \{Z_0 = Z_1 = \dots = Z_{\ell} := Z\}$ is a \mathcal{R}_{ℓ} -bisimulation between (\mathcal{T}, x) and (\mathcal{T}', x') . Thus, by Theorem 1, for every $\ell \geq 0$ we have that $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}}$. In other words, $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$.

Definition 13. *A pointed model (\mathcal{T}, x) is **finitely branching** if for all $k \geq 0$ the set $\text{Path}_k(\mathcal{T}, x)$ is finite.*

Theorem 2 (Hennessy-Milner’s style characterization). *If (\mathcal{T}, x) and (\mathcal{T}', x') are finitely branching, then $(\mathcal{T}, x) \rightleftharpoons^{\mathcal{R}} (\mathcal{T}', x')$ if and only if $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$.*

Proof.

(\Rightarrow) By Proposition 4.

(\Leftarrow) Consider the relation $Z \subseteq T \times T'$, which is defined such that $(z, z') \in Z$ iff $(\mathcal{T}, z) \equiv^{\mathcal{R}} (\mathcal{T}', z')$. The proof to show that Z is a \mathcal{R} -bisimulation is analogous to that which was given for Proposition 3 in order to see that \mathcal{Z} was a \mathcal{R}_ℓ -bisimulation. Now, we define $\Phi^{(j,i)} := \bigwedge_{\bar{\mu} \in P} \varphi_{\bar{\mu}}^{(j,i)}$ since the set P is finite because (\mathcal{T}', x') is finitely branching.

Remark 9. The classical counterexample for the modal logic with only one modality still works to see that the finitely branching hypothesis is required: take on one side an infinitely branching tree constructed with one branch of each finite length; on the other side, a copy of the previous tree with the addition of an infinitely long linear branch hanging from the root. For every $r \in \mathcal{R}$, we set in both trees that $R_r = \emptyset$. It can be seen that both \mathcal{R} -trees with their roots are logically equivalent but that any proposed Z fails to be a bisimulation.

From now on, we focus on XPath $_{\mathcal{R}}(\downarrow)$ as a fragment of the first-order logic FO($\sigma_{\mathcal{R}}$) with the natural signature $\sigma_{\mathcal{R}}$, and with a standard translation ST($-$) between formulas of XPath $_{\mathcal{R}}(\downarrow)$ and of first-order logic. For details, see the corresponding Definitions 16 and 17 in the Appendix.

The following lemmas are the key for proving the van Benthem's characterization style theorem. Before stating them, we need to give some definitions:

Definition 14. Let $\ell \geq 0$ and $\psi(u), \varphi(u) \in FO(\sigma_{\mathcal{R}})$ with one free variable u .

- $\psi(u)$ is **ℓ -local** if for all pointed model (\mathcal{T}, x) , $\mathcal{T} \models \psi[u \mapsto x]$ iff $\mathcal{T}|_{\ell}^x \models \psi[u \mapsto x]$.
- $\psi(u)$ is **$\equiv^{\mathcal{R}}$ -invariant** if for all (\mathcal{T}, x) and (\mathcal{T}', x') such that $\mathcal{T} \models \psi[u \mapsto x]$ and $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$ then $\mathcal{T}' \models \psi[u \mapsto x']$.
- $\psi(u)$ is **$\equiv_{\ell}^{\mathcal{R}}$ -invariant** if for all (\mathcal{T}, x) and (\mathcal{T}', x') such that $\mathcal{T} \models \psi[u \mapsto x]$ and $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ then $\mathcal{T}' \models \psi[u \mapsto x']$.
- $\psi(u)$ and $\varphi(u)$ are **semantically equivalent over trees** $\psi(u) \stackrel{\text{trees}}{\equiv} \varphi(u)$ if for all (\mathcal{T}, x) we have that $\mathcal{T} \models \psi[u \mapsto x]$ iff $\mathcal{T} \models \varphi[u \mapsto x]$.
- A pointed model (\mathcal{T}, x) has **depth at most ℓ** if $\text{Path}_{\ell}(\mathcal{T}, x) = \text{Path}_n(\mathcal{T}, x)$ for all $n \geq \ell$.

Lemma 2. For each $\equiv^{\mathcal{R}}$ -invariant $\psi(u) \in FO(\sigma_{\mathcal{R}})$ with one free variable u , there is $\ell \geq 0$ such that ψ is ℓ -local.

Proof. See Appendix.

Lemma 3. Suppose (\mathcal{T}, x) and (\mathcal{T}', x') have depth at most ℓ . Then $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$ if and only if $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$. Equivalently, $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$, if and only if, $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$.

Proof. The left-to-right direction is clear. For the other direction, suppose that we have $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ via $\mathcal{Z} = \{Z_k\}_{0 \leq k \leq \ell}$. Then we have that $Z := \bigcup_{k=0}^{\ell} Z_k$ is a \mathcal{R} -bisimulation between (\mathcal{T}, x) and (\mathcal{T}', x') .

Theorem 3 (van Benthem’s style characterization). *Let $\psi(u)$ be a formula in $FO(\sigma_{\mathcal{R}})$ with one free variable u . The following statements are equivalent:*

1. $\psi(u)$ is $\equiv^{\mathcal{R}}$ -invariant.
2. There exists a node expression φ such that $\psi(u) \stackrel{trees}{\equiv} ST(\varphi)(u)$

Proof.

(1 \Rightarrow 2) Suppose $\psi(u)$ is $\equiv^{\mathcal{R}}$ -invariant. By Lemma 2 we know that there is some $\ell \geq 0$ for which $\psi(u)$ is ℓ -local. Let us see that $\psi(u)$ is $\equiv_{\ell}^{\mathcal{R}}$ -invariant for such ℓ : given (\mathcal{T}, x) and (\mathcal{T}', x') such that $\mathcal{T} \models \psi[u \mapsto x]$ and $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ we want to show that $\mathcal{T}' \models \psi[u \mapsto x']$. On one side, by Remark 5, $(\mathcal{T}|_{\ell}^x, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x') \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}'|_{\ell}^{x'}, x')$. On the other side, by Lemma 3, $(\mathcal{T}|_{\ell}^x, x) \equiv^{\mathcal{R}} (\mathcal{T}'|_{\ell}^{x'}, x')$. Since $\mathcal{T} \models \psi[u \mapsto x]$ iff $\mathcal{T}|_{\ell}^x \models \psi[u \mapsto x]$, we conclude that $\mathcal{T}'|_{\ell}^{x'} \models \psi[u \mapsto x']$ iff $\mathcal{T}' \models \psi[u \mapsto x']$. From this, it is clear that $\varphi(u) \stackrel{trees}{\equiv} ST\left(\bigvee_{\mathcal{T} \models \varphi[u \mapsto x]} \chi_{(\mathcal{T}, x)}^{\ell}\right)(u)$ where $\chi_{(\mathcal{T}, x)}^{\ell}$ is given by Corollary 1.

(2 \Rightarrow 1) Suppose $\psi(u) \stackrel{trees}{\equiv} ST(\varphi)(u)$. Using Proposition 4, we can see that $ST(\varphi)(u)$ is $\equiv^{\mathcal{R}}$ -invariant. Indeed, if $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$, then, by Proposition 4, $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$; also if $\mathcal{T} \models ST(\varphi)[u \mapsto x]$, because $ST(-)$ is truth-preserving then $(\mathcal{T}, x) \models \varphi$. Hence, since $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$, we have that $(\mathcal{T}', x') \models \varphi$ which is equivalent to $\mathcal{T}' \models ST(\varphi)[u \mapsto x']$. So $\psi(u)$ is indeed $\equiv^{\mathcal{R}}$ -invariant.

So far, we have considered that \mathcal{R} had no fixed semantics on the universe, but for many cases it is reasonable to consider some restrictions, such that a particular binary relation is an equivalence relation, which is a reasonable assumption if we want to talk about \mathcal{R} -trees as a generalization of data trees.

Let \mathbb{A} be a subset of unary symbols from \mathcal{R} , and let $X, S,$ and T be three subsets of binary symbols from \mathcal{R} . We denote by $\mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$ the class of \mathcal{R} -trees \mathcal{T} satisfying that for every node $x \in T$ there is a unique symbol $a \in \mathbb{A}$ such that $x \in R_a$, and for all $r \in \mathcal{R}$ the relation R_r is reflexive if $r \in X$, symmetric if $r \in S$ and transitive if $r \in T$. It can be seen that each formula $\psi(u)$ in $FO(\sigma_{\mathcal{R}})$ that is invariant by bisimulation between structures in $\mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$ will be semantically equivalent (relative to $\mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$) to the translation of a node expression in $XPath_{\mathcal{R}}(\downarrow)$. The main reason is that the \mathcal{R} -trees \mathcal{T}_A and \mathcal{T}_B constructed in the proof of Lemma 2 are now in the class $\mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$, since each binary \tilde{R}_r inherits good properties from R_r as reflexivity, symmetry and transitivity (for the fresh nodes t_A and t_B , we can freely assign them to any single unary set R_a with $a \in \mathbb{A}$). The invariance of $\psi(u)$ between structures in this new class allows us to finish the proof. Thus, we can relativize van Benthem’s characterization to $\mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$. This gives us a generalization to the van Benthem’s characterization for $XPath_{=}(\downarrow)$, where the symbols in \mathbb{A} are the labels and there is a symbol $=_d \in X \cap S \cap T$ whose semantic is the pair of nodes with same data value. Therefore, we have:

Theorem 4. *Let $\psi(u)$ be a formula in $FO(\sigma_{\mathcal{R}})$ with one free variable u . The following statements are equivalent:*

1. *For all (\mathcal{T}, x) and (\mathcal{T}', x') with $\mathcal{T}, \mathcal{T}' \in \mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$, if $\mathcal{T} \models \psi[u \mapsto x]$ and $(\mathcal{T}, x) \equiv^{\mathcal{R}} (\mathcal{T}', x')$ then $\mathcal{T}' \models \psi[u \mapsto x']$*
2. *There exists a node expression $\varphi \in \text{XPath}_{\mathcal{R}}(\downarrow)$ such that for all (\mathcal{T}, x) with $\mathcal{T} \in \mathcal{U}_{\mathbb{A}, X, S, T}^{\mathcal{R}}$ it happens that $(\mathcal{T}, x) \models \psi(u)$ iff $(\mathcal{T}, x) \models ST(\varphi)(u)$.*

5 Conclusions and Future Work

We have provided (Definition 12) \mathcal{R} -bisimulation notions for generalizations of the logic XPath $_{=}$ (\downarrow) over the full universe of \mathcal{R} -trees. This notion coincides with that of [4] for an adequate \mathcal{R} (Remark 7). We have shown that \mathcal{R} -bisimulation coincides with \mathcal{R} -logical equivalence over finitely branching pointed trees (Theorem 2). We have also characterized XPath $_{\mathcal{R}}(\downarrow)$ as the fragment of first-order logic that is invariant by \mathcal{R} -bisimulations (Theorem 3). Finally, we have shown (Theorem 4) that our results also apply for universes of trees with some restricted semantics which contain the case of XPath $_{=}$ (\downarrow) over data trees.

In the future, we would like to further generalize the work done in this paper, for example allowing other navigational modalities (such as sibling operators) for trees.

Acknowledgments. We thank Román Sasyk (Departamento de Matemática, Universidad de Buenos Aires) for his helpful questions and observations which spurred the main topic of this work.

A Proofs and Definitions Omitted from the Main Text

Lemma 1. For every $\ell \geq 0$, there are finitely many equivalence classes (modulo semantic equivalence) of node expressions with downward depth at most ℓ .

Proof. Without loss of generality (by Proposition 1), we assume that every node expression φ is normal, and even more, of the form $\varphi = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$ for $* \in \mathcal{R} \cup \overline{\mathcal{R}}$, because every normal expression is a Boolean combination of these. Therefore, if the number of equivalence classes is finite for this type of node expressions, it will also be finite for their Boolean combinations.

The proof goes by induction over $\ell \geq 0$.

$\ell = 0$ For this case, as $\text{dd}(\varphi) = 0$ and φ is normal, necessarily $\varphi = \langle \varepsilon, \varepsilon, \dots, \varepsilon \rangle_*$ with $* \in \mathcal{R} \cup \overline{\mathcal{R}}$ (notice that we can not have path tests like, for instance, $\langle \{[\varepsilon]\}, \dots, \{[\varepsilon]\} \rangle_*$ because $\{[\varepsilon]\}$ is not direct). Since \mathcal{R} is finite, then there are finitely many φ .

$\ell > 0$ $\varphi = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$ for $* \in \mathcal{R} \cup \overline{\mathcal{R}}$ and $0 < \text{dd}(\varphi) \leq \ell$. As the α_i are direct (since φ is normal), each node test in one of the α_i has downward depth at most $\ell - 1$. By induction hypothesis, there are finitely many direct α_i modulo semantic equivalence, and, since \mathcal{R} is finite, also there are finitely many φ modulo semantic equivalence.

Corollary 1. For each $\ell \geq 0$ and a pointed model (\mathcal{T}, x) there is a node expression $\chi_{(\mathcal{T}, x)}^\ell \in \text{XPath}_{\mathcal{R}}^\ell(\downarrow)$ satisfying that for any pointed model (\mathcal{T}', x') , $(\mathcal{T}, x) \equiv_{\mathcal{R}}^\ell (\mathcal{T}', x')$ iff $(\mathcal{T}', x') \models \chi_{(\mathcal{T}, x)}^\ell$.

Proof. Let $\chi_{(\mathcal{T}, x)}^\ell$ be the conjunction of all the node expressions modulo equivalence $\varphi \in \text{XPath}_{\mathcal{R}}^\ell(\downarrow)$ that (\mathcal{T}, x) satisfies. Lemma 1 ensures that $\chi_{(\mathcal{T}, x)}^\ell$ is well-defined because it is a conjunction of finitely many node expressions (modulo semantic equivalence), and clearly, it satisfies the desired property.

Remark 10. One can redefine some satisfiability notions with the Definition 15:

1. $(\mathcal{T}, x, y) \models \alpha$, if and only if, there exists $\mu \in \text{Path}(\mathcal{T}, x)$ with $\text{end}(\mu) = y$ and $(\mathcal{T}, \mu) \models \alpha$.
2. If $*$ $\in \mathcal{R} \cup \overline{\mathcal{R}}$ then $(\mathcal{T}, x) \models \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$, if and only if, for all $j \in \{1, 2, \dots, n\}$ there are paths $\mu_j \in \text{Path}(\mathcal{T}, x)$ such that $(\mathcal{T}, \mu_j) \models \alpha_j$ and also $R_*(\text{end}(\mu_1), \dots, \text{end}(\mu_n))$

The following definition is a more concrete form of the satisfiability notion for path expressions in an \mathcal{R} -tree. It will be used in Lemma 4 and simplify its proof.

Definition 15. Given a path $\mu \in \text{Path}(\mathcal{T}, x)$, we define inductively the meaning of $(\mathcal{T}, \mu) \models \gamma$ for a path expression γ .

$$\begin{aligned} (\mathcal{T}, \mu) \models \varepsilon &\stackrel{\text{def}}{\iff} \text{len}(\mu) = 0 \\ (\mathcal{T}, \mu) \models [\varphi] &\stackrel{\text{def}}{\iff} \mu = x \text{ and } (\mathcal{T}, x) \models \varphi \\ (\mathcal{T}, \mu) \models \downarrow &\stackrel{\text{def}}{\iff} \text{len}(\mu) = 1 \\ (\mathcal{T}, \mu) \models \alpha\beta &\stackrel{\text{def}}{\iff} \text{there is a decomposition } \mu = \mu_\alpha \odot \mu_\beta \\ &\text{such that } (\mathcal{T}, \mu_\alpha) \models \alpha \text{ and } (\mathcal{T}, \mu_\beta) \models \beta \end{aligned}$$

Lemma 4. Let $\mathcal{Z} = \{Z_i\}_{0 \leq i \leq \ell}$ be a \mathcal{R}_ℓ -bisimulation between \mathcal{T} and \mathcal{T}' . For all $k \in \{0, 1, \dots, \ell\}$, any normal expression η with $\text{dd}(\eta) \leq k$, and paths $\mu \in \text{Path}_k(\mathcal{T}, x)$ and $\mu' \in \text{Path}_k(\mathcal{T}', x')$ such that $([\mu]_i, [\mu']_i) \in Z_{k-i}$ for all $i \in \{0, 1, \dots, k\}$, we have that:

1. If η is a node expression then $(\mathcal{T}, x) \models \eta$, if and only if, $(\mathcal{T}', x') \models \eta$.
2. If η is a path expression then $(\mathcal{T}, \mu) \models \eta$, if and only if, $(\mathcal{T}', \mu') \models \eta$.

Proof. The proof is by induction over $k \in \{0, 1, \dots, \ell\}$.

$k = 0$ For this case, note that $\mu = x, \mu' = x'$ and if η is a path expression then $\eta = \varepsilon$. So, the double implication at 2 is obvious. Let us see when η is a node expression. Necessarily $\eta = \langle \varepsilon, \varepsilon, \dots, \varepsilon \rangle_*$ with $*$ $\in \mathcal{R} \cup \overline{\mathcal{R}}$, and since $(x, x') \in Z_0$ we have that for all $*$ $\in \mathcal{R} \cup \overline{\mathcal{R}}$, $R_*(x, x, \dots, x)$ iff $R_*(x', x', \dots, x')$. It is the same that $(\mathcal{T}, x) \models \langle \varepsilon, \varepsilon, \dots, \varepsilon \rangle_*$, if and only if, $(\mathcal{T}', x') \models \langle \varepsilon, \varepsilon, \dots, \varepsilon \rangle_*$.

$k > 0$ Suppose η is a path expression with $0 < \text{dd}(\eta) \leq k$ then $\eta = \downarrow \xi \gamma$ where ξ is an empty string or a node test, and γ is an empty string or a normal path expression. Note that $\text{dd}(\xi \gamma) \leq k - 1$. Let us see the most interesting case when ξ and γ are not the empty string, so $\xi = [\varphi]$ for some normal node expression φ . Suppose $\mu = (x \downarrow y) \odot \nu$ and $\mu' = (x' \downarrow y') \odot \nu'$ with $\nu \in \text{Path}_{k-1}(\mathcal{T}, y)$ and $\nu' \in \text{Path}_{k-1}(\mathcal{T}', y')$. If $(\mathcal{T}, \mu) \models \eta$ then $(\mathcal{T}, y) \models \varphi$ and $(\mathcal{T}, \nu) \models \gamma$. By induction hypothesis, $(\mathcal{T}', y') \models \varphi$ and $(\mathcal{T}', \nu') \models \gamma$, and therefore $(\mathcal{T}', \mu') \models \eta$. Similarly, we can see that if $(\mathcal{T}', \mu') \models \eta$ then $(\mathcal{T}, \mu) \models \eta$.

Suppose now η is a node expression and for simplicity, of the form $\eta = \langle \downarrow [\varphi_1] \alpha_1, \downarrow [\varphi_2] \alpha_2 \rangle_*$ where $*$ $\in \mathcal{R} \cup \overline{\mathcal{R}}$, φ_1, φ_2 are normal node expressions and α_1, α_2 are normal path expressions non-equal to ε (the rest of the cases are proved by the same idea). Let x be a node in \mathcal{T} and x' a node in \mathcal{T}' such that $(x, x') \in Z_k$. If $(\mathcal{T}, x) \models \eta$ then there are paths $\mu_1, \mu_2 \in \text{Path}_k(\mathcal{T}, x)$ such that $(\mathcal{T}, \mu_1) \models \downarrow [\varphi_1] \alpha_1$, $(\mathcal{T}, \mu_2) \models \downarrow [\varphi_2] \alpha_2$ and $R_*(\text{end}(\mu_1), \text{end}(\mu_2))$. Since $(x, x') \in Z_k$, by Zig, for the paths μ_1 and μ_2 there are another paths $\mu'_1, \mu'_2 \in \text{Path}_k(\mathcal{T}', x')$ such that for all $j = 1, 2$ we have that $\text{len}(\mu_j) = \text{len}(\mu'_j)$, if $i \in \{0, 1, \dots, \text{len}(\mu_j)\}$ then $([\mu_j]_i, [\mu'_j]_i) \in Z_{k-i}$, and $R_*(\text{end}(\mu_1), \text{end}(\mu_2))$ iff $R'_*(\text{end}(\mu'_1), \text{end}(\mu'_2))$. As $\varphi_1, \varphi_2, \alpha_1, \alpha_2$ have downward depth at most $k - 1$, we can apply induction hypothesis and conclude that for all $j \in \{1, 2\}$ if $\mu_j = (x \downarrow y_j) \odot \nu_j$ for some $\nu_j \in \text{Path}_{k-1}(\mathcal{T}, y_j)$ and $\mu'_j = (x' \downarrow y'_j) \odot \nu'_j$ for some $\nu'_j \in \text{Path}_{k-1}(\mathcal{T}', y'_j)$ then $(\mathcal{T}, y_j) \models \varphi_j$ iff $(\mathcal{T}', y'_j) \models \varphi_j$, and $(\mathcal{T}, \nu_j) \models \alpha_j$ iff $(\mathcal{T}', \nu'_j) \models \alpha_j$. Thus, $(\mathcal{T}', x') \models \eta$. Similarly (by Zag), $(\mathcal{T}', x') \models \eta$ implies $(\mathcal{T}, x) \models \eta$.

Proposition 2. Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that if $(\mathcal{T}, x) \rightleftharpoons_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ then $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$.

Proof. Suppose $(\mathcal{T}, x) \rightleftharpoons_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ via $\mathcal{Z} = \{Z_0, Z_1, \dots, Z_{\ell}\}$ with $(x, x') \in Z_{\ell}$. We want to see that for all $\varphi \in \text{XPath}_{\mathcal{R}}^{\ell}(\downarrow)$ we have that $(\mathcal{T}, x) \models \varphi$ iff $(\mathcal{T}', x') \models \varphi$. It suffices to show this for all normal path test $\varphi = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$ with $*$ $\in \mathcal{R} \cup \overline{\mathcal{R}}$: if $(\mathcal{T}, x) \models \varphi$ then there are paths $\mu_1, \mu_2, \dots, \mu_n \in \text{Path}_{\ell}(\mathcal{T}, x)$ such that $(\mathcal{T}, \mu_i) \models \alpha_i$ for all $i \in \{1, 2, \dots, n\}$ and $R_*(\text{end}(\mu_1), \text{end}(\mu_2), \dots, \text{end}(\mu_n)))$. Because $(x, x') \in Z_{\ell}$, there are paths $\mu'_1, \mu'_2, \dots, \mu'_n \in \text{Path}_{\ell}(\mathcal{T}', x')$ satisfying Zig with respect to the paths $\mu_1, \mu_2, \dots, \mu_n$. Thus, by Lemma 4, if $i \in \{1, 2, \dots, n\}$ then $(\mathcal{T}, \mu_i) \models \alpha_i$ iff $(\mathcal{T}', \mu'_i) \models \alpha_i$. Since $R_*(\text{end}(\mu_1), \text{end}(\mu_2), \dots, \text{end}(\mu_n)))$, if and only if, $R'_*(\text{end}(\mu'_1), \text{end}(\mu'_2), \dots, \text{end}(\mu'_n)))$, we conclude that $(\mathcal{T}', x') \models \varphi$ via the paths $\mu'_1, \mu'_2, \dots, \mu'_n$. Similarly, by Zag, if $(\mathcal{T}', x') \models \varphi$ then $(\mathcal{T}, x) \models \varphi$.

Proposition 3. Given (\mathcal{T}, x) and (\mathcal{T}', x') we have that if $(\mathcal{T}, x) \equiv_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ then $(\mathcal{T}, x) \rightleftharpoons_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$.

Proof. First we define for all $k \in \{0, 1, \dots, \ell\}$ the relations $(z, z') \in Z_k$ iff $(\mathcal{T}, z) \equiv_k^{\mathcal{R}} (\mathcal{T}', z')$. We want to see that the family $\mathcal{Z} := \{Z_0, Z_1, \dots, Z_{\ell}\}$ is a \mathcal{R}_{ℓ} -bisimulation, and since by hypothesis $(x, x') \in Z_{\ell}$, we will have that $(\mathcal{T}, x) \rightleftharpoons_{\ell}^{\mathcal{R}} (\mathcal{T}', x')$ via \mathcal{Z} .

Given $Z_k \in \mathcal{Z}$ and $(z, z') \in Z_k$, we want to show that the Zig and Zag clauses are satisfied. Let us see only Zag (for Zig, the same idea works).

Let $(\mu'_1, \mu'_2, \dots, \mu'_n)$ be a sequence of paths in $\text{Path}_k(\mathcal{T}, z')$ and $* \in \mathcal{R} \cup \bar{\mathcal{R}}$ such that $R'_*(\text{end}(\mu'_1), \dots, \text{end}(\mu'_n))$, we consider the set

$$P := \{(\mu_1, \mu_2, \dots, \mu_n) \in \text{Path}_k(\mathcal{T}, z)^n \mid \forall i \in \{1, 2, \dots, n\} \\ \text{len}(\mu_i) = \text{len}(\mu'_i) \wedge R_*(\text{end}(\mu_1), \dots, \text{end}(\mu_n))\}$$

Note that $P \neq \emptyset$ because since $(z, z') \in Z_k$

$$(\mathcal{T}', z') \models \langle \downarrow^{\text{len}(\mu'_1)}, \downarrow^{\text{len}(\mu'_2)}, \dots, \downarrow^{\text{len}(\mu'_n)} \rangle_* \in \text{XPath}_{\mathcal{R}}^k(\downarrow) \\ \Leftrightarrow (\mathcal{T}, z) \models \langle \downarrow^{\text{len}(\mu'_1)}, \downarrow^{\text{len}(\mu'_2)}, \dots, \downarrow^{\text{len}(\mu'_n)} \rangle_*$$

where \downarrow^N is an abbreviation for the concatenation of N symbols \downarrow , and for convenience $\downarrow^0 := \varepsilon$.

Suppose Zag does not hold for the paths $\mu'_1, \mu'_2, \dots, \mu'_n$. Thus, we have that for all $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_n) \in P$ there must be some μ_j and some $[\mu_j]_i$ such that $([\mu_j]_i, [\mu'_j]_i) \notin Z_{k-i}$.

1. Fix $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_n) \in P$. We define a family of node expressions $\{\varphi_{\bar{\mu}}^{(j,i)}\}$ as follows, where $j \in \{1, 2, \dots, n\}$ and $i \in \{0, 1, \dots, \text{len}(\mu'_j)\}$: if (j_0, i_0) is the smallest pair (by the lexicographic order) with $([\mu_{j_0}]_{i_0}, [\mu'_{j_0}]_{i_0}) \notin Z_{k-i_0}$ then there exists some node expression $\psi \in \text{XPath}_{\mathcal{R}}^{k-i_0}(\downarrow)$ such that $(\mathcal{T}, [\mu_{j_0}]_{i_0}) \not\models \psi$ and $(\mathcal{T}', [\mu'_{j_0}]_{i_0}) \models \psi$. So, let $\varphi_{\bar{\mu}}^{(j_0, i_0)}$ be equal to such ψ , and for the rest of the pairs (j, i) let $\varphi_{\bar{\mu}}^{(j,i)} := \top$ where \top is some fixed tautological node expression with downward depth zero (such as that from Remark 1).
2. Now, for each $j \in \{1, 2, \dots, n\}$ and $i \in \{0, 1, \dots, \text{len}(\mu'_j)\}$ let $\Phi^{(j,i)}$ be a formula such that for every $(\tilde{\mathcal{T}}, \tilde{x})$, $(\tilde{\mathcal{T}}, \tilde{x}) \models \Phi^{(j,i)}$ iff for all $\varphi_{\bar{\mu}}^{(j,i)}$ with $\bar{\mu} \in P$ we have $(\tilde{\mathcal{T}}, \tilde{x}) \models \varphi_{\bar{\mu}}^{(j,i)}$ (informally, abusing the notation in the case that P is infinite $\Phi^{(j,i)} \equiv \bigwedge_{\bar{\mu} \in P} \varphi_{\bar{\mu}}^{(j,i)}$). This formula exists because the expressions $\varphi_{\bar{\mu}}^{(j,i)}$ have downward depth at most $k - i$, and thus by Lemma 1 they are finite modulo equivalence. Thus $\Phi^{(j,i)}$ can be defined as the conjunction of some witnesses from each of the (finite) equivalence classes.
3. Finally, for each $j \in \{1, 2, \dots, n\}$ consider

$$\alpha_j := [\Phi^{(j,0)}] \downarrow [\Phi^{(j,1)}] \downarrow \dots \downarrow [\Phi^{(j, \text{len}(\mu'_j))}]$$

By construction, $\text{dd}(\alpha_j) \leq k$ and $(\mathcal{T}', \mu'_j) \models \alpha_j$. Therefore, $(\mathcal{T}', x') \models \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$ and since $(\mathcal{T}, x) \equiv_{\mathcal{R}}^k (\mathcal{T}', x')$, it must be that $(\mathcal{T}, x) \models \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_*$. But, this means that there are paths $\mu_1, \dots, \mu_n \in \text{Path}_k(\mathcal{T}, x)$ satisfying $(\mathcal{T}, \mu_j) \models \alpha_j$ and $R_*(\text{end}(\mu_1), \dots, \text{end}(\mu_n))$. As $\mu := (\mu_1, \dots, \mu_n) \in P$, there must be some μ_j with $(\mathcal{T}, \mu_j) \not\models \alpha_j$. So, we obtain a contradiction by assuming that Zag is not satisfied.

Definition 16. Let $\sigma_{\mathcal{R}} := \{F_{\downarrow}\} \cup \{L_r\}_{r \in \mathcal{R}}$ be a signature where F_{\downarrow} is binary and for each $r \in \mathcal{R}$ the symbol L_r is $\mathcal{A}(r)$ -ary. We interpret the symbols in $\sigma_{\mathcal{R}}$ in an \mathcal{R} -tree $\mathcal{T} = \langle T, E_{\downarrow}, \{R_r\}_{r \in \mathcal{R}} \rangle$ as $\llbracket F_{\downarrow} \rrbracket_{\mathcal{T}} := E_{\downarrow}$ and $\llbracket L_r \rrbracket_{\mathcal{T}} := R_r$ for $r \in \mathcal{R}$.

Definition 17. If $FO(\sigma_{\mathcal{R}})$ is the set of first order formulas with signature $\sigma_{\mathcal{R}}$ and equality, we define recursively the truth-preserving **standard translation** $ST(-)$ from $XPath_{\mathcal{R}}(\downarrow)$ to $FO(\sigma_{\mathcal{R}})$ which maps node expressions to formulas with one free variable and path expression to formulas with two free variables.

$$\begin{aligned}
 ST(\varphi \wedge \psi)(u) &:= ST(\varphi)(u) \wedge ST(\psi)(u) & ST(\neg\varphi)(u) &:= \neg ST(\varphi)(u) \\
 ST(\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_r)(u) &:= (\exists v_1, \dots, v_n) (\bigwedge_{i=1}^n ST(\alpha_i)(u, v_i) \wedge L_r(v_1, \dots, v_n)) \\
 &\text{for } r \in \mathcal{R}, \text{ with } \mathcal{A}(r) = n \\
 ST(\langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_{\bar{r}})(u) &:= (\exists v_1, \dots, v_n) (\bigwedge_{i=1}^n ST(\alpha_i)(u, v_i) \wedge \neg L_r(v_1, \dots, v_n)) \\
 &\text{for } r \in \mathcal{R}, \text{ with } \mathcal{A}(r) = n \\
 ST(\varepsilon)(u, v) &:= (u = v) & ST(\downarrow)(u, v) &:= F_{\downarrow}(u, v) \\
 ST([\varphi])(u, v) &:= (u = v) \wedge ST(\varphi)(u) \\
 ST(\alpha\beta)(u, v) &:= (\exists w)(ST(\alpha)(u, w) \wedge ST(\beta)(w, v))
 \end{aligned}$$

Lemma 2. For each $\equiv^{\mathcal{R}}$ -invariant $\psi(u) \in FO(\sigma_{\mathcal{R}})$ with one free variable u , there is $\ell \geq 0$ such that ψ is ℓ -local.

Proof. We follow the Step 1 of the proof of van Benthem/Rosen’s theorem for the modal logic given by Martin Otto in [8].

Let $\ell := 2^q - 1 \geq 0$ where q is the quantifier rank of $\psi(u)$. We want to show that for an arbitrary pointed model (\mathcal{T}, x) , $\mathcal{T} \models \psi[u \mapsto x]$ iff $\mathcal{T}|_{\ell}^x \models \psi[u \mapsto x]$, and hence, $\psi(u)$ is ℓ -local.

Given a pointed model (\mathcal{T}, x) we define two new \mathcal{R} -trees \mathcal{T}_A and \mathcal{T}_B as follows:

- The underlying tree of \mathcal{T}_A is constructed by tying from a fresh node t_A : the original (\mathcal{T}, x) , q -copies of (\mathcal{T}, x) via a family \mathcal{I} of q isomorphisms (as $\sigma_{\mathcal{R}}$ -structures) from \mathcal{T} , and q -copies of $(\mathcal{T}|_{\ell}^x, x)$ via a family \mathcal{J} of q isomorphisms from $\mathcal{T}|_{\ell}^x$. For each $r \in \mathcal{R}$ of arity n , the relation \tilde{R}_r in \mathcal{T}_A is defined by extending the relation R_r in \mathcal{T} as $\tilde{R}_r(y_1, y_2, \dots, y_n)$ iff $R_r(x_1, x_2, \dots, x_n)$ where for all $k \in \{1, 2, \dots, q\}$ there exists $f \in \mathcal{I} \cup \mathcal{J}$ such that $f(x_k) = y_k$. Notice that t_A doesn’t play a relevant role because it is not related to any of the nodes, even itself.
- \mathcal{T}_B and t_B are constructed almost exactly like \mathcal{T}_A and t_A , but now we replace the original (\mathcal{T}, x) with $(\mathcal{T}|_{\ell}^x, x)$.

Clearly, $(\mathcal{T}_A, x) \equiv^{\mathcal{R}} (\mathcal{T}, x)$ and $(\mathcal{T}_B, x) \equiv^{\mathcal{R}} (\mathcal{T}|_{\ell}^x, x)$. Suppose $\mathcal{T} \models \psi[u \mapsto x]$, and since $\psi(u)$ is $\equiv^{\mathcal{R}}$ -invariant we have that $\mathcal{T}_A \models \psi[u \mapsto x]$. We affirm that $(\mathcal{T}_A, x) \equiv_q (\mathcal{T}_B, x)$ as $\sigma_{\mathcal{R}}$ -structures, i.e., they satisfy the same formulas with quantifier rank less than or equal to q . To see that, one can follow the idea given by Otto showing a winning strategy for player **II** in a Ehrenfeucht-Fraïssé game.

Therefore, as $\psi(u)$ has quantifier rank q , $\mathcal{T}_B \models \psi[u \mapsto x]$, and then $\mathcal{T}|_{\ell}^x \models \psi[u \mapsto x]$.

References

1. Clark, J., DeRose, S.: XML path language (XPath). Website (1999). W3C Recommendation
2. Gottlob, G., Koch, C., Pichler, R.: Efficient algorithms for processing XPath queries. *TODS* **30**, 444–491 (2005)
3. Bojańczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. *JACM* **56**, 1–48 (2009)
4. Figueira, D., Figueira, S., Areces, C.: Model theory of XPath on data trees. Part I: bisimulation and characterization. *JAIR* **53**, 271–314 (2015)
5. Abriola, S., Descotte, M.E., Figueira, S.: Model theory of XPath on data trees. Part II: binary bisimulation and definability. *Inf. Comput.* **255**, 195–223 (2017)
6. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press, Cambridge (2001)
7. van Benthem, J.: *Modal correspondence theory*. Ph.D. thesis, Universiteit van Amsterdam (1976)
8. Otto, M.: *Elementary proof of the van Benthem-Rosen characterisation theorem*. Technical report 2342, Fachbereich Mathematik, Technische Universität Darmstadt (2004)



Uniform Interpolation via Nested Sequents

Iris van der Giessen¹, Raheleh Jalali¹, and Roman Kuznets²(✉)

¹ Utrecht University, Utrecht, The Netherlands
{i.vandergiessen,r.jalalikesavarz}@uu.nl

² TU Wien, Vienna, Austria
roman@logic.at

Abstract. A modular proof-theoretic framework was recently developed to prove Craig interpolation for normal modal logics based on generalizations of sequent calculi (e.g., nested sequents, hypersequents, and labelled sequents). In this paper, we turn to uniform interpolation, which is stronger than Craig interpolation. We develop a constructive method for proving uniform interpolation via nested sequents and apply it to reprove the uniform interpolation property for normal modal logics K, D, and T. While our method is proof-theoretic, the definition of uniform interpolation for nested sequents also uses semantic notions, including bisimulation modulo an atomic proposition.

Keywords: Uniform interpolation · Modal logic · Nested sequents

1 Introduction

A propositional (modal) logic L admits the Craig interpolation property (CIP) if for any formulas φ and ψ such that $\vdash_L \varphi \rightarrow \psi$, there is an interpolant θ containing only atomic propositions that occur in both φ and ψ such that $\vdash_L \varphi \rightarrow \theta$ and $\vdash_L \theta \rightarrow \psi$. Logic L has the uniform interpolation property (UIP) if for each formula φ and each atomic proposition p there are uniform interpolants $\exists p\varphi$ and $\forall p\varphi$ built from atomic propositions occurring in φ except for p , such that for all formulas ψ not containing p :

$$\vdash_L \varphi \rightarrow \psi \Leftrightarrow \vdash_L \exists p\varphi \rightarrow \psi \quad \text{and} \quad \vdash_L \psi \rightarrow \varphi \Leftrightarrow \vdash_L \psi \rightarrow \forall p\varphi.$$

It is well known that this property is stronger than Craig interpolation.

To prove the CIP (UIP) constructively, one can use analytic (terminating) sequent calculi. Whereas for the CIP the syntactic proofs are often straightforward, the case of the UIP is more complicated. Pitts provided the first syntactic proof of this kind, establishing the UIP for IPC [19]. Bílková successfully adjusted the method to (re)prove the UIP for several modal logics including K, T, and GL [2]. Iemhoff provided a modular method for (intuitionistic)

Iris van der Giessen and Raheleh Jalali acknowledge the support of the Netherlands Organization for Scientific Research under grant 639.073.807. Roman Kuznets is funded by the Austrian Science Fund (FWF) ByzDEL project (P33600).

modal logics and intermediate logics based on sequent calculi consisting of the so-called focused rules, among others establishing the UIP for D [12, 13].

There are also algebraic and model-theoretic methods. The UIP for GL and K is due to Shavrukov [21] and Ghilardi [8] respectively. Interestingly, modal logics S4 and K4 do not enjoy the UIP [2, 9] despite enjoying the CIP. Visser provided semantic proofs for K, GL, and IPC based on bounded bisimulation up to atomic propositions [24]. This method was later used for the stronger Lyndon UIP [14]. The semantic interpretation of uniform interpolation is called bisimulation quantifiers, see [6] for an overview. Bisimulation will also play a role in this paper.

The proof-theoretic approach has two advantages. First, it enables one to find interpolants constructively rather than merely prove their existence.¹ Second, negative results were obtained in [12, 13] stating that logics without the UIP cannot have certain natural sequent calculi. As a consequence, K4 and S4 do not possess such proof systems. Similar negative results were obtained for modal and substructural logics in [22] and [23] using the CIP and UIP.

The goal of this paper is to extend the same line of research to multisequent formalisms starting with nested sequents². Multisequent formalisms, such as nested sequents, hypersequents, and labelled sequents, are (commonly believed to be) more expressive than sequents and offer modular and analytic calculi for a wide range of logics. E.g., S5 has well-known cut-free hypersequent calculi [1, 17] but no known cut-free sequent calculus while modal logics K5 and B possess cut-free nested sequent calculi, but no hypersequent calculi [5]. Nested sequent calculi were recently used to prove the CIP for modal logics [7]. A modular proof-theoretic framework encompassing them and other multisequents was provided in [15]. The same ideas, which combine syntactic and semantic reasoning, were extended to multisequent calculi for intermediate logics [16].

We provide a method to prove the UIP for K, D, and T using terminating nested sequent calculi from [5]. These calculi are used to construct uniform interpolants syntactically, whereas the correctness proof for the constructed interpolants relies on semantic reasoning, including model modifications and bisimulation. While the UIP for these three logics has been previously shown via analytic sequent calculi, our constructive method is also applicable to logics based on multisequent formalisms that lack a sequent representation. In particular, in an extended version [10] of this paper, we successfully adapted our method to hypersequents for S5.

Bílková [3] also provided a syntactic proof for the UIP for K based on nested sequents. The main difference with our method is that we exploit the treelike structure of nested sequents, thus reflecting the treelike models for K, by using semantic reasoning while the algorithm for the interpolants remains fully syntactic.

The paper is organized as follows. In Sect. 2 the nested sequent calculi for K, T, and D, as well as model modifications invariant under bisimulation, are

¹ More precisely, it enables one to find interpolants efficiently rather than by an exhaustive search that terminates due to the existence of the interpolant.

² Nested sequents are also known as tree-hypersequents [20] or deep sequents [5].

introduced. In Sect. 3, we prove uniform interpolation for K , T , and D . Finally, in Sect. 4 we summarize the results and outline future work. An extended version [10] of this paper provides more detailed proofs of these results and, in addition, includes a direct proof of the UIP for $S5$ via hypersequents.

2 Preliminaries

Definition 1. Modal formulas *in negation normal form* are defined by the grammar $\varphi ::= \perp \mid \top \mid p \mid \bar{p} \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \Box\varphi \mid \Diamond\varphi$ where \perp and \top are Boolean constants, p is an atomic proposition (atom) from a countable set \mathbf{Prop} , and \bar{p} is its negation. An element ℓ of the set \mathbf{Lit} of literals is either an atom or its negation. Literals and Boolean constants are atomic formulas.

We define $\bar{\varphi}$ (or $\neg\varphi$) recursively as usual using De Morgan’s laws to push the negation inwards. We define $\varphi \rightarrow \psi := \bar{\varphi} \vee \psi$ as usual.

Definition 2. A nested sequent Γ is recursively defined in the following form:

$$\varphi_1, \dots, \varphi_n, [\Gamma_1], \dots, [\Gamma_m]$$

where $\varphi_1, \dots, \varphi_n$ are modal formulas for $n \geq 0$ and $\Gamma_1, \dots, \Gamma_m$ are nested sequents for $m \geq 0$. We call brackets $[]$ a structural box. The formula interpretation ι of a nested sequent is defined recursively by

$$\iota(\varphi_1, \dots, \varphi_n, [\Gamma_1], \dots, [\Gamma_m]) := \varphi_1 \vee \dots \vee \varphi_n \vee \Box\iota(\Gamma_1) \vee \dots \vee \Box\iota(\Gamma_m).$$

One way of looking at a nested sequent is to consider a tree of ordinary (one-sided) sequents, i.e., multisets of formulas. Each structural box in the nested sequent creates a *child* in the tree. In order to be able to reason about formulas in a particular tree node, we introduce labels. A *label* is a finite sequence of natural numbers. We denote labels by σ, τ, \dots ; a label $\sigma * n$ denotes the label σ extended by the natural number n . We sometimes write σn instead of $\sigma * n$, unless it is ambiguous, as, e.g., for $1 * 2 * 3$, which is different from $1 * 23$.

Definition 3 (Labeling). For a nested sequent Γ and label σ we define a labeling function l_σ to recursively label structural boxes in nested sequents as follows:

$$l_\sigma(\varphi_1, \dots, \varphi_n, [\Gamma_1], \dots, [\Gamma_m]) := \varphi_1, \dots, \varphi_n, [l_{\sigma 1}(\Gamma_1)]_{\sigma 1}, \dots, [l_{\sigma m}(\Gamma_m)]_{\sigma m}.$$

Let $\mathcal{L}_\sigma(\Gamma)$ be the set of labels occurring in $l_\sigma(\Gamma)$ plus label σ (for formulas outside all structural boxes). Define $l(\Gamma) := l_1(\Gamma)$, and let $\mathcal{L}(\Gamma) := \mathcal{L}_1(\Gamma)$.³

Formulas in a nested sequent Γ are labeled according to the labeling of the structural boxes containing them. We write $1 : \varphi \in \Gamma$ iff the formula φ occurs in Γ outside all structural boxes. Otherwise, $\sigma : \varphi \in \Gamma$ whenever φ occurs in $l(\Gamma)$ within a structural box labeled σ .

³ Labeled nested sequents are closely related to labelled sequents from [18] but retain the nested notation.

$\text{id}_p \frac{}{\Gamma\{p, \bar{p}\}}$	$\text{id}_T \frac{}{\Gamma\{\top\}}$	$\vee \frac{\Gamma\{\varphi \vee \psi, \varphi, \psi\}}{\Gamma\{\varphi \vee \psi\}}$	$\wedge \frac{\Gamma\{\varphi \wedge \psi, \varphi\} \quad \Gamma\{\varphi \wedge \psi, \psi\}}{\Gamma\{\varphi \wedge \psi\}}$
$\Box \frac{\Gamma\{\Box\varphi, [\varphi]\}}{\Gamma\{\Box\varphi\}}$	$k \frac{\Gamma\{\Diamond\varphi, [\Delta, \varphi]\}}{\Gamma\{\Diamond\varphi, [\Delta]\}}$	$d \frac{\Gamma\{\Diamond\varphi, [\varphi]\}}{\Gamma\{\Diamond\varphi\}}$	$t \frac{\Gamma\{\Diamond\varphi, \varphi\}}{\Gamma\{\Diamond\varphi\}}$

Fig. 1. Terminating nested rules: the principal formula is not K- (D-, T-) saturated.

The set $\mathcal{L}(\Gamma)$ can be seen as the set of nodes of the corresponding tree of Γ , with 1 being the root. Often, we do not distinguish between a nested sequent Γ and its labeled sequent $l(\Gamma)$. For instance, we write $\sigma \in \Gamma$ if $\sigma \in \mathcal{L}(\Gamma)$.

Whether X is a formula, a sequence/set/multiset of formulas, a nested sequent/context, or some other formula-based object, we denote by $\text{Var}(X) \subseteq \text{Prop}$ the set of atoms occurring in X (note that p may also occur in the form of \bar{p}).

Recall that the normal modal logic **K** consists of all classical tautologies, the **k**-axiom $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$ and is closed under *modus ponens* (from $\varphi \rightarrow \psi$ and φ , infer ψ) and *necessitation* (from φ , infer $\Box\varphi$). Further, the modal logics **D** and **T** are defined as $\text{D} := \text{K} + \Box\varphi \rightarrow \Diamond\varphi$ and $\text{T} := \text{K} + \Box\varphi \rightarrow \varphi$.

The nested calculus **NK** for the modal logic **K** consists of the rules in the first row in Fig. 1 plus the rules \Box and **k**. This calculus is an extension of the multiset-based version from [5] to the language with Boolean constants \perp and \top , necessitating the addition of the rule id_T for handling these. The calculus **ND** (**NT**) for the logic **D** (**T**) is obtained by adding to **NK** the rule **d** (**t**). As shown in [5], the nested sequent calculi **NK**, **ND**, and **NT** are sound and complete for modal logics **K**, **D**, and **T** respectively.

Definition 4 (Saturation). Let $\Gamma = \Gamma'\{\theta\}_\sigma$, i.e., $\sigma : \theta \in \Gamma$. The formula θ is **K**-saturated in Γ if the following conditions hold based on the form of θ :

- θ is an atomic formula;
- if $\theta = \varphi \vee \psi$, then both $\sigma : \varphi \in \Gamma$ and $\sigma : \psi \in \Gamma$;
- if $\theta = \varphi \wedge \psi$, then either $\sigma : \varphi \in \Gamma$ or $\sigma : \psi \in \Gamma$;
- if $\theta = \Box\varphi$, then there is a label $\sigma n \in \mathcal{L}(\Gamma)$ such that $\sigma n : \varphi \in \Gamma$.

The formula $\theta = \Diamond\varphi$ is

- **K**-saturated in Γ w.r.t. $\sigma n \in \mathcal{L}(\Gamma)$ if $\sigma n : \varphi \in \Gamma$;
- **D**-saturated in Γ if there is some label $\sigma n \in \mathcal{L}(\Gamma)$;
- **T**-saturated in Γ if $\sigma : \varphi \in \Gamma$.

A nested sequent Γ is \mathbf{K} -saturated if (1) it is neither of the form $\Gamma'\{p, \bar{p}\}$ for some $p \in \mathbf{Prop}$ nor of the form $\Gamma'\{\top\}$; and (2) all its formulas $\sigma : \diamond\varphi$ are \mathbf{K} -saturated w.r.t. every child of σ ; and (3) all its other formulas are \mathbf{K} -saturated in Γ . A nested sequent is \mathbf{D} -saturated (\mathbf{T} -saturated) if it is \mathbf{K} -saturated and all its formulas $\sigma : \diamond\varphi$ are \mathbf{D} -saturated (\mathbf{T} -saturated) in Γ .

Theorem 5 ([5]). *The calculi \mathbf{NK} , \mathbf{ND} , and \mathbf{NT} in Fig. 1 are terminating.*

Definition 6. A Kripke model is a triple $\mathcal{M} = (W, R, V)$, where $W \neq \emptyset$, $R \subseteq W \times W$, and $V : \mathbf{Prop} \rightarrow 2^W$ is a valuation function. Define $\mathcal{M}, w \models \varphi$ as usual: $\mathcal{M}, w \models \top$ and $\mathcal{M}, w \not\models \perp$; for $p \in \mathbf{Prop}$, we have $\mathcal{M}, w \models p$ iff $w \in V(p)$ and $\mathcal{M}, w \models \bar{p}$ iff $w \notin V(p)$; we have $\mathcal{M}, w \models \varphi \wedge \psi$ ($\mathcal{M}, w \models \varphi \vee \psi$) iff $\mathcal{M}, w \models \varphi$ and (or) $\mathcal{M}, w \models \psi$; finally, $\mathcal{M}, w \models \Box\varphi$ iff $\mathcal{M}, v \models \varphi$ whenever wRv and $\mathcal{M}, w \models \diamond\varphi$ iff $\mathcal{M}, v \models \varphi$ for some wRv . A formula φ is valid in \mathcal{M} , denoted $\mathcal{M} \models \varphi$, when $\mathcal{M}, w \models \varphi$ for all $w \in W$.

A model $\mathcal{M}' = (W', R', V')$ is a submodel of $\mathcal{M} = (W, R, V)$ when $W' \subseteq W$, $R' = R \cap (W' \times W')$, and $V'(p) = V(p) \cap W'$ for each $p \in \mathbf{Prop}$. A submodel generated by $w \in W$, denoted $\mathcal{M}_w := (W_w, R_w, V_w)$, is the smallest submodel $\mathcal{M}' = (W', R', V')$ of \mathcal{M} such that $w \in W'$ and $v \in W'$ when xRv and $x \in W'$.

We will use models based on finite intransitive directed trees, usually denoting the root ρ . For \mathbf{K} , we require the accessibility relation R to be irreflexive, i.e., $\forall w \in W \neg(wRw)$. For \mathbf{T} , R is reflexive, i.e., $\forall w \in W wRw$. And for \mathbf{D} , R is serial, i.e., $\forall w \in W \exists v \in W wRv$. Note that seriality implies reflexivity of the leaves of the tree. We call these models \mathbf{K} -models, \mathbf{T} -models, and \mathbf{D} -models respectively.

Theorem 7 ([11, Sect. 4.20]). *If $\mathbf{L} \in \{\mathbf{K}, \mathbf{D}, \mathbf{T}\}$, then $\varphi \in \mathbf{L}$ iff $\mathcal{M} \models \varphi$ for each \mathbf{L} -model \mathcal{M} .*

Following [15], we extend definitions of truth and validity to nested sequents, recall relevant facts about bisimulation, and introduce some model modifications.

Definition 8. A (treelike) multiworld interpretation of a nested sequent Γ into a model $\mathcal{M} = (W, R, V)$ is a function $\mathcal{I} : \mathcal{L}(\Gamma) \rightarrow W$ from labels in Γ to worlds of \mathcal{M} such that $\mathcal{I}(\sigma)R\mathcal{I}(\sigma n)$ whenever $\{\sigma, \sigma n\} \subseteq \mathcal{L}(\Gamma)$. Then

$$\mathcal{M}, \mathcal{I} \models \Gamma \quad \iff \quad \mathcal{M}, \mathcal{I}(\sigma) \models \varphi \text{ for some } \sigma : \varphi \in \Gamma.$$

Γ is valid in \mathcal{M} , denoted by $\mathcal{M} \models \Gamma$, means that $\mathcal{M}, \mathcal{I} \models \Gamma$ for all multiworld interpretations \mathcal{I} of Γ into \mathcal{M} .

The following lemma, which can be easily proved by induction on the structure of Γ , implies completeness for validity of nested sequents.

Lemma 9. $\mathcal{M} \models \Gamma$ iff $\mathcal{M} \models \iota(\Gamma)$ for any nested sequent Γ and model \mathcal{M} .

We now define bisimulations modulo an atom p , similar to the ones from [6, 24], where uniform interpolation is studied on the basis of bisimulation quantifiers. While those papers focus on purely semantic methods, we embed the semantic tools of bisimulation into our constructive proof-theoretic approach in Sect. 3. Our bisimulations behave largely like standard bisimulations except they do not have to preserve the truth of formulas with occurrences of p .

Definition 10 (Bisimilarity). A bisimulation up to an atom p between models $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ is a non-empty relation $Z \subseteq W \times W'$ such that the following hold for all $w \in W$ and $w' \in W'$ with wZw' :

- atoms _{p} .** $w \in V(q)$ iff $w' \in V'(q)$ for all $q \in \text{Prop} \setminus \{p\}$;
- forth.** if wRv , then there exists $v' \in V'$ such that vZv' and $w'R'v'$; and
- back.** if $w'R'v'$, then there exists $v \in V$ such that vZv' and wRv .

When wZw' , we write $(\mathcal{M}, w) \sim_p (\mathcal{M}', w')$. Further, we write $(\mathcal{M}, \mathcal{I}) \sim_p (\mathcal{M}', \mathcal{I}')$ for $\mathcal{I} : X \rightarrow W$ and $\mathcal{I}' : X \rightarrow W'$ with a common domain X if there is a bisimulation Z up to p between \mathcal{M} and \mathcal{M}' such that $\mathcal{I}(\sigma)Z\mathcal{I}'(\sigma)$ for each $\sigma \in X$.

The main property of bisimulations is truth preservation for modal formulas. The following theorem is proved the same way as [4, Theorem 2.20].

Theorem 11. If $(\mathcal{M}, w) \sim_p (\mathcal{M}', w')$, then for all formulas φ with $p \notin \text{Var}(\varphi)$, we have $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}', w' \models \varphi$.

We are interested in manipulations of treelike models that preserve bisimulation up to p , in particular, in duplicating a part of a model or replacing it with a bisimilar model.

Definition 12 (Model transformations). Let $\mathcal{M} = (W, R, V)$ be an intransitive tree (possibly with some reflexive worlds), $\mathcal{M}_w = (W_w, R_w, V_w)$ be its subtree with root $w \in W$, and $\mathcal{N} = (W_N, R_N, V_N)$ be another tree with root $\rho_N \in W_N$. A model $\mathcal{M}' = (W', R', V')$ is the result of replacing the subtree \mathcal{M}_w with \mathcal{N} in \mathcal{M} if $W' := (W \setminus W_w) \sqcup W_N$, $V'(q) := (V(q) \setminus V_w(q)) \sqcup V_N(q)$ for all $q \in \text{Prop}$, and $R' := (R \cap (W \setminus W_w)^2) \sqcup R_N \sqcup \{(v, \rho_N) \mid vRw\}$.

A model $\mathcal{M}'' = (W'', R'', V'')$ is the result of duplicating (cloning) \mathcal{M}_w in \mathcal{M} if another copy⁴ $\mathcal{M}_w^c = (W_w^c, R_w^c, V_w^c)$ of \mathcal{M}_w is inserted alongside (as a subtree of) \mathcal{M}_w , i.e., if $W'' := W \sqcup W_w^c$, $V''(q) := V(q) \sqcup V_w^c(q)$ for all $q \in \text{Prop}$, and, in case of duplicating, $R'' := R \sqcup R_w^c \sqcup \{(v, w^c) \mid vRw\}$ (in case of cloning, $R'' := R \sqcup R_w^c \sqcup \{(w, w^c)\}$). Finally, for a reflexive world w , the result of unraveling \mathcal{M}_w in \mathcal{M} is obtained by first cloning \mathcal{M}_w in \mathcal{M} and then removing the reflexive loop (w, w) from the accessibility relation.

Lemma 13. In the setup from Definition 12, let $Z \subseteq W_N \times W_w$ be a bisimulation demonstrating that $(\mathcal{N}, \rho_N) \sim_p (\mathcal{M}_w, w)$. Then, for \mathcal{M}' obtained by

⁴ Here $v^c := (v, c)$, $W_w^c := \{v^c \mid v \in W_w\}$, $R_w^c := \{(v^c, u^c) \mid (v, u) \in R_w\}$, and $V_w^c(q) := \{v^c \mid v \in V_w(q)\}$.

replacing \mathcal{M}_w with \mathcal{N} in \mathcal{M} we have that $(\mathcal{M}', v) \sim_p (\mathcal{M}, v)$ for all $v \in W \setminus W_w$ and that $(\mathcal{M}', u_N) \sim_p (\mathcal{M}, u)$ whenever $u_N Z u$. Moreover, if both \mathcal{M} and \mathcal{N} are K-models (D-models, T-models), then so is \mathcal{M}' .

For \mathcal{M}'' obtained by duplicating \mathcal{M}_w in \mathcal{M} , we have $(\mathcal{M}'', v) \sim_p (\mathcal{M}, v)$ for all $v \in W$ and, in addition, $(\mathcal{M}'', u^c) \sim_p (\mathcal{M}, u)$ for all $u \in W_w$. If \mathcal{M} is a K-model (D-model, T-model) not rooted at w , so is \mathcal{M}'' . The same holds for cloning and unraveling if wRw except that unraveling does not preserve T-models.

Proof. It is easy to see that $Z' := \{(v, v) \mid v \in W \setminus W_w\} \sqcup Z$ for replacing or that $Z'' := \{(v, v) \mid v \in W\} \sqcup \{(u^c, u) \mid u \in W_w\}$ for duplicating, cloning, and unraveling witnesses all the stated bisimilarities in each respective case. Both the tree structure and reflexivity of worlds are preserved by all operations other than unraveling, which turns a reflexive w into an irreflexive world, violating T-model requirements. Seriality is preserved by all operations. \square

3 Uniform Interpolation for Nested Sequents

In this section we prove the UIP for K, T, and D via NK, NT, and ND. We define two new notions of UIPs for nested sequents that involve Kripke semantics: the nested-sequent UIP (NUIP) in Definition 22 that closely follows the structure of the UIP and the more convenient to use bisimulation NUIP (BNUIP) in Definition 25. Lemma 24 and Corollary 28 extract back the standard definition of the UIP.

Definition 14 (UIP). *A logic \mathbb{L} in a language containing an implication \rightarrow and Boolean constants \perp and \top (primary or defined) has the uniform interpolation property, or UIP, if for every formula φ in the logic and atom p , there exist formulas $\forall p\varphi$ and $\exists p\varphi$ such that*

- (i) $\text{Var}(\exists p\varphi) \subseteq \text{Var}(\varphi) \setminus \{p\}$ and $\text{Var}(\forall p\varphi) \subseteq \text{Var}(\varphi) \setminus \{p\}$,
- (ii) $\vdash_{\mathbb{L}} \varphi \rightarrow \exists p\varphi$ and $\vdash_{\mathbb{L}} \forall p\varphi \rightarrow \varphi$, and
- (iii) for each formula ψ with $p \notin \text{Var}(\psi)$:

$$\vdash_{\mathbb{L}} \varphi \rightarrow \psi \quad \Rightarrow \quad \vdash_{\mathbb{L}} \exists p\varphi \rightarrow \psi \quad \text{and} \quad \vdash_{\mathbb{L}} \psi \rightarrow \varphi \quad \Rightarrow \quad \vdash_{\mathbb{L}} \psi \rightarrow \forall p\varphi.$$

For classical-based logics, the existence of left-interpolants ensures the existence of right-interpolants, and vice versa (e.g., $\exists p\varphi := \neg \forall p\neg\varphi$). Thus, from now on, we focus on $\forall p\varphi$. In the following, we import some notation from [15].

Definition 15. *Multiformulas are defined by $\mathcal{U} ::= \sigma : \varphi \mid (\mathcal{U} \otimes \mathcal{U}) \mid (\mathcal{U} \oplus \mathcal{U})$, where σ is a label and φ is a formula. We write $\mathcal{L}(\mathcal{U})$ for the set of labels in \mathcal{U} .*

Definition 16 (Suitability). *A multiworld interpretation \mathcal{I} of a sequent Γ into a model \mathcal{M} is suitable for a multiformula \mathcal{U} if $\mathcal{L}(\mathcal{U}) \subseteq \mathcal{L}(\Gamma)$, in which case we call it a multiworld interpretation of \mathcal{U} into \mathcal{M} .*

Definition 17 (Truth for multiformulas). Let \mathcal{I} be a multiworld interpretation of a multiformula \mathcal{U} into a model \mathcal{M} . Define $\mathcal{M}, \mathcal{I} \models \mathcal{U}$ recursively as:

$$\begin{aligned} \mathcal{M}, \mathcal{I} \models \sigma : \varphi & \quad \text{iff} \quad \mathcal{M}, \mathcal{I}(\sigma) \models \varphi, \\ \mathcal{M}, \mathcal{I} \models \mathcal{U}_1 \otimes \mathcal{U}_2 & \quad \text{iff} \quad \mathcal{M}, \mathcal{I} \models \mathcal{U}_i \text{ for both } i = 1, 2, \\ \mathcal{M}, \mathcal{I} \models \mathcal{U}_1 \oplus \mathcal{U}_2 & \quad \text{iff} \quad \mathcal{M}, \mathcal{I} \models \mathcal{U}_i \text{ for at least one } i = 1, 2. \end{aligned}$$

Since $\mathcal{L}(\mathcal{U}_i) \subseteq \mathcal{L}(\mathcal{U})$, \mathcal{I} is also a multiworld interpretation of each \mathcal{U}_i into \mathcal{M} .

We define the label-erasing function from multiformulas to formulas, as well as multiformula equivalence and some of the latter's easily provable properties.

Definition 18. The label-erasing function from multiformulas to formulas is defined as follows: $\text{form}(\sigma : \varphi) := \varphi$, $\text{form}(\mathcal{U}_1 \otimes \mathcal{U}_2) := \text{form}(\mathcal{U}_1) \wedge \text{form}(\mathcal{U}_2)$, and $\text{form}(\mathcal{U}_1 \oplus \mathcal{U}_2) := \text{form}(\mathcal{U}_1) \vee \text{form}(\mathcal{U}_2)$.

Definition 19 (Multiformula equivalence). Multiformulas \mathcal{U}_1 and \mathcal{U}_2 are equivalent, denoted $\mathcal{U}_1 \equiv \mathcal{U}_2$, iff $\mathcal{L}(\mathcal{U}_1) = \mathcal{L}(\mathcal{U}_2)$ and $\mathcal{M}, \mathcal{I} \models \mathcal{U}_1 \Leftrightarrow \mathcal{M}, \mathcal{I} \models \mathcal{U}_2$ for any multiworld interpretation \mathcal{I} of \mathcal{U}_1 into a model \mathcal{M} .

Lemma 20 (Equivalence property). For any multiformula \mathcal{U} , label σ , and formulas φ and ψ , we have $\mathcal{U} \otimes \mathcal{U} \equiv \mathcal{U} \otimes \mathcal{U} \equiv \mathcal{U}$, and $\sigma : \varphi \otimes \sigma : \psi \equiv \sigma : (\varphi \wedge \psi)$, and $\sigma : \varphi \oplus \sigma : \psi \equiv \sigma : (\varphi \vee \psi)$.

Lemma 21 (Normal forms). For any multiformula \mathcal{U} , there is an equivalent multiformula \mathcal{U}^d (\mathcal{U}^c) in SDNF (SCNF) such that \mathcal{U}^d (\mathcal{U}^c) is a \otimes -disjunction (\otimes -conjunction) of \otimes -conjunctions (\otimes -disjunctions) of labeled formulas $\sigma : \varphi$ and each disjunct (conjunct) contains exactly one occurrence of each $\sigma \in \mathcal{L}(\mathcal{U})$.⁵

Proof. Since \otimes and \oplus behave classically, one can employ the standard transformation into the DNF/CNF. In order to ensure one label per disjunct/conjunct rule, multiple labels can be combined using Lemma 20, whereas missing labels can be added in the form of $\sigma : \perp$ ($\sigma : \top$). \square

We now introduce the uniform interpolation property for nested sequents. Here, the uniform interpolants are multiformulas instead of formulas.

Definition 22 (NUIP). Let a nested sequent calculus NL be sound and complete w.r.t. a logic L. We say that NL has the nested-sequent uniform interpolation property, or NUIP, if for each nested sequent Γ and atom p there exists a multiformula $A_p(\Gamma)$, called a nested uniform interpolant, such that

- (i) $\text{Var}(A_p(\Gamma)) \subseteq \text{Var}(\Gamma) \setminus \{p\}$ and $\mathcal{L}(A_p(\Gamma)) \subseteq \mathcal{L}(\Gamma)$,
- (ii) for each multiworld interpretation \mathcal{I} of Γ into an L-model \mathcal{M}

$$\mathcal{M}, \mathcal{I} \models A_p(\Gamma) \quad \text{implies} \quad \mathcal{M}, \mathcal{I} \models \Gamma,$$

⁵ Here 'S' in SDNF and SCNF stands for *special* to account for the additional requirement of one occurrence per label.

(iii) for each nested sequent Σ with $p \notin \text{Var}(\Sigma)$ and $\mathcal{L}(\Sigma) = \mathcal{L}(\Gamma)$ and for each multiworld interpretation \mathcal{I} of Γ into an L -model \mathcal{M} ,

$$\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma) \text{ and } \mathcal{M}, \mathcal{I} \not\models \Sigma \text{ imply } \mathcal{M}', \mathcal{I}' \not\models \Gamma \text{ and } \mathcal{M}', \mathcal{I}' \not\models \Sigma$$

for some multiworld interpretation \mathcal{I}' of Γ into some L -model \mathcal{M}' .

NUIP(i) ensures that interpretations of Γ are suitable for $A_p(\Gamma)$.

Remark 23. Břilková’s definition in [3] differs in several ways. Apart from a minor difference in NUIP(iii), our definition involves semantic notions and uses multi-formula interpolants instead of formulas.

Lemma 24. *If a nested calculus NL has the NUIP, then its logic L has the UIP.*

Proof. To show the existence of $\forall p\varphi$, consider a nested uniform interpolant $A_p(\varphi)$ of the nested sequent φ , with $\mathcal{L}(\varphi) = \{1\}$. By Lemma 21, w.l.o.g. we can assume that $A_p(\varphi) = 1 : \xi$. Let $\forall p\varphi := \xi$. We establish the UIP properties based on the corresponding NUIP properties. By NUIP(i), we have that $\text{Var}(\forall p\varphi) = \text{Var}(1 : \xi) \subseteq \text{Var}(\varphi) \setminus \{p\}$ which establishes UIP(i) (cf. Definition 14).

For UIP(ii) we use a semantic argument. Assume towards a contradiction that $\not\vdash_{\mathsf{L}} \xi \rightarrow \varphi$, in which case by completeness $\mathcal{M}, w \not\models \xi \rightarrow \varphi$ for some L -model $\mathcal{M} = (W, R, V)$ and $w \in W$. Consider a multiworld interpretation \mathcal{I} of sequent φ into \mathcal{M} such that $\mathcal{I}(1) := w$. Then $\mathcal{M}, \mathcal{I} \models 1 : \xi$ but $\mathcal{M}, \mathcal{I} \not\models \varphi$, in contradiction to NUIP(ii). Hence, $\vdash_{\mathsf{L}} \forall p\varphi \rightarrow \varphi$ as required.

Finally, for UIP(iii), let $p \notin \text{Var}(\psi)$ and suppose $\not\vdash_{\mathsf{L}} \psi \rightarrow \xi$. Once again, by completeness, $\mathcal{M}, w \not\models \psi \rightarrow \xi$ for some L -model $\mathcal{M} = (W, R, V)$ and $w \in W$. Consider the nested sequent $\bar{\psi}$, with $\mathcal{L}(\bar{\psi}) = \mathcal{L}(\varphi) = \{1\}$, and a multiworld interpretation \mathcal{I} of sequent φ into \mathcal{M} with $\mathcal{I}(1) := w$. Then $\mathcal{M}, \mathcal{I} \models 1 : \xi$ and $\mathcal{M}, \mathcal{I} \not\models \bar{\psi}$. By NUIP(iii), there must exist an L -model \mathcal{M}' and a multiworld interpretation \mathcal{I}' of sequent φ into \mathcal{M}' such that $\mathcal{M}', \mathcal{I}' \not\models \varphi$ and $\mathcal{M}', \mathcal{I}' \not\models \bar{\psi}$. In other words, $\mathcal{M}', \mathcal{I}'(1) \not\models \varphi$ and $\mathcal{M}', \mathcal{I}'(1) \models \psi$. Thus, by soundness of L , we have $\not\vdash_{\mathsf{L}} \psi \rightarrow \varphi$, thus completing the proof of UIP(iii). \square

We replace NUIP(iii) with a (possibly) stronger condition (iii)' that uses bisimulations up to p to find a model \mathcal{M}' :

Definition 25 (BNUIP). *A nested sequent calculus NL has the bisimulation nested-sequent uniform interpolation property, or BNUIP, if, in addition to conditions NUIP(i)–(ii) from Definition 22,*

(iii)' for each L -model \mathcal{M} and multiworld interpretation \mathcal{I} of Γ into \mathcal{M} , $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma)$, then there are an L -model \mathcal{M}' and multiworld interpretation \mathcal{I}' of Γ into \mathcal{M}' such that $(\mathcal{M}', \mathcal{I}') \sim_p (\mathcal{M}, \mathcal{I})$ and $\mathcal{M}', \mathcal{I}' \not\models \Gamma$.

It easily follows from Theorem 11 that, like formulas, both nested sequents and multiformulas are invariant under bisimulations:

Lemma 26. *Let $\Gamma (\mathcal{U})$ be a sequent (multiformula) not containing p and \mathcal{I} and \mathcal{I}' be multiworld interpretations of $\Gamma (\mathcal{U})$ into \mathcal{M} and \mathcal{M}' respectively such that $(\mathcal{M}, \mathcal{I}) \sim_p (\mathcal{M}', \mathcal{I}')$. Then $\mathcal{M}, \mathcal{I} \models \Gamma$ iff $\mathcal{M}', \mathcal{I}' \models \Gamma$ ($\mathcal{M}, \mathcal{I} \models \mathcal{U}$ iff $\mathcal{M}', \mathcal{I}' \models \mathcal{U}$).*

Lemma 27. *If $\Gamma, A_p(\Gamma)$ satisfy (iii)' of Definition 25, then they satisfy (iii) of Definition 22.*

Proof. Let Σ be a nested sequent with $p \notin \text{Var}(\Sigma)$ and $\mathcal{L}(\Sigma) = \mathcal{L}(\Gamma)$. Let $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma)$ and $\mathcal{M}, \mathcal{I} \not\models \Sigma$. By condition (iii)' we find an L-model \mathcal{M}' and \mathcal{I}' from Γ into \mathcal{M}' such that $(\mathcal{M}', \mathcal{I}') \sim_p (\mathcal{M}, \mathcal{I})$ and $\mathcal{M}', \mathcal{I}' \not\models \Gamma$. By Lemma 26, we also conclude $\mathcal{M}', \mathcal{I}' \not\models \Sigma$. \square

Corollary 28. *If a calculus NL has the BNUIP, then its logic L has the UIP.*

3.1 Uniform Interpolation For K

Now we present our method of constructing nested uniform interpolants satisfying the BNUIP for NK. Interpolants $A_p(\Gamma)$ are defined recursively on the basis of the terminating calculus from Fig. 1. If Γ is not K-saturated, $A_p(\Gamma)$ is defined recursively in Table 1 based on the form of Γ . For rows 3–5, we assume that the formula in the left column is not K-saturated in Γ , whereas in the last row we assume $\diamond\varphi$ not to be K-saturated w.r.t. σn in Γ .⁶ Each row in the table corresponds to a rule in the proof search.

Table 1. Recursive construction of $A_p(\Gamma)$ for NK for Γ that are not K-saturated.

Γ matches	$A_p(\Gamma)$ equals
$\Gamma' \{ \top \}_\sigma$	$\sigma : \top$
$\Gamma' \{ p, \bar{p} \}_\sigma$	$\sigma : \top$
$\Gamma' \{ \varphi \vee \psi \}$	$A_p(\Gamma' \{ \varphi \vee \psi, \varphi, \psi \})$
$\Gamma' \{ \varphi \wedge \psi \}$	$A_p(\Gamma' \{ \varphi \wedge \psi, \varphi \}) \otimes A_p(\Gamma' \{ \varphi \wedge \psi, \psi \})$
$\Gamma' \{ \Box\varphi \}_\sigma$	$\bigotimes_{i=1}^m \left(\sigma : \Box\delta_i \otimes \bigotimes_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right)$ <p>where n is the smallest integer such that $\sigma n \notin \mathcal{L}(\Gamma)$ and the SCNF</p> <p>of $A_p(\Gamma' \{ \Box\varphi, [\varphi]_{\sigma n} \})$ is $\bigotimes_{i=1}^m \left(\sigma n : \delta_i \otimes \bigotimes_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right)$,</p>
$\Gamma' \{ \diamond\varphi, [\Delta]_{\sigma n} \}$	$A_p(\Gamma' \{ \diamond\varphi, [\Delta, \varphi] \})$

⁶ Strictly speaking, this is a non-deterministic algorithm. Since the order does not affect our results, we do not specify it. However, it is more efficient to apply rows 1–2 of Table 1 first and row 5 last.

For K -saturated Γ , we define $A_p(\Gamma)$ recursively as follows:

$$A_p(\Gamma) := \bigvee_{\substack{\sigma: \ell \in \Gamma \\ \ell \in \text{Lit} \setminus \{p, \bar{p}\}}} \sigma : \ell \quad \otimes \quad \bigvee_{\substack{\tau \in \mathcal{L}(\Gamma) \\ (\exists \psi) \tau : \diamond \psi \in \Gamma}} \tau : \diamond A_p^{\text{form}} \left(\bigvee_{\tau : \diamond \psi \in \Gamma} \psi \right), \quad (1)$$

where $A_p^{\text{form}}(\Gamma) := \text{form}(A_p(\Gamma))$. Since we apply **form** to a multiformula \mathcal{U} with 1 being its only label, we have $\mathcal{M}, \mathcal{I} \models \mathcal{U}$ iff $\mathcal{M}, \mathcal{I}(1) \models \text{form}(\mathcal{U})$. As usual, we define the empty disjunction to be false, which here means $\bigvee \emptyset := 1 : \perp$. The construction of $A_p(\Gamma)$ is well-defined (modulo a chosen order) because it terminates w.r.t. the following ordering on nested sequents. For a nested sequent Γ , let $d(\Gamma)$ be the number of its distinct diamond subformulas. Let \ll be the ordering in which the rules of NK terminate (see Lemma 5). Consider the lexicographic ordering based on the pair (d, \ll) . For each row in Table 1, d stays the same but the recursive calls are for premise(s) lower w.r.t. ordering \ll . The recursive call in step (1) for K -saturated sequents, on the other hand, decreases d because the set of diamond subformulas of $\bigvee_{\tau : \diamond \psi \in \Gamma} \psi$ is strictly smaller than that of Γ . When $d(\Gamma) = 0$ for a K -saturated Γ , the second disjunct of the recursive call (1) is empty and, thus, no new recursive calls are generated.

Before we prove the main theorem, we provide some examples.

Example 29. Consider the sequent $\Box p, \Box \bar{p}$. We use Lemmas 20 and 21 as necessary. The algorithm for $A_p(\Box p, \Box \bar{p})$ calls the calculation of $A_p(\Box p, \Box \bar{p}, [p]_{11})$, which in turn calls $A_p(\Box p, \Box \bar{p}, [p]_{11}, [\bar{p}]_{12})$. The latter sequent is K -saturated, and the algorithm returns $1 : \perp \otimes 1 : \perp$, the first disjunct corresponding to the empty disjunction of literals other than p and \bar{p} and the second one representing the absent diamond formulas. Computing its SCNF we get

$$A_p(\Box p, \Box \bar{p}, [p]_{11}, [\bar{p}]_{12}) \equiv 1 : \perp \otimes 11 : \perp \otimes 12 : \perp.$$

Applying the transformation from the penultimate line of Table 1, we first get

$$A_p(\Box p, \Box \bar{p}, [p]_{11}) = 1 : \perp \otimes 11 : \perp \otimes 1 : \Box \perp \equiv 1 : \Box \perp \otimes 11 : \perp,$$

and finally $A_p(\Box p, \Box \bar{p}) = 1 : \Box \perp \otimes 1 : \Box \perp \equiv 1 : \Box \perp$. It is easy to check that $\Box \perp$ is indeed a uniform interpolant of $\Box p \vee \Box \bar{p}$.

Example 30. Consider the nested sequent $\Gamma = \bar{p}, \diamond q \wedge \diamond p, [q]$. In the absence of boxes, the algorithm amounts to processing the K -saturated sequents in the leaves of the proof search tree.

$$\frac{\frac{\bar{p}, \diamond q \wedge \diamond p, \diamond p, [q, p]_{11}}{\bar{p}, \diamond q \wedge \diamond p, \diamond p, [q]_{11}}}{\bar{p}, \diamond q \wedge \diamond p, [q]_{11}}$$

We have

$$\begin{aligned} A_p(\bar{p}, \diamond q \wedge \diamond p, \diamond q, [q]_{11}) &= 11 : q \otimes 1 : \diamond A_p^{\text{form}}(q), \\ A_p(\bar{p}, \diamond q \wedge \diamond p, \diamond p, [q, p]_{11}) &= 11 : q \otimes 1 : \diamond A_p^{\text{form}}(p). \end{aligned}$$

Since formulas $A_p^{\text{form}}(q)$ and $A_p^{\text{form}}(p)$ can be simplified to q and \perp respectively, we obtain $A_p(\Gamma) \equiv (11 : q \otimes 1 : \diamond q) \otimes (11 : q \otimes 1 : \diamond \perp)$, which is equivalent to $11 : q$ since $\diamond \perp$ can never be true. Again, it is easy to see that $11 : q$ is a bisimulation nested uniform interpolant of $\bar{p}, \diamond q \wedge \diamond p, [q]_{11}$ with respect to p .

Theorem 31. *The nested calculus NK has the BNUIP.*

Proof. BNUIP(i) is easily satisfied. To prove BNUIP(ii), let Γ be a nested sequent and \mathcal{I} a multiworld interpretation of Γ into a K-model $\mathcal{M} = (W, R, V)$ such that $\mathcal{M}, \mathcal{I} \models A_p(\Gamma)$ (by BNUIP(i) \mathcal{I} is suitable for $A_p(\Gamma)$). We show $\mathcal{M}, \mathcal{I} \models \Gamma$ by induction on the lexicographic ordering (d, \ll) . Considering the construction of $A_p(\Gamma)$, we treat the cases of Table 1 first and deal with the case of K-saturated Γ last. Cases in rows 1–2 of Table 1 are trivial. Those in rows 3, 4, and 6 are similar (see [10]), so we only discuss row 5:

Let $\Gamma = \Gamma' \{ \Box \varphi \}_\sigma$, and $A_p(\Gamma' \{ \Box \varphi, [\varphi]_{\sigma n} \}) \equiv \bigotimes_{i=1}^m \left(\sigma n : \delta_i \otimes \bigotimes_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right)$ for some $\sigma n \notin \mathcal{L}(\Gamma)$, and

$$\mathcal{M}, \mathcal{I} \models \bigotimes_{i=1}^m \left(\sigma : \Box \delta_i \otimes \bigotimes_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right). \tag{2}$$

For any v with $\mathcal{I}(\sigma)Rv$, define a multiworld interpretation $\mathcal{I}_v := \mathcal{I} \sqcup \{(\sigma n, v)\}$ of $\Gamma' \{ \Box \varphi, [\varphi]_{\sigma n} \}$ into \mathcal{M} . By (2) we have, for each i , either $\mathcal{M}, \mathcal{I}_v(\tau) \models \gamma_{i,\tau}$ for some $\tau \in \mathcal{L}(\Gamma)$ or $\mathcal{M}, \mathcal{I}_v(\sigma n) \models \delta_i$, meaning that $\mathcal{M}, \mathcal{I}_v \models A_p(\Gamma' \{ \Box \varphi, [\varphi]_{\sigma n} \})$. By the induction hypothesis, $\mathcal{M}, \mathcal{I}_v \models \Gamma' \{ \Box \varphi, [\varphi]_{\sigma n} \}$ whenever $\mathcal{I}(\sigma)Rv$. Clearly, $\mathcal{M}, \mathcal{I} \models \Gamma$ if $\mathcal{M}, \mathcal{I}(\sigma) \models \Box \varphi$. Otherwise there exists a v such that $\mathcal{I}(\sigma)Rv$ and $\mathcal{M}, v \not\models \varphi$. For this world $\mathcal{M}, \mathcal{I}_v \models \Gamma' \{ \Box \varphi, [\varphi]_{\sigma n} \}$ implies $\mathcal{M}, \mathcal{I}_v \models \Gamma' \{ \Box \varphi \}_\sigma$, which yields $\mathcal{M}, \mathcal{I} \models \Gamma$ because \mathcal{I}_v agrees with \mathcal{I} on all labels from Γ .

Finally, for the case when Γ is K-saturated, let $\mathcal{M}, \mathcal{I} \models A_p(\Gamma)$ from (1). Clearly, $\mathcal{M}, \mathcal{I} \models \Gamma$ if we have $\mathcal{M}, \mathcal{I}(\sigma) \models \ell$ for some $\sigma : \ell \in \Gamma$. Thus, it remains to consider the case when $\mathcal{M}, \mathcal{I}(\tau) \models \diamond A_p^{\text{form}}(\bigvee_{\tau : \diamond \psi \in \Gamma} \psi)$ for some $\tau \in \mathcal{L}(\Gamma)$. Then $\mathcal{M}, v \models A_p^{\text{form}}(\bigvee_{\tau : \diamond \psi \in \Gamma} \psi)$ for some v such that $\mathcal{I}(\tau)Rv$ and, accordingly, $\mathcal{M}, \mathcal{J} \models A_p(\bigvee_{\tau : \diamond \psi \in \Gamma} \psi)$ for $\mathcal{J} := \{(1, v)\}$. By induction hypothesis (for a smaller d), $\mathcal{M}, \mathcal{J} \models \bigvee_{\tau : \diamond \psi \in \Gamma} \psi$, and, hence, $\mathcal{M}, v \models \psi$ for some $\tau : \diamond \psi \in \Gamma$. Now $\mathcal{M}, \mathcal{I} \models \Gamma$ follows from $\mathcal{I}(\tau)Rv$. This case concludes the proof for BNUIP(ii).

It only remains to prove BNUIP(iii)'. Let \mathcal{I} be a multiworld interpretation of Γ into a K-model \mathcal{M} such that $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma)$. We must find another multiworld interpretation \mathcal{I}' into some K-model \mathcal{M}' such that $(\mathcal{M}', \mathcal{I}') \sim_p (\mathcal{M}, \mathcal{I})$ and $\mathcal{M}', \mathcal{I}' \models \Gamma$. We construct \mathcal{M}' and \mathcal{I}' while simultaneously proving BNUIP(iii)' by induction on the lexicographic order (d, \ll) .

Let Γ be K-saturated and $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma)$ for $A_p(\Gamma)$ from (1). The following steps are schematically depicted in Fig. 2 (see [10] for more details).

- (1) First, we make the interpretation injective. It is easy to see (though tedious to describe in detail) that by a breadth-first recursion on nodes σ in Γ , one

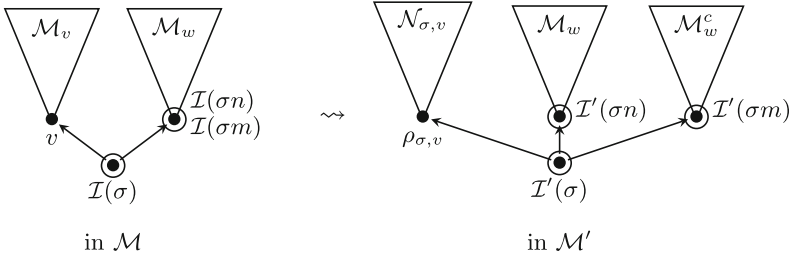


Fig. 2. Main transformations for constructing model \mathcal{M}' : circles are worlds in $\text{Range}(\mathcal{I})$.

can duplicate $\mathcal{M}_{\mathcal{I}(\sigma n)}$ according to Definition 12 whenever $\mathcal{I}(\sigma m) = \mathcal{I}(\sigma n)$ for some $n < m$ to obtain a model \mathcal{N} and an injective multiworld interpretation \mathcal{J} of Γ into it such that $(\mathcal{N}, \mathcal{J}) \sim_p (\mathcal{M}, \mathcal{I})$. Thus, $\mathcal{J}(\sigma) \neq \mathcal{J}(\tau)$ whenever $\sigma \neq \tau$ and $\mathcal{N}, \mathcal{J} \not\models A_p(\Gamma)$ by Lemma 26.

- (2) Then we deal with out-of-range children. A model \mathcal{N}' is constructed from \mathcal{N} by applying the following \diamond -processing step for each node $\tau \in \mathcal{L}(\Gamma)$ that contains at least one formula of the form $\diamond\varphi$ (nodes can be chosen in any order). Start by setting $\mathcal{N}^0 := \mathcal{N}$ and $j := 0$:

- **\diamond -processing step for τ :** Since $\mathcal{N}^j, \mathcal{J} \not\models A_p(\Gamma)$, it follows from (1) that $\mathcal{N}^j, \mathcal{J}(\tau) \not\models \diamond A_p^{\text{form}} \left(\bigvee_{\tau: \diamond\psi \in \Gamma} \psi \right)$. Thus, $\mathcal{N}^j, v \not\models A_p^{\text{form}} \left(\bigvee_{\tau: \diamond\psi \in \Gamma} \psi \right)$ for each child v of $\mathcal{J}(\tau)$ in \mathcal{N}^j , and, accordingly, $\mathcal{N}_v^j, \mathcal{I}_v \not\models A_p \left(\bigvee_{\tau: \diamond\psi \in \Gamma} \psi \right)$ for the multiworld interpretation $\mathcal{I}_v := \{(1, v)\}$ of sequent $\bigvee_{\tau: \diamond\psi \in \Gamma} \psi$ into the subtree \mathcal{N}_v^j of \mathcal{N}^j with root v . By the induction hypothesis for a smaller d , there exists a K-model $\mathcal{N}_{\tau,v}$ with root $\rho_{\tau,v}$ such that $(\mathcal{N}_v^j, v) \sim_p (\mathcal{N}_{\tau,v}, \rho_{\tau,v})$ and $\mathcal{N}_{\tau,v}, \rho_{\tau,v} \not\models \bigvee_{\tau: \diamond\psi \in \Gamma} \psi$. Let \mathcal{N}^{j+1} be the result of replacing each subtree \mathcal{N}_v^j for children v of $\mathcal{J}(\tau)$ not in $\text{Range}(\mathcal{J})$ with $\mathcal{N}_{\tau,v}$ in \mathcal{N}^j according to Definition 12. Note that all these subtrees are disjoint because the models are intransitive trees and, hence, these replacements do not interfere with one another. Note also that since $\text{Range}(\mathcal{J})$ is downward closed and the roots of the replaced subtrees are outside, no world from the range is modified. Thus, \mathcal{J} remains an injective interpretation into \mathcal{N}^{j+1} . Finally, it follows from Lemma 13 that $(\mathcal{N}^j, \mathcal{J}) \sim_p (\mathcal{N}^{j+1}, \mathcal{J})$. Hence, $\mathcal{N}^{j+1}, \mathcal{J} \not\models A_p(\Gamma)$. Let $\mathcal{N}' = (W', R', V')$ be the model obtained after replacements for all τ 's are completed (again they do not interfere with each other). Then we have $(\mathcal{N}, \mathcal{J}) \sim_p (\mathcal{N}', \mathcal{J})$ and, for each out-of-range child v of $\mathcal{J}(\tau)$ in \mathcal{N} , the world $\rho_{\tau,v}$ is a child of $\mathcal{J}(\tau)$ in \mathcal{N}' and $\mathcal{N}', \rho_{\tau,v} \not\models \bigvee_{\tau: \diamond\psi \in \Gamma} \psi$. This accounts for all children of $\mathcal{J}(\tau)$ in \mathcal{N}' .

- (3) It remains to adjust the truth values of p . We define $\mathcal{M}' := (W', R', V'_p)$ by modifying the valuation V' of \mathcal{N}' as follows. We define $V'_p(q) := V'(q)$ for $q \neq p$. And for $q = p$ we define:

$$V'_p(p) := V'(p) \cap (W' \setminus \text{Range}(\mathcal{J})) \sqcup \{v \in W' \mid \exists \sigma (v = \mathcal{J}(\sigma) \& \sigma: \bar{p} \in \Gamma)\}.$$

For $\mathcal{I}' := \mathcal{J}$, it immediately follows from the definition that

$$\mathcal{M}', \mathcal{I}'(\sigma) \not\models \bar{p} \text{ if } \sigma : \bar{p} \in \Gamma \quad \text{and} \quad \mathcal{M}', \mathcal{I}'(\sigma) \not\models p \text{ if } \sigma : p \in \Gamma. \quad (3)$$

Moreover, since subtrees $\mathcal{M}'_{\rho_{\tau,v}}$ are disjoint from $\text{Range}(\mathcal{I}')$,

$$\mathcal{M}', \rho_{\tau,v} \not\models \psi \text{ whenever } \tau : \diamond\psi \in \Gamma. \quad (4)$$

After these 3 steps, we have a model $(\mathcal{M}', \mathcal{I}') \sim_p (\mathcal{N}', \mathcal{J}) \sim_p (\mathcal{N}, \mathcal{J}) \sim_p (\mathcal{M}, \mathcal{I})$ that satisfies (3) and (4). It remains to prove that $\mathcal{M}', \mathcal{I}' \not\models \Gamma$ by showing that $\mathcal{M}', \mathcal{I}'(\sigma) \not\models \varphi$ for all $\sigma : \varphi \in \Gamma$, which is done by induction on the structure of φ . Each case, except for the \diamond case is easy (see [10]). So, let $\sigma : \diamond\psi \in \Gamma$. To falsify $\diamond\psi$ at $\mathcal{I}'(\sigma)$, we need to show that $\mathcal{M}', u \not\models \psi$ whenever $\mathcal{I}'(\sigma)R'u$. If $u = \mathcal{I}'(\sigma n)$ for some label $\sigma n \in \mathcal{L}(\Gamma)$, saturation ensures that $\sigma n : \psi \in \Gamma$, hence, $\mathcal{M}', u \not\models \psi$ by the induction hypothesis. The only other children of $\mathcal{I}'(\sigma)$ are $u = \rho_{\sigma,v}$, for which $\mathcal{M}', u \not\models \psi$ follows from (4). This completes the proof of BNUIP(iii)' for K-saturated sequents.

To conclude the proof of BNUIP(iii)', we have to treat all sequents that are not K-saturated based on Table 1. Here, the only non-trivial case is the \square case. The other cases are easy (see [10]). Assume $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma' \{ \square\varphi \}_\sigma)$, i.e.,

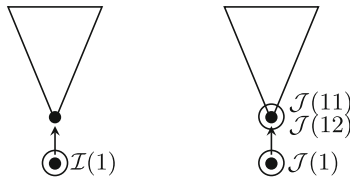
$$\mathcal{M}, \mathcal{I} \not\models \bigwedge_{i=1}^m \left(\sigma : \square\delta_i \otimes \bigvee_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right) \quad (5)$$

where

$$A_p(\Gamma' \{ \square\varphi, [\varphi]_{\sigma n} \}) \equiv \bigwedge_{i=1}^m \left(\sigma n : \delta_i \otimes \bigvee_{\tau \neq \sigma n} \tau : \gamma_{i,\tau} \right). \quad (6)$$

By (5), for some i we have $\mathcal{M}, \mathcal{I}(\sigma) \not\models \square\delta_i$ and $\mathcal{M}, \mathcal{I}(\tau) \not\models \gamma_{i,\tau}$ for all $\tau \neq \sigma n$. The former means that $\mathcal{M}, v \not\models \delta_i$ for some v such that $\mathcal{I}(\sigma)Rv$. Therefore, a multiworld interpretation $\mathcal{J} := \mathcal{I} \sqcup \{(\sigma n, v)\}$ of $\Gamma' \{ \square\varphi, [\varphi]_{\sigma n} \}$ into \mathcal{M} falsifies (6), and, by the induction hypothesis, there is a multiworld interpretation \mathcal{J}' into a K-model \mathcal{M}' such that $(\mathcal{M}', \mathcal{J}') \sim_p (\mathcal{M}, \mathcal{J})$ and $\mathcal{M}', \mathcal{J}' \not\models \Gamma' \{ \square\varphi, [\varphi]_{\sigma n} \}$. For $\mathcal{I}' := \mathcal{J}' \upharpoonright \text{Dom}(\mathcal{I})$, we have $(\mathcal{M}, \mathcal{I}) \sim_p (\mathcal{M}', \mathcal{I}')$ and $\mathcal{M}', \mathcal{I}' \not\models \Gamma' \{ \square\varphi \}_\sigma$ because all formulas from $\Gamma' \{ \square\varphi \}_\sigma$ are present in $\Gamma' \{ \square\varphi, [\varphi]_{\sigma n} \}$. \square

Example 32. As shown in Example 29, $A_p(\square p, \square \bar{p}) = 1 : \square \perp$. To see the importance of injectivity in BNUIP(iii)', suppose $\mathcal{M}, \mathcal{I} \not\models 1 : \square \perp$, i.e., $\mathcal{I}(1)$ has at least one child. Assume this is the only child, as in a model depicted on the left:



For a saturation $\square p, \square \bar{p}, [p]_{11}, [\bar{p}]_{12}$ of this sequent, we found an interpolant in SCNF: namely, $1 : \perp \otimes 11 : \perp \otimes 12 : \perp$. A multiworld interpretation \mathcal{J} mapping both 11 and 12 to the only child of $\mathcal{J}(1) := \mathcal{I}(1)$ yields the picture on

the right. Clearly, the SCNF is false, $\mathcal{M}, \mathcal{J} \not\models 1 : \perp \otimes 11 : \perp \otimes 12 : \perp$. But, without forcing \mathcal{J} to be injective, it is impossible to make $\Box p, \Box \bar{p}$ false at $\mathcal{J}(1)$: whichever truth value p has at $\mathcal{J}(11)$, it makes one of the boxes true.

3.2 Uniform Interpolation For D and T

The proof for K can be adjusted to prove the same result for D and T.

Theorem 33. *The nested sequent calculi ND and NT have the BNUIP.*

Proof. We follow the structure of the proof in Theorem 31 and only describe deviations from it. If Γ is not D-/T-saturated, then cases in Table 1 are appended with the bottom (top) row of Table 2, which is applied only if $\diamond\varphi$ is not D-/T-saturated in Γ . For D-/T-saturated Γ , define $A_p(\Gamma)$ by (1) as before. BNUIP(i) is clearly satisfied by either row in Table 2.

Table 2. Additional recursive rules for constructing $A_p(\Gamma)$ for Γ that are not T-saturated (top row) or not D-saturated (bottom row).

Γ matches	$A_p(\Gamma)$ equals
$\Gamma'\{\diamond\varphi\}$ in logic T	$A_p(\Gamma'\{\diamond\varphi, \varphi\})$
$\Gamma'\{\diamond\varphi\}_\sigma$ in logic D	$\bigotimes_{i=1}^m \left(\sigma : \diamond\delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$ where the SDNF of $A_p(\Gamma'\{\diamond\varphi, [\varphi]_{\sigma 1}\})$ is $\bigotimes_{i=1}^m \left(\sigma 1 : \delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$

Let us first show BNUIP(ii) for NT. Although T-models are reflexive, this does not affect the reasoning for either saturated sequents or non-saturated box formulas. The only new case is applying the top row of Table 2 to a non-T-saturated $\sigma : \diamond\varphi$ in Γ . Assume $\mathcal{M}, \mathcal{I} \models A_p(\Gamma'\{\diamond\varphi, \varphi\}_\sigma)$ for a T-model \mathcal{M} . By the induction hypothesis, $\mathcal{M}, \mathcal{I} \models \Gamma'\{\diamond\varphi, \varphi\}_\sigma$. Since $\mathcal{M}, \mathcal{I}(\sigma) \models \varphi$ implies $\mathcal{M}, \mathcal{I}(\sigma) \models \diamond\varphi$ by reflexivity, the desired $\mathcal{M}, \mathcal{I} \models \Gamma'\{\diamond\varphi\}_\sigma$ follows.

For BNUIP(iii)' for T-saturated sequents, we have to modify the construction in step (1) on p. 12 of an injective multiworld interpretation \mathcal{J} into a new T-model \mathcal{N} out of the given \mathcal{I} into \mathcal{M} where $\mathcal{M}, \mathcal{I} \not\models A_p(\Gamma)$. In the case of K, there could be only one situation of σm conflated with some already processed τ : namely, when $\tau = \sigma n$ is a sibling. This can still happen for T-models and is processed the same way. But, due to reflexivity, there is now another possibility: conflating with the parent $\tau = \sigma$. In this case, cloning is used instead of duplication, which produces a bisimilar T-model by Lemma 13. Having reflexive rather than irreflexive intransitive trees in step (2) on p. 12 does not affect the argument. The proof that $\mathcal{M}', \mathcal{I}' \not\models \Gamma$ for the given T-saturated Γ in step (3) on p. 13 requires an adjustment only for the case of $\sigma : \diamond\psi \in \Gamma$: it is additionally necessary to show that $\mathcal{M}', \mathcal{I}'(\sigma) \not\models \psi$ for the reflexive loop at $\mathcal{I}'(\sigma)$. This is

resolved by observing that $\sigma : \psi \in \Gamma$ due to T-saturation and, hence, ψ must also be false in $\mathcal{I}'(\sigma)$ by the induction hypothesis.

Finally, for BNUIP(iii)' for non-T-saturated sequents, a new case comes from the top row of Table 2, but $\mathcal{M}', \mathcal{I}' \not\models \Gamma' \{ \diamond\varphi, \varphi \}$ obtained by the IH directly implies $\mathcal{M}', \mathcal{I}' \not\models \Gamma' \{ \diamond\varphi \}$. This completes the proof of the BNUIP for NT.

For BNUIP(ii) for ND, the only new case is applying the bottom row of Table 2 to a non-D-saturated $\sigma : \diamond\varphi$ in $\Gamma = \Gamma' \{ \diamond\varphi \}_\sigma$. So let us assume

$\mathcal{M}, \mathcal{I} \models \bigvee_{i=1}^m \left(\sigma : \diamond\delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$ for some multiworld interpretation \mathcal{I} into a D-model $\mathcal{M} = (W, R, V)$ such that the SDNF of $A_p(\Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \})$ equals $\bigvee_{i=1}^m \left(\sigma 1 : \delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$. Therefore, for some i we have $\mathcal{M}, \mathcal{I}(\tau) \models \gamma_{i,\tau}$ for all $\tau \in \mathcal{L}(\Gamma)$ and $\mathcal{M}, \mathcal{I}(\sigma) \models \diamond\delta_i$. Therefore, $\mathcal{M}, v \models \delta_i$ for some v such that $\mathcal{I}(\sigma)Rv$. Formula $\diamond\varphi$ is not D-saturated in $\Gamma' \{ \diamond\varphi \}_\sigma$, so $\mathcal{I}_v := \mathcal{I} \sqcup \{ (\sigma 1, v) \}$ is a multiworld interpretation of $\Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \}$ into \mathcal{M} . Moreover, we have $\mathcal{M}, \mathcal{I}_v \models A_p(\Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \})$. By induction hypothesis, $\mathcal{M}, \mathcal{I}_v \models \Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \}$, from which it easily follows that $\mathcal{M}, \mathcal{I} \models \Gamma' \{ \diamond\varphi \}_\sigma$.

For BNUIP(iii)' for a D-saturated sequent Γ , we change step (1) in such a way that not only is the multiworld interpretation \mathcal{J} injective, but $\text{Range}(\mathcal{J})$ contains only irreflexive worlds. Injectivity is obtained in a similar way as done for T using duplication and cloning to obtain a bisimilar D-model \mathcal{M}'' by Lemma 13 with injective multiworld interpretation, say \mathcal{J} . So $(\mathcal{M}'', \mathcal{J}) \sim_p (\mathcal{M}, \mathcal{I})$, with injective \mathcal{J} . To ensure that the multiworld interpretation only maps to irreflexive worlds, we repeatedly unravel subtrees rooted in reflexive worlds from $\text{Range}(\mathcal{J})$ while keeping the same multiworld interpretation \mathcal{J} . Since each unraveling decreases the number of reflexive worlds in $\text{Range}(\mathcal{J})$, this process terminates yielding a model \mathcal{M}' that is a D-model and satisfies $(\mathcal{M}', \mathcal{J}) \sim_p (\mathcal{M}, \mathcal{I})$ by Lemma 13. The replacements of step (2) preserve D-models by Lemma 13 and step (3) requires no changes either. Note that in steps (2) and (3) we do not change the range of $\mathcal{I}' := \mathcal{J}$, so it still only maps to irreflexive worlds. We need this construction in the proof that $\mathcal{M}', \mathcal{I}' \not\models \Gamma$ for case $\sigma : \diamond\psi \in \Gamma$, where the argument for $\mathcal{M}', \mathcal{I}'(\sigma) \not\models \diamond\psi$ now works the same way as in K since $\mathcal{I}'(\sigma)$ is irreflexive by construction.

The only remaining new case is the application of the bottom row of Table 2 for a non-D-saturated $\sigma : \diamond\varphi$, i.e., when node σ is a leaf of the sequent tree, in BNUIP(iii)'. Let $\mathcal{M}, \mathcal{I} \not\models \bigvee_{i=1}^m \left(\sigma : \diamond\delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$. By seriality of \mathcal{M} , there is a world $v \in W$ such that $\mathcal{I}(\sigma)Rv$. Then $\mathcal{J} := \mathcal{I}' \sqcup \{ (\sigma 1, v) \}$ is a multiworld interpretation of $\Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \}$ into \mathcal{M} . Moreover, we have $\mathcal{M}, \mathcal{J} \not\models \bigvee_{i=1}^m \left(\sigma 1 : \delta_i \otimes \bigotimes_{\tau \neq \sigma 1} \tau : \gamma_{i,\tau} \right)$. By induction hypothesis, there is a multiworld interpretation \mathcal{J}' of $\Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \}$ into some D-model \mathcal{M}' such that $(\mathcal{M}', \mathcal{J}') \sim_p (\mathcal{M}, \mathcal{J})$ and $\mathcal{M}', \mathcal{J}' \not\models \Gamma' \{ \diamond\varphi, [\varphi]_{\sigma 1} \}$. Similar to the case of $\Box\varphi$ for K, restricting this \mathcal{J}' to the labels of Γ yields a multiworld interpretation bisimilar to \mathcal{I} and refuting $\Gamma = \Gamma' \{ \diamond\varphi \}_\sigma$. \square

Corollary 34. *Logics K, D, and T have the uniform interpolation property.*

4 Conclusion

We developed a constructive method of proving uniform interpolation based on nested sequent calculi. While this is an important and natural step to further utilize these formalisms, much remains to be done. This method works well for the non-transitive logics K, D, and T but meets with difficulties, e.g., for S5, which is also known to enjoy uniform interpolation. In [10], we successfully adapted the method to hypersequents to cover S5. There are other logics in the so-called modal cube between K and S5 with the UIP, for which it remains to find the right formalism and adaptation of our method. Another natural direction of future work is intermediate logics, where exactly seven logics are known to have the UIP.

Acknowledgements. Iris and Roman thank Björn Lellmann and Tim Lyon for enlightening discussions, detailed explanations, and a genuinely pleasant working atmosphere during Iris’s visit to Vienna. We also thank the anonymous reviewers for their careful and useful comments that led to several improvements of the paper.

References

1. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In *Logic: From Foundations to Applications*, pp. 1–32. Clarendon Press (1996)
2. Bílková, M.: Interpolation in modal logics. Ph.D. thesis, Charles University (2006). <https://dspace.cuni.cz/handle/20.500.11956/15732>
3. Bílková, M.: A note on uniform interpolation proofs in modal deep inference calculi. In: Bezhanishvili, N., Löbner, S., Schwabe, K., Spada, L. (eds.) *TbiLLC 2009. LNCS (LNAI)*, vol. 6618, pp. 30–45. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22303-7_3
4. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press (2001). <https://doi.org/10.1017/CBO9781107050884>
5. Brünnler, K.: Deep sequent systems for modal logic. *Arch. Math. Log.* **48**, 551–577 (2009). <https://doi.org/10.1007/s00153-009-0137-3>
6. D’Agostino, G.: Uniform interpolation, bisimulation quantifiers, and fixed points. In: ten Cate, B.D., Zeevat, H.W. (eds.) *TbiLLC 2005. LNCS (LNAI)*, vol. 4363, pp. 96–116. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75144-1_8
7. Fitting, M., Kuznets, R.: Modal interpolation via nested sequents. *Ann. Pure Appl. Log.* **166**, 274–305 (2015). <https://doi.org/10.1016/j.apal.2014.11.002>
8. Ghilardi, S.: An algebraic theory of normal forms. *Ann. Pure Appl. Log.* **71**, 189–245 (1995). [https://doi.org/10.1016/0168-0072\(93\)E0084-2](https://doi.org/10.1016/0168-0072(93)E0084-2)
9. Ghilardi, S., Zawadowski, M.: Undefinability of propositional quantifiers in the modal system S4. *Studia Logica* **55**, 259–271 (1995). <https://doi.org/10.1007/BF01061237>
10. van der Giessen, I., Jalali, R., Kuznets, R.: Uniform interpolation via nested sequents and hypersequents. E-print 2105.10930, arXiv (2021). [arXiv:2105.10930](https://arxiv.org/abs/2105.10930)

11. Goré, R.: Tableau methods for modal and temporal logics. In: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) *Handbook of Tableau Methods*, pp. 297–396. Springer, Dordrecht (1999). https://doi.org/10.1007/978-94-017-1754-0_6
12. Iemhoff, R.: Uniform interpolation and sequent calculi in modal logic. *Arch. Math. Log.* **58**, 155–181 (2019). <https://doi.org/10.1007/s00153-018-0629-0>
13. Iemhoff, R.: Uniform interpolation and the existence of sequent calculi. *Ann. Pure Appl. Log.* **170**(11), 102711 (2019). <https://doi.org/10.1016/j.apal.2019.05.008>
14. Kurahashi, T.: Uniform Lyndon interpolation property in propositional modal logics. *Arch. Math. Log.* **59**, 659–678 (2020). <https://doi.org/10.1007/s00153-020-00713-y>
15. Kuznets, R.: Multicomponent proof-theoretic method for proving interpolation properties. *Ann. Pure Appl. Log.* **169**, 1369–1418 (2018). <https://doi.org/10.1016/j.apal.2018.08.007>
16. Kuznets, R., Lellmann, B.: Interpolation for intermediate logics via hyper- and linear nested sequents. In: *Advances in Modal Logic*, vol. 12, pp. 473–492. College Publications (2018). <http://www.aiml.net/volumes/volume12/Kuznets-Lellmann.pdf>
17. Minc, G.E.: On some calculi of modal logic. In: *The Calculi of Symbolic Logic*. I, volume 98 (1968) of *Proceedings of the Steklov Institute of Mathematics*, pp. 97–124. AMS (1971)
18. Negri, S., von Plato, J.: *Proof Analysis*. Cambridge University Press (2011). <https://doi.org/10.1017/CBO9781139003513>
19. Pitts, A.M.: On an interpretation of second order quantification in first order intuitionistic propositional logic. *J. Symb. Log.* **57**, 33–52 (1992). <https://doi.org/10.2307/2275175>
20. Poggiolesi, F.: The method of tree-hypersequents for modal propositional logic. In: Makinson, D., Malinowski, J., Wansing, H. (eds.) *Towards Mathematical Philosophy*. TL, vol. 28, pp. 31–51. Springer, Dordrecht (2009). https://doi.org/10.1007/978-1-4020-9084-4_3
21. Shavrukov, V.Yu.: Subalgebras of diagonalizable algebras of theories containing arithmetic, volume 323 of *Dissertationes Mathematicae*. Polish Academy of Sciences (1993). <http://matwbn.icm.edu.pl/ksiazki/rm/rm323/rm32301.pdf>
22. Tabatabai, A.A., Jalali, R.: Universal proof theory: Semi-analytic rules and Craig interpolation. E-print 1808.06256, arXiv (2018). [arXiv:1808.06256](https://arxiv.org/abs/1808.06256)
23. Tabatabai, A.A., Jalali, R.: Universal proof theory: semi-analytic rules and uniform interpolation. E-print 1808.06258, arXiv (2018). [arXiv:1808.06258](https://arxiv.org/abs/1808.06258)
24. Visser, A.: Uniform interpolation and layered bisimulation. In: *Gödel 1996*, volume 6 of *Lecture Notes in Logic*, pp. 139–164. ASL (1996). <https://doi.org/10.1017/9781316716939.010>



Disjunction and Negation in Information Based Semantics

Vít Punčochář^(✉) and Andrew Tedder

Institute of Philosophy, Czech Academy of Sciences,
Jilská 1, 110 00 Prague, Czech Republic
puncochar@flu.cas.cz

Abstract. We investigate an information based generalization of the incompatibility-frame treatment of logics with non-classical negation connectives. Our framework can be viewed as an alternative to the neighbourhood semantics for extensions of lattice logic by various negation connectives, investigated by Hartonas. We set out the basic semantic framework, along with some correspondence results for extensions. We describe three kinds of constructions of canonical models and show that double negation law is not canonical with respect to any of these constructions. We also compare our semantics to Hartonas'.

Keywords: Non-classical logics · Incompatibility · Information · Relational semantics · Negation · Disjunction

1 Introduction

A characteristic feature of relational semantics is that it is based on a relation \models , which in any given model relates elements of the model and formulas of some language. For example, in the standard Kripke semantics for modal logics the elements of Kripke models represent *possible worlds* and \models , as a relation between possible worlds and formulas, is conceived of as the relation of *truth*. Motivated by this interpretation, the following semantic clauses for conjunction, disjunction, and negation are postulated:

- (\wedge) $s \models \alpha \wedge \beta$ iff $s \models \alpha$ and $s \models \beta$,
- (\vee) $s \models \alpha \vee \beta$ iff $s \models \alpha$ or $s \models \beta$,
- (\neg) $s \models \neg\alpha$ iff $s \not\models \alpha$.

As a consequence, these connectives behave in accordance with classical logic in Kripke semantics. To avoid various features of classical logic, relational semantic frameworks for various non-classical logics, like, for example, intuitionistic logic and relevant logics, replace possible worlds with some more general entities,

This paper is an outcome of the project Logical Structure of Information Channels, no. 21-23610M, supported by the Czech Science Foundation and realized at the Institute of Philosophy of the Czech Academy of Sciences.

which we might neutrally call *states*. With respect to these states negation does not behave in this simple way, and the question of how to interpret these more general states has posed a perennial issue. One suggestion that has appeared repeatedly in the literature is that they should be viewed as bodies of information, i.e. as information states.¹ The relation \models between such states and formulas should perhaps be called (informational) *support* rather than truth. But the crucial thing is that with respect to such interpretation the clause (\neg) for negation seems implausible because a body of information may be incomplete and thus support neither a given sentence nor its negation, or it can be inconsistent and thus support both a sentence and its negation. Consequently, one has to introduce a different semantic clause for negation. When the clause (\neg) for negation is abandoned the space for non-classical logics is opened.

It is quite common that even if possible worlds are replaced with abstract states and the semantics of negation is modified, the structure of the semantic clauses (\wedge) and (\vee) for conjunction and disjunction is preserved.² However, from the informational perspective there seems to be some asymmetry between conjunction and disjunction. The clause (\wedge) still makes a good sense even if we replace possible worlds with information states (a body of information supports a conjunction iff it supports both conjuncts). However, it has been pointed out many times that the clause (\vee) for disjunction is implausible in the informational interpretation, just like the clause (\neg) for negation.³ A body of information may support a disjunction even if it does not support any of its disjuncts. If, for instance, we conceive of a body of information as that being available to an agent in the course of some reasoning task, then we know from experience that we may have information that a disjunction holds without any information about which of the disjuncts holds.

Several alternative treatments of disjunction have been proposed in the literature. For example, while Kripke semantics for intuitionistic logic [17] accepts the clause (\vee) for disjunction, the so called Beth semantics [2] for intuitionistic logic treats disjunction in a more complex way.

In the informational perspective one can characterize disjunction in this way, which shall be our approach: the disjunction of α and β is what two states have in common if one of them supports α and the other one supports β . In other words, if an information state s supports $\alpha \vee \beta$ then it contains the common content of the information expressed by α and the information expressed by β . If this is taken as the characteristic property of disjunction, one can transform it into a semantic clause. Assume that $s \circ t$ represents the common (informational)

¹ The interpretations of frame semantics in terms of information states was most famously provided for intuitionistic logic in [17]. Such an interpretation was given for relational semantics for relevant logics in [24], and has been used extensively in work influenced by situation semantics [1], such as in [18].

² This is, for instance, the case in the ternary relation semantic framework for relevant logics, details about which can be found in [23].

³ See, e.g. [5, 14], or [20] for examples.

content of the states s and t and $s \leq u$ expresses that the information of s is contained in the information of u . Then we obtain the following:

(\vee)' $s \models \alpha \vee \beta$ iff there are states t, u such that $t \models \alpha$, $u \models \beta$, and $t \circ u \leq s$.

For example, assume that information states are conceived as sets of possible worlds (as is common in epistemic logic [8]) and a formula is supported by an information state iff it is true (in the sense of classical logic) in every possible world of the state. Then the common content of two states coincides with union (a formula is true in all worlds of s and all worlds of t iff it is true in all worlds of $s \cup t$) and informational inclusion \leq coincides with the superset relation. It indeed holds in this setting that a disjunction is supported by an information state iff the state is a subset of the union of two states, t and u , such that t supports the first disjunct and u supports the second disjunct. For instance, one can take t to be the set of all worlds in which the first disjunct is true and u the set of all worlds in which the second disjunct is true.

Dually, information states can be represented as theories (of some standard logical system like classical logic, intuitionistic logic, relevant logic or fuzzy logic) and a formula is supported by a state iff it belongs to the corresponding theory. In that case the common content coincides with intersection and informational inclusion coincides with the subset relation. Then it holds that a state s supports a disjunction iff there are two theories t and u (e.g. the theories generated by the two disjuncts) such that t supports the first disjunct, u supports the second disjunct, and $t \cap u \subseteq s$.

The clause (\vee)' can be viewed as capturing the general structure of these cases. Our framework will be based on this clause. A similar semantic treatment of disjunction has been employed for example in [6, 9, 16, 19].⁴

As regards negation, we will adopt the semantic characterization in terms of a primitive incompatibility relation, following work by Goldblatt [11] on ortholattices, and Dunn [7] and Restall [22] on relevant and paraconsistent negations, among other researchers working in non-classical logic. The central idea is that an information state supports $\neg\alpha$ iff the state is incompatible with any state that supports α . As mentioned, negation interpreted in this manner has been studied in the context of non-classical logic before, but we'll combine this treatment of negation with the treatment of disjunction mentioned above. The main aim of this paper is to investigate abstract features of negation characterized in this way in the context of information based semantics involving the mentioned treatment of disjunction, proving new completeness, frame correspondence, and canonicity results. Our framework is similar, in certain respects, to that developed by Hartonas [14], but ours has the advantage of being more concrete and, in some ways, simpler, allowing for an easier interpretation of the resulting class of models. In particular, we seek to interpret the points in the frame as state of information, and the various relations and operations thereon as dealing directly

⁴ Interestingly, this style of interpretation has been recently adapted to provide *exact verification* clause for *conjunction* in the context of truthmaker semantics – see [10] for this style of semantics applied to intuitionistic logic.

with their information, and for this purpose the more concrete approach here is a better fit than the more abstract approach taken in [14].

2 Semilattice Semantics

In this section we introduce a semilattice semantics for the language L which is built out of a set of atomic formulas At by conjunction \wedge and disjunction \vee .⁵ A pair of L -formulas $\langle \alpha, \beta \rangle$ will be called a consequence L -pair. We will write $\alpha \vdash \beta$ instead of $\langle \alpha, \beta \rangle$. Our logical basis will be the so called lattice logic LL understood as a set of consequence L -pairs generated by the system consisting of the following axioms and rules:

$$\begin{array}{l} \alpha \vdash \alpha \quad \alpha \wedge \beta \vdash \alpha \quad \alpha \wedge \beta \vdash \beta \quad \alpha \vdash \alpha \vee \beta \quad \beta \vdash \alpha \vee \beta \\ \alpha \vdash \beta, \beta \vdash \gamma / \alpha \vdash \gamma \quad \alpha \vdash \beta, \alpha \vdash \gamma / \alpha \vdash \beta \wedge \gamma \quad \alpha \vdash \gamma, \beta \vdash \gamma / \alpha \vee \beta \vdash \gamma \end{array}$$

The distributive lattice logic DLL is generated by the system for LL enriched with the following axiom:

$$\alpha \wedge (\beta \vee \gamma) \vdash (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

For the semantic treatment of the logic we will use the following structures.

Definition 1. *An information frame (IF) is a structure $\langle S, \circ, i \rangle$, where S is a non-empty set, \circ is an associative, commutative, idempotent binary operation on S , and i is an element of S such that $s \circ i = s$.*

The set S represents a space of information states. The operation \circ assigns to any two states t, u the common (informational) content of t and u , i.e. a state $t \circ u$ consisting of information that the two states have in common. The element i represents the state of the absolute inconsistency (or triviality). As pointed out in the introduction, the operation \circ can be conceived of either as union (of possible worlds), or as intersection (of theories). Both perspectives make good sense, so it is open to interpretation whether \circ is understood as join or rather as meet. We will fix the latter perspective, so that we introduce the order of the structure in the following way: $s \leq t$ iff $s \circ t = s$. In this perspective, i is the top element of the structure.

Definition 2. *Let $I = \langle S, \circ, i \rangle$ be an IF. A filter in I is any subset F of S such that $i \in F$, and for any $t, u \in S$ it holds that $t \circ u \in F$ iff $t \in F$ and $u \in F$. A valuation in I is a function that assigns to each atomic formula from At a filter in I . Given a valuation V in I , the pair $\langle I, V \rangle$ is called an information model (IM).*

⁵ It should be noted that the phrase “semilattice semantics” is often used to refer to the semantic framework developed by Urquhart [24] for logics containing a relevant implication connective. While our proposed semantics could be extended to incorporate some of that machinery (following the lead of [16]), it is not included in the basic setting we investigate here, for the sake of clarity. Having said this, a conditional (relevant, intuitionistic, or otherwise) could be included without much difficulty.

Equivalently, one can define filters as non-empty subsets of S that are upward closed (w.r.t. \leq) and closed under \circ . Relative to a given IM, we can introduce the following semantic clauses for L -formulas:

- (a) $s \models p$ iff $s \in V(p)$, for any atomic formula p ,
- (b) $s \models \alpha \wedge \beta$ iff $s \models \alpha$ and $s \models \beta$,
- (c) $s \models \alpha \vee \beta$ iff there are states t, u such that $t \models \alpha$, $u \models \beta$, and $t \circ u \leq s$.

If $s \models \alpha$ we say that the state s supports the formula α . Let M be an IM, and α an L -formula. Then $\|\alpha\|_M$ is the set of states in M that support α (the subscript will usually be omitted).

Proposition 1. $\|\alpha\|_M$ is a filter in M , for every L -formula α .

We say that a consequence L -pair $\alpha \vdash \beta$ is valid in an IF I if $\|\alpha\| \subseteq \|\beta\|$, for any IM $M = \langle I, V \rangle$. The proof of the following proposition proceeds by a canonical model construction as in [19].

Proposition 2. A consequence L -pair $\alpha \vdash \beta$ is valid in all IFs if and only if it is derivable in the system LL.

Definition 3. A distributive information frame (DIF) is an IF satisfying the following condition: if $t \circ u \leq s$ then there are t', u' such that $t \leq t'$, $u \leq u'$, and $t' \circ u' = s$.

In the distributive information frames the semantic clause for disjunction can be reformulated as in the following proposition.

Proposition 3. Let M be an IM based on a DIF, and α, β any L -formulas. Then it holds for any state s in M :

$$s \models \alpha \vee \beta \text{ iff there are states } t, u \text{ such that } t \models \alpha, u \models \beta, \text{ and } t \circ u = s.$$

Proposition 4. A consequence L -pair $\alpha \vdash \beta$ is valid in all DIFs if and only if it is derivable in the system DLL.

3 Incompatibility Relation in Information Frames

In this section we focus on general features of negation in the context of semilattice semantics. Our approach is based on the treatment of negation that has been extensively used in the literature on non-classical logics and that characterizes negation in terms of the following plausible semantic clause: $\neg\alpha$ is supported by a state s if and only if any state supporting α is incompatible with s (see [7, 11, 22]). We extend L with negation (\neg) and denote the resulting language as L^\neg . Moreover, we enrich our models with a primitive incompatibility relation \perp .

Definition 4. An incompatibility proto-IF is a tuple $\langle S, \circ, i, \perp \rangle$, where $\langle S, \circ, i \rangle$ is an IF, and \perp is a binary relation on S , satisfying the following monotonicity condition: if $t \perp u$ and $t \leq s$ then $s \perp u$. For any $s \in S$, the set $\{t \in S \mid s \perp t\}$ will be denoted as \perp_s , and the set $\{t \in S \mid t \perp s\}$ as \perp^s . An incompatibility proto-IM is an incompatibility proto-IF equipped with a valuation.

The semantic clause for negation is defined as follows:

$s \models \neg\alpha$ iff for any t , if $t \models \alpha$ then $t \perp s$.

Proposition 5. *Let $I = \langle S, \circ, i, \perp \rangle$ be an incompatibility proto-IF. Then $\|\alpha\|$ is a filter in $\langle I, V \rangle$, for every valuation V and every L^\neg -formula α , if and only if \perp_t is a filter, for every $t \in S$.*

Definition 5. *An incompatibility proto-IF where \perp_t is a filter, for every $t \in S$, will be called simply an incompatibility IF (IIF, for short). A filter F in an IIF will be called an incompatibility filter if there is a state $t \in S$ such that $F = \perp_t$.*

Proposition 6. *For any IIM M and all L^\neg -formulas α, β , if $\alpha \vdash \beta$ is valid in M then $\neg\beta \vdash \neg\alpha$ is valid in M .*

Definition 6. *By a logic Λ , we will understand a set of consequence L^\neg -pairs that (a) is closed under uniform substitution (b) contains all the axioms of the lattice logic, (c) is closed under the rules of the lattice logic, and under the following rule:*

(N) $\alpha \vdash \beta / \neg\beta \vdash \neg\alpha$.

Instead of $\alpha \vdash \beta \in \Lambda$, we will just write $\alpha \vdash_\Lambda \beta$. Given a logic Λ , a Λ -theory is defined as a non-empty set of L^\neg -formulas Δ satisfying the following conditions: (a) if $\alpha \in \Delta$ and $\beta \in \Delta$ then $\alpha \wedge \beta \in \Delta$, (b) if $\alpha \in \Delta$, and $\alpha \vdash_\Lambda \beta$ then $\beta \in \Delta$.

Proposition 7. *For any logic Λ and all L^\neg -formulas α, β , $\neg\alpha \vee \neg\beta \vdash_\Lambda \neg(\alpha \wedge \beta)$ and $\neg(\alpha \vee \beta) \vdash_\Lambda \neg\alpha \wedge \neg\beta$.*

Now, we will show that our framework is flexible especially in that it allows us to define for any logic various alternative canonical models. Fix an arbitrary logic Λ . We will construct three alternative canonical models for Λ . The first two constructions use Λ -theories. First, we define the following structure:

$$I_1^\Lambda = \langle S_1, \circ_1, i_1, \perp_1 \rangle$$

where S_1 is the set of non-empty⁶ Λ -theories, $\Delta \circ_1 \Gamma = \Delta \cap \Gamma$, i_1 is the set of all L^\neg -formulas, and \perp_1 is defined as follows:

$\Delta \perp_1 \Gamma$ iff there is an $\alpha \in \Delta$ such that $\neg\alpha \in \Gamma$.

We also define $M_1^\Lambda = \langle I_1^\Lambda, V_1 \rangle$, where $\Delta \in V_1(p)$ iff $p \in \Delta$.

Proposition 8. *The structure I_1^Λ is an IIF and M_1^Λ is an IIM.*

Proposition 9. *For any Λ -theory Δ , and any L^\neg -formula α :*

$\Delta \models \alpha$ in M_1^Λ iff $\alpha \in \Delta$.

As a consequence, for all $\alpha, \beta \in L^\neg$, $\alpha \vdash_\Lambda \beta$ if and only if $\|\alpha\| \subseteq \|\beta\|$ in M_1^Λ .

⁶ The requirement of non-emptiness is used in the proof of the following proposition.

Proposition 10. *For any logic Λ there is a class of IIMs \mathcal{C} such that Λ coincides with all the consequence L^\neg -pairs valid in \mathcal{C} .*

Proposition 11. *The set of all consequence pairs that are valid in all IIFs coincides with the minimal logic, i.e. the logic obtained by axioms and rules of the lattice logic plus the rule (N).*

A negation connective obeying only (N) has been called a “subminimal negation”, following Hazen [15]. The previous construction of canonical models generalizes Goldblatt’s canonical models for orthologic [11]. Interestingly there is a quite different alternative way how to define the incompatibility relation among theories. This construction was employed by Hartonas in [14] and it works also for our setting (a more detailed discussion of the connection between our and Hartonas’ frameworks will be provided in the following section).

Definition 7. *For any Λ -theory Δ we define*

$$\Delta^* = \{\alpha \in L^\neg \mid \text{there is } \beta \in L^\neg \text{ s.t. } \beta \vdash_\Lambda \delta, \text{ for all } \delta \in \Delta, \text{ and } \neg\beta \vdash \alpha\}.$$

Note that Δ^* is again a (possibly empty) Λ -theory. Now, consider the following structure:

$$I_2^A = \langle S_2, \circ_2, i_2, \perp_2 \rangle$$

where, S_2 is the set of all Λ -theories (including the empty one), $\Delta \circ_2 \Gamma = \Delta \cap \Gamma$, i_2 is (as before) the set of all L^\neg -formulas, and $\Delta \perp_2 \Gamma$ iff $\Delta^* \subseteq \Gamma$. Moreover, we define $M_2^A = \langle I_2^A, V_2 \rangle$, where $V_2 = V_1$.

Proposition 12. *The structure I_2^A is an IIF and M_2^A is an IIM.*

Proposition 13. *For any Λ -theory Δ , and any L^\neg -formula α :*

$$\Delta \vDash \alpha \text{ in } M_2^A \text{ iff } \alpha \in \Delta.$$

As a consequence, for all $\alpha, \beta \in L^\neg$, $\alpha \vdash_\Lambda \beta$ if and only if $\|\alpha\| \subseteq \|\beta\|$ in M_2^A .

Our third construction of a canonical model is based on the structure of the Lindenbaum-Tarski algebra of Λ . Let $[\alpha]$ denote the equivalence class determined by α , i.e. $[\alpha]$ is a set of L^\neg -formulas β such that $\alpha \vdash_\Lambda \beta$ and $\beta \vdash_\Lambda \alpha$. Let $Prop_\Lambda = \{[\alpha] \mid \alpha \in L^\neg\}$. We distinguish two cases. First assume that Λ contains an explosive formula, i.e. some L^\neg -formula ξ such that $\xi \vdash \beta$ for all $\beta \in L^\neg$. Then we define:

$$I_3^A = \langle S_3, \circ_3, i_3, \perp_3 \rangle,$$

where $S_3 = Prop_\Lambda$, $[\alpha] \circ_3 [\beta] = [\alpha \vee \beta]$, $i_3 = [\xi]$, $[\alpha] \perp_3 [\beta]$ iff $\beta \vdash_\Lambda \neg\alpha$. So, the structure consists of the Lindenbaum-Tarski algebra of Λ turned upside down. We further define $M_3^A = \langle I_3^A, V_3 \rangle$, where $[\alpha] \in V_3(p)$ iff $\alpha \vdash_\Lambda p$.

If Λ contains no explosive formula, the structure has to be enriched with a top element so that the definition of the structure $I_3^A = \langle S_3, \circ_3, i_3, \perp_3 \rangle$ is complicated in the following way:

- $S_3 = Prop_A \cup \{i_3\}$, where i_3 is a new element, i.e. $i_3 \notin Prop_A$,
- for $[\alpha], [\beta] \in Prop_A$, $[\alpha] \circ_3 [\beta] = [\alpha \vee \beta]$,
and for all $s \in S_3$ we fix $s \circ_3 i_3 = i_3 \circ_3 s = s$,
- for $[\alpha], [\beta] \in Prop_A$, $[\alpha] \perp_3 [\beta]$ iff $\beta \vdash_A \neg\alpha$,
and for all $s \in S_3$, we state that $i_3 \perp_3 s$ and $s \perp_3 i_3$.

Moreover, $M_3^A = \langle I_3^A, V_3 \rangle$, where $s \in V_3(p)$ iff $s = i_3$ or $s = [\alpha]$ and $\alpha \vdash_A p$. The following two propositions hold for both these constructions.

Proposition 14. *The structure I_3^A is an IIF and M_3^A is an IIM.*

Proposition 15. *For all L^\neg -formulas α, β :*

$$[\alpha] \models \beta \text{ in } M_3^A \text{ iff } \alpha \vdash_A \beta.$$

As a consequence, for $\alpha, \beta \in L^\neg$, $\alpha \vdash_A \beta$ if and only if $\|\alpha\| \subseteq \|\beta\|$ in M_3^A .

Definition 8. *We say that a set of consequence L^\neg -pairs K characterizes a class of IIFs \mathcal{C} if it holds for any IIF I that $I \in \mathcal{C}$ iff all L^\neg -pairs from K are valid in I . A consequence L^\neg -pair $\alpha \vdash \beta$ characterizes a class \mathcal{C} if $\{\alpha \vdash \beta\}$ characterizes \mathcal{C} .*

Proposition 16. *The following facts hold:*

- (a) $p \vdash \neg\neg p$ characterizes the class of IIFs with symmetric incompatibility relation⁷,
- (b) $\neg\neg p \vdash p$ characterizes the class of IIFs satisfying the following property: for every filter F and every state $s \notin F$ there is a state t such that $s \notin \perp_t$ and $F \subseteq \perp^t$.
- (c) $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ characterizes the class of IIFs where \perp is symmetric and where every filter is the intersection of a set of incompatibility filters, i.e., for every filter F there is a set of states $X \subseteq S$ such that $F = \bigcap_{s \in X} \perp_s$.
- (d) $\neg p \wedge \neg q \vdash \neg(p \vee q)$ characterizes the class of IIFs satisfying the following property: for any states s, t, u , if $u \perp s$ and $v \perp s$ then $u \circ v \perp s$.
- (e) $\neg(p \wedge q) \vdash \neg p \vee \neg q$ characterizes the class of IIFs satisfying the following property: for any state s and any filters F, G , if $F \cap G \subseteq \perp^s$ then there are states t, u such that $F \subseteq \perp^t$, $G \subseteq \perp^u$ and $t \circ u \leq s$.
- (f) $p \wedge \neg p \vdash q$ characterizes the class of IIFs satisfying the following property: for any state s , if $s \perp s$ then $s = i$.
- (g) $p \vdash q \vee \neg q$ characterizes the class of IIFs satisfying the following property: for any filter F and any state s , there are states t, u s.t. $t \in F$, $u \perp v$ for all $v \in F$, and $t \circ u \leq s$.

Definition 9. *Let $n \in \{1, 2, 3\}$ and let K be a set of consequence L^\neg -pairs that characterizes a class of IIFs \mathcal{C} . We say that K is canonical _{n} if for every logic Λ such that $K \subseteq \Lambda$, the frame I_n^A is in the class \mathcal{C} . A consequence L^\neg -pair $\alpha \vdash \beta$ is canonical _{n} if $\{\alpha \vdash \beta\}$ is canonical _{n} .*

⁷ Note that $\alpha \vdash \neg\neg\alpha$ is interderivable with the rule $\alpha \vdash \neg\beta / \beta \vdash \neg\alpha$, so ‘symmetric incompatibility’ frames are also characterized by this rule.

Note that canonicity gives us completeness results in the following sense: if a consequence L^\neg -pair $\alpha \vdash \beta$ is canonical _{n} for at least one $n \in \{1, 2, 3\}$ then the lattice logic enriched with $\alpha \vdash \beta$ is complete with respect to the class of IIFs that $\alpha \vdash \beta$ characterizes. More generally, if $\{\alpha_j \vdash \beta_j \mid j \in J\}$ is a set of consequence L^\neg -pairs such that for each $j \in J$, $\alpha_j \vdash \beta_j$ is canonical _{n} (for at least one $n \in \{1, 2, 3\}$) and characterizes a class of IIFs \mathcal{C}_j then the lattice logic enriched with $\{\alpha_j \vdash \beta_j \mid j \in J\}$ is complete with respect to the class $\bigcap_{j \in J} \mathcal{C}_j$. Here are some examples of canonicity.

Proposition 17. *Each of the consequence L^\neg -pairs $p \vdash \neg\neg p$, $\neg p \wedge \neg q \vdash \neg(p \vee q)$, $p \wedge \neg p \vdash q$ is canonical₁ and canonical₃.*

The previous proposition cannot be stated for canonicity₂. See for instance the proof of Proposition 18 in the appendix that shows that I_2^A for a logic A containing $p \vdash \neg\neg p$ does not have a symmetric incompatibility relation. As regards the cases with ‘non-first order conditions’, i.e. (b), (c), (e), and (g) of Proposition 16, we will focus only on the double negation law (c) and leave the remaining cases for future research. We also leave for future research whether it can be actually proved that these consequence L^\neg -pairs *cannot* be characterized by first-order conditions (for a related result, see [12]). However, we will show that double negation is not canonical for each of our canonical model constructions. Especially the case of non-canonicity₁ is interesting.

Proposition 18. *For each $n \in \{1, 2, 3\}$, $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ is not canonical _{n} .*

In the next section, we provide a comparison with Hartonas’ semantics. To facilitate the comparison we need another definition. This definition can be motivated by non-canonicity of double negation law, or by the fact that the algebra of propositions in an IIF, i.e. the algebra of possible values of formulas in the IIF, is always a complete lattice with an additional negation operation (to be more specific, it is always an algebraic lattice with a negation operation). To allow also for lattices that are not complete, we can restrict the space of possible valuations in the style of the so-called general frames known from modal logic (see [3, §1.4]). Given an IIF, let us define the following two operations on filters corresponding to the algebraic counterparts of disjunction and negation:

$$F \sqcup G = \{s \in S \mid \text{there are } t \in F \text{ and } u \in G \text{ such that } t \circ u \leq s\},$$

$$\neg F = \{s \in S \mid \text{for all } t \in F, t \perp s\}.$$

Definition 10. *A general IIF (GIIF, for short) is a pair $J = \langle I, P \rangle$ where I is an IIF, and P is a set of filters in I closed under the operations \cap, \sqcup, \neg . A valuation V in J is a valuation in I that assigns to every atomic formula an element of P . A GIIF equipped with a valuation is called a general IIM (GIIM).*

Note that every logic A is sound and complete for instance with respect to the GIIF obtained from I_3^A by restricting the space of valuations to principal filters, or with respect to the GIIFs obtained from I_1^A or I_2^A by restricting the valuations

to principal filters that are generated by a theory that is in turn generated by a single formula.

Before moving on, let's pause to note some points of difference between our semantics and the incompatibility semantics discussed by Dunn [7], and others. Restricted to language L^\neg , the difference comes down to two sequents which are immediate consequences of Dunn's approach, but which are not generally valid in our framework. These are distribution $\alpha \wedge (\beta \vee \gamma) \vdash (\alpha \wedge \beta) \vee \gamma$ and $\neg\alpha \wedge \neg\beta \vdash \neg(\alpha \vee \beta)$. Both of these are consequences of working only with 'prime' states, for which the usual disjunction truth condition (\vee) obtains. By moving to our semantic setting, we can reject these sequents, and perhaps there are compelling philosophical reasons to want to do so. For instance, there are well-rehearsed reasons to reject distribution coming out of the literature on quantum logic (see [21] for a classic philosophical discussion), and logics without distribution have also been defended in connection with understanding logical consequence in terms of meaning containment (see [4]).

4 Comparison with Hartonas' Approach

In this section we compare our semantics with the one presented in Hartonas' [14], which is based on his duality theory for lattices developed in [13]. We first reconstruct Hartonas' approach based on his neighborhood frames. Let $X \neq \emptyset$, and $\nu : X \rightarrow \mathcal{P}(X)$. The function ν determines the following two maps on $\mathcal{P}(X)$ defined as follows:

$$\begin{aligned} \lambda U &= \{x \in X \mid \text{for all } u \in U : x \in \nu(u)\}, \\ \rho U &= \{x \in X \mid \text{for all } u \in U : u \in \nu(x)\}. \end{aligned}$$

These maps form a Galois connection and their composition determines a closure operator $\Gamma_\nu = \lambda\rho$. The sets satisfying $U = \Gamma_\nu(U)$ are called ν -stable sets. Now we can introduce Hartonas neighborhood frames (models).

Definition 11. *A Hartonas neighborhood frame (HNF) is any structure $N = \langle X, Y, \iota, \nu, \perp \rangle$, where X is a non-empty set, $Y \subseteq X$, $\iota \in X$, $\nu : X \rightarrow \mathcal{P}(X)$, and $\perp \subseteq X \times X$ such that the following conditions are satisfied:*

- (a) $\nu(x) = \Gamma_\nu(\{x\})$, for any $x \in X$,
- (b) for any ν -stable $U \subseteq X$ there is $x \in X$ such that $U = \nu(x)$,
- (c) the ν -stable sets generated by the points of Y form a bounded sublattice of the complete lattice of ν -stable sets where the top element is X and the bottom element is the singleton set $\{\iota\}$; the elements of the sublattice are called regular subsets of X ,
- (d) the neighborhood function imposes a partial order on the carrier set of a frame by letting $x \leq_\nu y$ iff $\nu(y) \subseteq \nu(x)$,
- (e) if U is regular, then $\{x \in X \mid \text{for all } u \in U : u \perp x\}$ is also regular.

A valuation in N is a function that assigns to each atomic formula from At a regular subsets of X . Given a valuation V in N , the pair $\langle N, V \rangle$ is called a Hartonas neighborhood model (HNM).

Proposition 19. *In any HNF, the ν -stable sets are exactly the principal upsets with respect to the ordering \leq_ν and ν is a bijection between X and the lattice of ν -stable sets such that $\nu(x) = \{y \in X \mid x \leq_\nu y\}$, for every $x \in X$. Moreover, $\nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y)))$ is the greatest lower bound of x and y , and ι the top element w.r.t. \leq_ν .*

Relative to a given HNM, a relation of support \Vdash between the elements of X and L^- -formulas is defined in Hartonas' semantics as follows:

- (a) $x \Vdash p$ iff $x \in V(p)$, for any atomic formula p ,
- (b) $x \Vdash \alpha \wedge \beta$ iff $x \Vdash \alpha$ and $x \Vdash \beta$,
- (c) $x \Vdash \alpha \vee \beta$ iff $x \in \nu(y)$ for each y s.t. for any z if $z \Vdash \alpha$ or $z \Vdash \beta$ then $z \in \nu(y)$,
- (d) $x \Vdash \neg\alpha$ iff for any y , if $y \Vdash \alpha$ then $y \perp x$.

Hartonas' condition for disjunction, which is somewhat unintuitive and complicated, on first blush, is designed to lead to the following equivalence:

$$x \Vdash \alpha \vee \beta \text{ iff } x \in \Gamma_\nu(\{y \in X \mid y \Vdash \alpha \text{ or } y \Vdash \beta\}).$$

The following proposition is a simple consequence of the definitions.

Proposition 20. *Let M be a HNM. Then $\{x \in X \mid x \Vdash \alpha \text{ in } M\}$ is regular, for every L^- -formula α .*

Definition 12. *A HNF (HNM) is called normal (NHNF, NHNM) if it satisfies the following two conditions:*

- (a) *for every $x \in X$, the set $\{y \in X \mid x \perp y\}$ is ν -stable,*
- (b) *for every $x, y, z \in X$, if $z \in \nu(x)$ and $x \perp u$ then $z \perp u$.*

In the light of Proposition 19 we define $x \circ^\nu y = \nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y)))$. Now we can formulate the claim that NHNFs and NHNMs can be viewed, respectively, as special cases of GIIFs and GIIMs.

Proposition 21. *Let $N = \langle X, Y, \nu, \iota, \perp \rangle$ be a NHNF. Then the structure $I = \langle X, \circ^\nu, \iota, \perp \rangle$ is an IIF and $N' = \langle I, \nu(Y) \rangle$ is a GIIF. The valuations in N in the sense of the Definition 11 are exactly the valuations in N' in the sense of Definition 10. Moreover, for any such valuation V , any $x \in X$ and $\alpha \in L^-$ it holds that $x \Vdash \alpha$ in $\langle N, V \rangle$ iff $x \models \alpha$ in $\langle N', V \rangle$.*

We have shown that normal HNMs correspond exactly to particular cases of our GIIMs. One can observe that the restriction to normal HNMs is not a serious limitation. Hartonas' completeness proof [14, Theorem 3.12] proceeds by the canonical model method. The canonical model that Hartonas uses for a logic A is the structure $N^A = \langle X^A, Y^A, \iota^A, \nu^A, \perp^A \rangle$ where X^A is the set of all A -theories, Y^A is the set of those A -theories that are generated by a single formula, ι^A is the set of all formulas, $\Gamma \in \nu^A(\Delta)$ iff $\Delta \subseteq \Gamma$, and \perp^A is defined in the same way as our \perp_2 . In general the model N^A corresponds structurally with our M_2^A . Note that for any A , N^A is normal in the sense of Definition 12.

Hartonas also formulates an interesting modification of the framework presented above. In this modified semantics negation is captured in an operational way using an operation that we might call Hartonas star to contrast it with the so-called Routley star that is often used especially in relevant logics instead of the incompatibility relation (see [7]). As Hartonas points out, the Routley star is motivated by the fact that one can define a corresponding operation on prime theories in the canonical model. In order to work properly the Routley star requires ‘primeness’ and thus it is not suitable for the approaches based on theories rather than prime theories. To be more specific, there is no operation corresponding directly to Routley star that could be defined in canonical models based on all theories. Hartonas shows that his star operation is a feasible alternative for Routley star suitable for theory-based approaches. We have used this operation when we defined the incompatibility relation \perp_2 in the canonical model M_2^A .

The construction of the canonical model M_2^A within our framework actually shows that the Hartonas star can be adapted to our setting. Let us finish this section with a brief discussion of how this can be done.

Definition 13. *An information frame with a Hartonas star (IFHS) is a tuple $\langle S, \circ, i, * \rangle$, where $\langle S, \circ, i \rangle$ is an IF, and $*$ is an arbitrary unary operation on S . An information model with a Hartonas star (IMHS) is an IFHS equipped with a valuation.*

Relative to a IMHS, the semantic clauses for conjunction and disjunction are as before and the semantic clause for negation is as follows:

$$s \models \neg\alpha \text{ iff for any } t, \text{ if } t \models \alpha \text{ then } t^* \leq s.$$

This clause is less intuitive than the clause using incompatibility relation. This is however similar to the case of Routley star which is also difficult to justify on the intuitive basis (though see [22] for an attempt). Nevertheless, this treatment of negation has some technical merits.

In Hartonas’ framework, the star operation is regulated by several constraints. In our framework no restrictions are needed and any operation is admissible which is supported by the following two propositions.

Proposition 22. *For any IMHS M and any $\alpha \in L^\neg$, $\|\alpha\|_M$ is a filter in M .*

Proposition 23. *For any IMHS M and all $\alpha, \beta \in L^\neg$, if $\alpha \vdash \beta$ is valid in M then $\neg\beta \vdash \neg\alpha$ is valid in M .*

The previous proposition together with the fact that for any logic A the canonical model M_2^A can be viewed as an IMHS give us the following result.

Proposition 24. *For any logic A there is a class of IMHSs \mathcal{C} such that A coincides with all the consequence L^\neg -pairs valid in \mathcal{C} . Moreover, the set of all consequence pairs that are valid in all IFHSs coincides with the minimal logic, i.e. the logic obtained by axioms and rules of the lattice logic plus the rule (N).*

This comparison between our proposed semantics and Hartonas' is illustrative in showcasing some nice properties of our approach. In addition to this, however, we think that the comparison also speaks in favour of our framework at an intuitive and conceptual level. In particular, we think the use of \circ to interpret disjunction makes intuitive sense, and fits nicely into an approach to relational semantics which takes *information states* to be the basic entities. That we can obtain a similar kind of generality to Hartonas' framework, employing neighbourhood semantic machinery, suggests that our approach retains some of the generality of his framework, while, perhaps, employing intuitively clearer structures.

A Appendix

Proof (Proposition 5). First, assume that \perp_t is a filter, for every $t \in S$. We show the inductive step for negation, that is, we will assume that $\|\alpha\|$ is a filter, and show that then $\|\neg\alpha\|$ is a filter as well. It holds that $i \models \neg\alpha$, since for any $t \in \|\alpha\|$, $i \in \perp_t$. Next, assume that $s \models \neg\alpha$ and $s \leq t$. Then if $u \models \alpha$ then $s \in \perp_u$, and thus $t \in \perp_u$. So, $t \models \neg\alpha$. Assume that $s \models \neg\alpha$ and $t \models \neg\alpha$. Then for any $u \in \|\alpha\|$, $s \in \perp_u$ and $t \in \perp_u$, and so $s \circ t \in \perp_u$. Thus $s \circ t \models \neg\alpha$.

Second, assume that for some $t \in S$, \perp_t is not a filter. We will show that there is a valuation V in I , and a formula α such that $\|\alpha\|$ is not a filter in $\langle I, V \rangle$. We consider three cases: (a) $i \notin \perp_t$; (b) there are $s, u \in S$ such that $s \leq u$, $s \in \perp_t$, and $u \notin \perp_t$; (c) there are $s, u \in \perp_t$ such that $s \circ u \notin \perp_t$.

Consider a valuation V such that $V(p) = \{v \in S \mid t \leq v\}$. In the case (a), $t \models p$ but $i \notin \perp_t$. So, $i \notin \|\neg p\|$. In the case (b), if $v \models p$ then $t \leq v$, and thus $s \in \perp_v$. So, $s \models \neg p$. But $u \not\models \neg p$, for $t \models p$ and $u \notin \perp_t$. Hence, $\|\neg p\|$ is not a filter. In the case (c) $s \models \neg p$ and $u \models \neg p$ but $s \circ u \not\models \neg p$. Hence, again, $\|\neg p\|$ is not a filter. \square

Proof (Proposition 9). The proof is by induction on the complexity of α , and the only case we will consider is that where $\alpha = \neg\beta$. The right-to-left direction is immediate, so suppose that $\neg\beta \notin \Delta$. Fix $\Gamma = \{\gamma \mid \beta \vdash_A \gamma\}$, and note that $\Gamma \in S_1$ and $\Gamma \models \beta$. Suppose that $\Gamma \perp_1 \Delta$ holds, so that for some $\gamma \in \Gamma$, $\neg\gamma \in \Delta$; in that case, $\beta \vdash \gamma$, and so $\neg\gamma \vdash \neg\beta$, and thus $\neg\beta \in \Delta$, contrary to the assumption. Thus $\Delta \not\models \neg\beta$, as desired. \square

Proof (Proposition 13). Again the proof is by induction on the complexity of α . The inductive step for negation amounts to Lemma 3.13 in [14]. Since our notation is quite different, let us reconstruct this step. For the left-to-right direction assume that $\Delta \models \neg\beta$. Then for any $\Gamma \in S_2$, if $\Gamma \models \beta$, i.e. $\beta \in \Gamma$, then $\Gamma^* \subseteq \Delta$. Let us fix $\Gamma = \{\gamma \mid \beta \vdash_A \gamma\}$. Then $\gamma \in \Gamma^*$ iff for some δ , $\delta \vdash_A \beta$ and $\neg\delta \vdash_A \gamma$, which is equivalent to $\neg\beta \vdash_A \gamma$. Hence, $\Gamma^* = \{\gamma \mid \neg\beta \vdash_A \gamma\}$ and thus $\neg\beta \in \Delta$. For the right-to-left direction assume that $\neg\beta \in \Delta$. Take any Γ such that $\Gamma \models \beta$ and thus $\beta \in \Gamma$. Let $\gamma \in \Gamma^*$. Then there is δ such that $\delta \vdash_A \beta$ and $\neg\delta \vdash_A \gamma$. Hence, $\neg\beta \vdash_A \gamma$ and $\gamma \in \Delta$. We have shown that $\Gamma^* \subseteq \Delta$, i.e. $\Gamma \perp_2 \Delta$. It follows that $\Delta \models \neg\beta$. \square

Proof (Proposition 15). The inductive step for negation can be proved as follows: $[\alpha] \models \neg\beta$ iff for all γ , if $[\gamma] \models \beta$ then $[\gamma] \perp_3 [\alpha]$ iff for all γ , if $\gamma \vdash_A \beta$ then $\alpha \vdash_A \neg\gamma$ iff $\alpha \vdash_A \neg\beta$. \square

Proof (Proposition 16). (b) First, take any IIF I in which the condition is satisfied, i.e. for every filter F and every state $s \notin F$ there is a state t such that $s \notin \perp_t$ and $F \subseteq \perp^t$. Assume $s \not\equiv p$, i.e. $s \notin V(p)$. Thus, there is t such that $s \notin \perp_t$ and $V(p) \subseteq \perp^t$. Then $t \models \neg p$ (any state supporting p is incompatible with t) but since it is not the case that $t \perp s$ we obtain $s \not\equiv \neg\neg p$. We have shown that $\neg\neg p \vdash p$ is valid in I . Second, take an IIF I in which the condition is not satisfied, i.e. there is a filter F and a state $s \notin F$ such that for every state t , if $F \subseteq \perp^t$ then $s \in \perp_t$. Consider a valuation V such that $V(p) = F$. Then $s \not\equiv p$. Moreover, if $t \models \neg p$, then $V(p) \subseteq \perp^t$, and, consequently $t \perp s$. It follows that $s \models \neg\neg p$. We have shown that $\neg\neg p \vdash p$ is not valid in I .

(c) First, note that if \perp is symmetric, $\perp_t = \perp^t$ for any state t . Hence, it follows from (a) and (b) that $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ characterizes the class of IIFs where (i) the compatibility relation is symmetric and (ii) for any filter F and any state $s \notin F$ there is an incompatibility filter G such that $F \subseteq G$ and $s \notin G$. Now it suffices to show that (ii) is equivalent to the condition stating that every filter is the intersection of a set of incompatibility filters. Let $UpF = \{G \mid G \text{ is an incompatibility filter such that } F \subseteq G\}$. It is obvious that $F \subseteq \bigcap UpF$. It holds that there is F that is not the intersection of any set of incompatibility filters iff there is F that is a proper subset of $\bigcap UpF$ iff there is F and $s \notin F$ such that $s \in \bigcap UpF$ iff (ii) does not hold. \square

Proof (Prop. 17). We will just consider the case of $\neg p \wedge \neg q \vdash \neg(p \vee q)$ – the others are similar, and this is the axiom that distinguishes our basic setting from that employing a standard treatment of disjunction (and all ‘prime’ states). For canonicity₁, suppose that $\Gamma, \Sigma, \Delta \in S_1$, $\Gamma \perp_1 \Delta$, and $\Sigma \perp_1 \Delta$. It follows that there are $\alpha \in \Gamma$ and $\beta \in \Sigma$ such that $\neg\alpha, \neg\beta \in \Delta$. Thus $\neg\alpha \wedge \neg\beta \in \Delta$, and thus $\neg(\alpha \vee \beta) \in \Delta$. It is immediate that $\alpha \vee \beta \in \Gamma \circ_1 \Sigma = \Gamma \cap \Sigma$, and thus $\Gamma \circ_1 \Sigma \perp_1 \Delta$, as desired. For canonicity₃, we’ll work in the more complex setting, not assuming that there is an explosive formula. To that end, suppose that $s, t, u \in S_3$, $s \perp_3 u$, and $t \perp_3 u$. If any of s, t, u is i_3 then we’re done, as $i_3 \perp_3 v$, $v \perp_3 i_3$, and $v \circ_3 i_3 = i_3 \circ_3 v = i_3$ all hold for every v , so suppose that none is: i.e., that there are $\alpha, \beta, \gamma \in Prop_A$ such that $s = [\alpha]$, $t = [\beta]$, and $u = [\gamma]$. By the supposition, then, $\gamma \vdash \neg\alpha$ and $\gamma \vdash \neg\beta$, and thus $\gamma \vdash \neg\alpha \wedge \neg\beta$. It follows that $\gamma \vdash \neg(\alpha \vee \beta)$, and thus, since $s \circ_3 t = [\alpha \vee \beta]$, $s \circ_3 \perp_3 u$, as desired. \square

Proof (Proposition 18). First, we will show that $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ is not canonical₁. Let \mathcal{A} be the distributive lattice logic enriched with (N) and both $\neg\neg p \vdash p$ and $p \vdash \neg\neg p$. We will show the following: There is a filter of non-empty \mathcal{A} -theories F (i.e. a set of non-empty \mathcal{A} -theories that is closed under intersection and stronger \mathcal{A} -theories) and there is a non-empty \mathcal{A} -theory $\Delta \notin F$ such that $\Delta \perp_1 \Omega$ for all Ω such that $\Gamma \perp_1 \Omega$ for each $\Gamma \in F$. This will imply that F cannot be expressed as an intersection of incompatibility filters. Let us construct F and Δ satisfying the desired property. Take for each $i \in \mathbb{N}$ an infinite set of atomic

formulas $X_i = \{p_1^i, p_2^i, \dots\}$, assuming that if $i \neq j$ then X_i and X_j are disjoint. Let Γ_i be the Λ -theory generated by X_i . Let F be the filter generated by the set of Λ -theories $\{\Gamma_1, \Gamma_2, \dots\}$. Now we can construct Δ . Let Y be the set of all disjunctions of atomic formulas from $\bigcup_{i \in \mathbb{N}} X_i$ satisfying the following:

$p_{j_1}^{i_1} \vee \dots \vee p_{j_n}^{i_n} \in Y$ iff all i_1, \dots, i_n are distinct and it is not the case that there is k such that $j_1 = \dots = j_n = k$ and $i_1, \dots, i_n \leq k$.

This definition can be illustrated with the following table. Y contains all disjunctions of atomic formulas from different columns except those that are connected by a path built from the horizontal lines.

Γ_1	Γ_2	Γ_3	Γ_4	\dots
p_1^1	p_1^2	p_1^3	p_1^4	\dots
p_2^1	— p_2^2	p_2^3	p_2^4	\dots
p_3^1	— p_3^2	— p_3^3	p_3^4	\dots
p_4^1	— p_4^2	— p_4^3	— p_4^4	\dots
\vdots	\vdots	\vdots	\vdots	\ddots

Let Δ be the Λ -theory generated by Y . Let $\Gamma \in F$. Then there are $\Gamma_{i_1}, \dots, \Gamma_{i_n}$ such that $\Gamma_{i_1} \cap \dots \cap \Gamma_{i_n} \subseteq \Gamma$. Take $k = \max\{i_1, \dots, i_n\}$. Then $p_k^1 \vee \dots \vee p_k^n \in \Gamma$ but $p_k^1 \vee \dots \vee p_k^n \notin \Delta$, and thus $\Delta \neq \Gamma$. We have shown that $\Delta \notin F$.

Now assume $\Gamma \perp_1 \Omega$ for each $\Gamma \in F$. Then $p_{j_1}^1 \wedge \dots \wedge p_{j_n}^1 \vdash_{\Lambda} \alpha$ for some $p_{j_1}^1, \dots, p_{j_n}^1 \in X_1$ and some α such that $\neg\alpha \in \Omega$. Then $\neg(p_{j_1}^1 \wedge \dots \wedge p_{j_n}^1) \in \Omega$. Take $k = \max\{j_1, \dots, j_n\} + 1$. There must be some $p_{i_1}^k, \dots, p_{i_m}^k \in X_k$ such that $\neg(p_{i_1}^k \wedge \dots \wedge p_{i_m}^k) \in \Omega$. Hence $\neg((p_{j_1}^1 \wedge \dots \wedge p_{j_n}^1) \vee (p_{i_1}^k \wedge \dots \wedge p_{i_m}^k)) \in \Omega$. Moreover, k was selected in such a way that it also holds that $p_j^1 \vee p_i^k \in \Delta$, for each $j \in \{j_1, \dots, j_n\}$ and $i \in \{i_1, \dots, i_m\}$. Then $(p_{j_1}^1 \wedge \dots \wedge p_{j_n}^1) \vee (p_{i_1}^k \wedge \dots \wedge p_{i_m}^k) \in \Delta$, by distributivity, and thus $\Delta \perp_1 \Omega$ as desired.

Second, we will show that $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ is not canonical₂. Let Λ be the lattice logic enriched with (N) and both $\neg\neg p \vdash p$ and $p \vdash \neg\neg p$. We will show that \perp_2 is not symmetric. Let Γ be the Λ -theory generated by $\{p\}$ and Δ the Λ -theory generated by an infinite set of atomic formulas $X = \{q_1, q_2, \dots\}$ assuming that $p \notin X$. Then Γ^* is the Λ -theory generated by $\{\neg p\}$ and $\Delta^* = \emptyset$ (for there is no β such that $\beta \vdash_{\Lambda} \delta$ for each $\delta \in \Delta$). Hence, $\Delta \perp_2 \Gamma$ but not $\Gamma \perp_2 \Delta$.

Finally, we will show that $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ is not canonical₃. Let Λ be again the lattice logic enriched with (N) and with $\neg\neg p \vdash p$ and $p \vdash \neg\neg p$. Let $X = \{q_1, q_2, q_3, \dots\}$ be an infinite set of atomic formulas such that $p \notin X$. Let F be the filter in I_3^A generated by $\{[q_1], [q_2], [q_3], \dots\}$. Recall that I_3^A is given by the Lindenbaum-Tarski algebra turned upside down and enriched with a new top element i_3 . It holds that $[\alpha] \in F$ iff there are q_{i_1}, \dots, q_{i_n} such that $\alpha \vdash_{\Lambda} q_{i_1} \vee \dots \vee q_{i_n}$. It follows that $[p] \notin F$. In order to show that $\{p \vdash \neg\neg p, \neg\neg p \vdash p\}$ is not canonical₃ it is sufficient to observe that for any $t \in S_3$, if $F \subseteq \perp_3^t$ then

$[p]_{\perp_3} t$. Note that there is no β such that $\beta \vdash_A \neg\alpha$ for all $\alpha \in F$. This follows for example from the fact that A has the variable sharing property and so for any β we can take any q_i not occurring in β and then $\beta \not\vdash_A \neg q_i$. It follows that $F \subseteq \perp_3^t$ only if $t = i_3$. But it holds $[p]_{\perp_3} i_3$ which finishes the proof. \square

Proof (Proposition 19). First, we show that $\nu(x) = \{y \in X \mid x \leq_\nu y\}$. By the definition of Γ_ν as $\lambda\rho$ it generally holds that if $y \in \Gamma_\nu(\{x\})$ then $\Gamma_\nu(\{y\}) \subseteq \Gamma_\nu(\{x\})$. By (a) and (d) of Definition 11 we obtain: if $y \in \nu(x)$ then $x \leq_\nu y$, i.e. $\nu(x) \subseteq \{y \in X \mid x \leq_\nu y\}$. On the other side, if $x \leq_\nu y$ then $\nu(y) \subseteq \nu(x)$, and since $y \in \nu(y)$ (because in general $y \in \lambda\rho(\{y\})$), it follows that $y \in \nu(x)$, and thus $\{y \in X \mid x \leq_\nu y\} \subseteq \nu(x)$. Since any ν -stable set is of the form $\nu(x)$, for some $x \in X$, ν -stable sets are exactly the principle upsets w.r.t. \leq_ν . Since \leq_ν is a partial order, ν is a bijection between X and the lattice of ν -stable sets.

By (c) of Definition 11 we obtain $\{\iota\} \subseteq \nu(x)$, for every $x \in X$, which means (by (d)) that ι the top element w.r.t. \leq_ν . It remains to be shown that $\nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y)))$ is the greatest lower bound of x and y w.r.t. \leq_ν . It holds that $\nu(x) \subseteq \Gamma_\nu(\nu(x) \cup \nu(y))$ and thus $\nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y))) \leq_\nu x$. By the same reasoning $\nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y))) \leq_\nu y$, so $\nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y)))$ is a lower bound of $\{x, y\}$. Assume that $z \leq_\nu x$ and $z \leq_\nu y$. Then $\nu(x) \subseteq \nu(z)$ and $\nu(y) \subseteq \nu(z)$. So, $\nu(x) \cup \nu(y) \subseteq \nu(z)$ and since $\nu(z)$ is ν -stable we obtain $\Gamma_\nu(\nu(x) \cup \nu(y)) \subseteq \nu(z)$. It follows that $z \leq_\nu \nu^{-1}(\Gamma_\nu(\nu(x) \cup \nu(y)))$ which is what we needed to show. \square

Proof (Proposition 21). Let $N = \langle X, Y, \nu, \iota, \perp \rangle$ be a NHNF. Proposition 19 shows that $\langle X, \circ^\nu, \iota \rangle$ is an IF. Moreover, the conditions (a) and (b) from Definition 12 guarantee that $I = \langle X, \circ^\nu, \iota, \perp \rangle$ is an IIF. Since $\nu(x)$ is always a (principal) filter, the valuations in N are always the valuations in I . Let V be a valuation in N (in the sense of Definition 11). We will show by induction that for any $\alpha \in L^\neg$ it holds that $x \Vdash \alpha$ in $\langle N, V \rangle$ iff $x \vDash \alpha$ in $\langle I, V \rangle$. The case of atomic formulas and the inductive steps for conjunction and negation are immediate. Let us show the inductive step for disjunction. Assume that the claim holds for some L^\neg -formulas α, β . It follows from Propositions 19 and 20 that there are states z_α, z_β such that $\nu(z_\alpha) = \{x \in X \mid x \Vdash \alpha\}$ and $\nu(z_\beta) = \{x \in X \mid x \Vdash \beta\}$. Now we can observe that the following claims are equivalent:

- $x \vDash \alpha \vee \beta$,
- there are $u, v \in X$ such that $u \vDash \alpha$, $v \vDash \beta$ and $u \circ^\nu v \leq_\nu x$,
- there are $u, v \in X$ such that $u \Vdash \alpha$, $v \Vdash \beta$ and $\nu^{-1}(\Gamma_\nu(\nu(u) \cup \nu(v))) \leq_\nu x$,
- there are $u, v \in X$ such that $u \Vdash \alpha$, $v \Vdash \beta$ and $\nu(x) \subseteq \Gamma_\nu(\nu(u) \cup \nu(v))$,
- $\nu(x) \subseteq \Gamma_\nu(\nu(z_\alpha) \cup \nu(z_\beta))$,
- $x \in \Gamma_\nu(\nu(z_\alpha) \cup \nu(z_\beta))$,
- $x \Vdash \alpha \vee \beta$.

It follows that $\nu(Y)$ is closed under the required operation and thus $N' = \langle I, \nu(Y) \rangle$ is a GIIF such that the valuations in N coincide with the valuations in N' , which finishes the proof. \square

References

1. Barwise, J., Perry, J.: *Situations and Attitudes*. MIT Press, Cambridge (1983)
2. Beth, E.W.: Semantic construction of intuitionistic logic. *Mededlingen Koninklijke Nederlandse Akad. Wetenschappen Afd. Letterkunde ns* **19**(11), 357–388 (1956)
3. Blackburn, P., de Rijke, M., Venema, Y.: *Modal Logic*. Cambridge University Press, Cambridge (2001)
4. Brady, R.T., Meinander, A.: Distribution in the logic of meaning containment and in quantum mechanics. In: Tanaka, K., Berto, F., Mares, E., Paoli, F. (eds.) *Paraconsistency: Logic and Applications*, vol. 26, pp. 223–255. Springer, Dordrecht (2013). https://doi.org/10.1007/978-94-007-4438-7_13
5. Copeland, B.J.: On when a semantics is not a semantics: some reasons for disliking the Routley-Meyer semantics for relevance logic. *J. Philos. Log.* **8**, 399–413 (1979)
6. Došen, K.: Sequent-systems and groupoid models. II. *Stud. Log.* **48**(1), 41–65 (1989)
7. Dunn, J.M.: Star and perp: two treatments of negation. *Philos. Perspect.* **7**, 331–357 (1993)
8. Fagin, R., Halpern, J.Y., Moses, Y., Moshe, Y.V.: *Reasoning About Knowledge*. MIT Press, Cambridge (1995)
9. Fine, K.: Models for entailment. *J. Philos. Log.* **3**, 347–372 (1974)
10. Fine, K.: Truth-maker semantics for intuitionistic logic. *J. Philos. Log.* **43**, 549–577 (2014)
11. Goldblatt, R.I.: Semantic analysis of orthologic. *J. Philos. Log.* **3**, 19–35 (1974)
12. Goldblatt, R.I.: Orthomodularity is not elementary. *J. Symb. Log.* **49**, 401–404 (2014)
13. Hartonas, C.: Duality for lattice-ordered algebras and for normal algebraizable logics. *Stud. Log.* **58**, 403–450 (1997)
14. Hartonas, C.: Reasoning with incomplete information in generalized Galois logics without distribution: the case of negation and modal operators. In: Bimbó, K. (ed.) *J. Michael Dunn on Information Based Logics*. OCL, vol. 8, pp. 279–312. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29300-4_14
15. Hazen, A.: Subminimal negation. Philosophy Department Preprint 1/92, University of Melbourne (1992)
16. Humberstone, L.: Operational semantics for positive R. *Notre Dame J. Formal Log.* **29**(1), 61–80 (1988)
17. Kripke, S.A.: Semantical analysis of intuitionistic logic I. In: Dummett, M.A.E., Crossley, J.N. (eds.) *Studies in Logic and the Foundations of Mathematics*, vol. 40, pp. 92–130. Elsevier, Amsterdam (1965)
18. Mares, E.D.: *Relevant Logic: A Philosophical Interpretation*. Cambridge University Press, Cambridge (2004)
19. Punčochář, V.: Algebras of information states. *J. Log. Comput.* **27**, 1643–1675 (2017)
20. Punčochář, V.: A relevant logic of questions. *J. Philos. Log.* **49**(5), 905–939 (2020). <https://doi.org/10.1007/s10992-019-09541-9>
21. Putnam, H.: How to think quantum-logically. *Synthese* **29**(1), 55–61 (1974)
22. Restall, G.: Negation in relevant logics: how i stopped worrying and learned to love the routley star. In: Gabbay, D., Wansing, H. (eds.) *What is Negation? Volume 13 of Applied Logic Series*, pp. 53–76. Kluwer (1999)
23. Restall, G.: *An Introduction to Substructural Logics*. Routledge, London (2000)
24. Urquhart, A.: Semantics for relevant logics. *J. Symb. Log.* **37**(1), 159–169 (1972)



Algorithmically Broad Languages for Polynomial Time and Space

Daniel Leivant^{1,2}(✉)

¹ Computer Science, Indiana University, Bloomington, IN 47405, USA
leivant@indiana.edu

² IRIF, Université de Paris, 75205 Paris, France

Abstract. Flexible programming languages with built-in bounds on time or space resources are of obvious practical interest. Since we know that no programming language can capture exactly the PTime (or PSpace) algorithms, the challenge is to design languages that guarantee PTime while accommodating a broad spectrum of algorithmic methods. We propose here such languages for PTime and PSpace, based on size-sensitive imperative programming, parameterless procedures, and a retooling of the ramification method.

Keywords: Ramification · Imperative programming · Parameterless procedures · Variants · Bundles · PTime · PSpace · Non-size-increase

1 Introduction

The quest for machine-independent formalisms that characterize complexity classes has been pursued for over half a century, with three distinct motivations: providing new tools for separation results between complexity classes, exploring links between computational complexity and abstraction principles, and developing programming languages and systems with certified resource-bounds.

The potential value of a formalism for software development depends, however, on its algorithmic expressiveness. Unfortunately, Hajek's Theorem [15, Theorem 2] implies that, for any Turing-complete programming language, deciding whether a program runs in PTime (or PSpace) is a Σ_2^0 -complete problem, so no reasonable (i.e. semi-decidable) formalism can possibly capture exactly the PTime (respectively PSpace) algorithms.

The challenge is thus to construct formalisms that are sound for PTime, include an algorithm for every PTime function (i.e. are *extensionally* complete for PTime), and capture PTime algorithmic methods as broadly as possible. In seeking algorithmic breadth it seems reasonable to combine imperative paradigms with a useful form of recursive procedures. Towards that end we combine a number of ingredients. To obtain data genericity, we use as basic data-unit finite partial-functions (f-functions for short), and construe compound data, such as graphs or strings, as collections of f-functions, which we dub *bundles*. Size-change

takes center stage by adopting as basic operations the extension and contraction of f-functions. To facilitate direct memory management we use as basic actions updates of single entries of f-functions. As complexity concerns seem to naturally call for a finitistic approach to data, that is avoiding references to completed infinite objects, we stay away from infinite objects, such as functions over \mathbb{N} .

The framework above yields quite naturally an abstract characterization of primitive-recursive complexity [33]. By ramifying the framework we obtain languages for PTime and PSpace with broad algorithmic breadth.

2 Finite Functions as Generic Data

2.1 F-Functions

We pursue the approach of [29], which takes finite functions over “atoms” as generic building block for all finite data, and basic operations over these functions’ entries (i.e. elements of their graph) as fundamental mappings. The use of finite functions, rather than finite sets or relations, achieves at one fell swoop a built-in tupling operation and a streamlined way of naming and addressing atoms. Our basic program-operations are consequently abstract and generic analogs of micro-code; the intent is that other operations can be defined directly in terms of those operations.

Thus, we posit a denumerable set A of *atoms*, which are unspecified and unstructured objects. To accommodate in due course non-denoting terms we extend A to a set $A_{\perp} =_{\text{df}} A \cup \{\perp\}$, where \perp is a fresh object intended to denote “undefined.”

DEFINITION 1 A (*k*-ary) **f-function** (short for *finite function*) is a function $F : A_{\perp}^k \rightarrow A_{\perp}$ for which there is a finite set $S_F \subset A^k$ (the *source* of F) such that $F(a_1..a_k)$ is \perp iff $\langle a_1..a_k \rangle \notin S_F$. An *entry* of F is a tuple $\langle a_1 \dots a_k, b \rangle$ where $b = F(a_1, \dots, a_k) \neq \perp$. The *size* $|F|$ of F is its number of entries. The *image* of F is the set $I_F = \{b \in A \mid b = F(\mathbf{a}) \text{ for some } \mathbf{a} \in A^k\}$. \dashv

Function partiality provides a natural representation of finite relations over A by partial functions. Namely, a finite k -ary relation R over A ($k > 0$) is represented by the f-function $\xi_R(a_1, \dots, a_k) = \text{if } R(a_1, \dots, a_k) \text{ then } a_1 \text{ else } \perp$. In particular, a finite subset $B \subset A$ is represented by the identity function on B .

Conversely, any partial k -ary function F over A determines the k -ary relation $R_F = \{\langle \mathbf{a} \rangle \in A^k \mid F(\mathbf{a}) \text{ is defined}\}$. Consequently, we write $\mathbf{a} \in F$ for $F(\mathbf{a}) \neq \perp$.

2.2 Bundles

DEFINITION 2 A *vocabulary* is a finite list V of function-identifiers, referred to as *V-ids*, with each $\mathbf{f} \in V$ assigned an *arity* $\tau(\mathbf{f}) \geq 0$. We refer to nullary *V-ids* as *tokens* and to those of positive arity as *pointers*. A *bundle* over a vocabulary V , or *V-bundle* for short, is a mapping β that to each $\mathbf{f} \in V$ of arity r assigns an r -ary f-function $\beta(\mathbf{f})$, said to be a *component* of β . \dashv

DEFINITION 3 The *source* S_β of a bundle β is the union of the sources of its components as defined above, its *image* I_β is the union of the images of its components, its *universe* U_β is the union of its source and its image, and its *size* $|\beta|$ is the sum of the sizes of its components. If β is a V -bundle, and γ a W -bundle where $W \supseteq V$, then we say that γ is an *expansion* of β (to W) if the two structures have identical interpretations for each identifier in V .¹ The *trivial expansion* of β to W , denoted β^W , interprets every $f \in W - V$ as the empty f -function.

Given $f \in V$ and a V -bundle β , the *size of f in β* , denoted $|f|_\beta$, is the size of $\beta(f)$, that is the number of entries therein. For $F \subseteq V$ the *size of F in β* , denoted $|F|_\beta$, is $\sum\{|f|_\beta \mid f \in F\}$.

For a vocabulary V the set \mathbf{Tm}_V of V -terms is defined inductively: the reserved identifier ω is in \mathbf{Tm}_V ; and if $f \in V$ is of arity $k \geq 0$, and $t_1, \dots, t_k \in \mathbf{Tm}_V$, then $ft_1 \cdots t_k \in \mathbf{Tm}_V$. ⊣

Note that the traditional role of variables is taken over here by tokens. A term t without ω is *standard*.

We write function application in formal terms without parentheses and commas, as in fx or $f\vec{x}$ (thus stand-alone tokens are terms). We implicitly posit that the arity of a function-id matches the number of arguments displayed.

DEFINITION 4 A V -bundle β engenders a mapping $\hat{\beta} : \mathbf{Tm}_V \rightarrow U_\beta$, defined by recurrence on \mathbf{Tm}_V : $\hat{\beta}(\omega) = \perp$; and $\hat{\beta}(ft_1 \cdots t_k) = \beta(f)(\hat{\beta}(t_1), \dots, \hat{\beta}(t_k))$ ($\tau(f) = k \geq 0$). An *equation over V* , or *V -equation*, is a phrase $t \simeq q$ where t and q are V -terms. $t \simeq q$ is *true* in β if $\hat{\beta}(t)$ and $\hat{\beta}(q)$ are the same element of A_\perp . ⊣

2.3 Bundles for Inductive Data

Examples of bundles include not only finite data-structures, but also inductive data, such as natural numbers, strings, and lists. That is, elements of free algebras generated from a finite set of constructors. Defining natural numbers as particular finite structures goes back to Euclid, and underlies Church’s numerals and its generalization to all inductive data [10]. For example, the following bundles are the natural number 3 (construed as the term sss), and the string 011 (construed as 011)



Since bundles are all finite by definition, fundamental theorems about impossibility of first-order axiomatizations do not apply, and it is indeed possible to axiomatize, for each vocabulary V , the bundles representing V -terms, by the following statements.

- * **Separation:** If $c(a_1, \dots, a_k)$ and $c'(a'_1, \dots, a'_{k'})$ are the same atom, where c and c' are constructors, then c and c' are the same, $k = k'$, and $a_i = a'_i$ for all i .

¹ Unlike the traditional notion of structure expansion, here γ may have a larger universe than β .

- ★ **Decomposition:** If a is $\mathbf{c}(b_1 \dots b_r)$ for some \mathbf{c} and $b_i \in A$, then so is each b_i .
- ★ **Anchor:** There is exactly one atom a in the collective image of the constructors and not in their collective source. That is: $a = \mathbf{c}(\mathbf{b})$ for some constructor \mathbf{c} and atoms \mathbf{b} ; and no atom a' is of the form $\mathbf{c}(\mathbf{b})$ with a listed in \mathbf{b} .

It follows from these properties that if \mathbf{c} is a constructor of arity $r > 0$ then there are unary inverse-functions $\mathbf{c}^{-1}, \dots, \mathbf{c}^{-r}$, i.e. $\mathbf{c}^{-i}(\mathbf{c}(a_1, \dots, a_r)) = a_i$. A more general sequencing mechanism is also easy to define [29].

Finally, note that a tuple of bundles can be represented by a single bundle. The simplest approach is as follows. Given V_i -bundles β_i ($i = 1..k$) let V'_1, \dots, V'_k be a renaming of V_1, \dots, V_k , $V' = \cup_i V'_i$, and β'_i be the renaming of β_i to V'_i . The tuple $\langle \beta_1 \dots \beta_k \rangle$ is then represented by the V' -bundle β defined by $\beta(\mathbf{f}) = \beta'_i(\mathbf{f})$ for $\mathbf{f} \in V'_i$. For example, a pair of natural numbers, represented as bundles over the vocabulary $\langle \mathbf{z}, \mathbf{s} \rangle$, can be represented as a single bundle for the vocabulary $\langle \mathbf{z}, \mathbf{s}, \mathbf{z}', \mathbf{s}' \rangle$.

3 Programs for Bundle Transformation

3.1 Atomic Operations

We define an imperative language **BT** for the transformation of bundles, based on parameterless recursive procedures.

DEFINITION 5 *The following operations modify a bundle β into a bundle β' which is identical to β except as noted.*

- ★ **V -extension:** A phrase $\mathbf{ft}_1 \dots \mathbf{t}_k \downarrow \mathbf{t}'$ where \mathbf{t}' and each \mathbf{t}_i are standard V -terms. If $\beta(\mathbf{ft}_1 \dots \mathbf{t}_k) = \perp$, then $\beta'(\mathbf{ft}_1 \dots \mathbf{t}_k) = \beta(\mathbf{t}')$.
- ★ **V -contraction:** $\mathbf{ft}_1 \dots \mathbf{t}_k \uparrow$. It sets $\beta'(\mathbf{f})(\beta(\mathbf{t}_1), \dots, \beta(\mathbf{t}_k)) = \perp$.
- ★ **V -inception:** $\mathbf{c} \downarrow$, where \mathbf{c} is a V -token.² If $\beta(\mathbf{c}) = \perp$, then $\beta'(\mathbf{c})$ is any atom not in the universe of β . The choice of atom will have no bearing on the descriptive or computational properties of β' .

We refer to extensions and contractions as **revisions**, and to revisions and inceptions as **updates**. The identifier \mathbf{f} in each revision above is its **eigen-id**. A revision is **active (in a bundle β)** if the resulting structure β' is not identical to β . ¬

Remarks

1. An assignment $\mathbf{f} \bar{\mathbf{t}} := \mathbf{q}$ can be defined using an auxiliary token \mathbf{b} , as an abbreviation for the program $\mathbf{b} \downarrow \mathbf{q}; \mathbf{f} \bar{\mathbf{t}} \uparrow; \mathbf{f} \bar{\mathbf{t}} \downarrow \mathbf{b}; \mathbf{b} \uparrow$.³ When \mathbf{f} is a token we get the customary form of imperative assignment.

² A common alternative notation is $\mathbf{c} := \mathbf{new}$.

³ \mathbf{b} memorizes the atom denoted by \mathbf{q} , in case the contraction renders that atom inaccessible.

2. The inception operation does not have a dual, as atoms can be released from a bundle by repeated contractions.
3. A more general form of inception, $\mathbf{t} \Downarrow$ with \mathbf{t} a term, can be defined using an auxiliary token \mathbf{b} : $\mathbf{b} \Downarrow$; $\mathbf{t} \Downarrow \mathbf{b}$; $\mathbf{b} \Uparrow$

3.2 Programs

DEFINITION 6 A *guard* (over V) is a boolean combination G of V -equations; $\mathbf{!t}$ abbreviates $\mathbf{t} \neq \omega$. We posit an unbounded supply of reserved *procedure-identifiers* for which we use ψ as syntactic parameter.

The set of *programs over V* , or *V -programs* for short, is generated inductively, as follows. Updates, as well as **skip**, **abort**, and every PI are programs. And if P, Q are programs, then so are $P; Q$, $\mathbf{if}[G]\{P\}\{Q\}$, and $\mu\psi.\{P\}$. For a V -program P , we write $P[V']$ for a renaming V' of V .⁴ \dashv

DEFINITION 7 Given a V -program $P \equiv P[\psi]$, with its free PIs among $\psi_1 \dots \psi_m$, the *semantics* P is a mapping, denoted $\llbracket P \rrbracket$, from binary relations Ψ_1, \dots, Ψ_m between bundles, to a binary relation between bundles. $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)$ is defined by structural recurrence on P .

- * If P is an update then it is interpreted as described above;
- * $\llbracket \mathbf{skip} \rrbracket(\Psi)$ is the identity on bundles, and $\llbracket \mathbf{abort} \rrbracket(\Psi)$ is the empty mapping.
- * If P is ψ_i then $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)$ is Ψ_i .
- * For P obtained by composition or branching the semantics is standard.
- * If P is $\mu\psi.\{Q\}$ where $Q \equiv Q[\psi, \psi_1 \dots \psi_m]$, then $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)$ is the union $\cup_i \Xi_i$, where each Ξ_i is a binary relation on bundles, defined by recurrence on i : $\Xi_0 = \emptyset$; and $\Xi_{i+1} = \llbracket Q \rrbracket(\Xi_i, \Psi_1, \dots, \Psi_m)$ (This union a mapping by the Knaster-Tarski Theorem) \dashv

Note that although $\mu\psi.\{P\}$ is parameter-free, each call to P at an occurrence of ψ in P can be preceded by a suitable initialization of some V -ids. However, the value of those V -ids is not recovered upon termination of the recursive call, as would have been the case for formal procedure parameters.

Guarded iteration constructs, such as **while** and **until**, are definable in terms of parameterless recursion (compare [17, §9.1]). In particular, a while-loop $\mathbf{do}[G]\{P\}$ can be regarded as an abbreviation for $\mu\psi \{ \mathbf{if}[G]\{P; \psi\} \{ \mathbf{skip} \} \}$.

Guarded iterative programs are well known to be sound and complete for Turing computability. See [33] for a proof of Turing completeness adapted to **BT**.

3.3 Functions in BT

Based on the semantics of **BT** we define the semantics of programs with respect to classes of bundles. By a *class of V -bundles* we mean here a collection of

⁴ Recall that our vocabularies are lists.

V -bundles closed under isomorphism and renaming of the vocabulary V . In particular, for each V , if $\mathbb{A}(V)$ is the free algebra generated from V , as for example the free algebra $\mathbb{N} = \mathbb{A}(\mathbf{z}, \mathbf{s})$, then the collection of V -bundles for terms in $\mathbb{A}(V)$ is a class of V -bundles.

DEFINITION 8 We write $P[V_{in}; V_{out}]$ for a V -program P with two distinguished sublists V_{in} and V_{out} of V .⁵ A W -program $P[V_{in}; V_{out}]$ **computes** a partial-mapping $\Phi: \mathfrak{C} \rightarrow \mathfrak{C}'$ from a class \mathfrak{C} of V -bundles to a class \mathfrak{C}' of V' -bundles. if for every $\beta[V] \in \mathfrak{C}$ we have $\beta^W \Rightarrow_P \beta'$ where β^W is the trivial expansion⁶ of β to W , and β' is an expansion of V_{out} . \dashv

Note that the mappings Φ computed by programs are mappings over f-functions, not over atoms. As such, they are infinite objects. While not being named explicitly in programs, they can be referred to indirectly via the programs that compute them. Indeed, had we wished to refer to them explicitly, we'd need a separate syntactic sort for infinite mappings over f-functions (and bundles). That seems useful in defining more flexible and practical extensions of our languages, but is not called for here. The issue here is akin to the mappings underlying Fraenkel's Replacement Axiom of the formal set theory **ZF**, which as proper classes are first-class citizens in the NBG and MK set theories (see e.g. [8, 20]).

3.4 Examples of BT-programs

★ [**Successor:**] The following program $Succ[\mathbf{z}, \mathbf{s}; \mathbf{z}, \mathbf{s}]$ uses the same vocabulary for input and output.

$$c \Downarrow; sc \Downarrow z; z \Uparrow; z \Downarrow c; c \Uparrow$$

★ [**Predecessor:**]⁷ $c \Downarrow sz; sz \Uparrow; z \Downarrow c; c \Uparrow$

★ [**Addition**] A program $Sum[\mathbf{z}, \mathbf{s}, \mathbf{z}', \mathbf{s}'; \mathbf{z}, \mathbf{s}]$ for addition:

$$\begin{aligned} \mu\psi \{ & \text{if } [!s'z'] \\ & \{c := s'z'; s'z' \Uparrow; z' := c; Succ[\mathbf{z}, \mathbf{s}]; \psi\} \\ & \{\text{skip}\} \} \end{aligned}$$

4 Variants and Primitive-Recursive Complexity

4.1 Recursion with Variants

We proceed to modify **BT** by rephrasing recursion with a built-in termination condition, resulting in a language **BTV** conveying a generic form of primitive-recursive complexity, independently from any inductive type. Whereas a loop's

⁵ V_{in} and V_{out} need not be disjoint, and V need not be their union.

⁶ See Definition 3.

⁷ These successor and predecessor programs are in constant time, in contrast to the linear time obtained when acting on the final atom of the input.

guard can be construed as a liveness condition, evaluated locally at the loop's input bundle, a dual notion, expressing a safety condition, is evaluated for an entire pass through the loop body, and implying termination when successful.

DEFINITION 9 A *V-variant* is a subset T of V . ◄

The depletion of variants will be used to pace the iteration of procedure calls by the depletion of variants. Our variants are a restricted form of the traditional function-variants [13, 14, 45], with the variant-functions taken here to be the size of a portion of the underlying data.

DEFINITION 10 The *programs of BTV* are generated as for **BT**, but with recursion rephrased to account for variants: If P is a **BTV**-program, ψ a *PI*, and T a variant, then $\mu\psi:T\{P\}$ is a **BTV**-program.

For the iterative operation **do** (definable in term of recursion) we place a variant T as a second guard: **do**[G][T]{ α } ◄

4.2 Semantics of BTV

The semantics of **BTV** refers to the size-change of variants to trigger exit from procedure iteration: If the variant is empty at a procedure call, then the recursion is exited; if it has decreased in the last pass, then a recursive call is triggered; otherwise computation is aborted. Namely:

- ★ If $|\beta(T)| = 0$, then $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)(\beta) = \beta$.
- ★ If $|\beta(T)| > 0$ then $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)$ is the union $\cup_{i=1..m} \Xi_i$, where the Ξ_i 's are binary relations on bundles, defined by recurrence on i : $\Xi_0 = \emptyset$; and $\Xi_{i+1}(\beta) = \llbracket Q \rrbracket((\Xi_i \triangleleft \beta), \Psi_1, \dots, \Psi_m)(\beta)$ where

$$(\Xi_i \triangleleft \beta)(\gamma) = \text{if } |\gamma(T)| < |\beta(T)| \text{ then } \Xi_i(\gamma) \text{ (otherwise undefined)}$$

Note that primitive recursion over inductive types, such as strings or natural numbers, is a special case of this definition, in which the aborted termination never occurs, because the recurrence argument is reduced by the very statement of recurrence.

In [34] we proved that **BTV** characterizes primitive-recursive complexity. On the one hand, every primitive-recursive function over a free algebra is computable by a **BTV**-program (generalizing [36]); on the other hand, for every **BTV**-program P there is a primitive-recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that if $\beta \Rightarrow_P \beta'$ then $|\beta'| \leq f(|\beta|)$.

4.3 Examples of BTV-programs

- ★ [**Addition**] Referring to the **BT**-program of Sect. 3.4 for addition we take the second input as variant. This done, the guard used in the Sect. 3.4 program is no longer needed, because it is implied by the variant depletion. So $Sum[z, s, z', s'; z, s]$ becomes

$$\mu\psi:s' \{c := s'z'; s'z' \uparrow; z' := c; Succ[z, s]; \psi\}$$

Note that since s' is consumed by the iteration, via $s'z' \uparrow$, as required for the iteration to proceed, s' is not available for repeated use (e.g. for multiplication).

- ★ [**Duplication**] The program $Dup[z, s; z', s', z'', s'']$ outputs two copies of the input:

$$\begin{aligned} & z' \Downarrow; z'' \Downarrow; \\ & \mu\psi:s \{ c := sz; sz \uparrow; z := c; \\ & \quad Succ[z', s']; Succ[z'', s'']; \\ & \quad \psi \} \end{aligned}$$

A reduced form of Dup is a program $Rnm[z, s; z', s']$ that renames the input. Composing the two we get a program $Copy[z, s; z', s']$ with an output equal to the input, and the input unchanged.

- ★ [**Multiplication**] Program $Prod[z, s, z', s'; z'', s'']$:

$$\begin{aligned} & z'' \Downarrow; \\ & \mu\psi:s'. \{ c := s'z; s'z' \uparrow; z := c; \\ & \quad Copy[z, s; \bar{z}, \bar{s}]; \\ & \quad Sum[z'', s'', \bar{z}, \bar{s}; z'', s'']; \\ & \quad \psi \} \end{aligned}$$

- ★ [**Exponentiation**] The **BTV**-programs Dup and Sum can be iterated, yielding a **BTV** program for $n \mapsto 2^n$.

5 Ramification

5.1 The Ramification Method

Considerable work has been done on ramified recurrence, also known as tiered, stratified, predicative, or normal/safe. The method has been used to obtain machine-independent characterizations of several major complexity classes, such as polynomial time [4, 25] and polynomial space [30, 40], as well as alternating log time [9, 31], alternating poly-log time [9], NC [28, 39], linear space [16, 24, 26], NP [2, 41], the PTime hierarchy [3], and probabilistic polynomial time [23]. The method is all the more of interest given the roots of ramification in the foundations of mathematics [43, 44], thus bridging abstraction levels in set-theory and type-theory to computational complexity classes.

Ramification of imperative programs was tentatively explored in [35] and [32]. The former refers to PTime over arbitrary structures, via a combination of imperative programming and interpretations. It posits constructors (“operators”), that interpret the structure in hand in $\mathbb{W} = \{0, 1\}^*$ by non-increasing functions, thus embedding the ostensibly-generic data into \mathbb{W} , with functions over \mathbb{W} serving as variant-functions, and ramification applying to \mathbb{W} . Unfortunately, the resulting language falls short of being practical, and it characterizes quasi-PTime, i.e. programs whose number of distinct configurations for a given input is polynomially bounded by the size of the input (termination not guaranteed). Likewise, [32] characterizes quasi-PTime for programming over graphs, and is unfortunately confined to rather special cases that have failed to generalize.

We believe that a more successful use of ramification for imperative programs calls for more than formal analogies with the ramification of recursion equations. For one, the emphasis of the latter on a dichotomy between input and output values is no longer tenable for imperative programs, where data is not necessarily singled out as input or output. Also, the use of variables in declarative programming hides a form of free data duplication (which is also present, albeit implicitly, in the use of combinators in place of variables). For example, the defining equation for product, $\times(sn, x) = +(x, \times(n, x))$, refers to x twice in the definiendum, and any imperative implementation of this schema would have to duplicate x , since those two occurrences have distinct roles in the computation

Our approach here is to refer to loop variants as part of the language syntax, and to ramify those variants via a ramification of the entire vocabulary. We allow size-reduction of loop-variants of rank r to be counter-balanced by a size-increase of some ids of rank r that are not in the variant, provided the over-all size of V_r does not increase. The non-size-increase principle [1, 19] is a special case of this mechanism, for programs that use only rank 0.

5.2 Ramified Imperative Recursion

DEFINITION 11 A *ramified vocabulary* is a vocabulary V with each V -id assigned a natural number as its *rank*.⁸ We write V_r for the set of V -ids of rank r . A *variant (of rank r)* is a set $T \subseteq V_r$. ⊣

We define programming languages **BTR** and **BTR*** for PSpace and PTime algorithms, respectively. These languages impose rank-driven conditions on variants, and the two languages differ only in their semantic interpretation of variants’ “size decrease,” which refers to individual recursive calls for **BTR**, but to cumulative recursive calls for **BTR***, in a sense to be described below.

The programs of **BTR** and **BTR*** are generated like those of **BTV**, but with all variants ramified. As was the case for **BTV**, the intent resides here in the semantics.

⁸ Leaving some ids unranked is akin to assigning them 0.

Consider a program P of the form $\mu\psi:T.Q$, where T is a variant of rank r . For an input bundle β the semantics of **BTR** aborts the computation on a recursive call not only when the depletion condition fails, as is the case in **BTV**, but also when the ramification condition fails:

- ★ If $|\beta(T)| = 0$, then $\llbracket P \rrbracket(\Psi)(\beta) = \beta$.
- ★ If $|\beta(T)| > 0$ then $\llbracket P \rrbracket(\Psi_1, \dots, \Psi_m)$ is the union $\cup_i \Xi_i$, where the Ξ_i 's are binary relations on bundles, defined by recurrence on i : ($\Xi_0 = \emptyset$; and $\Xi_{i+1}(\beta) = \llbracket Q \rrbracket(\Xi_i \triangleleft \beta), \Psi_1, \dots, \Psi_m)(\beta)$ where

$$\begin{aligned} (\Xi_i \triangleleft \beta)(\gamma) = \\ \text{if } |\gamma(T)| < |\beta(T)| \quad \text{and} \quad |\gamma(V_j)| \leq |\beta(V_j)| \text{ for all } j \geq r \\ \text{then } \Xi_i(\gamma) \quad (\text{otherwise undefined}) \end{aligned}$$

BTR* differs semantically from **BTR** only in the ramification-condition, which is tightened here to the following. If $\psi_1 \dots \psi_\ell$ are the instances of ψ in Q , and the execution of Q with input β leads to bundles $\gamma_1 \dots \gamma_\ell$ at those instances (respectively, with γ_h empty if ψ_h is not reached by execution for input β), then $\sum_h |\gamma_h(T)| < |\beta(T)|$.

Remarks

1. Note that ranks are here properties of V -ids, and not of atoms, f-functions, or terms. Moreover, no ranking for atoms or functions is inherited from the ranking of function-ids: an f-function may be the value of two distinct ids, possibly of different ranks. Consequently, there is no rank-driven restriction on inceptions or extensions, as the function-entries created are not assigned any rank, e.g. an extension $\mathbf{fc} \downarrow \mathbf{q}$ may have \mathbf{f} of rank 0 whereas \mathbf{q} refers to arbitrarily large ranks.
2. Ramification is not trivially compatible with function composition: we might have **BTR**-programs $P[V; V']$ and $Q[W; W']$, in which pointers might be ramified differently. However, since ramification depends only on the order between ranks, it is easy to see that ranks in P and Q may be reconfigured to have ranks in V' and W match, as proved for ramified recurrence in [18].

5.3 Related Work in Static Analysis

Feasibility analysis of imperative programs goes back at least to the Meyer-Ritchie characterization of primitive recursion by imperative “loop”-programs over \mathbb{N} [36]. It is natural to explore restrictions on recurrence that imply stronger forms of termination, i.e. complexity bounds. This idea goes back to Ritchie and Cobham [11, 42], who introduced recurrence restricted explicitly by bounding conditions. Although the characterizations they obtained use one form of bounded resources (bounded recurrence) to delineate another form (bounded execution), they proved useful, for example in suggesting complexity measures

for higher-order functionals [12]. A potent line of research has sought algorithms for certifying the feasibility of such programs [5–7, 21, 22]. A different approach was pursued in [37, 38], where imperative programs with n variables are assigned “certificates of feasibility,” which are roughly $n \times n$ matrices over $\{0, 1, \infty\}$.

6 Completeness and Soundness Properties

6.1 Extensional Completeness of \mathbf{BTR}^* for PTime

While no programming language can be sound and complete for FPTime *algorithms* [15], as noted above, a useful sanity check remains *extensional completeness*, that is the property that every *function* computable in PTime is computable by some program.

An easy approach is to rely on existing implicit characterizations of PTime, such as ramified recurrence over strings [27]. Indeed, the coding in \mathbf{BTR}^* of ramified recurrence over strings is straightforward, yielding:

PROPOSITION 12 1. \mathbf{BTR}^* interprets ramified recurrence over strings, and by [27] is therefore extensionally complete for FPTime.

2. \mathbf{BTR} interprets ramified parameterized recurrence⁹ over strings, and by [30] is therefore extensionally complete for FPSpace.

More interesting are the extensional completeness of \mathbf{BTR} and \mathbf{BTR}^* for the PTime programs of \mathbf{BT} . Indeed, \mathbf{BT} is the Turing-complete complete computation model germane to our approach. We give the proof for \mathbf{BTR} .

THEOREM 13 Every \mathbf{BT} -program P running in PTime is extensionally equivalent to some \mathbf{BTR}^* -program P^\dagger ; i.e. P^\dagger computes the same mapping between bundles as P .

PROOF OUTLINE

Let P be a \mathbf{BT} -program over V , running within time $c \cdot n^\ell$. P^\dagger is defined by recurrence on the loop-nesting depth of P . P^\dagger is P if P is loop-free; $(Q; R)^\dagger$ is $Q^\dagger; R^\dagger$; and $(\mathbf{if}[G]\{Q\}\{R\})^\dagger$ is $\mathbf{if}[G]\{Q^\dagger\}\{R^\dagger\}$.

If P is $\mu\psi.\{Q\}$, let $n_0 < \dots < n_{k-1}$ be the ranks in Q^\dagger . We define a “clocking” program that maps a V -bundle β to a list of length $c \cdot |\beta|^\ell$: By [33, §5.1] (and the proof there) there is a \mathbf{BTR}^* -program E that maps β to a list e of length $|\beta|$. By our examples above of \mathbf{BTR}^* -programs there is a \mathbf{BTR}^* -program M that, maps e to a list d of length $c \cdot |\beta|^\ell$, and a program R that contracts d by one entry. Since ranks in E, M , and P can be lifted uniformly, without affecting their rank-correctness or semantics, we may assume without loss of generality that all components of d have a rank higher than all variants in Q^\dagger .

⁹ I.e. ramified recurrence over strings “with parameter substitution”.

Now define P^\dagger to be $E; M; \mu\psi:d. \{R; Q^\dagger\}$.

Since d is of higher rank of all ranks in Q^\dagger the semantics of Q^\dagger is the same in P^\dagger as in P , concluding the proof. -1

Remark. Note that our extensional completeness results for **BTR*** depend on the availability of all ranks. The fact that 2 ranks suffice in the Bellantoni-Cook system [4] is a consequence of allowing rank mismatch in “safe-composition”. This phenomenon is specific to a declarative setting, and cannot be replicated in an imperative language.

A proof of the extensional completeness of **BTR** for PSpace will be given in the full version of this paper.

6.2 Soundness of BTR and BTR* for Feasibility

THEOREM 14 (Soundness of **BTR*** for PTime)

Let P be a **BTR***-program with recursions of ranks $\leq \ell$.

1. For each $j \leq \ell$ there is a positive¹⁰ polynomial $Z_{P,j}[n_{j+1} \dots n_\ell]$ such that

$$\text{Space}_{P,j}(\beta) \leq |\beta|_j + Z_{P,j}[|\beta|_{j+1}, \dots, |\beta|_\ell]$$

2. There is a positive polynomial $M_P[n_0 \dots n_\ell]$ such that for all V -bundles β

$$\text{Time}_P(\beta) \leq M_P[|\beta|_0, \dots, |\beta|_\ell]$$

PROOF. Parts (1) and (2) are proved simultaneously by induction on P .

Proof of (1). Suppose $\beta \Rightarrow_P \beta'$, where $\beta = \beta_0 \Rightarrow_Q \beta_1 \dots \Rightarrow_Q \beta_k = \beta'$. Since T is decreasing in the aggregate in Q , we have $k \leq |T|_\beta \leq |\beta|_r$. V_j is non-increasing in P . For each $j \geq r$, so we take $Z_{P,j} \equiv 0$.

For $j = r - d$, $d = 0, \dots, r$, we proceed by a secondary induction on d . The induction base $d = 0$, i.e. $j = r$, is already proved above.

For the main induction step, we have

$$\begin{aligned} &\text{Space}_{P,r-(d+1)}(\beta) \\ &= \max_{i < k} \text{Space}_{Q,r-(d+1)}(\beta_i) \\ &\leq \max_{i < k} Z_{Q,r-(d+1)}[|\beta_i|_{r-d}, \dots, |\beta_i|_\ell] \quad (\text{by main IH}) \\ &\leq Z_{Q,r-(d+1)}[\text{Space}_{P,r-d}(\beta), \dots, \text{Space}_{P,\ell}(\beta)] \\ &\quad (\text{by definition of } \text{Space}_{P,j} \text{ and since each } Z_{Q,j} \text{ is positive}) \\ &\leq Z_{Q,r-(d+1)}[A_{r-d}, \dots, A_\ell] \\ &\quad (\text{by secondary IH}) \end{aligned}$$

where A_j stands for $|\beta|_j + Z_{P,j}[|\beta|_{j+1}, \dots, |\beta|_\ell]$.

So it suffices to take $Z_{P,r-(d+1)}[n_{r-d}, \dots, n_\ell] \equiv_{\text{df}} Z_{Q,r-(d+1)}[B_{r-d} \dots B_\ell]$,

where B_j stands for $n_j + Z_{P,j}[n_{j+1}, \dots, n_\ell]$.

This concludes the inductive step for (2).

¹⁰ I.e. defined without subtraction or negative integers.

Proof of (2). We have

$$\begin{aligned}
 \text{Time}_P(\beta) &\leq k + \sum_{i < k} \text{Time}_Q(\beta_i) \\
 &\leq k + \sum_{i < k} M_Q(|\beta_i|_0, \dots, |\beta_i|_\ell) \\
 &\quad \text{by IH} \\
 &\leq k + \sum_{i < k} M_Q[A_0, \dots, A_\ell] \\
 &\quad \text{with the } A_j\text{'s above} \\
 &\leq |\beta|_r (1 + M_Q[A_0, \dots, A_\ell])
 \end{aligned}$$

So it suffices to take $M_P(n) \equiv_{\text{df}} n_r(1 + M_Q[B_0, \dots, B_\ell])$ where the B_i 's are as above. ⊣

THEOREM 15 (Soundness of **BTR** for FPSpace)

Let P be a **BTR**-program with recursions of ranks $\leq \ell$. For each $j \leq \ell$ there is a positive polynomial $R_{P,j}[n_{j+1} \dots n_\ell]$ such that

$$\text{Space}_{P,j}(\beta) \leq |\beta|_j + R_{P,j}[|\beta|_{j+1}, \dots, |\beta|_\ell]$$

PROOF. The proof is analogous to that of Theorem 14, and simpler (no reference to time complexity is needed). In executing a program Q of the form $\mu\psi:V, \{P[q]\}$, with P satisfying the theorem, with a polynomial R_P , execution of $P_{n+1} \equiv P[P_n]$ might run P_n a number of times, but each time with the variant V smaller than its size n at call time. Each of these executions takes space $\leq R_P$, yielding polynomials $R_{Q,j}$ for Q , of degree 1 higher than the degree of $R_{P,j}$. ⊣

7 Examples of BTR*-programs

7.1 Fundamental Examples

★ [**Duplication**] The duplication program of Sect. 4.3 can be ramified with the input at rank r higher than either output. Recall that $\text{Dup}[z, s; z', s', z'', s'']$ outputs two copies of the input. Note that one output can be of rank r as well, since it is not part of the variant, so the total rank- r size of the vocabulary remains constant. Consequently, the program Rnm needs no ramification at all; but $Copy$ does require that the input be of higher rank than the output, since Dup requires it. Sum too needs no ramification.

★ [**Polynomials**] Consider the **BTV**-program $Prod$ of Sect. 4.3. The program $Copy[z, s; \bar{z}, \bar{s}]$ requires that s have rank higher than \bar{s} , And the recursion defining $Prod$ requires that s' have rank higher than s'' (since the latter grows within the body). So two ranks suffice for $Prod$, with both inputs of higher rank than the output (as is the case for the definition of multiplication by primitive-recursion).

Now consider a program for $n \mapsto n^3$. We cannot nest two multiplication, as

in the definition by ramified recurrence [24]. To repeat *Copy*, would require a variant of rank higher than the pointer being copied! Consequently, we'd need three ranks. For $n \mapsto n^4$ we can also make do with just 3 ranks, by composing the square function with itself.

7.2 Limitative Examples

- ★ [*Exponentiation*] The exponentiation program of Sect. 4.3 cannot be ramified, because each iteration of *Dup* requires that the input be of higher rank than the second copy. The very same hurdle prevents a definition of exponentiation by iterating multiplication.
- ★ [*Ackermann's Function*] Programming Ackermann's Function requires naming the function and referring to it in recursive calls within "actual parameters". As observed in Sect. 3.3, **BTV** uses names for mapping between atoms, but none for mappings between bundles (such as functions over \mathbb{N}), and no use of parameters in recursive programs. Indeed, Ackermann's function exceeds even **BTV**, let alone **BTR***

8 Conclusion and Directions

Much of current research on practical program verification has focused on programs with a strong imperative component and direct memory access. Techniques of implicit computational complexity, notably ramification, have seemed only tangentially relevant to such goals, as their application to imperative programs have seemed to be problematic. We propose here a new take on ramified imperative programming. Starting with finite functions over atoms as a building block for data, and with basic updating such functions as fundamental operations, we obtained imperative programming languages capturing PTime and PSpace respectively, with a promising potential for algorithmic breadth.

Still missing from this picture is the intergration into programs of methods of program verification based on the theory of bundles of [29]. This would hopefully be developed into automatic verification to guarantee that given programs do abort via depletion-failures or ramification-failures.

References

1. Aehlig, K., Berger, U., Hofmann, M., Schwichtenberg, H.: An arithmetic for non-size-increasing polynomial-time computation. *TCS* **318**(1–2), 3–27 (2004)
2. Bellantoni, S.: Predicative recursion and computational complexity. Ph.D. thesis, University of Toronto (1992)
3. Bellantoni, S.: Predicative recursion and the polytime hierarchy. In: Clote, P., Remmel, J. (eds.) *Feasible Mathematics II*, pp. 15–29. Birkhäuser (1994)
4. Bellantoni, S., Cook, S.A.: A new recursion-theoretic characterization of the poly-time functions. *Comput. Complex.* **2**, 97–110 (1992). <https://doi.org/10.1007/BF01201998>

5. Ben-Amram, A.M.: On decidable growth-rate properties of imperative programs. In: Baillot, P. (eds.) International Workshop on Developments in Implicit Computational Complexity. EPTCS, vol. 23, pp. 1–14 (2010)
6. Ben-Amram, A.M., Hamilton, G.W.: Tight worst-case bounds for polynomial loop programs. In: Bojańczyk, M., Simpson, A. (eds.) FoSSaCS 2019. LNCS, vol. 11425, pp. 80–97. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17127-8_5
7. Ben-Amram, A.M., Jones, N.D., Kristiansen, L.: Linear, polynomial or exponential? complexity inference in polynomial time. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 67–76. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69407-6_7
8. Bernays, P.: A system of axiomatic set theory - part I. *JSL* **2**, 65–77 (1937)
9. Bloch, S.A.: Functional characterizations of uniform log-depth and polylog-depth circuit families. In: Proceedings, Structure in Complexity Theory, pp. 193–206. IEEE Computer Society (1992)
10. Böhm, C., Berarducci, A.: Automatic synthesis of typed lambda-programs on term algebras. *Theor. Comput. Sci.* **39**, 135–154 (1985)
11. Cobham, A.: The intrinsic computational difficulty of functions. In: Bar-Hillel, Y. (ed.) Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science, North-Holland, Amsterdam, pp. 24–30 (1962)
12. Constable, R.: Type two computational complexity. In: Proceedings of the Fifth ACM Symposium on Theory of Computing, pp. 108–121 (1973)
13. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall, Hoboken (1976)
14. Gries, D.: The Science of Programming. Springer, Heidelberg (1981). <https://doi.org/10.1007/978-1-4612-5983-1>
15. Hájek, P.: Arithmetical hierarchy and complexity of computation. *Theoret. Comput. Sci.* **8**, 227–237 (1979)
16. Handley, W.G.: Bellantoni and Cook’s characterization of polynomial time functions. Typescript, August 1992
17. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
18. Hofmann, M.: Type systems for polynomial-time computation. Ph.D. thesis, Universität Darmstadt (1998)
19. Hofmann, M.: Linear types and non-size-increasing polynomial time computation. *Inf. Comput.* **183**(1), 57–85 (2003)
20. Kanamori, A.: Bernays and set theory. *Bull. Symb. Logic* **15**(1), 43–69 (2009)
21. Kristiansen, L.: The implicit computational complexity of imperative programming languages. Technical report, BRICS (2001)
22. Kristiansen, L., Niggl, K.-H.: On the computational complexity of imperative programming languages. *Theor. Comput. Sci.* **318**(1–2), 139–161 (2004)
23. Lago, U.D., Toldin, P.P.: A higher-order characterization of probabilistic polynomial time. *Inf. Comput.* **241**, 114–141 (2015)
24. Leivant, D.: Stratified functional programs and computational complexity. In: Twentieth Annual ACM Symposium on Principles of Programming Languages, New York, pp. 325–333. ACM (1993)
25. Leivant, D.: Predicative recurrence in finite types. In: Nerode, A., Matiyasevich, Y.V. (eds.) LFCSS 1994. LNCS, vol. 813, pp. 227–239. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58140-5_23
26. Leivant, B.: Ramified recurrence and computational complexity i: word recurrence and poly-time. In: Feasible Mathematics II, New York, pp. 320–343. Birkhauser-Boston (1994)

27. Leivant, D.: Ramified recurrence and computational complexity I: word recurrence and poly-time. In: Clote, P., Rémel, J. (eds.) *Feasible Mathematics II, Perspectives in Computer Science*, pp. 320–343. Birkhauser-Boston, New York (1994). www.cs.indiana.edu/~leivant/papers
28. Leivant, D.: A characterization of NC by tree recurrence. In: *Thirty Ninth FOCS*, pp. 716–724. IEEE Computer Society (1998)
29. Leivant, D.: A theory of finite structures. CoRR, abs/1808.04949 (2018)
30. Leivant, D., Marion, J.-Y.: Ramified recurrence and computational complexity II: substitution and poly-space. In: Pacholski, L., Tiuryn, J. (eds.) *CSL 1994*. LNCS, vol. 933, pp. 486–500. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0022277>
31. Leivant, D., Marion, J.-Y.: A characterization of alternating log time by ramified recurrence. *Theor. Comput. Sci.* **236**(1–2), 193–208 (2000)
32. Leivant, D., Marion, J.-Y.: Evolving graph-structures and their implicit computational complexity. In: *ICALP*, vol. 40, pp. 349–360 (2013)
33. Leivant, D., Marion, J.-Y.: Implicit complexity via structure transformation. CoRR, abs/1802.03115 (2018)
34. Leivant, D., Marion, J.-Y.: Primitive recursion in the abstract. *Math. Struct. Comput. Sci.* **30**(1), 33–43 (2019)
35. Marion, J.-Y.: A type system for complexity flow analysis. In: *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*, pp. 123–132 (2011)
36. Meyer, A., Ritchie, D.: The complexity of loop programs. In: *Proceedings of the 1967 22nd National Conference*, New York, NY, USA, pp. 465–469. ACM (1967)
37. Niggl, K.-H., Wunderlich, H.: Certifying polynomial time and linear/polynomial space for imperative programs. *SIAM J. Comput.* **35**(5), 1122–1147 (2006)
38. Niggl, K.-H., Wunderlich, H.: Implicit characterizations of FPTIME and NC revisited. *J. Log. Algebraic Methods Program.* **79**(1), 47–60 (2010)
39. Oitavem, I.: Characterizing nc with tier 0 pointers. *Math. Log. Q.* **50**(1), 9–17 (2004)
40. Oitavem, I.: Characterizing PSPACE with pointers. *Math. Log. Q.* **54**(3), 323–329 (2008)
41. Oitavem, I.: A recursion-theoretic approach to NP. *Ann. Pure Appl. Logic* **162**(8), 661–666 (2011)
42. Ritchie, R.W.: Classes of predictably computable functions. *Trans. AMS* **106**, 139–173 (1963)
43. Schütte, K.: *Proof Theory*. Springer, Berlin (1977). <https://doi.org/10.1007/978-3-642-66473-1>
44. Whitehead, A.N., Russell, B.: *Principia Mathematica*, vol. II. Cambridge University Press, Cambridge (1912)
45. Winskel, G.: *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge (1993)



A Pure View of Ecumenical Modalities

Sonia Marin¹, Luiz Carlos Pereira², Elaine Pimentel^{3(✉)}, and Emerson Sales⁴

¹ Department of Computer Science, University College London, London, UK

² Philosophy Department, PUC-Rio/UERJ, Rio de Janeiro, Brazil

³ Department of Mathematics, UFRN, Natal, Brazil

⁴ Gran Sasso Science Institute, L'Aquila, Italy

Abstract. Recent works about ecumenical systems, where connectives from classical and intuitionistic logics can co-exist in peace, warmed the discussion on proof systems for combining logics. This discussion has been extended to alethic modalities using Simpson’s meta-logical characterization: necessity is independent of the viewer, while possibility can be either *intuitionistic* or *classical*. In this work, we propose a *pure, label free* calculus for ecumenical modalities, **nEK**, where exactly one logical operator figures in introduction rules and every basic object of the calculus can be read as a formula in the language of the ecumenical modal logic **EK**. We prove that **nEK** is sound and complete w.r.t. the ecumenical birelational semantics and discuss fragments and extensions.

1 Introduction

Ecumenism can be seen as the search for *unicity*, that is, for different thoughts, ideas or points of view to coexist in harmony. In mathematical logic, ecumenical approaches for a peaceful coexistence of logical systems have been studied deeply, e.g. [Gir93, LM11].

More recently, Prawitz proposed a natural deduction system combining classical and intuitionistic logics [Pra15]. The fundamental question he addressed was: what makes a connective classical or intuitionistic? We will illustrate, with a simple example, some ways of answering this. Consider the following statement, where $x, y, z \in \mathbb{R}$ and $z \geq 0$:

$$\text{if } x + y = 2z \text{ then } x \geq z \text{ or } y \geq z$$

How should we interpret “if then” and “or” in this sentence for it to be valid? The answer is: it depends! If we view this sentence with the classical mathematician’s eyes (*CM*), the intuitionistic mathematician (*IM*) would not see a theorem. Since intuitionists can see classical tautologies through the lens of double negation, we could embed this classical interpretation in the intuitionistic setting as:

$$\text{not (not (if } x + y = 2z \text{ then } x \geq z \text{ or } y \geq z))$$

This would indeed be a valid statement for both *CM* and *IM*.

A finer possibility for guaranteeing the validity of the sentence is to give to the implication an intuitionistic interpretation and to the disjunction a classical one. Namely, the following statement is also a theorem for both *CM* and *IM*:

$$\text{if } x + y = 2z \text{ then not (not } (x \geq z \text{ or } y \geq z))$$

Prawitz’ ecumenism idea can be summarized as: pinpoint the exact places where the classical and intuitionistic views differ and signal it such that *IM* knows to read it with her “ecumenical glasses”, i.e. through a double negation filter. The example above shows that *CM* and *IM* can consider, for example, different connectives for disjunction \vee_c and \vee_i , respectively. Prawitz answered this question for all the first-order connectives in [Pra15] by presenting an ecumenical natural deduction system.

In [PPdP19], we justified some of Prawitz’ choices via pure proof theoretical reasoning, using sequent based systems. Consider the well known classical and intuitionistic sequent systems *G3c* and *G3i* [TS96]. Since all rules in *G3c* are invertible, no choices have to be made during a classical proof search: one can apply any rule bottom-up in any order. This is not the case in *G3i*: choices may have to be made for disjunction, implication and the existential quantifier. This suggests that *CM* and *IM* would share the universal quantifier, conjunction and the constant for the absurd (hence also negation) – *the neutral connectives*, but they would each have their own existential quantifier, disjunction and implication, with different meanings.

Following this discussion, the original statement is ecumenically translated as

$$(x + y = 2z) \rightarrow_i x \geq z \vee_c y \geq z$$

Now the classical mathematician would see everything just fine (since she cannot differentiate classical from intuitionistic), while the intuitionistic mathematician would put on her ecumenical glasses only when observing the disjunction, so they would both agree on the statement. *This* is the essence of ecumenism!

In [MPPS20], we have extended this discussion to modalities to address the question: how would *CM* and *IM* view such concepts as “necessity” and “possibility”? Using Simpson’s meta-logical characterization [Sim94], the answer is that, if something is necessarily true, then it is independent of the viewer. Possibility, on the other hand, can be either *intuitionistic*: in the sense that one should have a guarantee that something will eventually be true; or *classical*: in the sense that it is not the case that necessarily something will not be true. Hence *CM* and *IM* share the necessity connective \Box , but each would have their own possibility views, represented by \Diamond_c and \Diamond_i , respectively.

Our solution, however, was not entirely satisfactory since the ecumenical modal calculi presented so far are not *pure* [Dum91]: the introduction rules for some connectives depend on negation and other connectives. Moreover, the ecumenical modal systems in [MPPS20] make use of labels: the basic objects used in proofs are from a more expressive language than the logic itself, which partially encodes the logic’s semantics.

This paper tackles these issues, proposing a *pure label free* calculus for ecumenical modalities, where every basic object of the calculus can be read as a formula in the language of the logic. For that, we will use *nested systems* [Bull92, Kas94, Brü09, Pog09] with a *stoup* [Gir91], together with a new notion of *polarities* for ecumenical formulas. Nested systems are extensions of the sequent framework where each sequent is replaced by a tree of sequents. The stoup is a distinguished context containing a single formula. Finally, formulas can be polarized as *negative* if the main connective is classical or the negation, or as *positive* otherwise. This is unlike any other notion of polarities that we know of, but could well be related [Lau02]. The idea is that negative formulas are stored in the classical context, while positive formulas are decomposed in the stoup. This not only allows for establishing the meaning of modalities via the rules that determine their correct use (*logical inferentialism*), but also places the ecumenical system as a unifying framework for modalities of which well known modal systems are fragments.

Organization and Contributions. Section 2 reviews the notation for modal formulas, the labeled system **labEK** and the ecumenical birelational semantics; Sect. 3 introduces the ecumenical nested system **nEK** and its normalization procedure; in Sects. 4 and 5 soundness and completeness of **nEK** w.r.t. the ecumenical birelational semantics are proved; Sect. 6 identifies the classical and intuitionistic fragments of **nEK**; Sect. 7 discusses some modal extensions; and Sect. 8 concludes the paper.

2 Preliminaries

In [MPPS20] we proposed an ecumenical version of normal modal logic, where classical and intuitionistic modalities co-exist. The system adopts Simpson’s approach [Sim94], called *meta-logical characterization*, where a modal logic is characterized by the interpretation of modalities in a first-order meta-theory. We translated modalities into the ecumenical first-order logic **LE** [Pra15, PPdP19], justified similarly by the standard interpretation of alethic modalities in a model. The presence of classical and intuitionistic existential connectives in **LE** induces two *possibility* modalities, while the neutral universal quantifier in **LE** entails a neutral *necessity* modality.

Language \mathcal{A} of ecumenical modal formulas is generated by the following grammar:

$$A:: = p_i \mid p_c \mid \perp \mid \neg A \mid A \wedge A \mid A \vee_i A \mid A \vee_c A \mid A \rightarrow_i A \mid A \rightarrow_c A \mid \Box A \mid \Diamond_i A \mid \Diamond_c A$$

We use subscript c for the classical meaning and i for the intuitionistic one, dropping such subscripts when formulas/connectives can have either meaning. A classical version p_c and an intuitionistic version p_i of each propositional variable co-exist in \mathcal{A} : their meanings are different but related via double negation. The neutral logical connectives $\{\perp, \neg, \wedge, \Box\}$ are common for classical and intuitionistic fragments, while $\{\rightarrow_i, \vee_i, \Diamond_i\}$ and $\{\rightarrow_c, \vee_c, \Diamond_c\}$ are restricted to intuitionistic and classical interpretations, respectively.

The meta-logical characterization naturally induces a labeled proof system [Sim94]. The language \mathcal{L} of *labeled modal formulas* is determined by *labeled formulas* of the form $x : A$ with $A \in \mathcal{A}$ and *relational atoms* of the form xRy , where x, y range over a set of variables. *Labeled sequents* have the form $\Gamma \Rightarrow x : A$, where Γ is a multiset containing labeled modal formulas and relational atoms. In what follows, if \mathbf{L} is a sequent based calculus, we use $\vdash_{\mathbf{L}} \Gamma \Rightarrow A$ to denote that there is an \mathbf{L} -proof of $\Gamma \Rightarrow A$. The labeled ecumenical system labEK [MPPS20] is presented in Fig. 1.

$$\begin{array}{c}
 \frac{}{x : p_i, \Gamma \Rightarrow x : p_i} \text{init} \quad \frac{}{x : \perp, \Gamma \Rightarrow z : C} \perp L \quad \frac{\Gamma \Rightarrow y : \perp}{\Gamma \Rightarrow x : A} W \\
 \frac{x : p_i, \Gamma \Rightarrow z : \perp}{x : p_c, \Gamma \Rightarrow z : \perp} L_c \quad \frac{x : \neg p_i, \Gamma \Rightarrow x : \perp}{\Gamma \Rightarrow x : p_c} R_c \\
 \frac{x : A, x : B, \Gamma \Rightarrow z : C}{x : A \wedge B, \Gamma \Rightarrow z : C} \wedge L \quad \frac{\Gamma \Rightarrow x : A \quad \Gamma \Rightarrow x : B}{\Gamma \Rightarrow x : A \wedge B} \wedge R \quad \frac{x : \neg A, \Gamma \Rightarrow z : A}{x : \neg A, \Gamma \Rightarrow z : \perp} \neg L \\
 \frac{x : A, \Gamma \Rightarrow x : \perp}{\Gamma \Rightarrow x : \neg A} \neg R \quad \frac{x : A, \Gamma \Rightarrow z : C \quad x : B, \Gamma \Rightarrow z : C}{x : A \vee_i B, \Gamma \Rightarrow z : C} \vee_i L \quad \frac{\Gamma \Rightarrow x : A_j}{\Gamma \Rightarrow x : A_1 \vee_i A_2} \vee_i R_j \\
 \frac{x : A, \Gamma \Rightarrow z : \perp \quad x : B, \Gamma \Rightarrow z : \perp}{x : A \vee_c B, \Gamma \Rightarrow z : \perp} \vee_c L \quad \frac{\Gamma, x : \neg A, x : \neg B \Rightarrow x : \perp}{\Gamma \Rightarrow x : A \vee_c B} \vee_c R \\
 \frac{x : A \rightarrow_i B, \Gamma \Rightarrow x : A \quad x : B, \Gamma \Rightarrow z : C}{x : A \rightarrow_i B, \Gamma \Rightarrow z : C} \rightarrow_i L \quad \frac{x : A, \Gamma \Rightarrow x : B}{\Gamma \Rightarrow x : A \rightarrow_i B} \rightarrow_i R \\
 \frac{x : A \rightarrow_c B, \Gamma \Rightarrow x : A \quad x : B, \Gamma \Rightarrow z : \perp}{x : A \rightarrow_c B, \Gamma \Rightarrow z : \perp} \rightarrow_c L \quad \frac{x : A, x : \neg B, \Gamma \Rightarrow x : \perp}{\Gamma \Rightarrow x : A \rightarrow_c B} \rightarrow_c R \\
 \frac{xRy, y : A, x : \Box A, \Gamma \Rightarrow z : C}{xRy, x : \Box A, \Gamma \Rightarrow z : C} \Box L \quad \frac{xRy, \Gamma \Rightarrow y : A}{\Gamma \Rightarrow x : \Box A} \Box R \quad \frac{xRy, y : A, \Gamma \Rightarrow z : C}{x : \Diamond_i A, \Gamma \Rightarrow z : C} \Diamond_i L \\
 \frac{xRy, \Gamma \Rightarrow y : A}{xRy, \Gamma \Rightarrow x : \Diamond_i A} \Diamond_i R \quad \frac{xRy, y : A, \Gamma \Rightarrow z : \perp}{x : \Diamond_c A, \Gamma \Rightarrow z : \perp} \Diamond_c L \quad \frac{x : \Box \neg A, \Gamma \Rightarrow x : \perp}{\Gamma \Rightarrow x : \Diamond_c A} \Diamond_c R
 \end{array}$$

Fig. 1. Ecumenical modal system labEK. In rules $\Box R, \Diamond_i L, \Diamond_c L$, the eigenvariable y does not occur free in any formula of the conclusion. In the rule W , either $A \neq \perp$ or $x \neq y$.

Example 1. Below the derivation in labEK of the distributivity of the intuitionistic diamond w.r.t the intuitionistic disjunction (see axiom k_2 in Sect. 5).

$$\begin{array}{c}
 \frac{}{xRy, y : A \Rightarrow y : A} \text{init} \quad \frac{}{xRy, y : B \Rightarrow y : B} \text{init} \\
 \frac{}{xRy, y : A \Rightarrow x : \Diamond_i A} \Diamond_i R \quad \frac{}{xRy, y : B \Rightarrow x : \Diamond_i B} \Diamond_i R \\
 \frac{}{xRy, y : A \Rightarrow x : \Diamond_i A \vee_i \Diamond_i B} \vee_i R \quad \frac{}{xRy, y : B \Rightarrow x : \Diamond_i A \vee_i \Diamond_i B} \vee_i R \\
 \frac{}{xRy, y : A \vee_i B \Rightarrow x : \Diamond_i A \vee_i \Diamond_i B} \Diamond_i L \\
 \frac{}{x : \Diamond_i (A \vee_i B) \Rightarrow x : \Diamond_i A \vee_i \Diamond_i B} \rightarrow_i L \\
 \frac{}{\Rightarrow x : \Diamond_i (A \vee_i B) \rightarrow_i (\Diamond_i A \vee_i \Diamond_i B)} \rightarrow_i R
 \end{array}$$

2.1 Ecumenical Birelational Models

The ecumenical birelational Kripke semantics, which is an extension of the proposal in [PR17] to modalities, was presented in [MPPS20].

Definition 2. A birelational model [PS86] is a quadruple $\mathcal{M} = (W, \leq, R, V)$ with a poset (W, \leq) , a binary relation $R \subset W \times W$, a monotone valuation $V : \langle W, \leq \rangle \rightarrow \langle 2^{\mathcal{P}}, \subseteq \rangle$ and

- F1. For all worlds w, v, v' , if wRv and $v \leq v'$, there is a w' such that $w \leq w'$ and $w'Rv'$;
- F2. For all worlds w', w, v , if $w \leq w'$ and wRv , there is a v' such that $w'Rv'$ and $v \leq v'$.

An ecumenical modal model is a birelational model such that truth of an ecumenical formula at a point w is the smallest relation \models_E satisfying

$\mathcal{M}, w \models_E p_i$	iff	$p_i \in V(w)$;
$\mathcal{M}, w \models_E A \wedge B$	iff	$\mathcal{M}, w \models_E A$ and $\mathcal{M}, w \models_E B$;
$\mathcal{M}, w \models_E A \vee_i B$	iff	$\mathcal{M}, w \models_E A$ or $\mathcal{M}, w \models_E B$;
$\mathcal{M}, w \models_E A \rightarrow_i B$	iff	for all v such that $w \leq v$, $\mathcal{M}, v \models_E A$ implies $\mathcal{M}, v \models_E B$;
$\mathcal{M}, w \models_E \neg A$	iff	for all v such that $w \leq v$, $\mathcal{M}, v \not\models_E A$;
$\mathcal{M}, w \models_E \perp$		never holds;
$\mathcal{M}, w \models_E \Box A$	iff	for all v, w' such that $w \leq w'$ and $w'Rv$, $\mathcal{M}, v \models_E A$.
$\mathcal{M}, w \models_E \Diamond_i A$	iff	there exists v such that wRv and $\mathcal{M}, v \models_E A$.
$\mathcal{M}, w \models_E p_c$	iff	$\mathcal{M}, w \models_E \neg(\neg p_i)$;
$\mathcal{M}, w \models_E A \vee_c B$	iff	$\mathcal{M}, w \models_E \neg(\neg A \wedge \neg B)$;
$\mathcal{M}, w \models_E A \rightarrow_c B$	iff	$\mathcal{M}, w \models_E \neg(A \wedge \neg B)$;
$\mathcal{M}, w \models_E \Diamond_c A$	iff	$\mathcal{M}, w \models_E \neg\Box\neg A$.

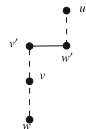
A formula A is valid in a model $\mathcal{M} = (W, \leq, R, V)$ if for all $w \in W$, $\mathcal{M}, w \models_E A$. A formula A is valid in a frame (W, \leq, R) if, for all valuations V , A is valid in the model (W, \leq, R, V) . Finally, we say that a formula is valid, if it is valid in all frames.

Since, restricted to intuitionistic and neutral connectives, \models_E is the usual birelational interpretation \models for IK [Sim94], and since the classical connectives are interpreted via the neutral ones using the double-negation translation, an ecumenical birelational model coincides with the standard birelational model for intuitionistic modal logic IK. Hence the following result easily holds from the similar result for IK.

Theorem 3 ([MPPS20]). *The system labEK is sound and complete w.r.t. the ecumenical modal semantics, that is, $\vdash_{\text{labEK}} x : A$ iff $\models_E A$.*

Remark 4. It is interesting to note that the relational semantics for the classical connectives is surprisingly more complex than for the intuitionistic ones. In fact, the definition of \models_E for the classical diamond is equivalent to

$$\mathcal{M}, w \models_E \Diamond_c A \text{ iff } \forall v \geq w. \exists u. v (\leq \circ R \circ \leq) u, \mathcal{M}, u \models_E A$$



where $v (\leq \circ R \circ \leq) u$ represents that there exist $v', w' \in W$ such that $v \leq v'$, $v' R w'$ and $w' \leq u$. Although intriguing, this kind of two-level semantics also appears in the relational semantics for classical logic in [ILH10], where the forcing relation is defined on top of the primitive notion of “strong refutation”.

3 A Nested System for Ecumenical Modal Logic

The two main criticisms regarding system labEK are: (i) it is not *pure*, in the sense that negation still plays an important role on interpreting classical connectives – for example, the rule $\diamond_c R$ introduces a classical diamond via its boxed negated version; and (ii) it includes *labels* in the technical machinery, hence allowing one to write sequents that cannot always be interpreted within the ecumenical modal language.

This section is devoted to tackle these points and propose a *pure label free* calculus for ecumenical modalities, where every basic object of the calculus can be translated as a formula in the language of the logic, with no use of auxiliary negations.

The inspiration comes from Girard’s notion of *stoup* [Gir91] and Straßburger’s *nested system* for IK [Str13]. The main idea is to let sequents of the form $\Sigma \Rightarrow \Pi$, with Σ, Π multisets of formulas, go through a two-phase refinement: the first one is to separate the succedent Π into two parts: one that is essentially classical; and another containing a single formula, the stoup. The second one is to add nested layers to sequents, which intuitively corresponds to worlds in a relational structure [Fit14, Brü09, Pog09].

The primary key concept to distinguish which formulas are allowed or not in the stoup is the following notion of polarity.

Definition 5. *A formula is called negative if its main connective is classical or the negation, and positive otherwise (we will use N for negative and P for positive formulas).*

The structure of a nested sequent for ecumenical modal logics is a tree whose nodes are multisets of formulas, just like in [Str13], with the relationship between parent and child in the tree represented by bracketing $[\cdot]$. The difference however is that the ecumenical formulas can be *left inputs* (in the left contexts – marked with a full circle \bullet), *right inputs* (in the classical right contexts – marked with a triangle ∇) or a *single right output* (the stoup – marked with a white circle \circ).

Definition 6. *Ecumenical nested sequents are defined in terms of a grammar of input sequents (written Λ) and full sequents (written Γ) where the left/right input formulas are denoted by A^\bullet and A^∇ , respectively, and A° denote the output formula. When the distinction between input and full sequents is not essential or cannot be made explicit, we will use Δ to stand for either case.*

$$\Lambda := \emptyset \mid A^\bullet, \Lambda \mid A^\nabla, \Lambda \mid [\Lambda] \qquad \Gamma := A^\circ, \Lambda \mid [\Gamma], \Lambda \qquad \Delta := \Lambda \mid \Gamma$$

As usual, we allow sequents to be empty, and we consider sequents to be equal modulo associativity and commutativity of the comma.

We write Γ^{\perp° for the result of replacing an output formula from Γ by \perp° , while Λ^{\perp° represents the result of adding anywhere of the input context Λ the output formula \perp° . Finally, Δ^* is the result of erasing an output formula (if any) from Δ .

Observe that full sequents Γ necessarily contain exactly one output-like formula, having the form $A_1, [A_2, [\dots, [A_n, A^\circ]] \dots]$.

Example 7. The nested sequent $\diamond_c A^\nabla, [\neg A^\circ]$ represents a tree of sequents where $\diamond_c A$ is in the right (classical) input context of the root sequent, while $\neg A$ is in the output context (stoup), in the leaf sequent.

The next definition (of contexts) allows for identifying subtrees within nested sequents, which is necessary for introducing inference rules in this setting.

Definition 8. An n -ary context $\Delta\{^1\} \dots \{^n\}$ is like a sequent but contains n pairwise distinct numbered holes $\{^i\}$ wherever a formula may otherwise occur. It is a full or a input context when $\Delta = \Gamma$ or Λ respectively.

Given n sequents $\Delta_1, \dots, \Delta_n$, we write $\Delta\{\Delta_1\} \dots \{\Delta_n\}$ for the sequent where the i -th hole in $\Delta\{^1\} \dots \{^n\}$ has been replaced by Δ_i (for $1 \leq i \leq n$), assuming that the result is well-formed, i.e., there is at most one output formula. If $\Delta_i = \emptyset$ the hole is removed.

Given two nested contexts $\Gamma^i\{^j\} = \Delta_1^i, [\Delta_2^i, [\dots, [\Delta_n^i, \{^j\}]] \dots]$, $i \in \{1, 2\}$, their merge¹ is

$$\Gamma^1 \otimes \Gamma^2\{^j\} = \Delta_1^1, \Delta_1^2, [\Delta_2^1, \Delta_2^2, [\dots, [\Delta_n^1, \Delta_n^2, \{^j\}]] \dots]$$

Figure 2 presents the nested sequent system nEK for ecumenical modal logic EK.

Example 9. Below left the nested proof corresponding to the labeled one in Example 1.

$$\frac{\frac{\frac{\overline{[A^\bullet, A^\circ]}}{\diamond_i A^\circ, [A^\bullet]} \text{init}}{\diamond_i A \vee_i \diamond_i B^\circ, [A^\bullet]} \diamond_i^\circ}{\diamond_i A \vee_i \diamond_i B^\circ, [A \vee_i B^\bullet]} \vee_i^\circ \quad \frac{\frac{\frac{\overline{[B^\bullet, B^\circ]}}{\diamond_i B^\circ, [B^\bullet]} \text{init}}{\diamond_i A \vee_i \diamond_i B^\circ, [B^\bullet]} \diamond_i^\circ}{\diamond_i A \vee_i \diamond_i B^\circ, [A \vee_i B^\bullet]} \vee_i^\circ}{\diamond_i(A \vee_i B)^\bullet, \diamond_i A \vee_i \diamond_i B^\circ} \diamond_i^\bullet \rightarrow_i^\circ \quad \frac{\frac{\frac{\overline{[A^\bullet, A^\nabla, \perp^\circ]}}{[A^\nabla, \neg A^\circ]} \text{init}_c}{\diamond_c A^\nabla, [\neg A^\circ]} \neg^\circ}{\square \neg A^\circ, \diamond_c A^\nabla} \diamond_c^\nabla}{\neg \square \neg A^\bullet, \diamond_c A^\nabla, \perp^\circ} \square^\circ}{\neg \square \neg A^\bullet, \diamond_c A^\circ} \neg^\bullet \text{sto}$$

The derivation above right shows part of the proof that \diamond_c can be defined from \square ($\diamond_c A \equiv \neg \square \neg A$). Note the instance of the classical general version of the initial axiom, init_c (see Theorem 11 in the next section). It also illustrates well

¹ As observed in [Pog09, Lel19], the merge is a “zipping” of the two nested sequents along the path from the root to the hole.

$$\begin{array}{c}
\frac{}{\Lambda\{p_i^\bullet, p_i^\circ\}} \text{init} \quad \frac{}{\Gamma\{\perp^\bullet\}} \perp^\bullet \quad \frac{\Gamma^\perp{}^\circ}{\Gamma} \text{W} \quad \frac{\Gamma^\perp{}^\circ\{p_i^\bullet\}}{\Gamma^\perp{}^\circ\{p_c^\bullet\}} p_c^\bullet \quad \frac{\Gamma^\perp{}^\circ\{p_i^\nabla\}}{\Gamma^\perp{}^\circ\{p_c^\nabla\}} p_c^\nabla \\
\frac{\Gamma\{A^\bullet, B^\bullet\}}{\Gamma\{A \wedge B^\bullet\}} \wedge^\bullet \quad \frac{\Lambda\{A^\circ\} \quad \Lambda\{B^\circ\}}{\Lambda\{A \wedge B^\circ\}} \wedge^\circ \quad \frac{\Gamma^*\{\neg A^\bullet, A^\circ\}}{\Gamma^\perp{}^\circ\{\neg A^\bullet\}} \neg^\bullet \quad \frac{\Gamma^\perp{}^\circ\{A^\bullet\}}{\Gamma^\perp{}^\circ\{\neg A^\nabla\}} \neg^\nabla \\
\frac{\Gamma\{A^\bullet\} \quad \Gamma\{B^\bullet\}}{\Gamma\{A \vee_i B^\bullet\}} \vee_i^\bullet \quad \frac{\Lambda\{A_j^\circ\}}{\Lambda\{A_j \vee_i A_2^\circ\}} \vee_{ij}^\circ \quad \frac{\Gamma^\perp{}^\circ\{A^\bullet\} \quad \Gamma^\perp{}^\circ\{B^\bullet\}}{\Gamma^\perp{}^\circ\{A \vee_c B^\bullet\}} \vee_c^\bullet \quad \frac{\Gamma^\perp{}^\circ\{A^\nabla, B^\nabla\}}{\Gamma^\perp{}^\circ\{A \vee_c B^\nabla\}} \vee_c^\nabla \\
\frac{\Gamma^*\{A \rightarrow_i B^\bullet, A^\circ\} \quad \Gamma\{B^\bullet\}}{\Gamma\{A \rightarrow_i B^\bullet\}} \rightarrow_i^\bullet \quad \frac{\Lambda\{A^\bullet, B^\circ\}}{\Lambda\{A \rightarrow_i B^\circ\}} \rightarrow_i^\circ \quad \frac{\Gamma^*\{A \rightarrow_c B^\bullet, A^\circ\} \quad \Gamma^\perp{}^\circ\{B^\bullet\}}{\Gamma^\perp{}^\circ\{A \rightarrow_c B^\bullet\}} \rightarrow_c^\bullet \\
\frac{\Gamma^\perp{}^\circ\{A^\bullet, B^\nabla\}}{\Gamma^\perp{}^\circ\{A \rightarrow_c B^\nabla\}} \rightarrow_c^\nabla \quad \frac{\Delta_1\{\Box A^\bullet, [A^\bullet, \Delta_2]\}}{\Delta_1\{\Box A^\bullet, [\Delta_2]\}} \Box^\bullet \quad \frac{\Lambda\{[A^\circ]\}}{\Lambda\{\Box A^\circ\}} \Box^\circ \quad \frac{\Gamma\{[A^\bullet]\}}{\Gamma\{\Diamond_i A^\bullet\}} \Diamond_i^\bullet \quad \frac{\Lambda_1\{[A^\circ, \Delta_2]\}}{\Lambda_1\{\Diamond_i A^\circ, [\Delta_2]\}} \Diamond_i^\circ \\
\frac{\Gamma^\perp{}^\circ\{[A^\bullet]\}}{\Gamma^\perp{}^\circ\{\Diamond_c A^\bullet\}} \Diamond_c^\bullet \quad \frac{\Delta_1^\perp{}^\circ\{\Diamond_c A^\nabla, [A^\nabla, \Delta_2^\perp{}^\circ]\}}{\Delta_1^\perp{}^\circ\{\Diamond_c A^\nabla, [\Delta_2^\perp{}^\circ]\}} \Diamond_c^\nabla \quad \frac{\Gamma^*\{P^\nabla, P^\circ\}}{\Gamma^\perp{}^\circ\{P^\nabla\}} \text{dec} \quad \frac{\Lambda\{N^\nabla, \perp^\circ\}}{\Lambda\{N^\circ\}} \text{sto}
\end{array}$$

Fig. 2. Nested ecumenical modal system nEK. P is a *positive* formula, N is a *negative* formula.

the relationship between nestings, classical inputs, and birelational structures: reading the proof bottom-up, the **sto** rule is a delay on applying rules over classical connectives. It corresponds to moving the formula up w.r.t. \leq in the birelational semantics. The rule \Box° , on the other hand, slides the formula to a fresh new world, related to the former one through the relation R . Finally, rule \rightarrow° also moves up the formula w.r.t. \leq . Compare this description with the image in Remark 4. In this paper, we will not explore *formally* the relationship between delays/negations/nestings and semantics.

3.1 Harmony

A logical connective is called *harmonious* in a certain proof system if there exists a certain balance between the rules defining it. For example, in natural deduction based systems, harmony is ensured when introduction/elimination rules do not contain insufficient/excessive amounts of information [DD20]. In sequent calculus, this property is often guaranteed by the admissibility of a general initial axiom (*identity-expansion*) and of the cut rule (*cut-elimination*). In the following, we will prove harmony, together with some intermediate results. We start with a proof theoretical result in nEK, which has a standard proof (see [PPdP19] and [MPPS20] for similar results).

Lemma 10. 1. In nEK, the rules $\vee_c^\bullet, \vee_c^\nabla, \rightarrow_c^\bullet, \rightarrow_c^\nabla, \neg^\bullet, \neg^\circ, p_c^\bullet, p_c^\nabla, \Diamond_c^\bullet, \Diamond_c^\nabla$ and **dec** are invertible, that is, in any application of such rules, if the conclusion is a provable nested sequent so are the premises.

2. The rules $\wedge^\bullet, \wedge^\circ, \vee_i^\bullet, \rightarrow_i^\circ, \Diamond_i^\bullet, \Box^\bullet, \Box^\circ$ and **sto** are totally invertible, that is, they are invertible and can be applied in any contexts.

3. The following rules are admissible in nEK

$$\frac{\Gamma}{\Lambda \otimes \Gamma} W_c \quad \frac{\Lambda \otimes \Lambda \otimes \Gamma}{\Lambda \otimes \Gamma} C_c$$

Proof. The proofs are by standard induction on the height of derivations. The proof of admissibility of W_c does not depend on any other result, while the admissibility of C_c depends on the invertibility results above.

The invertible but not totally invertible rules in nEK concern negative formulas, hence they can only be applied in the presence of empty stoups (\perp°). Note also that the rules W, \vee_i° , and \diamond_i° are not invertible, while \rightarrow_i^\bullet is invertible only w.r.t. the right premise.

Theorem 11. *The following rules are admissible in nEK*

$$\frac{}{\Lambda\{A^\bullet, A^\circ\}} \text{init}_i \quad \frac{}{\Gamma^{\perp^\circ}\{A^\bullet, A^\nabla\}} \text{init}_c$$

Proof. The proof of admissibility of the general initial axioms is by mutual induction. Below we show the modal cases where, by induction hypothesis, instances of the axioms hold for the premises.

$$\frac{\frac{\frac{}{\Gamma^{\perp^\circ}\{[A^\bullet, A^\nabla]\}} \text{init}_c}{\Gamma^{\perp^\circ}\{\diamond_c A^\nabla, [A^\bullet]\}} \diamond_c^\nabla}{\Gamma^{\perp^\circ}\{\diamond_c A^\bullet, \diamond_c A^\nabla\}} \diamond_c^\bullet \quad \frac{\frac{\frac{}{\Lambda\{[A^\bullet, A^\circ]\}} \text{init}_i}{\Lambda\{\Box A^\bullet, [A^\circ]\}} \Box^\bullet}{\Lambda\{\Box A^\bullet, \Box A^\circ\}} \Box^\circ$$

Proving admissibility of cut rules in sequent based systems with multiple contexts is often tricky, since the cut formulas can change contexts during cut reductions. This is the case for nEK. The proof is by mutual induction, with inductive measure (n, m) where m is the cut-height, the cumulative height of derivations above the cut, and n is the ecumenical weight of the cut-formula, defined as

$$\begin{aligned} \text{ew}(P_i) &= \text{ew}(\perp) = 0 & \text{ew}(A \star B) &= \text{ew}(A) + \text{ew}(B) + 1 \text{ if } \star \in \{\wedge, \rightarrow_i, \vee_i\} \\ \text{ew}(P_c) &= 4 & \text{ew}(\heartsuit A) &= \text{ew}(A) + 1 \text{ if } \heartsuit \in \{\neg, \diamond_i, \Box\} \\ \text{ew}(\diamond_c A) &= \text{ew}(A) + 4 & \text{ew}(A \circ B) &= \text{ew}(A) + \text{ew}(B) + 4 \text{ if } \circ \in \{\rightarrow_c, \vee_c\} \end{aligned}$$

Intuitively, the ecumenical weight measures the amount of extra information needed (the negations added) to define classical connectives from intuitionistic and neutral ones.

Theorem 12. *The following intuitionistic and classical cut rules are admissible in nEK*

$$\frac{\Lambda\{P^\circ\} \quad \Gamma\{P^\bullet\}}{\Lambda \otimes \Gamma\{\emptyset\}} \text{cut}^\circ \quad \frac{\Lambda^{\perp^\circ}\{N^\nabla\} \quad \Gamma\{N^\bullet\}}{\Lambda \otimes \Gamma\{\emptyset\}} \text{cut}^\nabla$$

Proof. The dynamic of the proof is the following: cut applications either move up in the proof, i.e. the cut-height is reduced, or are substituted by simpler cuts of the same kind, i.e. the ecumenical weight is reduced, as in usual cut-elimination reductions. The cut instances alternate between intuitionistic and classical (and vice-versa) in the principal cases, where the polarity of the subformulas flip. We sketch the main cut-reductions.

– Base cases. Consider the derivation below left

$$\frac{\frac{\Lambda\{p_i^\circ\}}{\Lambda \otimes \Gamma\{\emptyset\}} \frac{\overline{\Gamma\{p_i^\bullet\}}}{\text{init}}}{\text{cut}^\circ} \qquad \frac{\Lambda\{p_i^\circ\}}{\Lambda \otimes \Gamma^*\{p_i^\circ\}} W_c$$

If p_i^\bullet is principal, then $\Gamma\{p_i^\bullet\} = \Gamma^*\{p_i^\circ, p_i^\bullet\}$ and the reduction is the one above right.

If p_i^\bullet is not principal, then there is an atom q for which the pair q_i°, q_i^\bullet appears in $\Lambda \otimes \Gamma\{\emptyset\}$ and the reduction is a trivial one. Similar analyses hold for cut^∇ , when the left premise is an instance of init , and for the other axioms.

– Non-principal cases. In all the cases where the cut-formula is not principal in one of the premises, the cut moves upwards. We illustrate the most significant case, where a decide rule is applied, as in the derivation below left.

$$\frac{\frac{\frac{\Lambda\{P^\nabla, P^\circ\}\{N^\nabla\}}{\Lambda^{\perp^\circ}\{P^\nabla\}\{N^\nabla\}} \frac{\pi_1}{\text{dec}} \quad \frac{\pi_2}{\Gamma\{N^\bullet\}}}{\Lambda\{P^\nabla\} \otimes \Gamma\{\emptyset\}} \text{cut}^\nabla \qquad \frac{\frac{\Lambda\{P^\nabla, P^\circ\}\{N^\nabla\}}{\Lambda\{P^\nabla, P^\circ\} \otimes \Gamma^*\{\emptyset\}} \frac{\pi_1}{\text{dec}} \quad \frac{\pi_2}{\Gamma^{\perp^\circ}\{N^\bullet\}}}{\Lambda\{P^\nabla\} \otimes \Gamma^{\perp^\circ}\{\emptyset\}} \text{cut}^\nabla$$

The cut moves upwards in the right premise until N^\bullet is principal in the bottom-most step of π_2 , at which point $\Gamma = \Gamma^{\perp^\circ}$ and dec can move below the cut, obtaining the derivation above right.

– Principal cases. If the cut formula is principal in both premises, then we need to be extra-careful with the polarities. We illustrate below the reduction for case where $N = P \rightarrow_c Q$, with P, Q positive.

$$\frac{\frac{\frac{\Lambda^{\perp^\circ}\{P^\bullet, Q^\nabla\}}{\Lambda^{\perp^\circ}\{P \rightarrow_c Q^\nabla\}} \frac{\pi_1}{\rightarrow_c^\nabla} \quad \frac{\frac{\Gamma^*\{P \rightarrow_c Q^\bullet, P^\circ\}}{\Gamma^{\perp^\circ}\{P \rightarrow_c Q^\bullet\}} \frac{\pi_2}{\rightarrow_c^\bullet} \quad \frac{\pi_3}{\Gamma^{\perp^\circ}\{Q^\bullet\}}}{\Lambda \otimes \Gamma^{\perp^\circ}\{\emptyset\}} \text{cut}_0^\nabla$$

reduces to

$$\frac{\frac{\frac{\frac{\Lambda^{\perp^\circ}\{P^\bullet, Q^\nabla\}}{\Lambda^{\perp^\circ}\{P \rightarrow_c Q^\nabla\}} \frac{\pi_1}{\rightarrow_c^\nabla} \quad \frac{\Gamma^*\{P \rightarrow_c Q^\bullet, P^\circ\}}{\Lambda \otimes \Gamma^*\{P^\circ\}} \frac{\pi_2}{\text{cut}_2^\nabla} \quad \frac{\pi_3}{\Lambda^{\perp^\circ}\{P^\bullet, \neg Q^\bullet\}} \frac{\pi_1^\equiv}{\text{cut}^\circ}}{\Lambda^2 \otimes \Gamma^{\perp^\circ}\{\neg Q^\bullet\}} \text{cut}_1^\nabla}{\frac{\Lambda^2 \otimes \Gamma^* \otimes \Gamma^{\perp^\circ}\{\emptyset\}}{\Lambda \otimes \Gamma^{\perp^\circ}\{\emptyset\}} C_c}$$

where π_1^{\equiv} is the same as π_1 where every application of the rule **dec** over Q^∇ is substituted by an application of \neg^\bullet over $\neg Q^\bullet$. Observe that the cut-formula of cut_1^∇ has lower ecumenical weight than cut_0^∇ , while the cut-height of cut_2^∇ is smaller than cut_0^∇ . Finally, observe that this is a non-trivial cut-reduction: usually, the cut over the implication is replaced by a cut over Q first. Due to polarities, if Q is positive, then $\neg Q$ is negative and cutting over it will add to the left context the classical information Q , hence mimicking the behavior of formulas in the right input context.

4 Soundness

In this section we will show that all rules presented in Fig. 2 are sound w.r.t. the ecumenical birelational model. The idea is to prove that the rules of the system **nEK** preserve *validity*, in the sense that if the interpretation of the premises is valid, so is the interpretation of the conclusion.

The first step is to determine the interpretation of ecumenical nested sequents. In this section, we will present the translation of nestings to labeled sequents, hence establishing, at the same time, soundness of **nEK** and the relation between this system with **labEK**.

First of all, we observe that the entailment in ecumenical systems is intrinsically intuitionistic, in the sense that $\Gamma \Rightarrow B$ is valid iff $\bigwedge \Gamma \rightarrow_i B$ is valid [PPdP19]. Moreover, the classical connectives are defined semantically via the intuitionistic ones by sporadic double-negation. Another interesting aspect is that, in the labeled ecumenical modal system **labEK**, fresh world labels can be created (bottom-up) by the box operator in succedents and both diamond connectives in antecedents. Yet, once this new label is created, it is shared by all modal formulas, independently of their intuitionistic or classical nature.

This suggests the following interpretation of nested into labeled ecumenical sequents.

Definition 13. *Let $\Pi^\bullet, \Pi^\nabla, \Pi^\circ$ represent that all formulas in the each multiset are respectively input left, right, or output formulas. The underlying Π will represent in all cases the corresponding multiset of unmarked formula in \mathcal{A} . The translation $\llbracket \cdot \rrbracket_x$ from nested into labeled sequents is defined recursively by*

$$\llbracket \Pi_1^\bullet, \Pi_2^\nabla, \Pi_3^\circ, [\Delta_1], \dots, [\Delta_n] \rrbracket_x := (\{xR x_i\}_i, x : \Pi_1, x : \neg \Pi_2 \Rightarrow x : \Pi_3) \otimes \{\llbracket \Delta_i \rrbracket_{x_i}\}_i$$

where $1 \leq i \leq n$, x_i are fresh, and the merge operation on labeled sequents is defined as

$$(\Sigma_1 \Rightarrow \Pi_1) \otimes (\Sigma_2 \Rightarrow \Pi_2) := \Sigma_1, \Sigma_2 \Rightarrow \Pi_1, \Pi_2$$

Given \mathcal{R} a set of relational formulas, we will denote by xR^*z the fact that there is a path from x to z in \mathcal{R} , i.e., there are $y_j \in \mathcal{R}$ for $0 \leq j \leq k$ such that $x = y_0, y_{j-1}Ry_j$ and $y_k = z$.

That is, right input formulas are translated as negated left formulas in labeled sequents, and nestings correspond to relational atoms. The next result shows that, in fact, this interpretation is correct.

Theorem 14. *Let Γ be a nested sequent and x be any label, if $\vdash_{\text{nEK}} \Gamma$ then $\vdash_{\text{labEK}} \llbracket \Gamma \rrbracket_x$.*

Proof. The proof is by structural induction on the proof π of Γ . We will illustrate a classical and a modal case.

- If the last rule applied in π is \vee_c^∇ , by induction hypothesis,

$$\llbracket \Gamma^{\perp^\circ} \{A^\nabla, B^\nabla\} \rrbracket_x = \mathcal{R}, \Sigma, z : \neg A, z : \neg B \Rightarrow x : \perp$$

is provable for a set \mathcal{R} of relational atoms and a multiset Σ of labeled formulas, obtained by translating $\Gamma^{\perp^\circ} \{\emptyset\}$ and such that $xR^*z \in \mathcal{R}$. Hence:

$$\frac{\frac{\mathcal{R}, \Sigma, z : \neg A, z : \neg B \Rightarrow z : \perp}{\mathcal{R}, \Sigma \Rightarrow z : A \vee_c B} \vee_c R}{\mathcal{R}, \Sigma, z : \neg(A \vee_c B) \Rightarrow x : \perp} \neg L$$

- If the last rule applied in π is \diamond_c^∇ , by induction hypothesis,

$$\llbracket \Delta_1^{\perp^\circ} \{ \diamond_c A^\nabla, [A^\nabla, \Delta_2^{\perp^\circ}] \} \rrbracket_x = \mathcal{R}, zRy, \Sigma, z : \neg(\diamond_c A), y : \neg A \Rightarrow x : \perp$$

is provable for a set \mathcal{R} and a multiset Σ of relational and labeled formulas, resp., obtained by translating sequents $\Delta_1^{\perp^\circ}$ and $\Delta_2^{\perp^\circ}$, and where $xR^*z \in \mathcal{R}$. Hence:

$$\frac{\frac{\frac{\mathcal{R}, zRy, \Sigma, z : \neg(\diamond_c A), y : \neg A \Rightarrow z : \perp}{\mathcal{R}, zRy, \Sigma, z : \neg(\diamond_c A), z : \Box \neg A, y : \neg A \Rightarrow z : \perp} \Box L}{\mathcal{R}, zRy, \Sigma, z : \neg(\diamond_c A), z : \Box \neg A \Rightarrow z : \perp} \diamond_c R}{\mathcal{R}, zRy, \Sigma, z : \neg(\diamond_c A) \Rightarrow z : \diamond_c A} \neg L$$

Due to rule W in labEK, the label assigned to \perp on the right is irrelevant in both cases.

The proof above also establishes the relationship between *proofs* in nEK and labEK: the right input context stores *negative* formulas, which are in fact negated positive formulas (as in [Gir91]), and the decision rule **dec** in nEK is mimicked in labEK by applications of the left rule for negation. In this way, the use of nestings together with decision and store rules imposes a *discipline* on rule applications in labeled systems.

Theorems 3 and 14 immediately imply the following.

Corollary 15. *Nested system nEK is sound w.r.t. ecumenical birelational semantics.*

Finally, we observe that from labeled to nested sequents, on the other hand, is not a simple task, sometimes even impossible. In fact, although the relational

atoms of a sequent appearing in a **labEK** *proof* can be arranged so as to correspond to nestings, in general, if the relational context is not tree-like [GR12], the existence of such a translation is not clear. For instance, how should the sequent $xRy, yRx, x : A \Rightarrow y : B$ be interpreted in modal systems with symmetrical relations?

Hence, we will avoid the translation method for proving completeness, which will instead be proven in Sect. 5 with respect to the *Hilbert system*.

However, thanks to their tree shape, it is possible to interpret nested sequents as ecumenical modal formulas, and hence prove soundness in the same way as in [Str13]. This direct interpretation of nested sequents as ecumenical formulas means that **nEK** is a so-called *internal proof system*. We thus finish this section by sketching an alternative proof of soundness of **nEK** w.r.t. the ecumenical birelational semantics.

Definition 16. *The formula translation $\text{et}(\cdot)$ for ecumenical nested sequents is given by*

$$\begin{array}{ll} \text{et}(\emptyset) := \top & \text{et}(A^\bullet, A) := A \wedge \text{et}(A) \\ \text{et}(A^\nabla, A) := \neg A \wedge \text{et}(A) & \text{et}([\Lambda_1], \Lambda_2) := \diamond_i \text{et}(\Lambda_1) \wedge \text{et}(\Lambda_2) \\ \text{et}(A, A^\circ) := \text{et}(A) \rightarrow_i A & \text{et}(A, [\Gamma]) := \text{et}(A) \rightarrow_i \square \text{et}(\Gamma) \end{array}$$

where all occurrences of $A \wedge \top$ and $\top \rightarrow_i A$ are simplified to A . We say a sequent is valid if its corresponding formula is valid.

The following technical lemma holds in **nEK**, adapting the proof from **NIK**.

Lemma 17. [Str13, Lemmas 4.3 and 4.4] *Let Δ and Σ be input (resp. full) sequents, and $\Gamma\{\}$ be a full context (resp. $\Lambda\{\}$ be an input context). If $\text{et}(\Delta) \rightarrow_i \text{et}(\Sigma)$ is valid, then $\text{et}(\Gamma\{\Sigma\}) \rightarrow_i \text{et}(\Gamma\{\Delta\})$ and $\text{et}(\Lambda\{\Delta\}) \rightarrow_i \text{et}(\Lambda\{\Sigma\})$ are valid.*

The next theorem shows that the rules of **nEK** preserve validity in ecumenical modal frames w.r.t. the formula interpretation $\text{et}(\cdot)$.

Theorem 18. *Let*

$$\frac{\Gamma_1 \quad \dots \quad \Gamma_n}{\Gamma} r \quad n \in \{0, 1, 2\}$$

be an instance of the rule r in the system **nEK**. Then $\text{et}(\Gamma_1) \wedge \dots \wedge \text{et}(\Gamma_n) \rightarrow_i \text{et}(\Gamma)$ is valid in the birelational ecumenical semantics.

Proof. The proof for the intuitionistic propositional and modal connectives follows the same lines as in [Str13]. For the other cases, due to Lemma 17, it is sufficient to show that the following formulas are valid; all such proofs are straightforward.

1. for W : $\perp \rightarrow_i A$
2. for \neg^\bullet : $(\neg A \rightarrow_i A) \rightarrow_i (\neg\neg A)$
3. for \neg^∇ : $\neg A \rightarrow_i \neg A$
4. for \vee_c^\bullet : $(\neg A \wedge \neg B) \rightarrow_i (\neg(A \vee_c B))$

5. for \vee_c^∇ : $(\neg(\neg A \wedge \neg B)) \rightarrow_i (A \vee_c B)$
6. for \rightarrow_c^\bullet : $((A \rightarrow_c B) \rightarrow_i A) \wedge (\neg B) \rightarrow_i (\neg(A \rightarrow_c B))$
7. for \rightarrow_c^∇ : $(\neg(A \wedge \neg B)) \rightarrow_i (A \rightarrow_c B)$
8. for p_c^\bullet : $(\neg p_i) \rightarrow_i (\neg p_c)$
9. for p_c^∇ : $(\neg\neg p_i) \rightarrow_i p_c$
10. for \diamond_c^\bullet : $(\neg\diamond_i A) \rightarrow_i (\neg\diamond_c A)$
11. for \diamond_c^∇ : $(\Box\neg A) \rightarrow_i (\neg\diamond_c A)$

5 Completeness

Classical modal logic K is defined as propositional classical logic, extended with the *necessitation rule* (presented in Hilbert style) $A/\Box A$ and the *distributivity axiom* k : $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$.

There are, however, many variants of axiom k that induce logics that are classically, but not intuitionistically, equivalent (see [PS86, Sim94]). In fact, the following axioms follow from k via the De Morgan laws, but are intuitionistically independent

$$\begin{array}{ll} k_1 : \Box(A \rightarrow B) \rightarrow (\diamond A \rightarrow \diamond B) & k_2 : \diamond(A \vee B) \rightarrow (\diamond A \vee \diamond B) \\ k_3 : (\diamond A \rightarrow \Box B) \rightarrow \Box(A \rightarrow B) & k_4 : \diamond\perp \rightarrow \perp \end{array}$$

Combining axiom k with axioms $k_1 - k_4$ defines intuitionistic modal logic IK [PS86].

In the ecumenical setting, this discussion is even more interesting, since there are many more variants of k , depending on the classical or intuitionistic interpretation of implications and diamonds.

Theorems of *ecumenical modal logic* EK are defined as the formulas that are derivable from the axioms of intuitionistic propositional logic plus the definitions of classical operators using negation and the intuitionistic versions of the axioms $k-k_4$. We can show that all these EK axioms are provable in nEK (e.g. Example 9). Hence, in the presence of cut-elimination (Sect. 3.1), we can deduce completeness of nEK w.r.t. EK .

Theorem 19. *Every theorem of the logic EK is provable in nEK .*

Moreover, a formula is derivable in EK iff it is valid in all birelational frames (see [MPPS20]), which in turn implies completeness of nEK w.r.t. birelational semantics.

6 Extracting Fragments

In this section, we will study pure classical and intuitionistic fragments of nEK . For the sake of simplicity, negation will not be considered a primitive connective, it will rather take its respective intuitionistic or classical form.

Definition 20. An ecumenical modal formula C is classical (intuitionistic) if it is built from classical (intuitionistic) atomic propositions using only neutral and classical (intuitionistic) connectives but negation, which will be replaced by $A \rightarrow_c \perp$ ($A \rightarrow_i \perp$).

The first thing to observe is that, when only pure fragments are concerned, weakening is admissible. Observe that this is not the case for the whole system \mathbf{nEK} . In fact, $A \vee_c \neg A^\nabla, C^\circ$ is provable in \mathbf{nEK} for any formula C , but the proof necessarily starts with an application of the rule \mathbf{W} if, e.g., C is an atomic formula p_i .

Proposition 21. Let \mathbf{nEK}_i (\mathbf{nEK}_c) be the system obtained from $\mathbf{nEK} - \mathbf{W}$ by restricting the rules to the intuitionistic (classical) case (see Figs. 3 and 4). The rule \mathbf{W} is admissible in \mathbf{nEK}_i and \mathbf{nEK}_c .

Proof. For the intuitionistic fragment, the proof is standard, by induction on the height of derivations (considering all possible rule applications). The classical case is more involved. The idea is that classical formulas in the stoup are eagerly decomposed until either an axiom is applied, or the formula is stored in the classical input context and the stoup becomes empty. This is only possible because the rules \wedge° and \square° are totally invertible and all the other rules in \mathbf{nEK}_c are invertible (Lemma 10). Formally, the following proof strategy is complete for \mathbf{nEK}_c , when proving a nested sequent Γ :

- i. Apply the rules $\wedge^\bullet, \wedge^\circ, \square^\bullet, \square^\circ$ and \mathbf{sto} eagerly, obtaining leaves of the form $A\{\perp^\circ\}$.
- ii. Apply any rule of \mathbf{nEK}_c eagerly, until either finishing the proof with an axiom application or obtaining leaves of the form $A\{P^\circ\}$, where P is a positive formula in \mathbf{nEK}_c , that is, having as main connective \wedge or \square . Start again from step (i).

Observe that weakening is never applied, since a positive classical formula P° is totally decomposable into negative subformulas of the form N° , which are stored in the classical input context as N^∇ , or \perp° .

This result clarifies the role of weakening in \mathbf{nEK} : it serves as a bridge between intuitionistic and classical parts of a derivation and its application can be restricted to just below classical rules.

Since weakening is not present, \mathbf{nEK}_i matches exactly the system \mathbf{NIK} in [Str13].

Fact 22. The intuitionistic fragment of \mathbf{nEK} is Straßburger's system \mathbf{NIK} .

For the classical fragment, the discipline presented in the proof of Proposition 21 is interesting as it resembles focused search [LM11] in \mathbf{nEK}_c : neutral connectives are handled in the stoup, while rules on classical connectives are applied in classical context.

Yet, this discipline does not match the focusing defined in [CMS16], since in that work diamond is considered positive and box negative, while the ecumenical system enforces the opposite polarity assignment. The task of providing

$$\begin{array}{c}
\frac{}{\Lambda\{p_i^\bullet, p_i^\circ\}} \text{init} \quad \frac{}{\Gamma\{\perp^\bullet\}} \perp^\bullet \quad \frac{\Gamma\{A^\bullet, B^\bullet\}}{\Gamma\{A \wedge B^\bullet\}} \wedge^\bullet \quad \frac{\Lambda\{A^\circ\} \quad \Lambda\{B^\circ\}}{\Lambda\{A \wedge B^\circ\}} \wedge^\circ \\
\frac{\Gamma\{A^\bullet\} \quad \Gamma\{B^\bullet\}}{\Gamma\{A \vee_i B^\bullet\}} \vee_i^\bullet \quad \frac{\Lambda\{A_j^\circ\}}{\Lambda\{A_1 \vee_i A_2^\circ\}} \vee_{ij}^\circ \quad \frac{\Gamma^*\{A \rightarrow_i B^\bullet, A^\circ\} \quad \Gamma\{B^\bullet\}}{\Gamma\{A \rightarrow_i B^\bullet\}} \rightarrow_i^\bullet \quad \frac{\Lambda\{A^\bullet, B^\circ\}}{\Lambda\{A \rightarrow_i B^\circ\}} \rightarrow_i^\circ \\
\frac{\Delta_1\{\Box A^\bullet, [A^\bullet, \Delta_2]\}}{\Delta_1\{\Box A^\bullet, [\Delta_2]\}} \Box^\bullet \quad \frac{\Lambda\{[A^\circ]\}}{\Lambda\{\Box A^\circ\}} \Box^\circ \quad \frac{\Gamma\{[A^\bullet]\}}{\Gamma\{\Diamond A^\bullet\}} \Diamond_i^\bullet \quad \frac{\Lambda_1\{[A^\circ, \Delta_2]\}}{\Lambda_1\{\Diamond_i A^\circ, [\Delta_2]\}} \Diamond_i^\circ
\end{array}$$

Fig. 3. Intuitionistic fragment nEK_i.

$$\begin{array}{c}
\frac{}{\Gamma\{p_c^\bullet, p_c^\vee\}} \text{init} \quad \frac{}{\Gamma\{\perp^\bullet\}} \perp^\bullet \quad \frac{\Gamma\{A^\bullet, B^\bullet\}}{\Gamma\{A \wedge B^\bullet\}} \wedge^\bullet \quad \frac{\Lambda\{A^\circ\} \quad \Lambda\{B^\circ\}}{\Lambda\{A \wedge B^\circ\}} \wedge^\circ \\
\frac{\Gamma^{\perp^\circ}\{A^\bullet\} \quad \Gamma^{\perp^\circ}\{B^\bullet\}}{\Gamma^{\perp^\circ}\{A \vee_c B^\bullet\}} \vee_c^\bullet \quad \frac{\Gamma^{\perp^\circ}\{A^\vee, B^\vee\}}{\Gamma^{\perp^\circ}\{A \vee_c B^\vee\}} \vee_c^\vee \quad \frac{\Gamma^*\{A \rightarrow_c B^\bullet, A^\circ\} \quad \Gamma^{\perp^\circ}\{B^\bullet\}}{\Gamma^{\perp^\circ}\{A \rightarrow_c B^\bullet\}} \rightarrow_c^\bullet \\
\frac{\Gamma^{\perp^\circ}\{A^\bullet, B^\vee\}}{\Gamma^{\perp^\circ}\{A \rightarrow_c B^\vee\}} \rightarrow_c^\vee \quad \frac{\Delta_1\{\Box A^\bullet, [A^\bullet, \Delta_2]\}}{\Delta_1\{\Box A^\bullet, [\Delta_2]\}} \Box^\bullet \quad \frac{\Lambda\{[A^\circ]\}}{\Lambda\{\Box A^\circ\}} \Box^\circ \\
\frac{\Gamma^{\perp^\circ}\{[A^\bullet]\}}{\Gamma^{\perp^\circ}\{\Diamond_c A^\bullet\}} \Diamond_c^\bullet \quad \frac{\Delta_1^{\perp^\circ}\{\Diamond_c A^\vee, [A^\vee, \Delta_2^{\perp^\circ}]\}}{\Delta_1^{\perp^\circ}\{\Diamond_c A^\vee, [\Delta_2^{\perp^\circ}]\}} \Diamond_c^\vee \quad \frac{\Gamma^*\{P^\vee, P^\circ\}}{\Gamma^{\perp^\circ}\{P^\vee\}} \text{dec} \quad \frac{\Lambda\{N^\vee, \perp^\circ\}}{\Lambda\{N^\circ\}} \text{sto}
\end{array}$$

Fig. 4. Classical fragment nEK_c.

a fully focused system, as well as adding polarized versions of conjunction and disjunction, as done *e.g.* in [LM11, CMS16] is left for a future work.

7 Extensions

Depending on the application, several further modal logics can be defined as extensions of EK by simply restricting the class of frames we consider or, equivalently, by adding axioms over modalities. Many of the restrictions one can be interested in are definable as formulas of first-order logic, where the binary predicate $R(x, y)$ refers to the corresponding accessibility relation. Table 1 summarizes some of the most common logics, the corresponding frame property, together with the modal axiom capturing it [Sah75].

Since the intuitionistic fragment of nEK coincides with NIK, intuitionistic versions for the rules for the axioms **t**, **b**, **4**, and **5** match the rules (\bullet) and (\circ) presented in [Str13], and are depicted in Fig. 5.

For completing the ecumenical view, the classical (\vee) rules for extensions are justified via translation to the labeled system labEK. For example, the labeled

Table 1. Axioms and corresponding first-order conditions on R .

Axiom	Condition	First-Order Formula
d : $\Box A \rightarrow \Diamond A$	Seriality	$\forall x \exists y. R(x, y)$
t : $\Box A \rightarrow A \wedge A \rightarrow \Diamond A$	Reflexivity	$\forall x. R(x, x)$
b : $A \rightarrow \Box \Diamond A \wedge \Diamond \Box A \rightarrow A$	Symmetry	$\forall x, y. R(x, y) \rightarrow R(y, x)$
4 : $\Box A \rightarrow \Box \Box A \wedge \Diamond \Diamond A \rightarrow \Diamond A$	Transitivity	$\forall x, y, z. (R(x, y) \wedge R(y, z)) \rightarrow R(x, z)$
5 : $\Box A \rightarrow \Box \Diamond A \wedge \Diamond \Box A \rightarrow \Diamond A$	Euclideaness	$\forall x, y, z. (R(x, y) \wedge R(x, z)) \rightarrow R(y, z)$

$$\begin{array}{cccc}
\frac{\Gamma\{\Box A^*, A^*\}}{\Gamma\{\Box A^*\}} \mathbf{t}^* & \frac{\Delta_1\{[\Delta_2, \Box A^*], A^*\}}{\Delta_1\{[\Delta_2, \Box A^*]\}} \mathbf{b}^* & \frac{\Delta_1\{[\Delta_2, \Box A^*], \Box A^*\}}{\Delta_1\{[\Delta_2, \Box A^*]\}} \mathbf{4}^* & \frac{\Gamma\{[\Box A^*][\Box A^*]\}}{\Gamma\{[\Box A^*][\emptyset]\}} \mathbf{5}^* \\
\frac{\Lambda\{A^\circ\}}{\Lambda\{\Diamond_i A^\circ\}} \mathbf{t}^\circ & \frac{\Lambda_1\{[\Lambda_2], A^\circ\}}{\Lambda_1\{[\Lambda_2, \Diamond_i A^\circ]\}} \mathbf{b}^\circ & \frac{\Lambda_1\{[\Lambda_2, \Diamond_i A^\circ]\}}{\Lambda_1\{[\Lambda_2], \Diamond_i A^\circ\}} \mathbf{4}^\circ & \frac{\Lambda\{[\emptyset][\Diamond_i A^\circ]\}}{\Lambda\{[\Diamond_i A^\circ][\emptyset]\}} \mathbf{5}^\circ \\
\frac{\Gamma^{\perp^\circ}\{A^\nabla\}}{\Gamma^{\perp^\circ}\{\Diamond_c A^\nabla\}} \mathbf{t}^\nabla & \frac{\Delta_1^{\perp^\circ}\{[\Delta_2^{\perp^\circ}], A^\nabla\}}{\Delta_1^{\perp^\circ}\{[\Delta_2^{\perp^\circ}], \Diamond_c A^\nabla\}} \mathbf{b}^\nabla & \frac{\Delta_1^{\perp^\circ}\{[\Delta_2^{\perp^\circ}], \Diamond_i A^\circ\}}{\Delta_1^{\perp^\circ}\{[\Delta_2^{\perp^\circ}], \Diamond_c A^\nabla\}} \mathbf{4}^\nabla & \frac{\Gamma^{\perp^\circ}\{[\emptyset][\Diamond_c A^\nabla]\}}{\Gamma^{\perp^\circ}\{[\Diamond_c A^\nabla][\emptyset]\}} \mathbf{5}^\nabla
\end{array}$$

Fig. 5. Ecumenical modal extensions for axioms d, t, b, 4 and 5.

derivation on the left justifies the classical rule in the middle.

$$\frac{\frac{xRx, \mathcal{R}, \Sigma, x : \neg A \Rightarrow x : \perp}{xRx, \mathcal{R}, \Sigma, x : \Box \neg A \Rightarrow x : \perp} \Box L}{\frac{\mathcal{R}, \Sigma, x : \Box \neg A \Rightarrow x : \perp}{\mathcal{R}, \Sigma \Rightarrow x : \Diamond_c A} \Diamond_c R} \mathbf{T} \quad \frac{\Gamma^{\perp^\circ}\{A^\nabla\}}{\Gamma^{\perp^\circ}\{\Diamond_c A^\nabla\}} \mathbf{t}^\nabla \quad \frac{xRx, \Gamma \vdash z : C}{\Gamma \vdash z : C} \mathbf{T}$$

The rule \mathbf{T} above right is the labeled rule corresponding to the axiom **t** [Sim94]. The rules \mathbf{b}^∇ , $\mathbf{4}^\nabla$ and $\mathbf{5}^\nabla$, shown in Fig. 5, are obtained in the same manner. By mixing and matching these rules, we conjecture that we obtain ecumenical modal systems for most logics in the S5 modal cube [BRV01], i.e. those that are not defined with axiom **d**.

8 Conclusion

In this paper, we have presented a pure, nested proof system **nEK** for the ecumenical modal logic **EK**, together with pure fragments and extensions. We proved soundness of **nEK** w.r.t. the ecumenical birelational model via a translation to the labeled ecumenical modal system **labEK**. For completeness, we used the fact that **EK** axioms are provable in **nEK** and we proved cut-elimination for **nEK**. Finally, having an ecumenical nested system allowed for extracting well known systems as fragments.

First of all, it should be noted that combining classical and intuitionistic modalities *conservatively* in the same *pure* logical system is not trivial. In fact, the labeled system in [MPPS20] makes an extensive use of negations in order to keep classical information *persistent*. We have shown that this can be avoided by having an additional classical context to store negative formulas, similarly to Girard’s classical system LC [Gir91], henceforth solving the “impurity” issue. On the other hand, there seems to be no trivial solution to remove labels from intuitionistic modal sequent systems where the distributivity of the diamond w.r.t. the disjunction holds [Sim94]. The solution here was to adopt the framework to nested sequents, whose tree structure describes the corresponding path in the birelational semantics, a special case in which labels can be eliminated.

It turns out that this mix of classical context, polarities and nestings can be implosive, in the sense that adding a cut rule may lead to a collapse of the system to classical modal logic. For controlling the implosion, the cut rules must have a restricted use of polarities which, in turn, makes the cut-elimination proof non trivial.

There are many interesting ideas that can be explored for the proposed systems, axioms and semantics, as indicated throughout the text, and many lines to be pursued in this research direction. First of all, we have proposed a proof discipline for nEK, which does not correspond to focusing for modal systems as presented in [CMS16]. In fact, the presence of weakening is known to break focusing, we should therefore investigate alternative ways of having a fully focused system. Moreover, it should be interesting to study *typing* in ecumenical systems, in which fragments of already known modal type systems could be embedded. Finally, we plan to implement ecumenical provers, as well as to automate the cut-elimination proof in the L-Framework [OPR21].

Acknowledgements. This work was partially financed by CNPq, CAPES and by the UK’s EPSRC through research grant EP/S013008/1. We would like to thank the anonymous reviewers for their suggestions and comments.

References

- [Brü09] Brünnler, K.: Deep sequent systems for modal logic. Arch. Math. Log. **48**, 551–577 (2009). <https://doi.org/10.1007/s00153-009-0137-3>
- [Bull92] Bull, R.: Cut elimination for propositional dynamic logic without*. Zeitschr. f. math. Logik und Grundlagen d. Math. **38**, 85–100 (1992)
- [BRV01] Blackburn, P., de Rijke, M., de Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge (2001)
- [CMS16] Chaudhuri, K., Marin, S., Straßburger, L.: Modular focused proof systems for intuitionistic modal logics. In: FSCD 2016, no. 16, pp. 1–18 (2016)
- [DD20] Díaz-Caro, A., Dowek, G.: A new connective in natural deduction, and its application to quantum computing. CoRR, abs/2012.08994 (2020)
- [Dum91] Dummett, M.: The Logical Basis of Metaphysics. Harvard University Press, Cambridge (1991)

- [Fit14] Fitting, M.: Nested sequents for intuitionistic logics. *Notre Dame J. Formal Logic* **55**(1), 41–61 (2014)
- [GR12] Goré, R., Ramanayake, R.: Labelled tree sequents, tree hypersequents and nested (deep) sequents. *Adv. Modal Logic* **9**, 279–299 (2012)
- [Gir91] Girard, J.-Y.: A new constructive logic: classical logic. *Math. Struct. Comput. Sci.* **1**(3), 255–296 (1991)
- [Gir93] Girard, J.-Y.: On the unity of logic. *Ann. Pure Appl. Logic* **59**(3), 201–217 (1993)
- [ILH10] Ilik, D., Lee, G., Herbelin, H.: Kripke models for classical logic. *Ann. Pure Appl. Logic* **161**(11), 1367–1378 (2010)
- [Kas94] Kashima, R.: Cut-free sequent calculi for some tense logics. *Stud. Logica* **53**(1), 119–136 (1994)
- [Lau02] Laurent, O.: Étude de la Polarisation en Logique. Ph.D. thesis, Université Aix-Marseille II (2002)
- [Lel19] Lellmann, B.: Combining monotone and normal modal logic in nested sequents – with countermodels. In: Cerrito, S., Popescu, A. (eds.) *TABLEAUX 2019. LNCS (LNAI)*, vol. 11714, pp. 203–220. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29026-9_12
- [LM11] Liang, C., Miller, D.: A focused approach to combining logics. *Ann. Pure Appl. Logic* **162**(9), 679–697 (2011)
- [MP13] Miller, D., Pimentel, E.: A formal framework for specifying sequent calculus proof systems. *Theor. Comput. Sci.* **474**, 98–116 (2013)
- [MPPS20] Marin, S., Pereira, L.C., Pimentel, E., Sales, E.: Ecumenical modal logic. In: Martins, M.A., Sedlár, I. (eds.) *DaLí 2020. LNCS*, vol. 12569, pp. 187–204. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-65840-3_12
- [OPR21] Olarte, C., Pimentel, E., Rocha, C.: A rewriting logic approach to specification, proof-search, and meta-proofs in sequent systems. *CoRR*, abs/2101.03113 (2021)
- [Pog09] Poggiolesi, F.: The method of tree-hypersequents for modal propositional logic. In: Makinson, D., Malinowski, J., Wansing, H. (eds.) *Towards Mathematical Philosophy. TL*, vol. 28, pp. 31–51. Springer, Dordrecht (2009). https://doi.org/10.1007/978-1-4020-9084-4_3
- [PPdP19] Pimentel, E., Pereira, L.C., de Paiva, V.: An ecumenical notion of entailment. *Synthese* (2019). <https://doi.org/10.1007/s11229-019-02226-5>
- [PR17] Pereira, L.C., Rodriguez, R.O.: Normalization, soundness and completeness for the propositional fragment of Prawitz’ ecumenical system. *Rev. Port. Filos.* **73**(3–3), 1153–1168 (2017)
- [Pra15] Prawitz, D.: Classical versus intuitionistic logic. Why is this a Proof? *Festschrift for Luiz Carlos Pereira*, **27**, 15–32 (2015)
- [PS86] Plotkin, G.D., Stirling, C.P.: A framework for intuitionistic modal logic. In: Halpern, J.Y. (ed.) *1st Conference on Theoretical Aspects of Reasoning About Knowledge. Morgan Kaufmann, Burlington* (1986)
- [Res06] Restall, G.: Comparing Modal Sequent Systems. Draft manuscript (2006)
- [Sah75] Sahlqvist, H.: Completeness and correspondence in first and second order semantics for modal logic. In: Kanger, S. (eds.) *Proceedings of the Third Scandinavian Logic Symposium*, pp. 110–143 (1975)
- [Sim94] Simpson, A.K.: The proof theory and semantics of intuitionistic modal logic. Ph.D. thesis, College of Science and Engineering, School of Informatics, University of Edinburgh (1994)

- [Str13] Straßburger, L.: Cut elimination in nested sequents for intuitionistic modal logics. In: Pfenning, F. (ed.) FoSSaCS 2013. LNCS, vol. 7794, pp. 209–224. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37075-5_14
- [TS96] Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge Univ. Press, Cambridge (1996)



Provability Games for Non-classical Logics

Mezhirov Game for MPC, KD!, and KD

Alexandra Pavlova^(✉) 

Institute of Logic and Computation, TU Wien, Vienna, Austria
alexandra@logic.at

Abstract. Game semantics provides an alternative view on basic logical concepts. Provability games, i.e., games for the validity of a formula provide a link between proof systems and semantics. We present a new type of provability games, namely the Mezhirov game, for minimal propositional logic and two modal systems: the logic of functional frames and the logic of serial frames, i.e., **KD** and prove their adequacy. The games are finite resulting in a finite search for winning strategies.

Keywords: Provability game · Modal logic · Minimal logic

1 Introduction

A game attitude to logic in the broad sense has a long tradition [1, 2]. The underlining idea consists in providing a game-related explanation to various concepts, like truth, validity, bisimulation. Currently, there exists a grand variety of different types of games, e.g., semantic games, provability games, model comparison games, games on consistency, and others [1, 3]. One of the most prominent examples of *provability games*, i.e., those checking validity of the formula, is the *dialogue logic* introduced by Lorenzen [4] and later developed by S. Rahman [5] and Pavlova [21]. This framework seeks to explain and represent the logical concept of validity as a result of a structured argumentative dispute between two participants, or players, *Proponent* (**P**) and *Opponent* (**O**). Disjunctive games [6, 7], represent an alternative approach to provability and build upon *semantic games* (initially proposed by J. Hintikka [8, 9]) aimed at determining the truth of a formula in a given model by step-wise reduction to atomic formulae. In a disjunctive game, a game state is a disjunction of states from semantic games and **P** can duplicate states to be able to backtrack. The idea is to play a game over all possible models, but instead of playing an infinite (for some logics, e.g., Gödel logic [7]) number of semantic games, players engage in one disjunctive game.

We present another type of provability game called *Mazhirov's game* initially proposed by Iliya Mezhirov for intuitionistic logic **IPC** as well as *Grzegorzcyk*

Research supported by FWF project W1255-N23. The author is grateful to the anonymous referees for comments which led to improvements in this paper.

modal logic *Grz* [10,11,20] as a type of game semantics characterizing validity. An important feature of the game is its finiteness and explicit reference to truth values. We extend this approach to several other logics. The main results are new games that not only are finite but also shed light on the relation between the semantic and the syntactic approaches to validity. The paper is structured as follows: the game for minimal propositional logic is presented in Sect. 2, Sect. 3 is dedicated to Modal logic with the game for the logic of functional frames **KD!** being presented in Subsect. 3.1, and the game for the logic of serial frames **KD** discussed in Subsect. 3.2.

2 Game for Minimal Logic

A distinguishing feature of Minimal propositional logic **MPC**¹ compared to intuitionistic logic **IPC**, is the failure of the *principle of explosion*, i.e., $\perp \not\vdash \psi$ where ψ is an arbitrary formula. **IPC** = **IPC**⁺ + *principle of explosion*, **IPC**⁺ being the positive fragment of intuitionistic calculus. One of the metalogical features of **MPC** is that \perp is treated just like an ordinary constant so it does not have the same properties as the intuitionistic \perp ². There are two possible ways to formulate minimal logic [13], namely:

1. adding *falsum* to the positive fragment of intuitionistic logic **IPC**⁺ (respective language \mathcal{L}^+) s.t. $\neg\varphi =_{def} \varphi \rightarrow \perp$. Then $\mathcal{L}_{\mathbf{MPC}} = \mathcal{L}^+ \cup \{\perp\}$ and **MPC** has the same set of axioms as **IPC**⁺. We denote this system as **MPC**_⊥;
2. taking \neg as a primitive connective, i.e., $\mathcal{L}^+ \cup \{\neg\}$. There is a separate axiom for \neg such that **MPC**_¬ = **IPC**⁺ + $((p \rightarrow q) \wedge (p \rightarrow \neg q)) \rightarrow \neg p$ ³.

For both versions of minimal logic, completeness w.r.t. Kripke semantics was proved [13,14]. We shall use **MPC**_⊥ system and denote it simply as **MPC**.

Definition 1. *Let Prop be a countable set of atomic propositions. The language \mathcal{L}_m for minimal propositional logic **MPC** is generated by the following BNF:*

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \perp$$

where $p \in Prop$. The negation is an abbreviation: $\neg\varphi =_{def} \varphi \rightarrow \perp$.

MPC Axioms

1. $\varphi \rightarrow (\psi \rightarrow \varphi)$;
2. $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$;
3. $(\varphi \wedge \psi) \rightarrow \varphi$;

¹ **MPC** is alternatively known as **J** after Ingebrigt Johansson who proposed it in [12].

² To distinguish between the two, sometimes *f* is used as *falsum* constant for minimal logic. We, however, will stick to \perp . Whenever we compare the intuitionistic \perp with the minimal one, we shall use the lower indexes *i* and *m* respectively.

³ This version was originally proposed by Johansson. One can even trace it back to the famous paper by Kolmogorov.

4. $(\varphi \wedge \psi) \rightarrow \psi$;
5. $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$;
6. $\varphi \rightarrow (\varphi \vee \psi)$;
7. $\psi \rightarrow (\varphi \vee \psi)$;
8. $(\varphi \rightarrow \chi) \rightarrow ((\psi \rightarrow \chi) \rightarrow ((\varphi \vee \psi) \rightarrow \chi))$

Kripke Semantics for MPC: We give a standard definition of Kripke semantic for minimal logic [13–15]. $\mathcal{M}_m = \langle \mathcal{W}, R, v, \mathcal{Q} \rangle$ ⁴ which is a generalization of an intuitionistic model \mathcal{M}_i , namely: \mathcal{W} is non-empty set of possible worlds, R is a partial order (i.e., a reflexive, antisymmetric and transitive relation) on \mathcal{W} , and the variable valuation function $v : Prop \times \mathcal{W} \rightarrow \{0, 1\}$. The function v is required to be *monotonic* w.r.t. R : if xRy , then $v(p, x) \leq v(p, y)$ for any $p \in Prop$. Thus, if $v(p, x) = 1$ and xRy , then $v(p, y) = 1$. Let $R(x) = \{y | xRy\}$ be an upward closed set. An upward closed set S of a partially ordered set (X, \leq) is a subset $S \subseteq X$ with the following property: if $s \in S$ and if $x \in X$ is larger than s (i.e., if $s \leq x$), then $x \in S$.

To get a model for minimal logic, \mathcal{M}_i is augmented with the set $\mathcal{Q} \subseteq \mathcal{W}$ s.t. $\mathcal{Q} = \{w \in \mathcal{W} \mid w \Vdash \perp\}$ (sometimes called *abnormal worlds*) and it is an upward closed set. By $\mathfrak{F}_m = \{\mathcal{W}, R\}$ we denote a Kripke frame for **MPC**. We write $\mathcal{M}, x \Vdash \psi$ if formula ψ is true in the world x of \mathcal{M} . This is called the *forcing relation* which is defined as follows:

1. $\mathcal{M}, x \Vdash p$ iff $v(p, x) = 1$;
2. $\mathcal{M}, x \Vdash \varphi \wedge \psi$ iff $\mathcal{M}, x \Vdash \varphi$ and $\mathcal{M}, x \Vdash \psi$;
3. $\mathcal{M}, x \Vdash \varphi \vee \psi$ iff $\mathcal{M}, x \Vdash \varphi$ or $\mathcal{M}, x \Vdash \psi$;
4. $\mathcal{M}, x \Vdash \varphi \rightarrow \psi$ iff for all $y \in R(x)$ either $\mathcal{M}, y \not\Vdash \varphi$ or $\mathcal{M}, y \Vdash \psi$;
5. $\mathcal{M}, x \Vdash \perp$ iff $x \in \mathcal{Q}$.

Forcing relation is also monotonic w.r.t. R , i.e. if xRy and $x \Vdash \varphi$, then $y \Vdash \varphi$ (proof by induction on the construction of φ). **MPC** has *finite model property*, i.e. for every formula not derivable in **MPC** there exists a finite countermodel.

Proposition 1. *A formula is derivable in MPC iff it is true in all finite frames.*

Proof: The proof uses the filtration method. Assume that $\not\vdash_{\mathbf{MPC}} \varphi$. Then there exists a model $\mathcal{M} = \langle \mathcal{W}, R, v, \mathcal{Q} \rangle$ such that $\mathcal{M}, x_0 \not\Vdash \varphi$. Let \mathcal{F} be a set of all subformulae of φ . \mathcal{F} is finite. Since the definition of forcing refers only to the formulae from \mathcal{F} , if two worlds force the same formulae from \mathcal{F} , we can consider them equivalent and join them into one world. This is the idea behind the notion of filtration. Formally, we define an equivalence relation (reflexive, transitive and symmetric) on \mathcal{W} as follows: $x \sim_{\mathcal{F}} y$ iff for all $\psi \in \mathcal{F}$: $x \Vdash \psi \iff y \Vdash \psi$. Now we define a new model $\mathcal{M}_{/\sim_{\mathcal{F}}} = \langle \mathcal{W}_{/\sim_{\mathcal{F}}}, R^*, v^*, \mathcal{Q}^* \rangle$ where $\mathcal{W}_{/\sim_{\mathcal{F}}}$ is the set of equivalence classes of worlds from \mathcal{W} w.r.t. $\sim_{\mathcal{F}}$. The equivalence class of $x \in \mathcal{W}$ is the set $[x]_{\sim_{\mathcal{F}}} = \{y \mid y \sim_{\mathcal{F}} x\}$. Therefore, $x \sim_{\mathcal{F}} z \iff [x]_{\sim_{\mathcal{F}}} \sim_{\mathcal{F}} [z]_{\sim_{\mathcal{F}}}$. Therefore, $\mathcal{Q}^* \in \mathcal{W}^*$ is a set of abnormal equivalence classes. Now we define the accessibility

⁴ We shall generally omit the lower index if it does not lead to confusion.

relation. $[x]_{\sim_{\mathcal{F}}} R^* [y]_{\sim_{\mathcal{F}}}$ iff $x \Vdash \chi$ implies $y \Vdash \chi$ for every $\chi \in \mathcal{F}$. Note that, since in equivalent worlds the same formulae from \mathcal{F} are true, this definition does not depend on what particular elements we take from $[x]_{\sim_{\mathcal{F}}}$. The new relation R^* is reflexive and transitive by definition. R^* is also antisymmetric (Assume that $[x]_{\sim_{\mathcal{F}}} R^* [y]_{\sim_{\mathcal{F}}}$ and $[y]_{\sim_{\mathcal{F}}} R^* [x]_{\sim_{\mathcal{F}}}$. Then, due to monotonicity of forcing, $[x]_{\sim_{\mathcal{F}}}$ and $[y]_{\sim_{\mathcal{F}}}$ force exactly the same formulae. Hence, they represent the same equivalence class). It only remains to define the valuation v^* . $v^*(p, [x]_{\sim_{\mathcal{F}}}) = v(p, x)$ for all $p \in \mathcal{F}$. To preserve monotonicity let us set all the variables that are not in \mathcal{F} to be always false. The filtered model $\mathcal{M}_{/\sim_{\mathcal{F}}}$ is finite since there is a finite number of φ 's subformulae and a finite number of possible valuations for formulae from \mathcal{F} . We need to prove that it preserves forcing for formulae from \mathcal{F} , i.e. if $\psi \in \mathcal{F}$, then:

$$\mathcal{M}, x \Vdash \psi \iff \mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash \psi$$

Proof: By induction on the structure of ψ :

- $\psi \in Prop$. Then by definition of v^* : $\mathcal{M}, x \Vdash p \iff \mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash p$.
- $\psi := \perp$, then $\mathcal{M}, x \Vdash \perp$ iff $x \in \mathcal{Q}$. But then $x \in [x_i]_{/\sim_{\mathcal{F}}}$ such that $[x_i]_{/\sim_{\mathcal{F}}} \in \mathcal{Q}^*$. Then $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x_i]_{/\sim_{\mathcal{F}}} \Vdash \perp$ (since in all the worlds in the class $[x_i]_{/\sim_{\mathcal{F}}}$ all formulae get the same value)
- $\psi := \psi_1 \wedge \psi_2$. $\mathcal{M}, x \Vdash \psi_1 \wedge \psi_2$ iff $\mathcal{M}, x \Vdash \psi_1$ and $\mathcal{M}, x \Vdash \psi_2$. By IH, $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash \psi_1$ and $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash \psi_2$. Thus, $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash \psi_1 \wedge \psi_2$ by def. of forcing. The case for \vee is similar.
- $\psi := \psi_1 \rightarrow \psi_2$. $\mathcal{M}, x \Vdash \psi_1 \rightarrow \psi_2$ iff for all y such that xRy : $\mathcal{M}, y \nVdash \psi_1$ or $\mathcal{M}, y \Vdash \psi_2$. By IH, $\mathcal{M}_{/\sim_{\mathcal{F}}}, [y]_{/\sim_{\mathcal{F}}} \nVdash \psi_1$ or $\mathcal{M}_{/\sim_{\mathcal{F}}}, [y]_{/\sim_{\mathcal{F}}} \Vdash \psi_2$ for all $[y]_{/\sim_{\mathcal{F}}}$ such that $y \in [y]_{/\sim_{\mathcal{F}}}$. Then $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \Vdash \psi_1 \rightarrow \psi_2$. \square

By applying it to φ , we get that $\mathcal{M}_{/\sim_{\mathcal{F}}}, [x]_{/\sim_{\mathcal{F}}} \nVdash \varphi$, which is our goal. \square

Mezhirov Game for MPC. The game $\mathcal{G}(\varphi)$ played over the formula φ contains the following elements:

- φ is the initial formula of the game.
- \mathcal{F} is the set of all subformulae of the initial formula φ .
- the set of players $\mathcal{A} = \{\mathbf{O}, \mathbf{P}\}$ where \mathbf{P} is *Proponent* and \mathbf{O} is *Opponent*;
- players' corresponding sets of moves will be denoted by \mathcal{P} and \mathcal{O} .
- a position, or a game state, C is a pair $(\mathcal{P}, \mathcal{O})$, where \mathcal{O} and \mathcal{P} are sets of subformulae of φ . The number of possible positions is finite, since there number of formulae is finite. The starting position is $C_0 = (\varphi, \emptyset)$. By \mathcal{C} we denote the set of all game states.
- game valuation function $v : \mathcal{F} \times \mathcal{C} \longrightarrow \{0, 1\}$ defined recursively. Game valuation of each subformula of φ is calculated for every position in the game, hence we denote valuations in particular games as v_i read as the valuation of ψ at some game state i . \mathcal{V} is the set of all possible valuations. For all game position C and all $\psi_i, \psi_j \in \mathcal{F}$, $p \in Prop$, $\star \in \{\wedge, \vee, \rightarrow\}$:

$$v(\perp, C) = 1 \text{ iff } \perp \in \mathcal{O} \tag{1}$$

$$v(p, C) = 1 \text{ iff } p \in \mathcal{O} \tag{2}$$

$$v(\psi_i \star \psi_j, C) = 1 \iff \begin{cases} (\psi_i \star \psi_j) \in (\mathcal{O} \cup \mathcal{P}) \\ v(\psi_i) \star_B v(\psi_j) = 1 \end{cases} \tag{3}$$

where \star_B is the Boolean function associated with \star . The curly bracket should be read as a conjunction.

The last condition can be read as follows: a formula that is not in $\mathcal{O} \cup \mathcal{P}$ (also called a non-marked) is always false, and a marked formula behaves according to its classical truth table.

The game proceeds as follows:

- * Players *move* by adding a formula to their respective sets (\mathcal{P} and \mathcal{O}) which is called *marking* a formula.
- * If a player has marked a formula and its game valuation is 0, let us say that this formula is his **mistake**. If **O** has no mistakes but **P** has, then it's **P**'s turn to move. Otherwise, it's **O**'s turn.
- * The game terminates when a player whose turn it is cannot move. A player *loses* when he cannot move.

Mezhirov game for intuitionistic logic can be seen as a special restricted case of the game for minimal logic. Hence, to get Mezhirov game for Minimal Logic, we need to allow players to mark \perp and assign it the value 1. Since \perp is a constant, it gets the value 1 iff it is marked by **O**.

Each game can be viewed as a finite tree of game positions, rooted in the initial formula φ and branching according to the rules of Mezhirov game until all leaves reach a terminal state. Its branches are the possible runs of the game. A *strategy* σ for **P** is a subtree of the game tree obtained from pruning all but one **P**-labelled outgoing branches (i.e., where its **P**'s turn to move) from every node in the tree, while keeping **O**-labelled branches intact. The remaining tree specifies **P**'s moves at the given state, while all possible choices of **O** are still recorded. σ is a *winning strategy* (w.s.), for **P** if all leaf nodes of σ are winning states for **P**.

Example 1. Consider the following example of a run of the game for the formula $p \rightarrow (\neg p \rightarrow q)$. Actions that do not influence the final valuation (so **P** cannot correct his mistakes) are in red.

\mathcal{F}	\mathcal{O}	\mathcal{P}	\mathcal{V}_0	\mathcal{V}_1	\mathcal{V}_2	\mathcal{V}_3	\mathcal{V}_4	\mathcal{V}_5	\mathcal{V}_6	\mathcal{V}_7	\mathcal{V}_8
$p \rightarrow (\neg p \rightarrow q)$		0.X	1	0	1	1	0	0	0	0	0
$\neg p \rightarrow q$		2.X	0	0	1	1	0	0	0	0	0
$\neg p := p \rightarrow \perp$	3.X	8.X	0	0	0	0	1	1	1	1	1
p	1.X	6.X	0	1	1	1	1	1	1	1	1
q		5.X	0	0	0	0	0	0	0	0	0
\perp	4.X	7.X	0	0	0	0	1	1	1	1	1

It is quite easy to see that not only **P** loses this particular run of the game, but in fact, **P** does not have a winning strategy for a game over the formula $p \rightarrow (\neg p \rightarrow q)$. Whereas **P** still has a winning strategy in a game over minimally valid formula $p \rightarrow (\neg p \rightarrow \neg q)$.

Theorem 1. $\Vdash_{\mathbf{MPC}} \varphi$ iff **P** has a winning strategy in the game for minimal logic over φ .

Proof: Since minimal logic uses the same frame structures as intuitionistic logic (intuitionistic models are just models for minimal logic where $\mathcal{Q} = \emptyset$) and it has finite model property, we can refer to the proof in [16] for the intuitionistic case and use it for both directions.

\implies -direction: Given that $\not\Vdash_{\mathbf{MPC}} \varphi$, there exists at least one finite counter-model \mathcal{M} . Let **O** play according to that model as described in the proof in [16]. The only thing that we should modify in the proof is to show that $\mathcal{I}_w(\perp) = v(\perp)$ when **O** has marked all the formulae true in the world w (that **O** has in mind) since for all other formulae the proof is the same: $\mathcal{I}_w(\perp) = 1$ iff **O** has marked \perp by the construction of **O**'s strategy. By definition of v , $v(\perp) = 1$ iff $\perp \in \mathcal{O}$. The rest of the proof is the same as for the intuitionistic case.

\impliedby -direction: Again, take the proof in [16] for the intuitionistic case. We construct the model the same way. Since $\mathcal{Q} \in \mathcal{W}$, the proof that our construction is a model is analogous to the one in [16]. We need to update the proof by induction that $\mathcal{I}_{x_0}(\theta) = v(\theta)$ by adding the case of \perp . Since \perp is treated as a special variable, i.e. $\mathcal{I}_{x_0}(\theta) = 1$ if and only if θ is in \mathcal{O} by the construction, thus, $v(\theta)$ by definition of game valuation. This works in both directions. \square

3 Game for Modal Logic

In this section, we consider two variations of the game for classical modal logic. The frames are no longer required to be partial orders. Before we proceed with game specification, let us first define the notion of *modal depth* of a formula, denoted as $MD(\varphi)$ and of the *depth of a frame* (sometimes we refer to it as *semantic depth*).

Definition 2. Let $Prop$ be a countable set of atomic propositions and $p \in Prop$. The language \mathcal{L}_{Mc} for classical modal logic is generated by the following BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box\varphi$$

$$\Diamond\varphi := \neg\Box\neg\varphi.$$

Definition 3. Let \mathfrak{F} be a frame. We call the *depth* of \mathfrak{F} , indicated by $d(\mathfrak{F})$, the maximal number n such that no directed path⁵ in \mathfrak{F} is longer than n . $d(\mathcal{M})$ is the *semantic depth* of a model \mathcal{M} based on a frame \mathfrak{F} such that $d(\mathcal{M}) = d(\mathfrak{F})$.

⁵ A directed path in a directed graph is a finite or infinite sequence of edges which joins a sequence of distinct vertices with the additional restriction that the edges should be all directed in the same direction. The length of a path is the number of edges it contains.

Definition 4. *Modal depth of a formula, $MD(\varphi)$ is defined recursively as follows:*

1. $MD(p) = 0$ where $p \in Prop$;
2. $MD(\neg\varphi) = MD(\varphi)$;
3. $MD(\varphi_1 \star \varphi_2) = \max(MD(\varphi_1), MD(\varphi_2))$ where $\star \in \{\wedge, \vee, \rightarrow\}$;
4. $MD(\Box\varphi) = MD(\varphi) + 1$.

We denote by $\Box^n\varphi$ the formula $\underbrace{\Box \dots \Box}_{n \text{ times}}\varphi$.

Definition 5. *A pointed frame $\mathfrak{F} = \langle \mathcal{W}, R \rangle$ with distinguished state $w \in \mathcal{W}$ is a tree with root w if \mathfrak{F} is rooted at w and every state $u \in \mathcal{W}$ is reachable from w by a unique path. Accordingly, every Kripke structure over (\mathfrak{F}, w) is a tree structure.*

Claim 1 (Lemma 36 in [17]). Let $n \in \mathbb{N}$. Every finite pointed Kripke structure (\mathcal{M}, y) is bisimilar to a finite pointed structure (\mathcal{M}_n, x_0) whose restriction to depth n from the distinguished node x_0 is a tree structure.

Proof. Cf. proof in [17].

3.1 Mezhirov Game for Modal Logic of Functional Frames

We introduce Mezhirov game for the logic *functional* frames, known as **KD!**, **KDD_c**, or **DAIt**. For the model-theoretic account of **KD!** see [18]; proof-theory is studied in [19]. Logic **KD!** can be axiomatised in several ways:

- **KD!** = **K** + $\Diamond\varphi \leftrightarrow \Box\varphi$ where **D**-axiom $\Box\varphi \rightarrow \Diamond\varphi$ corresponds to *seriality* $(\forall x\exists y : xRy)$, and axiom **D_c**: $\Diamond\varphi \rightarrow \Box\varphi$ corresponds to *partial functionality* $(\forall x, y, z : (wRy \ \& \ wRz) \implies y = z)$.
- It can be alternatively axiomatised as **KD!** = **K** + $\neg\Box\varphi \leftrightarrow \Box\neg\varphi$.

KD! Semantics: The corresponding frame condition is functionality, i.e., $\forall w\exists!u : wRu$ meaning that every world has a unique successor. From this follows the linearity of the frame. Functional frames can either end in a loop (of arbitrary size) or be an infinite chain. Let $\mathcal{M}_{\mathbf{KD}!} = \langle \mathcal{W}, R, v \rangle$ where R is functional. Then:

1. $\mathcal{M}, x \models p$ where $p \in Prop$ iff $v(p, x) = 1$;
2. $\mathcal{M}, x \models \neg\varphi$ iff $\mathcal{M}, x \not\models \varphi$;
3. $\mathcal{M}, x \models \varphi \wedge \psi$ iff $\mathcal{M}, x \models \varphi$ and $\mathcal{M}, x \models \psi$;
4. $\mathcal{M}, x \models \varphi \vee \psi$ iff $\mathcal{M}, x \models \varphi$ or $\mathcal{M}, x \models \psi$;
5. $\mathcal{M}, x \models \varphi \rightarrow \psi$ iff $\mathcal{M}, x \not\models \varphi$ or $\mathcal{M}, x \models \psi$;
6. $\mathcal{M}, x \models \Box\varphi$ iff for all $y \in \mathcal{W}$ s.t. xRy : $\mathcal{M}, y \models \varphi$.

Mezhirov Game for KD! is based on the game for **Grz** logic presented by I. Mezhirov in [10]. Let φ be the initial formula over which the game is played. Since the game is played to determine the validity (hence φ should be valid in

every world), we can add without loss of generality \Box as the main operator of the initial formula. Let \mathcal{F} be defined as in Sect. 2. From now on, ‘a formula’ will mean a formula from \mathcal{F}^* , and ‘a variable’ will mean a variable occurring in φ . So there is a finite number of formulae and a finite number of variables. We recursively define the set \mathcal{F}^* as follows:

1. $\varphi^0 := (\Box(\varphi))^0$;
2. if $(\neg\psi)^i \in \mathcal{F}^*$, then $\psi^i \in \mathcal{F}^*$;
3. if $(\psi \star \chi)^i \in \mathcal{F}^*$, then $\psi^i \in \mathcal{F}^*$ and $\chi^i \in \mathcal{F}^*$ where $\star \in \{\wedge, \vee, \rightarrow\}$;
4. if $(\Box\psi)^i \in \mathcal{F}^*$, then $\psi^{i+1} \in \mathcal{F}^*$.

This means that we start with $\varphi^0 := (\Box(\varphi))^0$. Then for each boxed formula⁶ with the upper index n a copy of its *immediate subformulae* with $n + 1$ as its upper index is added.

Let \mathcal{F}_\Box^* be the boxed formulae of \mathcal{F}^* (namely: $\mathcal{F}_\Box^* \subset \mathcal{F}^*$ such that $\psi^i \in \mathcal{F}_\Box^*$ iff $\psi^i \in \mathcal{F}^*$ and $\psi^i := (\Box\chi)^i$). Both \mathcal{F}_\Box^* and \mathcal{F}^* play their role in the game. The game is played over the set \mathcal{F}_\Box^* and propositional variables $p^i \in \mathcal{F}^*$.

A *game state*, or position, is a tuple $C = (\mathcal{P}, \mathcal{O}, V^{Pr})$ where \mathcal{P} is a set of formulae marked by **P**, $\mathcal{P} \subseteq \mathcal{F}_\Box^*$, \mathcal{O} is a set of formulae marked by **O**, $\mathcal{O} \subseteq \mathcal{F}_\Box^*$, V^{Pr} is a set of indexed atomic formulae.

– **P** moves by:

- marking formulae from \mathcal{F}_\Box^* ;
- **P** can unmark a previously marked boxed formula that is an indexed proper subformula of φ (deleting a formula from \mathcal{P}), but not the initial formulae φ . No position can be visited twice during the game⁷.

– **O** moves by:

- marking formulae from \mathcal{F}_\Box^* ;
- adding a propositional variable $p^i \in \mathcal{F}^*$ to the set V^{Pr} . (The initial valuation of all indexed propositional variables is 0)

Remark 1. An alternative restriction on unmarking that “for every $(\Box\psi)^i$ **P** can unmark it at most once” requires a different proof of completeness.

Formulae from \mathcal{F}^* are used to calculate game valuations for each state. As previously, the game starts by **P** marking the initial formula φ^0 ($C_0 = (\varphi^0, \emptyset, \emptyset)$) and ends when a player whose turn it is cannot move. This player loses.

Game valuation v depends on the position of the game we are currently in. Let V be the set of game valuations as in Sect. 2. V is not a part of a game state. For all game position C and all $\psi^i, \chi^i \in \mathcal{F}^*$, $p^j \in Prop$, $\star \in \{\wedge, \vee, \rightarrow\}$:

⁶ By boxed formula we understand a formula the main operator of which is \Box , i.e., formula of the form $\Box\psi$.

⁷ To be more formal and precise, we can formulate this as follows: “for every game position C_k and every $(\Box\psi)^i \in \mathcal{P}_k$ s.t. $(\Box\psi)^i \neq \varphi^0$: **P** can unmark $(\Box\psi)^i$ at the game step $k + 1$ iff $C_{k+1} \neq C_l$ where $l \leq k$ ”.

$$v(\perp^j) = 0 \tag{4}$$

$$v(p^j) = 1 \text{ iff } p^j \in V^{pr} \tag{5}$$

$$v((\neg\psi)^i) = 1 \iff v(\psi^i) = 0 \tag{6}$$

$$v((\psi \star \chi)^i) = 1 \iff v(\psi^i) \star_B v(\chi^i) = 1 \tag{7}$$

where \star_B is the Boolean function associated with \star .

$$v((\Box\psi)^i) = 1 \iff \begin{cases} (\Box\psi)^i \in (\mathcal{O} \cup \mathcal{P}) \\ v(\psi^{i+1}) = 1 \end{cases} \tag{8}$$

If **P** has a mistake and **O** has none, then it is **P**'s turn to move. Otherwise, moves **O**. *Mistakes* and winning conditions are defined as in Sect. 2.

Example 2. Let's have a look at an example of a run of the game.

\mathcal{F}	\mathcal{O}	\mathcal{P}	\mathcal{V}_0	\mathcal{V}_1	\mathcal{V}_2	\mathcal{V}_3	\mathcal{V}_4
$(\Box(\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)))^0$		0.X	1	1	1	0	1
$(\Box(p \rightarrow q))^1$	1.X		0	1	1	1	1
$(\Box p)^1$	2.X		0	0	0	1	1
$(\Box q)^1$		4.X	0	0	0	0	1
Variable Valuations	3. $V^{pr} = \{p^2, q^2\}$						

One can see that in this run of the game **O** loses (even if we play the run up to the end when **O** marks all the formulae). **O** cannot change the valuation because he will get a mistake then. It is easy to see, that **P** has a winning strategy in the game over the *K*-axiom against any move by **O**.

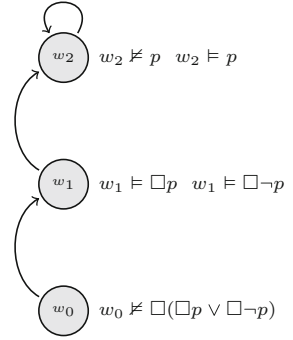
Example 3.

\mathcal{F}	\mathcal{O}	\mathcal{P}	\mathcal{V}_0	\mathcal{V}_1	\mathcal{V}_2
$(\Box(\Box p \rightarrow p))^0$		0.X	1	1	0
$(\Box p)^1$	1.X		0	0	1
Variable Valuations	2. $V^{pr} = \{p^2\}$				

For the initial formula to be true after the game state C_2 , p^1 should be in V^{pr} , but **P** cannot enforce it, so he loses. It is easy to see that **O** not only wins the above run of the game but also has a winning strategy.

Example 4. This example shows why we need **P** to have a possibility to unmark a formula once. By $del(m.X)$ we mean that **P** unmarks the formula marked at step m . The run of the game goes as follows: **P** marks the initial formula, but since the main connective of $(\Box p \vee \Box \neg p)^1$ is a disjunction, it evaluates to 0, hence **P** marks $(\Box \neg p)^1$. This forces **O** to move by marking $(\Box p)^1$ and adding p^2 to V^{pr} hence making $v_2(p^2) = 1$. then **P** unmarks formula $(\Box \neg p)^1$ (denoted as $4.del(1.X)$). The red colour emphasises the unmarked formula.

\mathcal{F}	\mathcal{O}	\mathcal{P}	\mathcal{V}_0	\mathcal{V}_1	\mathcal{V}_2	\mathcal{V}_3	\mathcal{V}_4
$(\Box(\Box p \vee \Box \neg p))^0$		0.X	0	1	0	1	1
$(\Box p)^1$		3.X	0	0	0	1	1
$(\Box \neg p)^1$		4.del(1.X)	0	1	0	0	0
Variable Valuations	2.V ^{pr} = {p ² }						



Remark 2. Assume we do not allow \mathbf{P} to unmark formulae, then if φ is a theorem of \mathbf{P} and φ is \Box -free, then \mathbf{P} has a w.s. for $\Box^n \varphi$ where $n \geq 0$. But \mathbf{P} does not have a w.s. for $\Box(\Box p \vee \Box \neg p)$.

Adequacy: We are proving the adequacy of the Mezhirov game defined in this section with respect to Kripke semantics for the logic $\mathbf{KD!}$. Since $\mathbf{KD!}$ has *finite model property* [18], we can follow the previous semantic proof strategy here.

Theorem 2. \mathbf{P} has a winning strategy in Mezhirov game for $\mathbf{KD!}$ over some formula φ iff $\vDash_{\mathbf{KD!}} \varphi$.

Lemma 1. If $\not\vDash_{\mathbf{KD!}} \varphi$, then \mathbf{O} has a winning strategy in Mezhirov game for $\mathbf{KD!}$ over φ .

Proof. Assume that $\not\vDash_{\mathbf{KD!}} \varphi$ and $MD(\varphi) = n - 1$, then there exists a finite countermodel $\mathcal{M} = \langle \mathcal{W}, R, \vDash \rangle$ and a world x_0 s.t. $\mathcal{M}, x_0 \not\vDash \varphi$. From Claim 1 follows that we can use a tree-structured countermodel of depth n . So let $\mathcal{W} = \{x_0, \dots, x_{n-1}\}$ and R be a functional relation such that $x_0 R x_1 R \dots x_{n-2} R x_{n-1}$ and $x_{n-1} R x_{n-1}$.

\mathbf{O} should build up his strategy following that countermodel, starting with the world x_0 falsifying the initial formula, i.e., $\mathcal{M}, x_0 \not\vDash_{\mathbf{KD!}} \varphi$. We need to relate the worlds in the model to the upper indexes in the game so that \mathbf{O} knows how to form his strategy. Let $ind(y)$ be the game index related to the world y . Note that function $ind(y)$ is injective:

- $ind(x_0) = 0$;
- for $x_i R x_j$ such that $x_i \neq x_j$ & $\neg x_j R x_i$ ⁸ and $ind(x_i) = i$: $ind(x_j) = i + 1$.

Since our model does not contain symmetric relations, the definition works fine. \mathbf{O} should play as follows:

1. \mathbf{O} marks $(\Box \psi)^i$ iff $\mathcal{M}, x_i \vDash \Box \psi$.
2. add an atomic formula p^j to the set V^{pr} iff $\mathcal{M}, x_j \vDash p$.

⁸ This condition indicates that x_i is not the final world and if x_j is the final world the accessibility relation between x_i and x_j is not symmetric.

Note that according to these rules, **O** can only add propositional variables to the set V^{Pr} , but he cannot delete them. Why is it enough? We need to make sure that this restriction does not spoil **O**'s strategies and does not give additional advantage to **P** in games over formulae that are not valid in functional frames. We need to show that **O** can follow the strategy described above in a game over a non-valid formula φ . This is due to the functionality of the frame⁹:

1. The game starts with $V^{Pr} = \emptyset$, hence for all $p^i \in \mathcal{F}^*$: $v(p^i) = 0$. Assume **O** starts building the strategy with the root x_0 . According to the construction, **O** adds the variable p^0 to V^{Pr} iff $\mathcal{M}, x_0 \models p$. Since there is only one world associated to the variables with index 0, **O** does not need to revise the subset $Y^0 \subseteq \mathcal{F}^*$ s.t. $Y^0 = \{p^0 \in Prop\}$.
2. Assume that we have proven this fact for the propositional variables with index i . Since there is always at most one world accessible from x_i , the same argument as for x_0 applies. Since our model has exactly n worlds and $MD(\varphi) = n - 1$ no issues with the world x_{n-1} arise.

Claim 2. If **O** has marked all the formulae according to the instruction above, then $\mathcal{M}, x_i \models \psi$ iff $v(\psi^i) = 1$ (for any $\psi^i \in \mathcal{F}^*$).

Proof. Proof by induction on the structure of ψ .

1. $\psi^i \in Prop$. $\mathcal{M}, x_i \models \psi$ iff **O** has added the formula ψ^i to the set V^{Pr} (by strategy construction), and thus by the def. of game valuation $v(\psi^i) = 1$.
2. $\psi^i := (\psi_1 \wedge \psi_2)^i$. $\mathcal{M}, x_i \models \psi_1 \wedge \psi_2$ iff $[\mathcal{M}, x_i \models \psi_1 \text{ and } \mathcal{M}, x_i \models \psi_2]$ iff (by IH) $[v(\psi_1^i) = 1 \text{ and } v(\psi_2^i) = 1]$ iff by def. of game valuation $v(\psi^i) = 1$. The cases of disjunction and implication are analogous.
3. $\psi^i := \neg\psi_1^i$. $\mathcal{M}, x_i \models \neg\psi_1$, iff $\mathcal{M}, x_i \not\models \psi_1$ iff (by IH) $v(\psi_1^i) = 0$ iff (by def. of game valuation) $v(\neg\psi_1^i) = 1$.
4. $\psi^i := (\Box\psi_1)^i$.
 \implies : If $\mathcal{M}, x_i \models \Box\psi_1$, then in world x_{i+1} s.t. $x_i R x_{i+1}$: $\mathcal{M}, x_{i+1} \models \psi_1$. By IH, $v(\psi_1^{i+1}) = 1$. By def. of game valuation $v((\Box\psi_1)^i) = 1$ since **O** has marked $(\Box\psi_1)^i$.
 \longleftarrow : $\psi^i := (\Box\psi_1)^i$. If $v((\Box\psi_1)^i) = 1$, then $v(\psi_1^{i+1}) = 1$. By IH, $\mathcal{M}, x_{i+1} \models \psi_1$, then $\mathcal{M}, x_i \models \Box\psi_1$ for the unique x_{i+1} s.t. $x_i R x_{i+1}$.

If **O** has marked all formulae according to his strategy, then he has no mistakes, i.e. for all $\psi \in \mathcal{O}$: $v(\psi) = 1$, by Claim 2. **P**, on the other hand, has at least one mistake, namely the initial formula φ . By the definition of the game, from this point on, it cannot be **O**'s turn again.

Lemma 2. If **O** has a winning strategy in Mezhirrov game for **KD!** over φ , then $\not\models_{\mathbf{KD!}} \varphi$.

⁹ The same argument is applicable to show that **O** does not need to be able to unmark formulae.

Proof. Assume that \mathbf{O} has a winning strategy ζ . Given such a winning strategy for \mathbf{O} , let us build a model $\mathcal{M} = \langle \mathcal{W}, R, v \rangle$. We later prove that it is a countermodel of φ , i.e., $w_0 \not\models_{\mathbf{KD}!} \varphi$. Let φ be of modal depth $MD(\varphi) = n - 1$. Hence, in our game, we have n indexes, namely $\langle 0, \dots, n - 1 \rangle$. For every index i we create a world w_i . Then $\mathcal{W} = \{w_0, \dots, w_{n-1}\}$ will be the set of all worlds in the constructed model \mathcal{M} . Let us define the accessibility relation as follows: $w_i R w_{i+1}$ iff $i \neq n - 1$ and $w_{n-1} R w_{n-1}$ where $n - 1$ is the maximal world (i.e., it has the largest index number). It is easy to see that R is a functional relation and that the model has n worlds. Finally, we need to add valuation. Since \mathbf{O} has a winning strategy, it can have branching (for instance, when it depends on \mathbf{P} 's action). Since the strategy is winning, \mathbf{P} has at least one mistake at each final game state of the strategy. However, we cannot just take any arbitrary final state.

Definition 6. For all game positions (states) x where it is \mathbf{P} 's turn to move (\mathbf{P} has m mistakes), we say that \mathbf{P} plays rationally if either $v_x(\varphi^0) = 0$ or in the following m moves \mathbf{P} unmarks all his mistakes.

Remark 3. \mathbf{P} can always keep playing rationally (i.e., at any game state x either $v_x((\varphi)^0) = 0$ or \mathbf{P} is allowed to unmark all his m number of mistakes in the next m rounds) if \mathbf{P} was always playing rationally earlier in the game run.

Proof. The case when \mathbf{P} has the initial formula as a mistake is trivial. We prove the remark by induction on the number of \mathbf{P} 's mistakes, i.e., number m of formulae $(\Box\chi)^i \in \mathcal{P}_x$ s.t. $v_x((\Box\chi)^i) = 0$. Assume that $v_x(\varphi^0) = 1$.

1. $m = 1$. Then there exists exactly one formula $(\Box\chi)^i$ such that $(\Box\chi)^i \in \mathcal{P}_x$, $v_x((\Box\chi)^i) = 0$ and $(\Box\chi)^i \neq \varphi^0$. Assume that \mathbf{P} cannot unmark $(\Box\chi)^i$. Then there exists a state (position) C_y such that $C_y = (\mathcal{P}_x \setminus \{(\Box\chi)^i\}, \mathcal{O}_x, V_x^{pr})$. Note that since $v_x((\Box\chi)^i) = v_y((\Box\chi)^i) = 0$ and marking and set of propositional valuations of all other formulae coincide in states y and x , the sets of game valuations at game positions y and x are identical, i.e., for all formulae $(\xi)^j \in \mathcal{F}^*$: $v_y((\xi)^j) = v_x((\xi)^j)$. Hence, \mathbf{P} does not have any mistakes in the game position y , since its only mistake in the state x was $(\Box\chi)^i$, hence it is \mathbf{O} 's turn to move. \mathbf{O} can only either mark a formula, but then $\mathcal{O}_{y+1} \neq \mathcal{O}_x$ which, given that \mathbf{O} cannot unmark formulae (monotonicity of \mathcal{O}), contradicts the fact that $\mathcal{O}_y = \mathcal{O}_x$; or add a propositional variable to V^{pr} , but it contradicts the assumption that $V_x^{pr} = V_y^{pr}$ given that \mathbf{O} cannot delete formulae from V^{pr} (monotonicity of V^{pr}). Hence, our assumption that \mathbf{P} cannot unmark $(\Box\chi)^i$ leads to a contradiction.
2. $m = k + 1$ (induction step). Assume we have proven the remark for the case with k mistakes. We prove that it also hold for the case $k + 1$. Assume that it is not the case that \mathbf{P} can unmark all its mistakes at position x . Then there exists at least one formula $(\Box\chi)^i$ such that $(\Box\chi)^i \in \mathcal{P}_x$, $v_x((\Box\chi)^i) = 0$, and $(\Box\chi)^i$ cannot be unmarked. then there exists a game position C_y such that $C_y = (\mathcal{P}_x \setminus \{(\Box\chi)^i\}, \mathcal{O}_x, V_x^{pr})$. Note that since $v_x((\Box\chi)^i) = v_y((\Box\chi)^i) = 0$ and marking and set of propositional valuations of all other formulae coincide in

states y and x , the sets of game valuations at game positions y and x are identical, i.e., for all formulae $(\xi)^j \in \mathcal{F}^*$: $v_y((\xi)^j) = v_x((\xi)^j)$. Hence, at game position y , \mathbf{P} has k mistakes. By assumption $v_x(\varphi^0) = 1$, thus $v_y(\varphi^0) = 1$. By the induction hypothesis, \mathbf{P} can unmark all k mistakes. Let \mathbf{P} play rationally, i.e., in k moves, \mathbf{P} unmarks all mistakes¹⁰. Then at game position C_{y+k} , \mathbf{P} does not have any mistakes and $\mathcal{O}_x = \mathcal{O}_y = \mathcal{O}_{y+k}$ and $V_x^{pr} = V_y^{pr} = V_{y+k}^{pr}$ and it is \mathbf{O} 's turn to move. Then either $\mathcal{O}_x \subset \mathcal{O}_{y+k+1}$, which contradicts monotonicity of \mathcal{O} , or $V_x^{pr} \subset V_{y+k+1}^{pr}$, which contradicts monotonicity of V^{pr} . Hence, our assumption that $v_x(\varphi^0) = 1$ was wrong.

Since we have proven that \mathbf{P} can enforce rationality of his play, we can consider without loss of generality only those final states that result from \mathbf{P} playing rationally. Let's call it C_f and define the valuation as follows: for all $p \in Prop$: $v(p, w_i) = v_f(p^i)$ at the state C_f . It is easy to see that the valuation in the worlds is the same as the game valuation at the state C_f (since \mathbf{P} has marked all possible formulae because \mathbf{P} cannot avoid mistakes).

Claim 3. For any world w_i from \mathcal{M} and any formula $\psi^i \in \mathcal{F}^*$: $\mathcal{M}, w_i \models \psi$ iff $v_f(\psi^i) = 1$ where v_f is the valuation at some final state C_f of the strategy (i.e., \mathbf{O} has marked all the formulae according to his strategy).

Proof. Proof by induction on the structure of the formula ψ .

1. $\psi \in Prop$. $\mathcal{M}, w_i \models \psi$ iff by def. of countermodel valuation, $v: v_f(\psi^i) = 1$.
2. $\psi := \psi_1 \wedge \psi_2$. $\mathcal{M}, w_i \models \psi_1 \wedge \psi_2$ iff $[\mathcal{M}, w_i \models \psi_1$ and $\mathcal{M}, w_i \models \psi_2]$ iff (by IH) $[v_f(\psi_1^i) = 1$ and $v_f(\psi_2^i) = 1]$ iff (by def.) $v_f((\psi_1 \wedge \psi_2)^i)$. The cases for disjunction and implication are analogous.
3. $\psi := \neg\psi_1$. $\mathcal{M}, w_i \models \neg\psi_1$ iff $\mathcal{M}, w_i \not\models \psi_1$ iff (by IH) $v_f(\psi_1^i) = 0$ iff (by def.) $v_f(\neg\psi_1^i) = 1$.
4. $\psi := \Box\psi_1$.
 \implies : if $\mathcal{M}, w_i \models \Box\psi_1$, then $\mathcal{M}, w_{i+1} \models \psi_1$ ($w_i R w_{i+1}$ since $(\Box\psi_1)^{n-1} \notin \mathcal{F}^*$).
 By IH, $v_f(\psi_1^{i+1}) = 1$, then by def. of game valuation $v_f((\Box\psi_1)^i) = 1$.
 \impliedby : if $v_f((\Box\psi_1)^i) = 1$, then $v_f(\psi_1^{i+1}) = 1$ (by game construction there cannot be a boxed formula with the maximum index). By IH, $\mathcal{M}, w_{i+1} \models \psi_1$, then $\mathcal{M}, w_i \models \Box\psi_1$ (since w_{i+1} is the only world accessible from w_i and by countermodel construction there exists w_i).

Claim 4. If \mathbf{O} has a winning strategy $\zeta(\varphi)$ in Mezhirrov game for **KD!** over formula φ , then in every final state of the game where \mathbf{P} always plays rationally f of the *w.s.* $\zeta(\varphi)$: $v_f(\varphi^0) = 0$.

Proof. Let $\varphi := \Box\psi$ and f an arbitrary final state of \mathbf{O} 's winning strategy $\zeta(\varphi)$ in game runs where \mathbf{P} plays rationally. Consider the following two cases:

¹⁰ Note that we are proving that \mathbf{P} can always keep playing rationally meaning that if \mathbf{P} was not playing rationally, then \mathbf{P} could have lost the opportunity to do so.

1. ψ is a propositional formula, i.e., does not contain any \Box -formulae (\Box -free). Since \mathbf{O} has a w.s., \mathbf{P} has at least one mistake. Since there is only one \Box -formula that can be marked, namely $\varphi^0 := (\Box\psi)^0$, $v_f((\Box\psi)^0) = 0$. Since f is arbitrary, it holds for any final state of \mathbf{O} 's strategy $\zeta(\varphi)$.
2. ψ contains \Box -formulae as its subformulae, namely formulae of the form $(\Box\xi)$. Since f is one of the final positions of \mathbf{O} 's strategy ζ , then \mathbf{P} cannot unmark any of his mistakes. Since \mathbf{P} plays rationally by assumption, $v_f(\varphi^0) = 0$ from Remark 4.

From Claim 3 and Claim 4 it follows that $\mathcal{M}, w_0 \not\models \varphi$, i.e., $\not\models_{\mathbf{KD}!} \varphi$.

There are alternative ways to prove this direction of the adequacy theorem, for instance, to show that if $\models_{\mathbf{KD}!} \varphi$ then \mathbf{P} has a winning strategy by setting \mathbf{P} 's strategy to mark only those formulae that are logical consequences of \mathcal{O} .

Adjusting the game to capture validity in frames that are linear, serial, and reflexive is rather straightforward by making the game valuation function to be:

$$v((\Box\psi)^i) = 1 \iff \begin{cases} (\Box\psi)^i \in (\mathcal{O} \cup \mathcal{P}) \\ v(\psi^i) = 1 \text{ and } v(\psi^{i+1}) = 1 \end{cases} \quad (4^*)$$

3.2 Mezhirova Game for Modal Logic of Serial Frames

In this section, we present a modified version of the game that is adequate for \mathbf{KD} normal modal logic, which is the logic of *serial frames*. The corresponding logical axiom is $\Box\varphi \rightarrow \Diamond\varphi$, called the **D**-axiom. Thus, syntactically $\mathbf{KD} = \mathbf{K} + \mathbf{D}$. The semantics is defined in a standard way, cf. the Section for $\mathbf{KD}!$, but in the case of \mathbf{KD} , there is only one restriction on the accessibility relation R , namely, *seriality* i.e., $\forall x \exists y : xRy$ where $x, y \in W$. Valuation and forcing relation are defined in a standard way.

The Game is Mezhirova game for $\mathbf{KD}!$ modulo moves for \mathbf{O} . \mathbf{O} moves by:

- (a) marking a formula from \mathcal{Fl}_{\Box} (adding it to \mathcal{O}) or
- (b) changing the valuation of propositional formulae in V^{Pr} (iff it forces \mathbf{P} to move next¹¹) or
- (c) unmarking a previously marked formula (deleting it from \mathcal{O}). No position can be visited twice during the game.

Why cannot we restrict players' move only to marking? Then \mathbf{P} would have winning strategies over some formulae that are not valid in \mathbf{KD} . For example, consider formula $\Box((\Box(\Box p \vee \Box q) \rightarrow \Box\Box p) \vee (\Box(\Box p \vee \Box q) \rightarrow \Box\Box q))$ which is not valid in \mathbf{KD} , but if players are not allowed to unmark formulae, then \mathbf{P} has a winning strategy for the corresponding game.

There exists a more efficient way to bound the number of repetitions (of marking/unmarking) based on the structure of φ^0 . However, it is more involved and requires additional definitions tightly related to countermodel construction and are left outside the scope of this work.

¹¹ Move (b) can be made iff it forces \mathbf{P} to have a mistake and \mathbf{O} to have none.

Example 5. Variables $\{p^0, p^1, p^2, p^3\}$.

\mathcal{F}	\mathcal{O}	\mathcal{P}	\mathcal{V}_0	\mathcal{V}_1	\mathcal{V}_2	\mathcal{V}_3	\mathcal{V}_4	\mathcal{V}_5
$\Box(\Box p \rightarrow \Box\Box p)^0$		0.X	1	1	0	0	0	0
$(\Box\Box p)^1$		3.X	0	0	0	0	0	0
$(\Box p)^1$	1.X	5.X	0	0	1	1	1	1
$(\Box p)^2$		4.X	0	0	0	0	0	0
Variable Valuations	$2.V^{pr} = \{p^2\}$							

Adequacy of Mezhiro Game for KD w.r.t. Serial Frames

Theorem 3. \mathbf{P} has a w.s. in Mezhiro game for KD over φ iff $\models_{\mathbf{KD}} \varphi$.

Proof: We omit some very technical details to provide the general idea of the proof and the relation between the game and Kripke structures.

\implies -direction: Proof by contraposition: Assume that $\not\models_{\mathbf{KD}} \varphi$, then there exists a model \mathcal{M} and a world x_0 such that $\mathcal{M}, x_0 \not\models \varphi$. By Claim 1, there exists a serial model \mathcal{M}_n with $d(\mathcal{M}_n) = MD(\varphi) + 1 = n$ and with the root node x_0 s.t. $\mathcal{M}_n, x_0 \not\models \varphi$. \mathcal{M}_n has at least one simple path of the length n where $x_{n-1}Rx_{n-1}$. We shall refer to the worlds of this path as x_i where $0 \leq i \leq n - 1$. To refer to any world of the model, we shall use variable w_j .

We relate the model and the formulae used in the game. Each world we assign a set X_i which intuitively corresponds to how deep in the model this world is w.r.t. the world x_0 :

1. $X^0 = \{x_0\}$;
2. $X^k = \{w_i \mid w_jRw_i \text{ and } w_j \in X^{k-1}\}$.

Let us relate the sets of worlds to the upper indexes in the game. Note that the $ind(X^i) : \{X^0, \dots, X^{n-1}\} \longrightarrow \{0, \dots, n - 1\}$ is a bijection where $ind(X^k) = k$.

Let us build \mathbf{O} 's strategy as follows:

1. \mathbf{O} fixes up a path π_0 of the length $\leq n$ in the countermodel.
2. \mathbf{O} marks $(\Box\psi)^i$ iff $\mathcal{M}, x_i \models \Box\psi$ where $x_i \in \pi_0$;
3. \mathbf{O} evaluates all atomic formulae ψ^i to 1 iff $\mathcal{M}_n, x_i \models \psi$ where $x_i \in \pi_0$;
4. If \mathbf{O} has marked all the formulae according to 1–3 but \mathbf{P} does not have a mistake, \mathbf{O} chooses another path π_1 of the length $\leq n$ such that there exists a formula $(\Box\psi)^i \in \mathcal{P}$ s.t. $\mathcal{M}_n, x_i \not\models \Box\psi$ where $x_i \in \pi_1$.
5. \mathbf{O} unmarks formulae $(\Box\chi)^j \in \mathcal{O}$ s.t. $\mathcal{M}_n, x_j \not\models \Box\chi$ where $x_j \in \pi_1$. Then repeat 2–3 w.r.t. the path π_1 .
6. If the current \mathbf{O} 's fixed path is π and \mathbf{P} has unmarked all the formulae $(\Box\psi)^i \in \pi$ s.t. $\mathcal{M}_n, w_i \not\models \psi$ where $w_i \in \pi$, then \mathbf{O}^{12} fixes a new path π^* and proceeds with actions 5, 2, and 3;

\mathbf{O} keeps repeating the same tactic. We need to prove the following:

Claim 5. If there are no formulae left for \mathbf{O} to choose when following this strategy, i.e., all formulae that are true in π fixed by \mathbf{O} 's have been marked and the game reached the terminal position, then it is \mathbf{P} 's move.

¹² We prove that it is always possible in what follows.

Proof Sketch: The idea is to prove that **P** has at least one mistake. Unlike in the case of Mezhirov game for **KD!**, here the mistake is not necessarily the initial formula. Two cases should be considered:

1. In case when the mistake is in the initial formula, we can use the proof of Claim 2 since it does not require branching.
2. In case when branching in the countermodel is required, the observation that **P** cannot unmark the mistakes that are semantically related to the previous branch used in **O**'s strategy is used. We leave out the technical details, but rather explain the argument: The idea is that the branching is finite, hence there are only finitely many paths π to fix. Hence, after a finite number of iterations, **O** will come back to marking the set of formulae as he has previously marked ($\mathcal{O}_f = \mathcal{O}_i$ s.t. $i \leq f$ where \mathcal{O}_f is **O**'s set at the terminal position). However, given that **P** has additional formulae in \mathcal{P} , **O** can do that. Each time **O** has marked all the formulae according to his strategy and a fixed path π : $v(\psi^i) = 1$ for all $\psi^i \in \mathcal{O}$, so **O** does not have mistakes. On the contrary, **P** won't be able to unmark all the mistakes since it would lead to the repetition of a previous game position. Hence, **P** has at least one mistake and it is **P**'s move indeed. \square

If **P** has used up all the possibilities to mark/unmark it, then **P** loses.

\implies -direction: This direction is analogous to the proof of Lemma 2 modulo changes similar to those in the proof of \implies -direction. The major technical complication is related to the fact that **P**'s mistake is not necessarily the initial formula φ^0 . Hence, constructing a countermodel from **O**'s winning strategy becomes more involved. \square

4 Conclusion and Future Work

In this paper, we proposed a new game-theoretic semantics for several logics: Johansson's Minimal logic, logic of functional frames **KD!**, and logic **KD** (of serial frames). The choice of logics is dictated by the idea of extending the framework to deal with intuitionistic (and potentially minimal) modal logic. The game is played over the validity of a formula, i.e., truth in every model. We have proved the adequacy of the provability games for **MPC** and **KD!** as well as provided a proof sketch for the **KD** case¹³: a formula φ is valid in one of the mentioned logics if and only if **P** has a winning strategy in a game over φ for that logic. Since the games are finite and the logics have the finite model property under intended Kripke frames, so is the search for winning strategies.

We have looked at two different modal logics: one without branching (**KD!**, the logic of functional frames) and one allowing branching, namely **KD**. This shows that the approach is not limited to the logics with bounded branching or reflexive and transitive like in **Grz** case. Hence, we plan to extend the approach to a wider range of normal modal logics since we have established the relation

¹³ The full formal proof is subject to a separate presentation due to its technicality.

between structural properties of the games and Kripke frames of corresponding logics. The game shows a tight connection between winning strategies and countermodel contraction. We hypothesise that for every logic having finite model property, a variant of Mezhirov game can be proposed.

Another promising application direction for the game that we plan to investigate is a provability game for intuitionistic and minimal modal logic. Since the framework is rather flexible, we are working on integrating intuitionistic (minimal) and modal features. One of the logics of our interest is Intuitionistic Epistemic Logic *IEL*. Kripke semantics for *IEL* comprises two accessibility relations, one being a subset of another. We see the potential of Mezhirov game in dealing with various intuitionistic modal logics.

Since the game is closely related to Kripke semantics, it is of interest to relate **P**'s winning strategies to the proof-theoretic concept of inference. The possibility of translation between winning strategies and proofs in labelled sequent calculi is one of the major open questions.

References

1. van Benthem, J.: Logic in Games. The MIT Press, Cambridge (2014)
2. Väänänen, J.: Models and Games. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge (2011)
3. Benthem, J.: Logic Games: From Tools to Models of Interaction, pp. 183–216, March 2011
4. Lorenzen, P., Lorenz, K.: Dialogische Logik. Wissenschaftliche Buchgesellschaft (1978)
5. Rahman, S., Tulenheimo, T.: From games to dialogues and back. In: Majer, O., Pietarinen, A.V., Tulenheimo, T. (eds.) Games: Unifying Logic, Language, and Philosophy. LEUS, vol. 15, pp. 153–208. Springer, Dordrecht (2009). https://doi.org/10.1007/978-1-4020-9374-6_8
6. Fermüller, C.G., Metcalfe, G.: Giles's game and the proof theory of Lukasiewicz logic. Stud. Logica. **92**(1), 27–61 (2009)
7. Fermüller, C., Lang, T., Pavlova, A.: From truth degree comparison games to sequents-of-relations calculi for Gödel logic. In: Lesot, M.-J., et al. (eds.) IPMU 2020. CCIS, vol. 1237, pp. 257–270. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50146-4_20
8. Hintikka, J.: Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic. Clarendon Press, Oxford (1973)
9. Hintikka, J.: The Principles of Mathematics Revisited. Cambridge University Press, Cambridge (1996)
10. Mezhirov, I.: A game semantics for Grz. J. Logic Comput. **16**(5), 663–669 (2006). eprint: <https://academic.oup.com/logcom/article-pdf/16/5/663/6282443/exl029.pdf>
11. Mezhirov, I., Vereshchagin, N.: On game semantics of the affine and intuitionistic logics. In: Hodges, W., de Queiroz, R. (eds.) WoLLIC 2008. LNCS (LNAI), vol. 5110, pp. 28–42. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69937-8_4
12. Johansson, I.: Der minimalkalkül, ein reduzierter intuitionistischer formalismus. Compos. Math. **4**, 119–136 (1937)

13. Colacito, A., de Jongh, D., Vargas, A.L.: Subminimal negation. *Soft. Comput.* **21**(1), 165–174 (2016). <https://doi.org/10.1007/s00500-016-2391-8>
14. Odintsov, S., Rybakov, V.: Unification and admissible rules for paraconsistent minimal Johanssons' logic j and positive intuitionistic logic IPC+. *Ann. Pure Appl. Log.* **164**, 771–784 (2013)
15. Odintsov, S.P.: *Constructive Negations and Paraconsistency*. Springer, Dordrecht (2008)
16. de Jongh, D., Bezhanishvili, N.: *Intuitionistic logic: Lecture notes* (2006)
17. Goranko, V., Otto, M.: Model theory of modal logic. In: *Handbook of Modal Logic*, pp. 255–325, January 2006
18. Segerberg, K.: Modal logics with functional alternative relations. *Notre Dame J. Formal Log.* **27**(4), 504–522 (1986)
19. Standefer, S.: Proof theory for functional modal logic. *Stud. Logica.* **106**(1), 49–84 (2017). <https://doi.org/10.1007/s11225-017-9725-0>
20. Mezhirov, I.: Game semantics for intuitionistic and modal (Grz) logic. Master's thesis, Lomonosov Moscow State University, Department of Mechanics and Mathematics (2006)
21. Pavlova, A.: Dialogue games for minimal logic. *Logic Log. Philos.* **30**(2), 281–309 (2021)

Author Index

- Abriola, Sergio 319
Akbar Tabatabai, Amirhossein 287
Avron, Arnon 167
- Balbiani, Philippe 219
Barlag, Timon 16
- Catta, Davide 269
Ciobăcă, Ștefan 150
- Fernández González, Saúl 219
Freiman, Robert 133
From, Asta Halkjær 1
- Galliani, Pietro 47
Ghosh, Sujata 201
González, Nicolás 319
- Iemhoff, Rosalie 287
Iordache, Viorel 150
- Jalali, Raheleh 287, 337
- Konečný, Michal 252
Kontinen, Juha 302
Kuznets, Roman 337
- Leivant, Daniel 372
Levi, Nissan 167
Li, Dazhu 201
Liu, Fenrong 201
- Marin, Sonia 388
- Park, Sewon 252
Pavlova, Alexandra 408
Pereira, Luiz Carlos 388
Pezlar, Ivo 100
Pimentel, Elaine 388
Punčochář, Vít 355
- Rohani, Atefeh 64
- Sales, Emerson 388
Sandström, Max 302
Stevens-Guille, Symon Jory 269
Studer, Thomas 64
- Tedder, Andrew 355
Thies, Holger 252
Thompson, Declan 235
Tu, Yaxin 201
- van der Giessen, Iris 337
van Ditmarsch, Hans 31
Veltri, Niccolò 184
Vollmer, Heribert 16
- Wen, Xuefeng 117
- Zhong, Shengyang 82