LNAI 14278

Revantha Ramanayake
Josef Urban (Eds.)

Automated Reasoning with Analytic Tableaux and Related Methods

32nd International Conference, TABLEAUX 2023
Prague, Czech Republic, September 18–21, 2023
Proceedings

Springer

OPEN ACCESS

Revantha Ramanayake · Josef Urban
Editors

Automated Reasoning with Analytic Tableaux and Related Methods

32nd International Conference, TABLEAUX 2023
Prague, Czech Republic, September 18–21, 2023
Proceedings

Springer

Editors
Revantha Ramanayake
University of Groningen
Groningen, The Netherlands

Josef Urban
Czech Technical University in Prague
Prague, Czech Republic

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-031-43512-6 ISBN 978-3-031-43513-3 (eBook)
https://doi.org/10.1007/978-3-031-43513-3

LNCS Sublibrary: SL7 – Artificial Intelligence

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

Paper in this product is recyclable.

Preface

TABLEAUX, the *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, is a conference series that started in 1992 and has been held every year since then. The series brings together researchers interested in all aspects - theoretical foundations, implementation techniques, systems development and applications - of the mechanization of reasoning with tableaux and related methods. Since 1995, proceedings of TABLEAUX have been published in Springer's LNCS/LNAI series.

TABLEAUX 2023 was the 32nd edition of the conference series and it was an in-person conference hosted by the Czech Technical University in Prague, Czech Republic, September 18–21, 2023. It was co-located with the 14th International Symposium on Frontiers of Combining Systems (FroCoS 2023).

The Program Committee received a total of 43 submissions, comprising 33 research papers and 10 short papers. Each submission received on average three reviews in a single-blind process and was evaluated during program committee discussions. Eventually 20 research papers and 5 short papers were accepted for presentation at the conference.

This volume includes all the accepted research papers and short papers of TABLEAUX 2023. These include papers on proof theory, with deductive mechanisms ranging from tableaux, sequent calculi and extensions, and non-wellfounded proofs. Their objects of inquiry encompass a range of modal logics, including in the non-normal, intuitionistic, constructive and temporal settings, linear logic, MV-algebras, separation logic, first-order logics and results on cut-elimination, termination and complexity of proof search, term-forming operators and proof-theoretic semantics. Investigations also delve into formalised proofs, automated theorem proving for classical and non-classical logics, and their integration with machine learning and SMT solvers. In addition to the main track, this year's edition hosted a special track on Artificial Intelligence and Theorem Proving (AITP), inviting papers combining machine learning and related AI methods with standard TABLEAUX topics.

This volume also includes abstracts of invited talks presented at TABLEAUX 2023. The five invited speakers, chosen by the Program Committee, were:

– Marta Bílková (Czech Academy of Sciences, Czechia) *joint with FroCoS*
– Chad E. Brown (Czech Technical University in Prague, Czechia) *joint with FroCoS*
– Valentin Goranko (Stockholm University, Sweden) *joint with FroCoS*
– Rosalie Iemhoff (Utrecht University, The Netherlands)
– Roman Kuznets (Technische Universität Wien, Austria)

The following papers were selected by the Program Committee for awards:

– **Best Paper.** Ian Shillito, Iris van der Giessen, Rajeev Gore and Rosalie Iemhoff. *A new calculus for intuitionistic Strong Löb logic: strong termination and cut-elimination, formalised.*

– **Best Junior Researcher Paper.** Bahareh Afshari, Lide Grotenhuis, Graham Leigh and Lukas Zenger. *Ill-founded Proof Systems For Intuitionistic Linear-time Temporal Logic.*

The two awards were presented at the conference.

We thank all the people who contributed to making TABLEAUX 2023 a success. We thank the Programme Committee and all additional reviewers for the time, professional effort and expertise they invested to deliver the high scientific standards of the conference and these proceedings. We thank the local organizers for making this event happen. We thank the invited speakers for their inspiring talks, and the Steering Committee for their helpful advice. We thank all the authors for their excellent contributions. Special thanks to Jens Otten who supported us with advice through all phases of the conference.

We would also like to thank Springer for sponsoring the conference and publishing these proceedings, University of Innsbruck for providing the registration system, and the Czech Institute of Informatics, Robotics, and Cybernetics (CIIRC-CTU) for hosting and supporting the conference and its organization.

July 2023

Revantha Ramanayake
Josef Urban

Organization

Program Committee Chairs

Revantha Ramanayake	University of Groningen, The Netherlands
Josef Urban	Czech Technical University in Prague, Czechia

Steering Committee

Agata Ciabattoni	Technische Universität Wien, Austria
Anupam Das	University of Birmingham, UK
Cláudia Nalon	University of Brasília, Brazil
Hans de Nivelle	Nazarbayev University, Kazakhstan
Jens Otten	University of Oslo, Norway
Dirk Pattinson	Australian National University, Australia
Elaine Pimentel	Federal University of Rio Grande do Norte, Brazil
Andrei Popescu	University of Sheffield, UK

Program Committee

Bahareh Afshari	University of Gothenburg, Sweden, and University of Amsterdam, The Netherlands
Carlos Areces	Universidad Nacional de Córdoba, Argentina
Peter Baumgartner	Data61/CSIRO, Australia
Serenella Cerrito	Université Paris-Saclay, Université d'Evry, France
Kaustuv Chaudhuri	Inria, France
Anupam Das	University of Birmingham, UK
Stéphane Demri	CNRS, France
Clare Dixon	University of Manchester, UK
Christian Fermüller	Technische Universität Wien, Austria
Camillo Fiorentini	Universitá degli Studi di Milano, Italy
Ulrich Furbach	University of Koblenz, Germany
Didier Galmiche	Université de Lorraine, France
Silvio Ghilardi	Universitá degli Studi di Milano, Italy
Marianna Girlando	University of Amsterdam, The Netherlands
Charles Grellois	Université de Bordeaux, France
Andrzej Indrzejczak	University of Łódź, Poland

Cezary Kaliszyk	University of Innsbruck, Austria
Hidenori Kurokawa	Kanazawa University, Japan
Stepan Kuznetsov	Russian Academy of Sciences, Russia
Timo Lang	University College London, UK
Stéphane Graham-Lengrand	SRI International, USA
Sonia Marin	University of Birmingham, UK
Neil Murray	University at Albany, USA
Cláudia Nalon	University of Brasília, Brazil
Sara Negri	University of Genoa, Italy
Hans de Nivelle	Nazarbayev University, Kazakhstan
Eugenio Orlandelli	University of Bologna, Italy
Jens Otten	University of Oslo, Norway
Alessandra Palmigiano	Vrije Universiteit Amsterdam, The Netherlands
Dirk Pattinson	Australian National University, Australia
Nicolas Peltier	CNRS, France
Frank Pfenning	Carnegie Mellon University, USA
Elaine Pimentel	University College London, UK
Gian Luca Pozzato	University of Turin, Italy
Michael Rawson	Technische Universität Wien, Austria
Reuben Rowe	Royal Holloway, University of London, UK
Katsuhiko Sano	Hokkaido University, Japan
José Espírito Santo	University of Minho, Portugal
Lutz Straßburger	Inria, France
Thomas Studer	University of Bern, Switzerland
Yoni Zohar	Bar-Ilan University, Israel
Zsolt Zombori	Alfréd Rényi Institute of Mathematics, Hungary

Local Organizers

Karel Chvalovský	Czech Technical University in Prague, Czechia
Jan Jakubův	Czech Technical University in Prague, Czechia
Cezary Kaliszyk	University of Innsbruck, Austria
Martin Suda	Czech Technical University in Prague, Czechia
Josef Urban	Czech Technical University in Prague, Czechia

Additional Reviewers

Stefano Aguzzoli
Martín Diéguez
Andrea De Domenico
Mauro Ferrari
Guido Fiorino
Pietro Galliani
Anton Gnatenko
Giuseppe Greco
Sean Holden
Etienne Lozes
Tim Lyon
Sergei Odintsov
Edi Pavlovic
Florian Rabe
Atefeh Rohani
Tor Sandqvist
Apostolos Tzimoulis
Dominik Wehr
Junhua Yu
Lukas Zenger

Abstracts of Invited Talks

Epistemic Logics of Structured Intensional Groups: Agents - Groups - Names - Types

Marta Bílková

Czech Academy of Sciences, Czechia

In the overwhelming majority of contributions to multi-agent epistemic, doxastic, and coalition logic, a group is reduced to its extension, i.e., the set of its members. This has a counter-intuitive consequence that groups change identity when their membership changes, and rules out uncertainty regarding who is a member of a given group. Additionally, this idealization does not reflect the structure of groups, or the structured way in which collective epistemic attitudes emerge, in the intended application of logical models. We will outline an abstract framework in which we can lift this idealisation, namely replacing agent or group labels of epistemic modalities with names, or providing them with an algebraic structure relevant to types of collective epistemic attitudes in question. The resulting formalisms are essentially two-sorted, combining the language of labels of modalities and the language of epistemic statements. A fully abstract account of such epistemic logics can be given, linking two-sorted algebras (involving propositions and group labels/types of knowledge) with monotone neighborhood frame semantics, in terms of an algebraic duality. This can further be applied to obtain, e.g., a definability theorem or to design a multi-type proof theory for the basic logic. We further discuss several particular examples of algebraic signatures giving rise to interesting and useful variants of group knowledge.

First-Order Instantiation-Based Tableau

Chad E. Brown

Czech Technical University in Prague, Czechia

We present a tableau calculus for first-order logic with equality. The calculus is a fragment of the higher-order calculus that is the theoretical basis for the award winning higher-order automated theorem prover Satallax and its successor Lash. A key aspect of the calculus is that universal quantifiers only need to be instantiated with terms that occur on one side of a disequation on the current open branch. This makes the search instantiation-based (as no metavariables are introduced and no unification is used). We will give an overview of the completeness proof and how the completeness proof can be modified to justify various modifications to the calculus. Both Satallax and Lash make use of the SAT solver MiniSat to determine when the search is complete (i.e., when every branch of the tableau is closed). Superposition provers like Vampire and E and SMT solvers like CVC5 and Z3 outperform Lash on typical first-order TPTP problems (used in the CASC competition). However, we will present a set of first-order clausal problems on which Lash significantly outperforms other provers.

Combining Semantic Tableaux

Valentin Goranko

Stockholm University, Sweden

Semantic tableaux for combined logical systems are usually constructed ad hoc and the question of developing more general methodologies for combining tableaux is yet to be systematically explored.

In this talk I will address that question and will outline a methodological approach for combining tableaux. I will discuss the questions of transfer of soundness, completeness, and termination from the components to the combined tableaux, both in general and in the context of some important special cases, including multi-agent epistemic and temporal epistemic logics.

Proof Systems and Termination

Rosalie Iemhoff

Utrecht University, The Netherlands

In the study of logics, proof systems are a useful tool, and proof systems that are terminating even more so. Termination comes in degrees, where the strongest form of termination arguably requires that any backwards proof search in the proof system terminates. Not every application in which a proof system is involved needs this strong form of termination, but some applications seem to do so. In this talk I discuss the role of termination in proof theory, and connect it in particular to counter model constructions and interpolation.

Always Look on Both Sides of Proof: Syntax and Semantics as the Yin and Yang of Structural Proof Theory

Roman Kuznets

Technische Universität Wien, Austria

Proof theory provides a purely syntactic way of reasoning, without the need to resort to semantics. This is especially true of internal proof calculi where proof objects are interpreted as formulas, as opposed to external calculi that also exploit semantic elements. On the other hand, tableau formalisms suggest that the distinction between pure and "impure" syntax, between internal and external calculi is, perhaps, more superficial than commonly believed. Indeed, tableaus are typically isomorphic to some internal sequent-like calculus, despite themselves being described in largely semantic terms.

I argue that the choice between embracing and avoiding semantic elements is a false one, that the two sides of proof formalisms mutually enrich rather than oppose each other. As an illustration of such successful interplay, I will discuss how semantic intuitions have been instrumental in developing several proof formalisms, including those used for solving two open problems: (1) the Lyndon interpolation property for Gödel-Dummett Logic and (2) decidability for the intuitionistic modal logic S4.

Supported by the Austrian Science Fund (FWF) project ByzDEL (P33600).

Contents

Tableau Calculi

Range-Restricted and Horn Interpolation through Clausal Tableaux

Christoph Wernhard(✉)

University of Potsdam, Potsdam, Germany
info@christophwernhard.com

Abstract. We show how variations of range-restriction and also the Horn property can be passed from inputs to outputs of Craig interpolation in first-order logic. The proof system is clausal tableaux, which stems from first-order ATP. Our results are induced by a restriction of the clausal tableau structure, which can be achieved in general by a proof transformation, also if the source proof is by resolution/paramodulation. Primarily addressed applications are query synthesis and reformulation with interpolation. Our methodical approach combines operations on proof structures with the immediate perspective of feasible implementation through incorporating highly optimized first-order provers.

1 Introduction

We show how variations of range-restriction and also the Horn property can be passed from inputs to outputs of Craig interpolation in first-order logic. The primarily envisaged application field is synthesis and reformulation of queries with interpolation [5,39,56]. Basically, the sought target query R is understood there as the right side of a definition of a given query Q within a given background knowledge base K, i.e., it holds that $K \models (Q \leftrightarrow R)$, where the vocabulary of R is in a given set of permitted target symbols. In first-order logic, the formulas R can be characterized as the Craig interpolants of $K \land Q$ and $\neg K' \lor Q'$, where K, Q are copies of K', Q' with the symbols not allowed in R replaced by fresh symbols [14]. Formulas R exist if and only if the entailment $K \land Q \models \neg K' \lor Q'$ holds. They can be constructed as Craig interpolants from given proofs of the entailment in a suitable calculus.

In databases and knowledge representation, syntactic fragments of first-order logic ensure desirable properties, for example domain independence. Typically, for given K and Q in some such fragment, also R must be in some specific fragment to be usable as a query or as a knowledge base component. Our work addresses this by showing for certain such fragments how membership is passed on to interpolants and thus to the constructed right sides of definitions. The

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495. The work was supported by the North-German Supercomputing Alliance (HLRN).

fragment in focus here is a variant of range-restriction from [59], known as a rather general syntactic condition to ensure domain independence [1, p. 97]. It permits conversion into a shape suitable for "evaluation" by binding free and quantified variables successively to the members of given predicate extensions. Correspondingly, if the vocabulary is relational, a range-restricted formula can be translated into a relational algebra expression. First-order representations of widely-used classes of integrity constraints, such as tuple-generating dependencies, are sentences that are range-restricted in the considered sense.

As proof system we use *clausal tableaux* [26,29–31,33], devised in the 1990s to take account of automated first-order provers that may be viewed as enumerating tree-shaped proof structures, labeled with instances of input clauses.[1] Such systems include the Prolog Technology Theorem Prover [53], SETHEO [32], leanCoP [42,43] and CMProver [16,45,60,61]. As shown in [62], a *given* closed clausal tableau is quite well-suited as a proof structure to extract a Craig interpolant. Via the translation of a resolution deduction tree [12] to a clausal tableau in cut normal form [31,62] this transfers also to interpolation from a given resolution/paramodulation proof.

Since the considered notion of range-restriction is based on prenexing and properties of both a CNF and a DNF representation of the formula, it fits well with the common first-order ATP setting involving Skolemization and clausification and the ATP-oriented interpolation on the basis of clausal tableaux, where in a first stage the propositional structure of the interpolant is constructed and in a second stage the quantifier prefix.

Our strengthenings of Craig interpolation are induced by a specific restriction of the clausal tableau structure, which we call *hyper*, since it relates to the proof structure restrictions of hyperresolution [46] and hypertableaux [2]. However, it is considered here for tree structures with rigid variables. A proof transformation that converts an arbitrary closed clausal tableau to one with the hyper property shows that the restriction is w.l.o.g. and, moreover, allows the prover unhampered search for the closed clausal tableaux or resolution/paramodulation proof underlying interpolation.

Structure of the Paper. Section 2 summarizes preliminaries, in particular interpolation with clausal tableaux [62]. Our main result on strengthenings of Craig interpolation for range-restricted formulas is developed in Sect. 3. Section 4 discusses Craig interpolation from a Horn formula, also combined with range-restriction. The proof transformation underlying these results is introduced in Sect. 5. We conclude in Sect. 6 with discussing related work, open issues and perspectives.

[1] Alternate accounts and views are provided by model elimination [34] and the connection method [7,8].

Proofs of nontrivial claims that are not proven in the body of the paper are supplemented in the preprint version [63]. An implementation with the PIE environment [60,61][2] is in progress.

2 Notation and Preliminaries

2.1 Notation

We consider *formulas* of first-order logic. An *NNF formula* is a quantifier-free formula built up from *literals* (atoms or negated atoms), truth-value constants $\top, \bot$, conjunction and disjunction. A *CNF formula*, also called *clausal formula*, is an NNF formula that is a conjunction of disjunctions (*clauses*) of literals. A *DNF formula* is an NNF formula that is a disjunction of conjunctions (*conjunctive clauses*) of literals. The complement of a literal L is denoted by $\overline{L}$. An occurrence of a subformula in a formula has positive (negative) *polarity*, depending on whether it is in the scope of an even (odd) number of possibly implicit occurrences of negation. Let F be a formula. $\mathcal{V}ar(F)$ is set of its free variables. $\mathcal{V}ar^{+}(F)$ ($\mathcal{V}ar^{-}(F)$) is the set of its free variables with an occurrence in an atom with positive (negative) polarity. $\mathcal{F}un(F)$ is the set of functions occurring in it, including constants, regarded here throughout as 0-ary functions. $\mathcal{P}red^{\pm}(F)$ is the set of pairs $\langle p, pol \rangle$, where p is a predicate and $pol \in \{+, -\}$, such that an atom with predicate p occurs in F with the polarity indicated by pol. $\mathcal{V}oc^{\pm}(F)$ is $\mathcal{F}un(F) \cup \mathcal{P}red^{\pm}(F)$. A *sentence* is a formula without free variables. An NNF is *ground* if it has no variables. If S is a set of terms, we call its members S-terms. The $\models$ symbol expresses semantic entailment.

2.2 Clausal First-Order Tableaux

A *clausal tableau* (briefly *tableau*) *for* a clausal formula F is a finite ordered tree whose nodes N with exception of the root are labeled with a literal $\mathsf{lit}(N)$, such that for each node N the disjunction of the literals of all its children in their left-to-right order, $\mathsf{clause}(N)$, is an instance of a clause in F. A branch of a tableau is *closed* iff it contains nodes with complementary literals. A node is *closed* iff all branches through it are closed. A tableau is *closed* iff its root is closed. A node is *closing* iff it has an ancestor with complementary literal. With a closing node N, a particular such ancestor is associated as *target of* N, written $\mathsf{tgt}(N)$. A tableau is *regular* iff no node has an ancestor with the same literal and is *leaf-closing* iff all closing nodes are leaves. A closed tableau that is leaf-closing is called *leaf-closed*. Tableau simplification can convert any tableau to a regular and leaf-closing tableau for the same clausal formula, closed iff the original tableau is so. Regularity is achieved by repeating the following operation [31, Sect. 2.1.3]: Select a node N with an ancestor that has the same literal, remove the edges originating in the parent of N and replace them with the edges originating in N. The leaf-closing property is achieved by repeatedly selecting an inner node

[2] http://cs.christophwernhard.com/pie.

N that is closing and removing the edges originating in N. All occurrences of variables in (the literal labels of) a tableau are free and their scope spans the whole tableau. That is, we consider *free-variable tableaux* [30, p. 158ff] with *rigid* variables [26, p. 114]. A tableau without variables is called *ground*. The universal closure of a clausal formula F is unsatisfiable iff there exists a closed clausal tableau for F. This holds also if *clausal tableau* is restricted by the properties *ground*, *regular* and *leaf-closing* in arbitrary combinations.

2.3 Interpolation with Clausal Tableaux

Craig's interpolation theorem [13,15] along with Lyndon's observation on the preservation of predicate polarities [35] ensures for first-order logic the existence of *Craig-Lyndon interpolants*, defined as follows. Let F, G be formulas such that $F \models G$. A *Craig-Lyndon interpolant* of F and G is a formula H such that (1) $F \models H$ and $H \models G$. (2) $\mathcal{V}oc^{\pm}(H) \subseteq \mathcal{V}oc^{\pm}(F) \cap \mathcal{V}oc^{\pm}(G)$. (3) $\mathcal{V}ar(H) \subseteq \mathcal{V}ar(F) \cap \mathcal{V}ar(G)$. The perspective of validating an entailment $F \models G$ by showing unsatisfiability of $F \land \neg G$ is reflected in the notion of *reverse Craig-Lyndon interpolant* of F and G, defined as Craig-Lyndon interpolant of F and $\neg G$.

Following [62], our interpolant construction is based on a generalization of clausal tableaux where nodes have an additional *side* label that is shared by siblings and indicates whether the tableau clause is an instance of an input clause derived from the formula F or of the formula G of the statement $F \land G \models \bot$ underlying the reverse interpolant. Thus, a *two-sided clausal tableau for* clausal formulas F *and* G is a tableau for $F \land G$ whose nodes N with exception of the root are labeled additionally with a *side* $\mathsf{side}(N) \in \{\mathsf{F}, \mathsf{G}\}$, such that (1) if N and N' are siblings, then $\mathsf{side}(N) = \mathsf{side}(N')$; (2) if N has a child N' with $\mathsf{side}(N') = \mathsf{F}$, then $\mathsf{clause}(N)$ is an instance of a clause in F, and if N has a child N' with $\mathsf{side}(N') = \mathsf{G}$, then $\mathsf{clause}(N)$ is an instance of a clause in G. We also refer to the side of the children of a node N as *side of* $\mathsf{clause}(N)$. For *side* $\in \{\mathsf{F}, \mathsf{G}\}$ define $\mathsf{path}_{side}(N) \overset{\text{def}}{=} \bigwedge_{N' \in Path \text{ and } \mathsf{side}(N')=side} \mathsf{lit}(N')$, where *Path* is the union of the set of the ancestors of N and $\{N\}$.

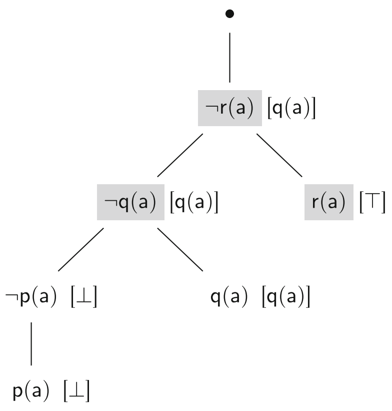


Fig. 1. A two-sided clausal tableau.

Let N be a node of a leaf-closed two-sided clausal tableau. The value of $\mathsf{ipol}(N)$ is an NNF formula, defined inductively as specified with the tables below, the left for the base case where N is a leaf, the right for the case where N is an inner node with children $N_1, \ldots, N_n$.

side(N)	side(tgt(N))	ipol(N)
F	F	$\bot$
F	G	$\mathsf{lit}(N)$
G	F	$\overline{\mathsf{lit}(N)}$
G	G	$\top$

side(N_1)	ipol(N)
F	$\bigvee_{i=1}^{n} \mathsf{ipol}(N_i)$
G	$\bigwedge_{i=1}^{n} \mathsf{ipol}(N_i)$

Example 1. Figure 1 shows a two-sided tableau for $F = \mathsf{p(a)} \land (\neg \mathsf{p(a)} \lor \mathsf{q(a)})$ and $G = (\neg \mathsf{q(a)} \lor \mathsf{r(a)}) \land \neg \mathsf{r(a)}$. Side G is indicated by gray background. For each node the value of ipol, after truth-value simplification, is annotated in brackets. The clauses of the tableau are $\neg \mathsf{r(a)}$ and $\neg \mathsf{q(a)} \lor \mathsf{r(a)}$, which have side G, and $\neg \mathsf{p(a)} \lor \mathsf{q(a)}$ and $\mathsf{p(a)}$, which have side F. If N is the node shown bottom left, labeled with $\mathsf{p(a)}$, then $\mathsf{path_F}(N) = \neg \mathsf{p(a)} \land \mathsf{p(a)}$ and $\mathsf{path_G}(N) = \neg \mathsf{r(a)} \land \neg \mathsf{q(a)}$.

If N_0 is the root of a two-sided tableaux for clausal *ground* formulas F and G, then $\mathsf{ipol}(N_0)$ is a Craig-Lyndon interpolant of F and $\neg G$.[3] The CTIF (*Clausal Tableau Interpolation for First-Order Formulas*) procedure (Fig. 2) [62] extends this to a two-stage [9,24] (inductive construction and lifting) interpolation method for full first-order logic. It is complete (yields a Craig-Lyndon interpolant for all first order formulas F and G such that $F \models G$) under the assumption that the method for tableau computation in Step 3 is complete (yields a closed tableau for all unsatisfiable clausal formulas). Some steps leave room for interpolation-specific heuristics: In step 4 the choice of the terms used for grounding; in step 5 the choice of the side assigned to clauses that are an instance of both a clause in F' and a clause in G'; and in step 7 the quantifier prefix, which is constrained just by a partial order.

Example 2. Let $F \overset{\text{def}}{=} \forall x\, \mathsf{p}(x) \land \forall x\, (\neg \mathsf{p}(x) \lor \mathsf{q}(x))$ and let $G \overset{\text{def}}{=} \forall x\, (\neg \mathsf{q}(x) \lor \mathsf{r}(x)) \rightarrow \mathsf{r(a)}$. Clausifying F and $\neg G$ then yields $F' = \mathsf{p}(x) \land \neg \mathsf{p}(x) \lor \mathsf{q}(x)$ and $G' = \neg \mathsf{q}(x) \lor \mathsf{r}(x) \land \neg \mathsf{r(a)}$. The tableau from Fig. 1 is a leaf-closed ground tableau for F' and G' and we obtain $\mathsf{q(a)}$ as H_{GRD}. Lifting for $\mathcal{F} = \{\}$ and $\mathcal{G} = \{\mathsf{a}\}$ yields the interpolant $H = \forall v_1\, \mathsf{q}(v_1)$.

Example 3. Let $F \overset{\text{def}}{=} \forall x \forall y\, \mathsf{p}(x, \mathsf{f}(x), y)$ and let $G \overset{\text{def}}{=} \exists x \mathsf{p}(\mathsf{a}, x, \mathsf{g}(x))$. Clausifying yields $F' = \mathsf{p}(x, \mathsf{f}(x), y)$ and $G' = \neg \mathsf{p}(\mathsf{a}, z, \mathsf{g}(z))$. We obtain $\mathsf{p(a, f(a), g(f(a)))}$ as H_{GRD}. Lifting is for $\mathcal{F} = \{\mathsf{f}\}$ and $\mathcal{G} = \{\mathsf{a}, \mathsf{g}\}$ with $t_1 = \mathsf{a}$, $t_2 = \mathsf{f(a)}$ and $t_3 = \mathsf{g(f(a))}$. It yields $H = \forall v_1 \exists v_2 \forall v_3\, \mathsf{p}(v_1, v_2, v_3)$.

3 Interpolation and Range-Restriction

We now develop our main result on strengthenings of Craig interpolation for range-restricted formulas.

[3] So far, the interpolation method is a variation of well-known methods for sequent systems [52,55] and analytic tableaux [20] when restricted to propositional formulas.

3.1 CNF and DNF with Some Assumed Syntactic Properties

Following [59] we will consider a notion of range-restriction defined in terms of properties of two prenex formulas that are equivalent to the original formula, have both the same quantifier prefix but matrices in CNF and DNF, respectively.

INPUT: First-order formulas F and G such that $F \models G$.

METHOD:

1. *Free variables to placeholder constants.* Let F_c and G_c be the sentences obtained from F and G by replacing each free variable with a dedicated fresh constant.
2. *Skolemization and clausification.* Apply there conversion to prenex form and second-order Skolemization independently to F_c and to $\neg G_c$, resulting in disjoint sets of fresh Skolem functions $\mathcal{F}', \mathcal{G}'$, clausal formulas F', G', and sets $\mathcal{U}' = \mathcal{V}ar(F'), \mathcal{V}' = \mathcal{V}ar(G')$ of variables such that

   (a) $F_c \equiv \exists \mathcal{F}' \forall \mathcal{U}' F'$ and $\neg G_c \equiv \exists \mathcal{G}' \forall \mathcal{V}' G'$.
   (b) $\mathcal{V}oc^{\pm}(F') \subseteq \mathcal{V}oc^{\pm}(F_c) \cup \mathcal{F}'$ and $\mathcal{V}oc^{\pm}(\neg G') \subseteq \mathcal{V}oc^{\pm}(G_c) \cup \mathcal{G}'$.
   (c) $\forall \mathcal{U}' \forall \mathcal{V}' (F' \land G') \models \bot$.

   In case F' or G' contains the empty clause, exit with result $H \overset{\text{def}}{=} \bot$ or $H \overset{\text{def}}{=} \top$, respectively.
3. *Tableau computation.* Compute a leaf-closed clausal tableau for the clausal formula $F' \land G'$. This can be obtained, for example, from a clausal tableaux prover for clausal first-order formulas.
4. *Tableau grounding.* Instantiate all variables of the tableau with ground terms built up from functions in $F' \land G'$ and possibly also fresh functions $\mathcal{S} = \mathcal{S}_1 \uplus \mathcal{S}_2$. Observe that the grounded tableau is still a leaf-closed tableau for $F' \land G'$.
5. *Side assignment.* Convert the ground tableau to a two-sided tableau for F' and G' by attaching appropriate *side* labels to all nodes except the root. This is always possible because every clause of the tableau is an instance of a clause in F' or in G'.
6. *Ground interpolant extraction.* Let H_{GRD} be the value of $\mathsf{ipol}(N_0)$, where N_0 is the root of the tableau.
7. *Interpolant lifting.* Let $\mathcal{F} \overset{\text{def}}{=} \mathcal{F}' \cup (\mathcal{F}un(F) \setminus \mathcal{F}un(G)) \cup \mathcal{S}_1$ and let $\mathcal{G} \overset{\text{def}}{=} \mathcal{G}' \cup (\mathcal{F}un(G) \setminus \mathcal{F}un(F)) \cup \mathcal{S}_2$. Let $\mathcal{FG}$ stand for $\mathcal{F} \cup \mathcal{G}$. An *$\mathcal{FG}$-maximal occurrence* of an $\mathcal{FG}$-term in a formula is an occurrence that is not within another $\mathcal{FG}$-term. Let $\{t_1, \ldots, t_n\}$ be the set of the $\mathcal{FG}$-terms with an $\mathcal{FG}$-maximal occurrence in H_{GRD}, ordered such that if t_i is a subterm of t_j, then $i < j$. Let $\{v_1, \ldots, v_n\}$ be a set of fresh variables. For $i \in \{1, \ldots, n\}$ define the quantifiers Q_i as $\exists$ if $t_i \in \mathcal{F}$-terms and as $\forall$ if $t_i \in \mathcal{G}$-terms. Let

   $$H_c \overset{\text{def}}{=} Q_1 v_1 \ldots Q_n v_n \, H'_{\text{GRD}},$$

   where H'_{GRD} is obtained from H_{GRD} by replacing all $\mathcal{FG}$-maximal occurrences of terms t_i with variable v_i, simultaneously for all $i \in \{1, \ldots, n\}$.
8. *Placeholder constants to free variables.* Let H be H_c after replacing any constants that were introduced in step 1 with their corresponding variables.

OUTPUT: Return H, a Craig-Lyndon interpolant of the input formulas F and G.

Fig. 2. The CTIF Procedure for Craig-Lyndon Interpolation [62].

Although not syntactically unique, we refer to them functionally as $\mathsf{cnf}(F)$ and $\mathsf{dnf}(F)$ since we only rely on specific – easy to achieve – syntactic properties that are stated in the following Proposition 4–6.

Proposition 4. *For all formulas F it holds that $\mathcal{V}ar(\mathsf{cnf}(F)) \subseteq \mathcal{V}ar(F)$; $\mathcal{V}oc^{\pm}(\mathsf{cnf}(F)) \subseteq \mathcal{V}oc^{\pm}(F)$; $\mathcal{V}ar(\mathsf{dnf}(F)) \subseteq \mathcal{V}ar(F)$; $\mathcal{V}oc^{\pm}(\mathsf{dnf}(F)) \subseteq \mathcal{V}oc^{\pm}(F)$.*

For prenex formulas F with an NNF matrix let $\mathsf{dual}(F)$ be the formula obtained from F by switching quantifiers $\forall$ and $\exists$, connectives $\land$ and $\lor$, truth-value constants $\top$ and $\bot$, and literals with their complement.

Proposition 5. *For all formulas F it holds that $\mathsf{cnf}(F) = \mathsf{dual}(\mathsf{dnf}(\neg F))$; $\mathsf{dnf}(F) = \mathsf{dual}(\mathsf{cnf}(\neg F))$; $\mathsf{cnf}(\neg F) = \mathsf{dual}(\mathsf{dnf}(F))$; $\mathsf{dnf}(\neg F) = \mathsf{dual}(\mathsf{cnf}(F))$.*

Proposition 6. *Let $F_1, F_2, \ldots, F_n$ be NNF formulas. Then* (i) *Each clause in $\mathsf{cnf}(\bigwedge_{i=1}^{n} F_i)$ is in some $\mathsf{cnf}(F_j)$.* (ii) *Each conjunctive clause in $\mathsf{dnf}(\bigvee_{i=1}^{n} F_i)$ is in some $\mathsf{dnf}(F_j)$.* (iii) *Formulas F_j that are literals are in each clause in $\mathsf{cnf}(\bigvee_{i=1}^{n} F_i)$.* (iv) *Formulas F_j that are literals are in each conjunctive clause in $\mathsf{dnf}(\bigwedge_{i=1}^{n} F_i)$.* (v) *If S is a set of variables such that for all $i \in \{1, \ldots, n\}$ and clauses C in $\mathsf{cnf}(F_i)$ it holds that $\mathcal{V}ar(C) \cap S \subseteq \mathcal{V}ar^{-}(C)$, then for all clauses C in $\mathsf{cnf}(\bigvee_{i=1}^{n} F_i)$ it holds that $\mathcal{V}ar(C) \cap S \subseteq \mathcal{V}ar^{-}(C)$.* (vi) *If S is a set of variables such that for all $i \in \{1, \ldots, n\}$ and conjunctive clauses D in $\mathsf{dnf}(F_i)$ it holds that $\mathcal{V}ar(D) \cap S \subseteq \mathcal{V}ar^{+}(D)$, then for all conjunctive clauses D in $\mathsf{dnf}(\bigwedge_{i=1}^{n} F_i)$ it holds that $\mathcal{V}ar(D) \cap S \subseteq \mathcal{V}ar^{+}(D)$.*

3.2 Used Notions of Range-Restriction

The following definition renders the characteristics of the range-restricted formulas as considered by Van Gelder and Topor in [59, Theorem 7.2] (except for the special consideration of equality in [59]).

Definition 7. A formula F with free variables $\mathcal{X}$ is called *VGT-range-restricted* if $\mathsf{cnf}(F) = Q\, M_{\mathsf{C}}$ and $\mathsf{dnf}(F) = Q\, M_{\mathsf{D}}$, where Q is a quantifier prefix (the same in both formulas) upon universally quantified variables $\mathcal{U}$ and existentially quantified variables $\mathcal{E}$ (in arbitrary order), and M_{C}, M_{D} are quantifier-free formulas in CNF and DNF, respectively, such that

1. For all clauses C in M_{C} it holds that $\mathcal{V}ar(C) \cap \mathcal{U} \subseteq \mathcal{V}ar^{-}(C)$.
2. For all conjunctive clauses D in M_{D} it holds that $\mathcal{V}ar(D) \cap \mathcal{E} \subseteq \mathcal{V}ar^{+}(D)$.
3. For all conjunctive clauses D in M_{D} it holds that $\mathcal{X} \subseteq \mathcal{V}ar^{+}(D)$.

For VGT-range-restricted formulas it is shown in [59] that these can be translated via two intermediate formula classes to a relational algebra expression. Related earlier results include [17,18,40,41]. The constraint on universal variables is also useful on its own as a weaker variation of range-restriction, defined as follows.

Definition 8. A formula F is called *U-range-restricted* if $\mathsf{cnf}(F) = Q\, M_{\mathsf{C}}$ where Q is a quantifier prefix upon of the universally quantified variables $\mathcal{U}$ (there may also be existentially quantified variables in Q) and M_{C} is a quantifier-free formula in CNF such that for all clauses C in M_{C} it holds that $\mathcal{V}ar(C) \cap \mathcal{U} \subseteq \mathcal{V}ar^{-}(C)$.

For formulas without free variables, U-range-restriction and VGT-range-restriction are related as follows.

Proposition 9. *Let F be a sentence. Then* (i) *F is VGT-range-restricted iff F and $\neg F$ are both U-range-restricted.* (ii) *If F is universal (i.e., in prenex form with only universal quantifiers), then F is VGT-range-restricted iff F is U-range-restricted.* (iii) *If F is existential (i.e., in prenex form with only existential quantifiers), then F is VGT-range-restricted iff $\neg F$ is U-range-restricted.*

U-range-restriction covers well-known restrictions of knowledge bases and inputs of bottom-up calculi for first-order logic and fragments of it that are naturally represented by clausal formulas [3]. First-order representations of tuple-generating dependencies (TGDs) are VGT-range-restricted sentences: conjunctions of sentences of the form $\forall \mathcal{X}\mathcal{Y}\,(A(\mathcal{X}\mathcal{Y}) \rightarrow \exists \mathcal{Z}\, B(\mathcal{Y}\mathcal{Z}))$, where A is a possibly empty conjunction of relational atoms, B is a nonempty conjunction of relational atoms and the free variables of A and B are exactly those in the sequences $\mathcal{X}\mathcal{Y}$ and $\mathcal{Y}\mathcal{Z}$, respectively. Also certain generalizations, e.g., to disjunctive TGDs, where B is built up from atoms, $\land$ and $\lor$, are VGT-range-restricted.

3.3 Results on Range-Restricted Interpolation

The following theorem shows three variations for obtaining range-restricted interpolants from range-restricted inputs.

Theorem 10 (Interpolation and Range-Restriction). *Let F and G be formulas such that $F \models G$.*

(i) *If F is U-range-restricted, then there exists a U-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*
(ii) *If F and G are sentences such that F and $\neg G$ are U-range-restricted, then there exists a VGT-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*
(iii) *If F and $\neg G$ are U-range-restricted, $\mathcal{V}ar(F) = \mathcal{V}ar(G) = \mathcal{X}$, and (1) no clause in $\mathsf{cnf}(F)$ has only negative literals; (2) for all clauses C in $\mathsf{cnf}(\neg G)$ with only negative literals it holds that $\mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$; (3) for all clauses C in $\mathsf{cnf}(\neg G)$ it holds that $\mathcal{V}ar(C) \cap \mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$, then there exists a VGT-range-restricted Craig-Lyndon interpolant H of F and G. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

Observe that Theorem 10.i requires range-restriction only for F, the first of the two interpolation arguments. Theorem 10.iii aims at applications for query reformulation that in a basic form are expressed as interpolation task for input formulas $F = K \land Q(\mathcal{X})$ and $G = \neg K' \lor Q'(\mathcal{X})$. Here K expresses background knowledge and constraints as a U-range-restricted sentence and $Q(\mathcal{X})$ represents a query to be reformulated, with free variables $\mathcal{X}$. Formulas K' and Q' are copies of K and Q, respectively, where predicates not allowed in the interpolant are replaced by primed versions. If the query Q is Boolean, i.e., $\mathcal{X}$ is empty, and Q is VGT-range-restricted, then Theorem 10.ii already suffices to justify the construction of a VGT-range-restricted interpolant. If $\mathcal{X}$ is not empty, the fine-print preconditions of Theorem 10.iii come into play. Precondition (1) requires that $\mathsf{cnf}(K)$ does not have a clause with only negative literals, which is satisfied if K represents TGDs. Also $\mathsf{cnf}(Q)$ is not allowed to have a clause with only negative literals. By precondition (2) all the free variables $\mathcal{X}$ must occur in all those clauses of $\mathsf{cnf}(\neg Q)$ that only have negative literals, which follows if Q meets condition (3.) of the VGT-range-restriction (Definition 7). By precondition (3) for all clauses C in $\mathsf{cnf}(\neg Q)$ it must hold that $\mathcal{V}ar(C) \cap \mathcal{X} \subseteq \mathcal{V}ar^{-}(C)$. A sufficient condition for Q to meet all these preconditions is that $\mathsf{dnf}(Q)$ has a purely existential quantifier prefix and a matrix with only positive literals where each query variable, i.e., member of $\mathcal{X}$, occurs in each conjunctive clause.

3.4 Proving Range-Restricted Interpolation – The Hyper Property

We will prove Theorem 10 by showing how the claimed interpolants can be obtained with CTIF. As a preparatory step we match items from the specification of CTIF (Fig. 2) with the constraints of range-restriction. The following notion gathers intermediate formulas and sets of symbols of CTIF.

Definition 11. An *interpolation context* is a tuple $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$, where F, G are formulas, F', G' are clausal formulas, $\mathcal{C}$ is a set of constants, $\mathcal{F}, \mathcal{G}$ are sets of functions, and $\mathcal{E}, \mathcal{U}, \mathcal{V}$ are sets of terms such that the following holds. (i) $F \models G$. (ii) Let F_c and G_c be F and G after replacing each free variable with a dedicated fresh constant. Let $\mathcal{C}$ be those constants that were used there to replace a variable that occurs in both F and G. F' and G' are the matrices of $\mathsf{cnf}(F_c)$ and of $\mathsf{cnf}(\neg G_c)$, after replacing existentially quantified variables with Skolem terms. (iii) $\mathcal{F}$ is the union of the set of the Skolem functions introduced for existential quantifiers of $\mathsf{cnf}(F_c)$, the set of functions occurring in F_c but not in G_c and, possibly, further functions freshly introduced in the grounding step of CTIF. Analogously, $\mathcal{G}$ is the union of the set of the Skolem functions introduced for $\mathsf{cnf}(\neg G_c)$, the set of functions occurring in G_c but not in F_c, and, possibly, further functions introduced in grounding. (iv) $\mathcal{E}$ and $\mathcal{U}$ are the sets of all terms with outermost function symbol in $\mathcal{F}$ and $\mathcal{G}$, respectively. (v) $\mathcal{V}$ is $\mathcal{E} \cup \mathcal{U} \cup \mathcal{C}$.

The following statements about an interpolation context are easy to infer.

Lemma 12. *Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ be an interpolation context. Then* (i) *No member of $\mathcal{G}$ occurs in F'.* (ii) *No member of $\mathcal{F}$ occurs in G'.* (iii) *If F is U-range-restricted, then for all clauses C in F' it holds that if a variable occurs in C in a position that is not within an $\mathcal{E}$-term it occurs in C in a negative literal, in a position that is not within an $\mathcal{E}$-term.* (iv) *If $\neg G$ is U-range-restricted, then for all clauses C in G' it holds that if a variable occurs in C in a position that is not within an $\mathcal{U}$-term, it occurs in C in a negative literal, in a position that is* *not within an $\mathcal{U}$-term.* (v) *If G satisfies condition (3) of Theorem 10.iii, then for all clauses C in G' it holds that any member of $\mathcal{C}$ that occurs in C in a position that is not within an $\mathcal{U}$-term occurs in C in a negative literal in a position that is not within an $\mathcal{U}$-term.*

CTIF involves conversion of terms to variables at lifting (step 7) and at replacing placeholder constants (step 8). We introduce a notation to identify those terms that will be converted there to variables. It mimics the notation for the set of free variables of a formula but applies to a set of terms, those with occurrences that are "maximal" with respect to a given set S of terms, i.e., are not within another term from S. For NNF formulas F define $S\text{-}\mathcal{M}ax(F)$ as the set of S-terms that occur in F in a position other than as subterm of another S-term. Define $S\text{-}\mathcal{M}ax^{+}(F)$ ($S\text{-}\mathcal{M}ax^{-}(F)$, respectively) as the set of S-terms that occur in F in a positive (negative, respectively) literal in a position other than as subterm of another S-term. We can now conclude from Lemma 12 the following properties of instances of clauses used for interpolant construction.

Lemma 13. *Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ be an interpolation context. Then*

(i) *If F is U-range-restricted, then for all instances C of a clause in F' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$.*
(ii) *If $\neg G$ is U-range-restricted, then for all instances C of a clause in G' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$.*
(iii) *If condition (1) of Theorem 10.iii holds, then no instance C of a clause in F' has only negative literals.*
(iv) *If condition (2) of Theorem 10.iii holds, then for all instances C of a clause in G' with only negative literals it holds that $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$.*
(v) *If $\neg G$ is U-range-restricted and condition (3) of Theorem 10.iii holds, then for all instances C of a clause in G' it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$.*

The following proposition adapts Props. 6.v and 6.vi to $S\text{-}\mathcal{M}ax$.

Proposition 14. *Let $F_1, F_2, \ldots, F_n$ be NNF formulas and let T be a set of terms. Then* (i) *If S is a set of terms such that for all $i \in \{1, \ldots, n\}$ and clauses C in $\mathsf{cnf}(F_i)$ it holds that $T\text{-}\mathcal{M}ax(C) \cap S \subseteq T\text{-}\mathcal{M}ax^{-}(C)$, then for all clauses C in $\mathsf{cnf}(\bigvee_{i=1}^{n} F_i)$ it holds that $T\text{-}\mathcal{M}ax(C) \cap S \subseteq T\text{-}\mathcal{M}ax^{-}(C)$.* (ii) *If S is a set of terms such that for all $i \in \{1, \ldots, n\}$ and conjunctive clauses D in $\mathsf{dnf}(F_i)$ it holds that $T\text{-}\mathcal{M}ax(D) \cap S \subseteq T\text{-}\mathcal{M}ax^{+}(D)$, then for all conjunctive clauses D in $\mathsf{dnf}(\bigwedge_{i=1}^{n} F_i)$ it holds that $T\text{-}\mathcal{M}ax(D) \cap S \subseteq T\text{-}\mathcal{M}ax^{+}(D)$.*

The key to obtain range-restricted interpolants from CTIF is that the tableau must have a specific form, which we call *hyper*, as it resembles proofs by hyper-resolution [46] and hypertableaux [2].

Definition 15. A clausal tableau is called *hyper* if the nodes labeled with a negative literal are exactly the leaf nodes.

While hyperresolution and related approaches, e.g., [2,3,11,36,46], consider DAG-shaped proofs with non-rigid variables, aiming at interpolant extraction we consider the hyper property for tree-shaped proofs with rigid variables. The *hyper* requirement is w.l.o.g. because arbitrary closed clausal tableaux can be converted to tableaux with the hyper property, as we will see in Sect. 5.

The proof of Theorem 10 is based on three properties that invariantly hold for all nodes, or for all inner nodes, respectively, stated in the following lemma.

Lemma 16. *Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V\rangle$ be an interpolation context and assume a leaf-closed and hyper two-sided clausal ground tableau for F' and G'.*

(i) *If F is U-range-restricted, then for all nodes N the property $\mathsf{INV_C}(N)$ defined as follows holds: $\mathsf{INV_C}(N) \overset{\text{def}}{=}$ For all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N))$.*

(ii) *If $\neg G$ is U-range-restricted, then for all nodes N the property $\mathsf{INV_D}(N)$ defined as follows holds: $\mathsf{INV_D}(N) \overset{\text{def}}{=}$ For all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_G}(N))$.*

(iii) *If $\neg G$ is U-range-restricted and conditions (1)–(3) Theorem 10.iii hold, then for all inner nodes N the property $\mathsf{INV_X}(N)$ defined as follows holds: $\mathsf{INV_X}(N) \overset{\text{def}}{=}$ For all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N))$ it holds that $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(D) \cup \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_G}(N))$.*

Each of Lemma 16.i, 16.ii and 16.iii can be proven independently by an induction on the tableau structure, but for the same tableau, such that the properties claimed by them can be combined. In proving these three sub-lemmas it is sufficient to use their respective preconditions only to justify the application of matching sub-lemmas of Lemma 13. That lemma might thus be seen as an abstract interface that delivers everything that depends on these preconditions and is relevant for Theorem 10.

We show here the proof of Lemma 16.i. Lemma 16.ii can be proven in full analogy. The proof of Lemma 16.iii is deferred to [63, App. A]. In general, recall that the tableau in Lemma 16 is a two-sided tableau for F' and G' that is leaf-closed and hyper. Hence literal labels of leaves are negative, while those of inner nodes are positive. All tableau clauses are ground and with an associated *side* in $\{\mathsf{F}, \mathsf{G}\}$ such that a tableau clause with side F is an instance of a clause in F' and one with side G is an instance of a clause in G'.

Proof (Lemma 16.i). By induction on the tableau structure.

Base case where N is a leaf. If N and $\mathsf{tgt}(N)$ have the same side, then $\mathsf{ipol}(N)$ is a truth value constant, hence $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) = \emptyset$, implying $\mathsf{INV_C}(N)$. If N has side F and $\mathsf{tgt}(N)$ has side G, then $\mathsf{ipol}(N) = \mathsf{lit}(N)$, which, because N is a leaf, is a negative literal. Thus $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) = \mathcal{V}\text{-}\mathcal{M}ax^{-}(\mathsf{ipol}(N))$, which implies $\mathsf{INV_C}(N)$. If N has side G and $\mathsf{tgt}(N)$ has side F, then $\mathsf{ipol}(N) = \mathsf{lit}(\mathsf{tgt}(N))$, which, because N is a leaf, is a positive literal. Thus $\mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N)) \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N))$, implying $\mathsf{INV_C}(N)$.

Induction Step. Let $N_1, \ldots, N_n$, where $1 \leq n$, be the children of N. Assume as induction hypothesis that for $i \in \{1, \ldots, n\}$ it holds that $\mathsf{INV_C}(N_i)$. Consider the case where the side of the children is F. Then

(1) $\mathsf{ipol}(N) = \bigvee_{i=1}^{n} \mathsf{ipol}(N_i)$.

Assume that $\mathsf{INV_C}(N)$ does not hold. Then there exists a clause K in $\mathsf{cnf}(\mathsf{ipol}(N))$ and a term t such that (2) $t \in \mathcal{U}$; (3) $t \in \mathcal{V}\text{-}\mathcal{M}ax(K)$; (4) $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{-}(K)$; (5) $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N))$. To derive a contradiction, we first show that given (2), (4) and (5) it holds that

(6) For all children N' of N: $t \notin \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N'))$.

Statement (6) can be proven as follows. Assume to the contrary that there is a child N' of N such that $t \in \mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N'))$. By (5) it follows that $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{lit}(N'))$ and $\mathsf{lit}(N')$ is positive. By Lemma 13.i and (2) there is another child N'' of N such that $\mathsf{lit}(N'')$ is negative and $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{lit}(N''))$. Since the tableau is closed, it follows from (5) that $\mathsf{tgt}(N'')$ has side G, which implies that $\mathsf{ipol}(N'') = \mathsf{lit}(N'')$. Hence $t \in \mathcal{V}\text{-}\mathcal{M}ax(\mathsf{ipol}(N''))$. Since $\mathsf{ipol}(N'')$ is a negative literal and a disjunct of $\mathsf{ipol}(N)$, it follows from (1) and Prop. 6.iii that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N))$ it holds that $t \in \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$, contradicting assumption (4). Hence (6) must hold.

From (6), (2) and the induction hypothesis it follows that for all children N' of N and clauses C' in $\mathsf{cnf}(\mathsf{ipol}(N'))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C') \cap \{t\} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C')$. Hence, by (1) and Prop. 14.i it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \{t\} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^{-}(C)$. This, however, contradicts our assumption of the existence of a clause K in $\mathsf{cnf}(\mathsf{ipol}(N))$ that satisfies (3) and (4). Hence $\mathsf{INV_C}(N)$ must hold.

We conclude the proof of the induction step for $\mathsf{INV_C}(N)$ by considering the case where the side of the children of N is G. Then

(7) $\mathsf{ipol}(N) = \bigwedge_{i=1}^{n} \mathsf{ipol}(N_i)$.
(8) For all children N' of N: $\mathsf{path_F}(N) = \mathsf{path_F}(N')$.

$\mathsf{INV_C}(N)$ follows from the induction hypothesis, (8), (7) and Prop. 6.i. □

The invariant properties of tableau nodes shown in Lemmas 16.i–16.iii apply in particular to the tableau root. We now apply this to prove Theorem 10.

Proof (Theorem 10). Interpolants with the stated properties are obtained with CTIF, assuming w.l.o.g. that the CNF computed in step 2 meets the requirement of Sect. 3.1, and that the closed clausal tableau computed in step 3 is leaf-closed and has the hyper property. That CTIF constructs a Craig-Lyndon interpolant has been shown in [62]. It remains to show the further claimed properties of the interpolant. Let $\langle F, G, F', G', \mathcal{F}, \mathcal{G}, \mathcal{E}, \mathcal{U}, \mathcal{C}, V \rangle$ be the interpolation context for the input formulas F and G and let N_0 be the root of the tableau computed in step 3. Since N_0 is the root, $\mathsf{path_F}(N_0) = \mathsf{path_G}(N_0) = \top$ and thus the expressions $\mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_F}(N_0))$ and $\mathcal{V}\text{-}\mathcal{M}ax^{+}(\mathsf{path_G}(N_0))$ in the specifications of $\mathsf{INV_C}(N_0)$, $\mathsf{INV_D}(N_0)$ and $\mathsf{INV_X}(N_0)$ all denote the empty set. The claims made in the particular sub-theorems can then be shown as follows.

(10.i) By Lemma 16.i it follows that $\mathsf{INV}_\mathsf{C}(N_0)$. Hence, for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$. It follows that the result of the interpolant lifting (step 7) of CTIF applied to $\mathsf{ipol}(N_0)$ is U-range-restricted. Placeholder constant replacement (step 8) does not alter this.

(10.ii) As for Theorem 10.i it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$. By Lemma 16.ii it follows that $\mathsf{INV}_\mathsf{D}(N_0)$. Hence, for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^+(D)$. It follows that the result of the interpolant lifting of CTIF applied to $\mathsf{ipol}(N_0)$ is U-range-restricted. Since F and G have no free variables, placeholder constant replacement has no effect.

(10.iii) As for Theorem 10.ii it follows that for all clauses C in $\mathsf{cnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(C) \cap \mathcal{U} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^-(C)$ and for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{V}\text{-}\mathcal{M}ax(D) \cap \mathcal{E} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^+(D)$. By Lemma 16.iii it follows that $\mathsf{INV}_\mathsf{X}(N_0)$. Hence, for all conjunctive clauses D in $\mathsf{dnf}(\mathsf{ipol}(N_0))$ it holds that $\mathcal{C} \subseteq \mathcal{V}\text{-}\mathcal{M}ax^+(D)$. It follows that the result of the interpolant lifting of CTIF applied to $\mathsf{ipol}(N_0)$ followed by placeholder constant replacement, now applied to $\mathcal{C}$, is VGT-range-restricted. ◻

4 Horn Interpolation

A *Horn clause* is a clause with at most one positive literal. A *Horn formula* is built up from Horn clauses with the connectives $\wedge$, $\exists$ and $\forall$. Horn formulas are important in countless theoretical and practical respects. Our interpolation method on the basis of clausal tableaux with the hyper property can be applied to obtain a Horn interpolant under the precondition that the first argument formula F of the interpolation problem is Horn. The following theorem makes this precise. It can be proven by an induction on the structure of a clausal tableau with the hyper property (see [63, App. B]).

Theorem 17 (Interpolation from a Horn Formula). *Let F be a Horn formula and let G be a formula such that $F \models G$. Then there exists a Craig-Lyndon interpolant H of F and G that is a Horn formula. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

An apparently weaker property than Theorem 17 has been shown in [38, § 4] with techniques from model theory: For *two* universal Horn formulas F and G there exists a universal Horn formula that is like a Craig interpolant, except that function symbols are not constrained. A *universal* Horn formula is there a prenex formula with only universal quantifiers and a Horn matrix. For CTIF, the corresponding strengthening of the interpolant to a universal formula can be read-off from the specification of interpolant lifting (step 7 in Fig. 2).

The following corollary shows that Theorem 17 can be combined with Theorem 10 to obtain interpolants that are both Horn and range-restricted.

Corollary 18 (Range-Restricted Horn Interpolants). *Theorems 10.i, 10.ii and 10.iii can be strengthened: If F is a Horn formula, then there exists* *a Craig-Lyndon interpolant H with the properties shown in the respective theorem and the additional property that it is Horn. Moreover, H can be effectively constructed from a clausal tableau proof of $F \models G$.*

Proof. Can be shown by combining the proof of Theorem 10.i, 10.ii and 10.iii , respectively, with the proof of interpolation from a Horn sentence, Theorem 17. The combined proofs are based on inductions on the same closed tableau with the hyper property. □

5 Obtaining Proofs with the Hyper Property

Our new interpolation theorems, Theorems 10 and 17, depend on the hyper property of the underlying closed clausal tableaux from which interpolants are extracted. We present a proof transformation that converts any closed clausal tableau to one with the hyper property. The transformation can be applied to a clausal tableau as obtained directly from a clausal tableaux prover. Moreover, it can be also be indirectly applied to a resolution proof. To this end, the resolution deduction *tree* [12] of the binary resolution proof is first translated to a closed clausal ground tableau in *cut normal form* [31, Sect. 7.22]. There the inner clauses are atomic cuts, tautologies of the form $\neg p(t_1, \ldots, t_n) \lor p(t_1, \ldots, t_n)$ or $p(t_1, \ldots, t_n) \lor \neg p(t_1, \ldots, t_n)$, corresponding to literals upon which a (tree) resolution step has been performed. Clauses of nodes whose children are leaves are instances of input clauses. Our hyper conversion can then be applied to the tableau in cut normal form. It is easy to see that a regular leaf-closed tableau with the hyper property can not have atomic cuts. Hence the conversion might be viewed as an elimination method for these cuts.

We specify the hyper conversion in Fig. 3 as a procedure that destructively manipulates a tableau. A *fresh copy* of an ordered tree T is there an ordered tree T' with fresh nodes and edges, related to T through a bijection c such that any node N of T has the same labels (literal label and side label) as node $c(N)$ of T' and such that the i-th edge originating in node N of T ends in node M if and only if the i-th edge originating in node $c(N)$ of T' ends in node $c(M)$. The procedure is performed as an iteration that in each round chooses an inner node with negative literal label and then modifies the tableau. Hence, at termination there is no inner node with negative literal, which means that the tableau is hyper. Termination of the procedure can be shown with a measure that strictly decreases in each round (Prop. 20 in [63, App. C]). Figures 4 and 5 show example applications of the procedure.

Since the hyper conversion procedure copies parts of subtrees it is not a polynomial operation.[4] To get an idea of its practical feasibility, we experimented with an unbiased set of proofs of miscellaneous problems. For this we took those 112 *CASC-J11* [54] problems that could be proven with Prover9 [37] in 400 s per

[4] A thorough complexity analysis should take calculus- or strategy-dependent properties of the input proofs into account. And possibly also the blow-up from resolution to tree resolution underlying the cut normal form tableaux.

INPUT: A closed clausal tableau.

METHOD: Simplify the tableau to leaf-closing and regular form (Sect. 2.2). Repeat the following operations until the resulting tableau is hyper.

1. Let N' be the first node visited in pre-order with a child that is an inner node with a negative literal label. Let N be the leftmost such child.
2. Create a fresh copy U of the subtree rooted at N'. In U remove the edges that originate in the node corresponding to N.
3. Replace the edges originating in N' with the edges originating in N.
4. For each leaf descendant M of N' with $\mathsf{lit}(M) = \overline{\mathsf{lit}(N)}$: Create a fresh copy U' of U. Change the origin of the edges originating in the root of U' to M.
5. Simplify the tableau to leaf-closing and regular form (Sect. 2.2).

OUTPUT: A leaf-closed, regular and hyper clausal tableau whose clauses are clauses of the input tableau.

Fig. 3. The *hyper conversion* proof transformation procedure.

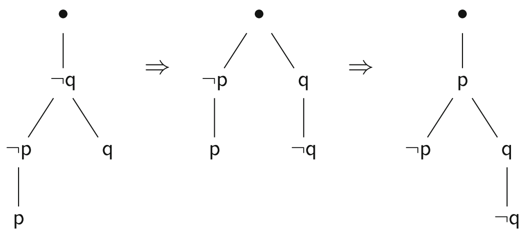


Fig. 4. Hyper conversion of a closed clausal tableau in two rounds.

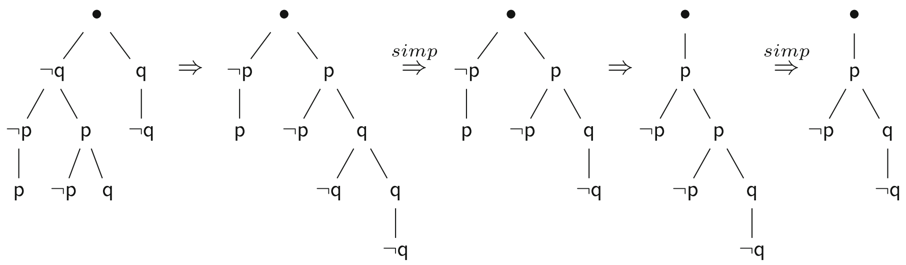


Fig. 5. Hyper conversion of a closed clausal tableau in cut normal form in two rounds. For each round the result after procedure steps 1–4 is shown and then the result after step 5, simplification, applied here to achieve regularity.

problem, including a basic proof conversion with Prover9's tool Prooftrans.[5] The hyper conversion succeeded on 107 (or 96%) of these, given 400 s timeout per proof, where the actual median of used time was only 0.01 s. It was applied to a tableau in cut normal form that represents the proof tree of Prover9's proof. The two intermediate steps, translation of paramodulation to binary resolution and expansion to cut normal form, succeeded in fractions of a second, except for one case where the expansion took 121 s and two cases where it failed due to memory exhaustion. The hyper conversion then failed in three further cases. For all except two proofs the hyper conversion reduced the proof size, where the overall median of the size ratio hyper-to-input was 0.39. See [63, App. D] for details.

6 Conclusion

We conclude with discussing related work, open issues and perspectives. Our interpolation method CTIF [62] is complete for first-order logic with function symbols. Vampire's native interpolation [22,23], targeted at verification, is like all local methods incomplete [28]. Princess [10,47] implements interpolation with a sequent calculus that supports theories for verification and permits uninterpreted predicates and functions. Suitable proofs for our approach can currently be obtained from CMProver (clausal tableaux) and Prover9 (resolution/paramodulation). With optimized settings, Vampire [27] and E [49] as of today only output proofs with gaps. This seems to improve [48] or might be overcome by re-proving with Prover9 using lemmas from the more powerful systems.

So far we did not address special handling of equality in the context of range-restriction, a topic on its own, e.g., [3,59]. We treat it as predicate, with axioms for reflexivity, symmetry, transitivity and substitutivity. CTIF works smoothly with these, respecting polarity constraints of equality in interpolants [62, Sect. 10.4]. With exception of reflexivity these axioms are U-range-restricted. We do not interfere with the provers' equality handling and just translate in finished proofs paramodulation into binary resolution with substitutivity axioms.

The potential bottleneck of conversion to clausal form in CTIF may be remedied with structure-preserving (aka *definitional*) normal forms [19,44,50,58].

Our *hyper* property might be of interest for proof presentation and exchange, since it gives the proof tree a constrained shape and in experiments often shortens it. Like hyperresolution and hypertableaux it can be generalized to take a "semantics" into account [51] [12, Chap. 6] [26, Sect. 4.5]. To shorten interpolants, it might be combined with proof reductions (e.g., [64]).

For query reformulation, interpolation on the basis of general first-order ATP was so far hardly considered. Most methods are sequent calculi [6,56] or analytic tableaux systems [5,21,25,57]. Experiments with ATP systems and propositional inputs indicate that requirements are quite different from those

[5] On a Linux notebook with 12th Gen Intel® Core™ i7-1260P CPU and 32 GB RAM.

in verification [4]. An implemented system [25,57] uses analytic tableaux with dedicated refinements for enumerating alternate proofs/interpolants corresponding to query plans for heuristic choice. In [5] the focus is on interpolants that are sentences respecting binding patterns, which, like range-restriction, ensures database evaluability. Our interpolation theorems show fine-grained conditions for passing variations of range-restriction and the Horn property on to interpolants. Matching these with the many formula classes considered in knowledge representation and databases is an issue for future work. A further open topic is adapting recent synthesis techniques for nested relations [6] to the clausal tableaux proof system.

Methodically, we exemplified a way to approach operations on proof structures while taking efficient automated first-order provers into account. Feasible implementations are brought within reach, for practical application and also for validating abstract claims and conjectures with scrutiny. The prover is a black box, given freedom on optimizations, strategy and even calculus. For interfacing, the overall setting incorporates clausification and Skolemization. Requirements on the proof structure do not hamper proof search, but are ensured by transformations applied to proofs returned by the efficient systems.

Acknowledgments. The author thanks Michael Benedikt for bringing the subtleties of range-restriction in databases to attention, Cécilia Pradic for insights into subtleties of proof theory, and anonymous reviewers for helpful suggestions to improve the presentation.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley, Boston (1995)
2. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. In: Alferes, J.J., Pereira, L.M., Orlowska, E. (eds.) JELIA 1996. LNCS, vol. 1126, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61630-6_1
3. Baumgartner, P., Schmidt, R.A.: Blocking and other enhancements for bottom-up model generation methods. J. Autom. Reasoning **64**, 197–251 (2020). https://doi.org/10.1007/11814771_11
4. Benedikt, M., Kostylev, E.V., Mogavero, F., Tsamoura, E.: Reformulating queries: theory and practice. In: Sierra, C. (ed.) IJCAI 2017, pp. 837–843. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/116
5. Benedikt, M., Leblay, J., ten Cate, B., Tsamoura, E.: Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation. Morgan & Claypool, San Rafael (2016). https://doi.org/10.1007/978-3-031-01856-5
6. Benedikt, M., Pradic, C., Wernhard, C.: Synthesizing nested relational queries from implicit specifications. In: PODS '23, pp. 33–45 (2023). https://doi.org/10.1145/3584372.3588653
7. Bibel, W.: Automated Theorem Proving, 2nd edn. Vieweg, Braunschweig (1987). https://doi.org/10.1007/978-3-322-90102-6. First edition 1982
8. Bibel, W., Otten, J.: From Schütte's formal systems to modern automated deduction. In: The Legacy of Kurt Schütte, pp. 217–251. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49424-7_13

9. Bonacina, M.P., Johansson, M.: On interpolation in automated theorem proving. J. Autom. Reasoning **54**(1), 69–97 (2014). https://doi.org/10.1007/s10817-014-9314-0
10. Brillout, A., Kroening, D., Rümmer, P., Wahl, T.: Beyond quantifier-free interpolation in extensions of Presburger arithmetic. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 88–102. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18275-4_8
11. Bry, F., Yahya, A.H.: Positive unit hyperresolution tableaux and their application to minimal model generation. J. Autom. Reasoning **25**(1), 35–82 (2000). https://doi.org/10.1023/A:1006291616338
12. Chang, C.L., Lee, R.C.T.: Symbolic Logic and Automated Theorem Proving. Academic Press, Cambridge (1973)
13. Craig, W.: Linear reasoning. A new form of the Herbrand-Gentzen theorem. J. Symb. Log. **22**(3), 250–268 (1957). https://doi.org/10.2307/2963593
14. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. J. Symb. Log. **22**(3), 269–285 (1957). https://doi.org/10.2307/2963594
15. Craig, W.: The road to two theorems of logic. Synthese **164**(3), 333–339 (2008). https://doi.org/10.1007/s11229-008-9353-3
16. Dahn, I., Wernhard, C.: First order proof problems extracted from an article in the Mizar mathematical library. In: Bonacina, M.P., Furbach, U. (eds.) FTP'97, pp. 58–62. RISC-Linz Report Series No. 97–50, Joh. Kepler Univ., Linz (1997). https://www.logic.at/ftp97/papers/dahn.pdf
17. Demolombe, R.: Syntactical characterization of a subset of domain independent formulas. Technical report, ONERA-CERT, Toulouse (1982)
18. Demolombe, R.: Syntactical characterization of a subset of domain independent formulas. JACM **39**, 71–94 (1992). https://doi.org/10.1145/147508.147520
19. Eder, E.: An implementation of a theorem prover based on the connection method. In: Bibel, W., Petkoff, B. (eds.) AIMSA'84, pp. 121–128. North-Holland (1985)
20. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Springer, Cham (1995). https://doi.org/10.1007/978-1-4612-2360-3
21. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation over databases with first-order and description logics ontologies. JAIR **48**, 885–922 (2013). https://doi.org/10.1613/jair.4058
22. Hoder, K., Holzer, A., Kovács, L., Voronkov, A.: Vinter: a Vampire-based tool for interpolation. In: Jhala, R., Igarashi, A. (eds.) APLAS 2012. LNCS, vol. 7705, pp. 148–156. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35182-2_11
23. Hoder, K., Kovács, L., Voronkov, A.: Interpolation and symbol elimination in Vampire. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 188–195. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14203-1_16
24. Huang, G.: Constructing Craig interpolation formulas. In: Du, D.-Z., Li, M. (eds.) COCOON 1995. LNCS, vol. 959, pp. 181–190. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0030832
25. Hudek, A., Toman, D., Weddell, G.: On enumerating query plans using analytic tableau. In: De Nivelle, H. (ed.) TABLEAUX 2015. LNCS (LNAI), vol. 9323, pp. 339–354. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24312-2_23
26. Hähnle, R.: Tableaux and related methods. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chap. 3, pp. 101–178. Elsevier (2001). https://doi.org/10.1016/b978-044450813-3/50005-9

27. Kovács, L., Voronkov, A.: First-order theorem proving and VAMPIRE. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
28. Kovács, L., Voronkov, A.: First-order interpolation and interpolating proof systems. In: Eiter, T., Sands, D. (eds.) LPAR-21. EPiC, vol. 46, pp. 49–64. EasyChair (2017). https://doi.org/10.29007/1qb8
29. Letz, R.: Clausal tableaux. In: Bibel, W., Schmitt, P.H. (eds.) Automated Deduction - A Basis for Applications, vol. I, pp. 43–72. Kluwer Academic Publishers (1998)
30. Letz, R.: First-order tableau methods. In: D'Agostino, A., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 125–196. Springer, Dordrecht (1999)
31. Letz, R.: Tableau and Connection Calculi. Structure, Complexity, Implementation. Habilitationsschrift, TU München (1999). http://www2.tcs.ifi.lmu.de/~letz/habil.pdf. Accessed 19 July 2023
32. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: a high-performance theorem prover. J. Autom. Reasoning **8**(2), 183–212 (1992). https://doi.org/10.1007/BF00244282
33. Letz, R., Stenz, G.: Model elimination and connection tableau procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, pp. 2015–2114. Elsevier (2001)
34. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. North-Holland, Amsterdam (1978)
35. Lyndon, R.: An interpolation theorem in the predicate calculus. Pac. J. Math. **9**, 129–142 (1959). https://doi.org/10.2140/pjm.1959.9.129
36. Manthey, R., Bry, F.: SATCHMO: A theorem prover implemented in Prolog. In: Lusk, E., Overbeek, R. (eds.) CADE 1988. LNCS, vol. 310, pp. 415–434. Springer, Heidelberg (1988). https://doi.org/10.1007/BFb0012847
37. McCune, W.: Prover9 and Mace4 (2005–2010). http://www.cs.unm.edu/~mccune/prover9. Accessed 19 July 2023
38. McNulty, G.F.: Fragments of first order logic, I: universal Horn logic. J. Symb. Log. **42**(2), 221–237 (1977). https://doi.org/10.2307/2272123
39. Nash, A., Segoufin, L., Vianu, V.: Views and queries: determinacy and rewriting. ACM Trans. Database Syst. **35**(3), 1–41 (2010). https://doi.org/10.1145/1806907.1806913
40. Nicolas, J.M.: Logics for improving integrity checking in relational data bases. Technical report, ONERA-CERT, Toulouse (1979)
41. Nicolas, J.M.: Logics for improving integrity checking in relational data bases. Acta Informatica **18**(3), 227–253 (1982). https://doi.org/10.1007/BF00263192
42. Otten, J.: Restricting backtracking in connection calculi. AI Commun. **23**(2–3), 159–182 (2010). https://doi.org/10.3233/AIC-2010-0464
43. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. J. Symb. Comput. **36**(1–2), 139–161 (2003). https://doi.org/10.1016/S0747-7171(03)00037-3
44. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. **2**, 293–304 (1986). https://doi.org/10.1016/S0747-7171(86)80028-1
45. Rawson, M., Wernhard, C., Zombori, Z., Bibel, W.: Lemmas: generation, selection, application. In: Ramanayake, R., Urban, J. (eds.) TABLEAUX 2023. LNCS (LNAI), vol. 14278, pp. 153–174. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-43513-3_9

46. Robinson, J.A.: Automatic deduction with hyper-resolution. Int. J. Comput. Math. **1**(3), 227–234 (1965)
47. Rümmer, P.: A constraint sequent calculus for first-order logic with linear integer arithmetic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 274–289. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_20
48. Schulz, S.: Credo Quia absurdum (?) – proof generation and challenges of proof generation. In: PAMLTP/DG4D^3 (2023), workshop presentation. https://europroofnet.github.io/_pages/WG5/Prague23/pres/Schulz.pdf
49. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
50. Scott, D.: A decision method for validity of sentences in two variables. J. Symb. Log. **27**(4), 477 (1962)
51. Slagle, J.R.: Automatic theorem proving with renamable and semantic resolution. JACM **14**(4), 687–697 (1967). https://doi.org/10.1145/321420.321428
52. Smullyan, R.M.: First-Order Logic. Springer, New York (1968). also republished with corrections by Dover publications (1995)
53. Stickel, M.E.: A Prolog technology theorem prover: implementation by an extended Prolog compiler. J. Autom. Reasoning **4**(4), 353–380 (1988). https://doi.org/10.1007/BF00297245
54. Sutcliffe, G., Desharnais, M.: The 11th IJCAR automated theorem proving system competition - CASC-J11. AI Commun. (2023). https://doi.org/10.3233/AIC-220244
55. Takeuti, G.: Proof Theory, second edn. North-Holland (1987)
56. Toman, D., Weddell, G.: Fundamentals of Physical Design and Query Compilation. Morgan & Claypool, San Rafael (2011). https://doi.org/10.1007/978-3-031-01881-7
57. Toman, D., Weddell, G.: An interpolation-based compiler and optimizer for relational queries (system design report). In: Eiter, T., Sands, D., Sutcliffe, G., Voronkov, A. (eds.) IWIL 2017 Workshop and LPAR-21 Short Presentations. Kalpa, vol. 1. EasyChair (2017). https://doi.org/10.29007/53fk
58. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Slisenko, A.O. (ed.) Studies in Constructive Mathematics and Mathematical Logic, vol. Part II, pp. 115–125. Steklov Mathematical Institute (1970)
59. Van Gelder, A., Topor, R.W.: Safety and translation of relational calculus queries. ACM Trans. Database Syst. **16**(2), 235–278 (1991). https://doi.org/10.1145/114325.103712
60. Wernhard, C.: The PIE system for proving, interpolating and eliminating. In: Fontaine, P., Schulz, S., Urban, J. (eds.) PAAR 2016. CEUR Workshop Proc., vol. 1635, pp. 125–138. CEUR-WS.org (2016). http://ceur-ws.org/Vol-1635/paper-11.pdf
61. Wernhard, C.: Facets of the *PIE* environment for proving, interpolating and eliminating on the basis of first-order logic. In: Hofstedt, P., Abreu, S., John, U., Kuchen, H., Seipel, D. (eds.) INAP/WLP/WFLP -2019. LNCS (LNAI), vol. 12057, pp. 160–177. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46714-2_11
62. Wernhard, C.: Craig interpolation with clausal first-order tableaux. J. Autom. Reasoning **65**(5), 647–690 (2021). https://doi.org/10.1007/s10817-021-09590-3
63. Wernhard, C.: Range-restricted and Horn interpolation through clausal tableaux. CoRR abs/2306.03572 (2023). https://doi.org/10.48550/arXiv.2306.03572

64. Wernhard, C., Bibel, W.: Learning from Łukasiewicz and Meredith: investigations into proof structures. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 58–75. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_4

Non-Classical Logics in Satisfiability Modulo Theories

Clemens Eisenhofer[1](✉), Ruba Alassaf[2], Michael Rawson[1], and Laura Kovács[1]

[1] TU Wien, Vienna, Austria
{clemens.eisenhofer,michael.rawson,laura.kovacs}@tuwien.ac.at
[2] University of Manchester, Manchester, UK
ruba.alassaf@manchester.ac.uk

Abstract. We show that tableau methods for satisfiability in non-classical logics can be supported naturally in SMT solving via the framework of user-propagators. By way of demonstration, we implement the description logic $\mathcal{ALC}$ in the Z3 SMT solver and show that working with user-propagators allows us to significantly outperform encodings to first-order logic with relatively little effort. We promote user-propagators for creating solvers for non-classical logics based on tableau calculi.

Keywords: SMT · Non-Classical Logics · User-Propagators · Tableaux

1 Introduction

Satisfiability modulo theory (SMT) solvers, e.g. [4,14,29], mostly implement CDCL($\mathcal{T}$) [6,27] to combine propositional satisfiability (SAT) solving with theory-specific decision procedures. Due to the modular nature of the underlying CDCL($\mathcal{T}$) algorithm, not only can SMT solvers reason in combinations of theories, but it is even possible to add and control custom first-order theories by attaching new decision procedures, as recently introduced in the user-propagator framework [8]. The underlying logic in the SMT solving community is classical first-order logic. When moving towards non-classical logics, such as modal or description logics [2,9,21], tableau calculi provide common ground [13]. The resulting proof procedures behave very differently to SMT solvers [16,22].

In this paper, we argue that *it is time to join forces*. We show that tableau methods can be integrated naturally into SMT solving (Sect. 3). In so doing, we promote user-propagators [8] for guiding non-classical reasoning within SMT solving. We demonstrate our work within the Z3 SMT solver [29] and show that this approach outperforms two standard Z3 implementations based on quantification (Sect. 4). Finally, we discuss an alternative encoding for non-boolean based logics capable of dealing with explicit non-containment (Sect. 5).

We thank Nikolaj Bjørner for discussions on this topic. We acknowledge funding from the ERC Consolidator Grant ARTIST 101002685, the TU Wien SecInt Doctoral College, and the FWF SFB project SpyCoDe F8504.

Related Work. SAT/SMT solving driven by instantiation rules from modal and description logic tableaux have been investigated [1,20,33], as has porting classical tableau rules to SMT [10], as has intuitionistic logic [12,15]. Our work applies user propagation as a *framework for implementing non-classical logics*, but also for *theories* that have tableau rules, such as strings [26] or finite sets [3]. MetTeL 2 [37,38] can automatically synthesize solvers from tableau rules expressed in a domain-specific input language: complex features that cannot be expressed in the input language can be implemented by manually changing the output program generated by the tool.

Another approach to non-classical logics translates non-classical input to SAT/SMT [11,23], first-order or higher-order logic [18,19,31,32,35,36] via a shallow embedding. After translation, a SAT/SMT solver or automatic theorem provers (ATPs) can be used for reasoning. ATPs typically work poorly esspecially on satisfiable instances from such translations [25,39,40]. Solvers do not usually take into account meta-logical properties of the considered non-classical logic. If at all, such properties are communicated to a solver via further lemmas or fine-tuning the solver's configuration. Our approach allows us to directly encode expert knowledge of the considered logic. Additionally, our approach allows reasoning in multiple non-classical logics simultaneously and supports theory reasoning.

2 Background and Challenges

Background. We assume familiarity with basics of classical first-order logic [34], SMT solving [7] , and the description logic $\mathcal{ALC}$ [2]. To avoid confusion with first-order quantifiers, we use modal syntax to write $\mathcal{ALC}$ formulas φ as

$$\varphi ::= \top \mid A \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Box_r \varphi$$

where A is a (theory[1]) atom and r a modality/role. The logical connectives $\Rightarrow$, $\wedge$, and $\bot$ are defined as usual. The modal operator $\Diamond_r$ is defined as the dual of $\Box_r$. We assume a problem in $\mathcal{ALC}$ is given by a *knowledge base* $\langle TBox, ABox \rangle$. Elements in $TBox$ are of the form $global(\varphi)$[2] and are intended to be true in all worlds. Elements in $ABox$ are of the form $w_i : \varphi$, asserting "φ holds in world w_i"; or $r_k : (w_i, w_j)$, asserting "r_k relates worlds w_i and w_j". In case no $ABox$ is given, we assume the existence of an implicit world w_0. The truth-value of a formula φ under such a Kripke interpretation is given as in [2].

SMT Challenges for First-Order Translation of Description Logics. We motivate our work by considering the $\mathcal{ALC}$ knowledge base

$$TBox = \{global(\Diamond_r(A \wedge \Diamond_r \neg A))\}. \tag{1}$$

[1] this is an addition to the classical definition of $\mathcal{ALC}$.

[2] we write the more usual form $\varphi_1 \sqsubseteq \varphi_2$ as $global(\varphi_1 \Rightarrow \varphi_2)$.

$$\text{rule:}\ \frac{\text{Some conditions } P_1, \ldots, P_n}{\begin{array}{ccc} \mathit{sign}_{1,1} : \varphi_{1,1} \in \mathcal{L}(w_{1,1}) & \ldots & \mathit{sign}_{n,1} : \varphi_{n,1} \in \mathcal{L}(w_{n,1}) \\ \ldots & \ldots & \ldots \\ \mathit{sign}_{1,m_1} : \varphi_{1,m_1} \in \mathcal{L}(w_{1,m_1}) & \ldots & \mathit{sign}_{n,m_n} : \varphi_{n,m_n} \in \mathcal{L}(w_{n,m_n}) \end{array}}$$

Fig. 1. Abstract tableau calculus rule.

One may reason about this formula by (i) translating it into classical first-order logic via the *standard translation* [9]; and (ii) using a decision procedure handling uninterpreted functions and quantifiers to establish satisfiability of the translated formula. In particular, step (i) translates (1) into the first-order formula

$$\forall x(\exists y(\mathit{reach}^r(x,y) \land A(y) \land \exists z(\mathit{reach}^r(y,z) \land \neg A(z)))) \tag{2}$$

where reach^r is an uninterpreted function symbol. Then, in step (ii) SMT solving over (2) instantiates the universally-quantified variable x with w_0, using for example model-based quantifier instantiation (MBQI) [17]. Skolemization introduces two new constants w_1 and w_2, which results in the quantifier-free instance:

$$\mathit{reach}^r(w_0,w_1) \land \mathit{reach}^r(w_1,w_2) \land A(w_1) \land \neg A(w_2), \tag{3}$$

from which the partial interpretation

$$\mathit{reach}^r(x,y): \text{ if } (((x = w_0 \land y = w_1) \lor (x = w_1 \land y = w_2))) \text{ then } \top \text{ else } *. \tag{4}$$

can be deduced. The symbol $*$ is undetermined and represents an arbitrary Boolean value. Assume that the SMT solver sets $*$ to $\bot$ in order to complete the partial model (4) for checking (2): As the solver cannot derive equalities among the world constants w_0, w_1, w_2, the solver has to check all three constants with respect to the universal quantifier of (2). As w_1 and w_2 violate the universal quantifier, further constants are generated by Skolemization, but (2) remains violated and the sequence of MBQI steps repeat indefinitely. Choosing $\top$ for $*$ avoids such failure, but increases the burden of SMT solving, as the solver must consider all potential relations among all constants (here, w_0, w_1 and w_2) and eliminate such relations stepwise again as they lead to conflicts. Randomly choosing $\top$ or $\bot$ for completing the partial model (4) of (2) is not a solution either, as it combines the disadvantages of both approaches.

3 Tableau as a Decision Procedure in CDCL($\mathcal{T}$)

Addressing the above challenges, we advocate user-propagators for tailored SMT solving, providing efficient implementations of custom tableau reasoners. We propose using the lemma generation process of CDCL($\mathcal{T}$), explained below, to simulate rule application of tableau calculi.

In a nutshell, the CDCL($\mathcal{T}$) infrastructure [6] introduces fresh Boolean variables to name theory atoms of an input formula; the resulting propositional

$$\neg\ \textit{rule}: \frac{1:\neg\varphi \in \mathcal{L}(w)}{0:\varphi \in \mathcal{L}(w)} \qquad \neg\ \textit{rule}: \frac{0:\neg\varphi \in \mathcal{L}(w)}{1:\varphi \in \mathcal{L}(w)}$$

$$\wedge\ \textit{rule}: \frac{1:\varphi_1 \wedge \varphi_2 \in \mathcal{L}(w)}{\begin{array}{c}1:\varphi_1 \in \mathcal{L}(w)\\ 1:\varphi_2 \in \mathcal{L}(w)\end{array}} \qquad \wedge\ \textit{rule}: \frac{0:\varphi_1 \wedge \varphi_2 \in \mathcal{L}(w) \text{ and } 0:\varphi_1,\varphi_2 \notin \mathcal{L}(w)}{0:\varphi_1 \in \mathcal{L}(w) \qquad 0:\varphi_2 \in \mathcal{L}(w)}$$

$$\Box\ \textit{rule}: \frac{1:\Box_r\varphi \in \mathcal{L}(w_i) \text{ and } 1:r(w_j) \in \mathcal{L}(w_i) \text{ and } w_i \text{ not blocked}}{1:\varphi \in \mathcal{L}(w_j)}$$

$$\Box\ \textit{rule}: \frac{0:\Box_r\varphi \in \mathcal{L}(w_i) \text{ and } \nexists w_j(1:r(w_j) \in \mathcal{L}(w_i) \wedge \varphi \in \mathcal{L}(w_j)) \text{ and } w_i \text{ not blocked}}{\begin{array}{c}0:\varphi \in \mathcal{L}(w_j)\ \text{ with fresh } w_j\\ 1:r(w_j) \in \mathcal{L}(w_i)\end{array}}$$

$$\textit{global rule}: \frac{1:\textit{global}(\varphi) \in \mathcal{L}, w \text{ not blocked, and } w \text{ occuring in some } \mathcal{L}(w')}{1:\varphi \in \mathcal{L}(w)}$$

$$\textit{individual rule}: \frac{1:(w:\varphi) \in \mathcal{L}}{1:\varphi \in \mathcal{L}(w)} \qquad \textit{reach rule}: \frac{1:(r:(w_i,w_j)) \in \mathcal{L}}{1:r(w_j) \in \mathcal{L}(w_i)}$$

with w being blocked iff there is a (transitive) predecessor *pred* such that $\mathcal{L}(w) \subseteq \mathcal{L}(\textit{pred})$

Fig. 2. Rules for the $\mathcal{ALC}$ Description Logic.

skeleton is then solved by an ordinary SAT solver. If a propositional model is found, theory solvers are asked if the model is correct with respect to theory atoms. These specialized procedures may introduce further "lemma" formulas to the Boolean abstraction or report conflicts directly, forcing the SAT solver to "correct" the Boolean interpretation. This is repeated until all theory solvers agree on the Boolean assignment or the Boolean abstraction becomes unsatisfiable.

User-Propagators in CDCL($\mathcal{T}$) with Tableau Methods. Our solution builds a custom reasoner using the user-propagator framework [8]. Algorithm 1 shows underlined parts relevant for the following discussion. The custom reasoner is implemented by providing the methods `push`, `pop`, `fixed` and `final` in some programming language. The method `abstr(f)` is a method to be applied *a priori* solving. All other methods are those of the SMT solver.

We can simulate a tableau calculus whose rules are of the abstract form shown in Fig. 1. We use *signed formulas* of the form *sign* : $\circ(\bar{\varphi})$, where *sign* is a member of a fixed set, usually truth values, and $\circ$ is a logical operator applied to operands/subformulas $\bar{\varphi}$. Each P_i asserts that a signed formula is (not) contained in a *label* $\mathcal{L}(w)$. Labels are sets of signed formulas with known sign at some node w on the current branch. Rules may only add signed formulas to labels and create new branches. We assume the input is satisfiable, in case no more rule is applicable.

This means, we consider *sound, confluent, and non-destructive tableaux with signed formulas* [34] and *explicit labelled nodes* [24], which are straightforward in

Algorithm 1: Simple CDCL($\mathcal{T}$) Algorithm.
Methods that can be provided by a user-propagator are underlined.

```
1  Method CDCLT($f$):
2      $f \leftarrow$ abstr($f$)                                          ▷ Sect. 3.2
3      Loop
4          if conflict($f$) then
5              if backtrack($f$) = failed then return UNSAT
6              foreach $s \in \mathcal{T}$-solvers do $s$.$\underline{\text{pop}}$()          ▷ Sect. 3.5
7          while can_unit_propagate($f$) do assign(get_up($f$))
8          if contains_unassigned($f$) then
9              foreach $s \in \mathcal{T}$-solvers do $s$.$\underline{\text{push}}$()         ▷ Sect. 3.5
10             assign(guess_variable($f$))
11         else
12             foreach $s \in \mathcal{T}$-solvers do $s$.$\underline{\text{final}}$()       ▷ Sect. 3.4
13             if ¬new_formulas_propagated() then return SAT
14 Method assign($x$, $value$):
15     foreach $s \in \mathcal{T}$-solvers do
16         if is_associated($s$, $x$) ∧ is_relevant($x$) then
17             $s$.$\underline{\text{fixed}}$($x$, $value$)                              ▷ Sect. 3.3
```

our framework. Many calculi [13], including those for propositional logics, first-order logics, various modal/description logics, and several many-valued logics, can naturally be expressed within Fig. 1. The main steps of our work towards integrating tableau reasoning in SMT solving can be illustrated using a running example in $\mathcal{ALC}$. The tableaux rules for $\mathcal{ALC}$ in our notation are detailed in Fig. 2.

Example 1 (Running Example). Consider the $\mathcal{ALC}$ knowledge base:

$$TBox = \{global(Hum \Rightarrow (\Box_p(Alive \Rightarrow age \leq recordLifespan) \wedge \Diamond_p Hum))\}$$
$$ABox = \{eva : Hum \vee \Diamond_f \neg Hum, \quad par : (eva, paul)\}$$

where $Alive$ (Alive), Hum (Human), and age depend on the current world, but $recordLifespan$ does not; age and $recordLifespan$ are of integral sort; p (parent) and f (friend) denote roles; and eva and $paul$ are named worlds.

3.1 SMT-LIB Encoding and Custom SMT Theory

To enable SMT-based tableau reasoning, we encode non-classical logic features directly in an extension of the SMT-LIB input standard [5]. In particular, we encode non-classical logic symbols with the help of uninterpreted function symbols and sorts, yielding an SMT theory of non-classical logic.

Example 2 ($\mathcal{ALC}$ Knowledge Base in SMT-LIB). For $\mathcal{ALC}$, we introduce the uninterpreted *Relation* and *World* sorts and the following functions:

$$\begin{array}{llll} box: & Relation \times B \mapsto B & dia: & Relation \times B \mapsto B \\ global: & B \mapsto B & world: & \emptyset \mapsto W \\ reachable: & Relation \times W \times W \mapsto B & & \end{array}$$

where B is the sort of Booleans and *world* represents the current world[3]. Functions may have an extra "World" argument to denote their dependency on some world. With these syntactic features on top of SMT-LIB, Example 1 is encoded as

```
(declare-fun Hum (World) Bool)      (declare-fun Alive (World) Bool)
(declare-fun age (World) Int)       (declare-const recordLifespan Int)
(declare-const eva World)           (declare-const paul World)
(declare-const p Relation)          (declare-const f Relation)
(assert (global
    (=> (Hum world) (and
        (box p (=> (Alive world) (<= (age world) recordLifespan)))
        (dia p (Hum world))))))
(assert (global (=> (= world eva) (or (Hum world) (dia f (Rob world))))))
(assert (reachable p eva paul))
```

3.2 Preprocessing (`Abstr`)

Next, we traverse the syntax tree of the parsed problem and introduce fresh user-function symbols to abstract away subformulas we want to observe. All instances of introduced user-functions are automatically *associated* with our user-propagator and thus Boolean assignments to those instances might be reported by the SMT core by calling the `fixed` method. We might add a node parameter of an uninterpreted sort to user-functions to store additional information, such as the current world in Kripke semantics. As we go, we build a tree-shaped *abstraction* data structure for keeping track of abstracted subformulas and efficiently applying tableau rules. Only the root of the abstraction is passed to the SMT solver. Furthermore, we apply (logic-specific) simplifications.

Example 3 (Preprocessing and Abstraction). Recall Example 1. We replace all operators handled by tableau rules by fresh user-functions: here, for the occurrences of $\Box_r\varphi$, $global(\varphi)$, and for theory atoms. World-dependent terms and some operators, such as $\Box$, require a node argument denoting the world in which they are evaluated. To ease instantiating multiple instances of the formulas, we use an unbounded variable x as the node argument. We obtain the SMT abstraction of Example 1 given in Fig. 3. G denotes applications of the *global*-rule, M^r applications of $\Box_r$, and T arbitrary theory atoms. *ABox* elements are encoded directly by instantiating the node arguments accordingly (e.g., $\neg M_1^f(eva)$).

[3] which will be eliminated during preprocessing.

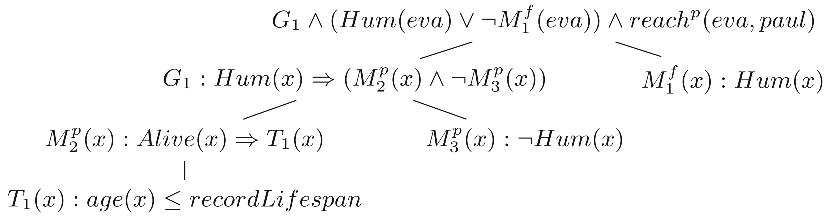

$G_1 \wedge (Hum(eva) \vee \neg M_1^f(eva)) \wedge reach^p(eva, paul)$
$G_1 : Hum(x) \Rightarrow (M_2^p(x) \wedge \neg M_3^p(x))$
$M_1^f(x) : Hum(x)$
$M_2^p(x) : Alive(x) \Rightarrow T_1(x)$
$M_3^p(x) : \neg Hum(x)$
$T_1(x) : age(x) \leq recordLifespan$

Fig. 3. Abstraction tree for Example 1. For simplicity, we rewrote $\Box_r A$ as $\neg\Diamond_r\neg A$.

3.3 Populating Languages (Fixed)

Whenever the SAT core assigns a variable $V_i(w) \mapsto value$, we look up the operator $\circ$ and its operands abstracted by V_i during preprocessing. We add $\circ$, together with the auxiliary symbol and its operands $\bar{\varphi}_i$, to the respective label set[4] such that $\hat{\mathcal{L}}(w) := \hat{\mathcal{L}}(w) \cup \{(value : \circ, V_i, \bar{\varphi}_i)\}$ As the user-propagator reports only assignments to formulas that were previously abstracted away by user-functions, we might also need to abstract away other formulas for which we are not interested in adding additional rules, in order to be notified when these elements are added to some labels. For example, if we must observe 0: $(\varphi_1 \wedge \varphi_2) \in \mathcal{L}(w)$, we can replace $\wedge$ by a user-function. Usually, the tableau is closed (i.e. conflict) automatically if we have formulas of different sign. If the calculus has more complicated closing conditions, they can be reported explicitly by propagating a conflict.

Example 4 (Tracking Assignments to Arbitrary Subformulas). To keep track of all relevant Boolean assignments to atoms, we replace all atoms by user-functions, including complex theory atoms such as $age(w) \leq recordLifespan$ as shown in Fig. 3. To preserve semantics, we add the definitions of the abstracted atoms by propagation For example, within Example 1 we might eagerly propagate

$$T_1(w) = value \vdash ((age(w) \leq recordLifespan) = value),$$

as soon as $T_1(w)$ is assigned the Boolean *value*.

3.4 Rule Application (Final)

Whenever the solver found a Boolean assignment such that the propositional abstraction of its extended SMT problem (Sect. 3.1) is satisfied, we apply logic-specific tableau rules by iterating over the set $\hat{\mathcal{L}}(w)$ for every node w until no more tableau rules are applicable. A *propagation claim* is of the form $J_1, \ldots, J_m \vdash C$. An arbitrary number of them can be added by the user-propagator within *fixed* and *final*, indicating that the SAT core needs to assign $C \mapsto 1$ justified by the expressions $J_1, \ldots, J_m$; here, C may be an arbitrary Boolean expression.

[4] $\hat{\mathcal{L}}(w)$ are sets maintained by the user-propagator code to simulate $\mathcal{L}(w)$.

Consider a tableau rule R as in Fig. 1 and assume that R is applied because $\{P'_1, \ldots, P'_m\} \subseteq \{P_1, \ldots, P_n\}$ are satisfied, obtaining

$$Just(P'_1), \ldots, Just(P'_m) \vdash C, \tag{5}$$

where $Just(P'_i)$ is J_i. We give C as a formula in disjunctive normal form (DNF)

$$\bigvee_{1 \le i \le n} \bigwedge_{1 \le j \le m_i} (\varphi_{i,j}(w_{i,j}) = sign_{i,j}) \tag{6}$$

simulating application of the rule R. We note that by using relevancy propagation [28] SMT solving may enjoy tableau-style branching, such that only one disjunct of the above DNF is chosen and reported assigned; unnecessary Boolean assignments are not reported to the user-propagator. We distinguish between two types of P'_i in (5): (i) those asserting elements are in the label, where P'_i is $sign : \circ(\bar{\varphi}) \in \mathcal{L}(w)$; and (ii) those that assert the opposite, where P'_i is $sign : \circ(\bar{\varphi}) \notin \mathcal{L}(w)$.

Justifying (i) is straightforward, as there must be an auxiliary user-function denoting that the respective element is contained in the label. We therefore have $sign : \circ(\bar{\varphi}), V, \bar{\varphi} \in \hat{\mathcal{L}}(w)$ and define $Just(P'_i)$ to be the equality $V = sign$. Case (ii) cannot be justified in general in our encoding because some assignments might not have been reported due to relevancy propagation. However, justifications for non-containment constraints may be omitted in the following scenarios:

1. The expression C can be simplified to $\top$ with respect to the current SAT assignment and hence Lemma (5) and its justifications are irrelevant. Consider $F(w) \mapsto 0$ where $F(w)$ is a user-function used to replace $A \wedge B$ in some node w (see $\wedge$ rule in Fig. 2) and $0 : A \in \hat{\mathcal{L}}(w)$. Propagating $F(w) \vdash A(w) = \bot \vee B(w) = \bot$ has no effect, as the SMT solver detects that the consequent is already satisfied and ignores (5).
2. Applying R without satisfying the negative containment condition does not affect soundness or completeness and we make sure that we do not apply R infinitely often. Consider $F(w) \mapsto 0$ where $F(w)$ replaces $\Box A$ in some node w (see $\Box$ rule in Fig. 2). Applying this rule once or finitely often does not affect soundness or completeness in $\mathcal{ALC}$.

In either scenario, we do not justify that the respective conditions P'_i are satisfied, but only check P'_i before application of R (e.g. checking if a world is blocked). We hence set $Just(P'_i)$ to $\top$.

Example 5 (Applying Rules). Recall Example 1. Consider 1: $M_2^p \in \hat{\mathcal{L}}(eva)$, 0: $M_3^p \in \hat{\mathcal{L}}(eva)$ and 1: $G \in \hat{\mathcal{L}}$. SMT solving may propagate in `final`

$$M_3^p(eva) = \bot \vdash (\neg Hum(mary)) = \bot \wedge reach^p(eva, mary) = \top$$

by a 0: $\Box$-rule instance of Fig. 1, where *mary* is a fresh world. The next `final` callback might then propagate (because of the 1: $\Box$ and 1: *global* rules)

$$\begin{aligned} M_2^p(eva) = \top \wedge reach^p(eva, mary) = \top \\ \vdash (Alive(mary) \Rightarrow T_1(mary) = \top \\ G_1 = \top \wedge reach^p(eva, mary) = \top \\ \vdash (Hum(mary) \Rightarrow (M_2^p(mary) \wedge \neg M_3^p(mary))) = \top. \end{aligned}$$

3.5 Backtracking (`Push+pop`)

Backtracking in the CDCL core of SMT solving uses justifications provided for propagation claims. Our SMT-based tableau reasoner has to reset (*pop*) its state to a previously-saved state (*push*), by restoring the value of $\hat{\mathcal{L}}(w)$ to the one it had in the previous state. However, unlike tableau calculi, subformulas introduced by rule application may persist after backtracking because of conflict learning and similar techniques, which can result in the solver assigning these atoms unnecessarily. These spurious assignments correspond to adding elements to some label $\mathcal{L}(w)$ without a respective rule being applicable and hence, it might happen that $\hat{\mathcal{L}}(w) \neq \mathcal{L}(w)$. We can nonetheless apply rules resulting from spurious assignments as if they were not spurious: mostly, the solver will either justify the spurious elements anyway later or, in the case of a conflict, backtrack and undo these assignments.

Example 6 (Spurious Assignments). Recall Example 1. Suppose *paul* has a parent *mary*, generated by $M_3^p(paul) \mapsto 0$ using the 0: $\Box$-rule. Further, assume *mary* has a parent *sam*, generated by $M_3^p(mary) \mapsto 0$. On conflict, the SMT solver might backtrack to a state before assigning $M_3^p(paul) \mapsto 0$. The tableau-based theory solver removes $reach^p(sam)$ from $\hat{\mathcal{L}}(mary)$, as well as $reach^p(mary)$ from $\hat{\mathcal{L}}(paul)$. However, the solver may not "forget" the existence of atoms $M_3^p(mary)$ and $M_3^p(paul)$. It may therefore happen that $M_3^p(mary)$ is assigned later without first generating *mary* via $M_3^p(paul) \mapsto 0$. We ignore this spurious assignment, as the solver may later again assign $M_3^p(paul) \mapsto 0$, *ex post facto* justifying the existence of *mary*. If this justification is not given later and we encounter a conflict, the solver backtracks and removes the spurious assignment. If it leads to a model, we ignore everything in the model resulting from the spurious assignment.

4 Implementation and Experiments

We implemented[5] our tableau reasoning approach from Sect. 3 in the Z3 SMT solver [29]. We compare our implementation applying user propagation over the custom SMT theory of Sect. 3.1 against our implementation using two translations of modal logic to first-order logic, *viz.* the standard translation [9] and iterative deepening using cardinality assumptions. We considered altogether 400

[5] https://github.com/CEisenhofer/ModalZ3.

Table 1. Experimental results for benchmarks in the modal logic K.

	satisfiable (400)	unsatisfiable (185)	total (585)
standard translation	221 (55.3%)	81 (43.8%)	302 (51.6%)
model building	219 (54.8%)	78 (42.2%)	297 (50.8%)
user-propagator	**269 (67.3%)**	**132 (71.4%)**	**401 (68.5%)**

satisfiable and 185 unsatisfiable benchmarks in the modal logic K [30]. Our initial experiments using a 60-second timeout are summarized in Table 1, showing that applying our user-propagator framework performs the best. This is partially so because quantifier reasoning in Z3 comes with MBQI overhead (Sect. 2). Finite model building performs poorly for large minimal models.

5 Conclusion and Discussion

We introduce an SMT-based reasoning framework for tableau methods, encoding tableau rules directly in SMT and applying user-propagators for custom reasoning. When implemented and evaluated using the Z3 SMT solver, our results outperform alternative encodings of the modal logic K. However, implementing logics via user-propagators *requires further knowledge about the considered non-classical logics* for tailored support towards, e.g., conflict learning and theory reasoning.

Beyond the Boolean Basis and Alternative Encodings. We so far considered an assignment $V \mapsto value$ to denote that $value : V \in \mathcal{L}(w)$ and only capture $value : V \notin \mathcal{L}(w)$ implicitly. This can be generalized to n mutually-exclusive truth values by using $\lceil log_2(n) \rceil$ Boolean variables. If, on the other hand, we need to justify that some element is *not* in our label, we can use a different encoding with each potential value encoded by a single Boolean. In this case, we use $bit_{sign}(V) = true$ to represent $V \in \mathcal{L}(w)$ instead of $V = sign$.

Example 7 (Ternary Logic). Consider a three-valued logic with values true, false, and undefined. The first encoding represents each truth value as a list of two bits where 00 represents false, 01 true, and 10 undefined respectively. The case of 11 is invalid. The second uses a list of three bits, one for each potential value. For each introduced subformula, we additionally propagate the cardinality constraint that exactly one bit has to be set to 1. This encoding incorporates the usual assumption that $value_1 : \circ \in \mathcal{L}(w)$ and $value_2 : \circ \in \mathcal{L}(w)$ with $value_1 \neq value_2$ represents a conflict, but could be dropped in cases where this is not desired.

Theories and Non-Classical Logic A challenging question arises when considering theories in combination with non-Boolean based logics. As we abstract away theory atoms (Example 3) and add them again on demand (Example 4), we can customize what and how theory atoms are passed to the SMT solver. For ternary logic, we might propagate the theory atom positively when assigned true, for false its negation, and nothing when the value is undefined.

References

1. Areces, C., Fontaine, P., Merz, S.: Modal satisfiability via SMT solving. In: Software, Services, and Systems, pp. 30–45 (2015). https://doi.org/10.1007/978-3-319-15545-6_5
2. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic (2017)
3. Bansal, K., Barrett, C.W., Reynolds, A., Tinelli, C.: Reasoning with finite sets and cardinality constraints in SMT. Log. Methods Comput. Sci. **14**(4), 1–31 (2018). https://doi.org/10.23638/LMCS-14(4:12)2018
4. Barbosa, H., et al.: cvc5: a versatile and industrial-strength SMT solver. In: TACAS 2022. LNCS, vol. 13243, pp. 415–442. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99524-9_24
5. Barrett, C., Fontaine, P., Tinelli, C.: The Satisfiability Modulo Theories Library (SMT-LIB) (2016). http://SMT-LIB.org
6. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, 2nd edn., vol. 336, pp. 1267–1329 (2021)
7. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, 2nd edn., pp. 1267–1329 (2021). https://doi.org/10.3233/FAIA201017
8. Bjørner, N.S., Eisenhofer, C., Kovács, L.: Satisfiability modulo custom theories in Z3. In: VMCAI, pp. 91–105 (2023). https://doi.org/10.1007/978-3-031-24950-1_5
9. Blackburn, P., van Benthem, J.: Modal logic: a semantic perspective. In: Handbook of Modal Logic, pp. 1–84 (2007). https://doi.org/10.1016/s1570-2464(07)80004-8
10. Bury, G., Cruanes, S., Delahaye, D.: SMT solving modulo tableau and rewriting theories. In: SMT (2018)
11. Caridroit, T., Lagniez, J., Berre, D.L., de Lima, T., Montmirail, V.: A sat-based approach for solving the modal logic s5-satisfiability problem. In: AAAI, pp. 3864–3870 (2017). https://doi.org/10.1609/aaai.v31i1.11128
12. Claessen, K., Rosén, D.: SAT modulo intuitionistic implications. In: LPAR, pp. 622–637 (2015). https://doi.org/10.1007/978-3-662-48899-7_43
13. D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J.: Handbook of tableau methods (2013). https://doi.org/10.1007/978-94-017-1754-0
14. Dutertre, B.: Yices 2.2. In: CAV, pp. 737–744 (2014). https://doi.org/10.1007/978-3-319-08867-9_49
15. Fiorentini, C., Goré, R., Graham-Lengrand, S.: A proof-theoretic perspective on smt-solving for intuitionistic propositional logic. In: TABLEAUX, pp. 111–129 (2019). https://doi.org/10.1007/978-3-030-29026-9_7
16. Fitting, M.: Tableau methods of proof for modal logics. Notre Dame J. Formal Log. **13**(2), 237–247 (1972). https://doi.org/10.1305/ndjfl/1093894722
17. Ge, Y., de Moura, L.M.: Complete instantiation for quantified formulas in satisfiability modulo theories. In: CAV, pp. 306–320 (2009). https://doi.org/10.1007/978-3-642-02658-4_25
18. Gleißner, T., Steen, A.: The MET: the art of flexible reasoning with modalities. In: RuleML+RR, pp. 274–284 (2018). https://doi.org/10.1007/978-3-319-99906-7_19
19. Gleißner, T., Steen, A., Benzmüller, C.: Theorem provers for every normal modal logic. In: LPAR, pp. 14–30 (2017). https://doi.org/10.29007/jsb9
20. Goré, R., Kikkert, C.: CEGAR-Tableaux: improved modal satisfiability via modal clause-learning and SAT. In: TABLEAUX, pp. 74–91. https://doi.org/10.1007/978-3-030-86059-2_5

21. Goré, R., Nguyen, L.A.: Analytic cut-free tableaux for regular modal logics of agent beliefs. In: CLIMA, pp. 268–287 (2007). https://doi.org/10.1007/978-3-540-88833-8_15
22. Goré, R., Olesen, K., Thomson, J.: Implementing tableau calculi using BDDs: BDDTab system description. In: IJCAR, pp. 337–343 (2014). https://doi.org/10.1007/978-3-319-08587-6_25
23. Haarslev, V., Sebastiani, R., Vescovi, M.: Automated reasoning in $\mathcal{ALCQ}$ via SMT. In: CADE, pp. 283–298 (2011). https://doi.org/10.1007/978-3-642-22438-6_22
24. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for expressive description logics. In: LPAR, pp. 161–180 (1999). https://doi.org/10.1007/3-540-48242-3_11
25. Horrocks, I., Voronkov, A.: Reasoning support for expressive ontology languages using a theorem prover. In: FoIKS, pp. 201–218 (2006). https://doi.org/10.1007/11663881_12
26. Liang, T., Reynolds, A., Tsiskaridze, N., Tinelli, C., Barrett, C., Deters, M.: An efficient SMT solver for string constraints. Formal Methods Syst. Des. **48**(3), 206–234 (2016). https://doi.org/10.1007/s10703-016-0247-6
27. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT Solvers. In: Handbook of Satisfiability, 2nd edn., vol. 336, pp. 133–182 (2021)
28. de Moura, L., Bjørner, N.: Relevancy Propagation. Technical Report MSR-TR-2007-140, Microsoft Research, Technical Report (2007), https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2007-140.pdf
29. de Moura, L.M., Bjørner, N.S.: Z3: an efficient SMT solver. In: TACAS, pp. 337–340 (2008). https://doi.org/10.1007/978-3-540-78800-3_24
30. Nalon, C., Hustadt, U., Papacchini, F., Dixon, C.: Local reductions for the modal cube. In: IJCAR, pp. 486–505 (2022). https://doi.org/10.1007/978-3-031-10769-6_29
31. Schmidt, R.A., Hustadt, U.: The axiomatic translation principle for modal logic. ACM Trans. Comput. Log. **8**(4), 19 (2007). https://doi.org/10.1145/1276920.1276921
32. Schneider, M., Sutcliffe, G.: Reasoning in the OWL 2 full ontology language using first-order automated theorem proving. In: CADE, pp. 461–475 (2011). https://doi.org/10.1007/978-3-642-22438-6_35
33. Sebastiani, R.: From KSAT to delayed theory combination: exploiting DPLL outside the SAT domain. In: FroCoS, pp. 28–46 (2007). https://doi.org/10.1007/978-3-540-74621-8_2
34. Smullyan, R.M.: First-order logic (1995). https://doi.org/10.1007/978-3-642-86718-7
35. Steen, A.: An extensible logic embedding tool for lightweight non-classical reasoning (short paper). In: PAAR (2022)
36. Steen, A., Fuenmayor, D., Gleißner, T., Sutcliffe, G., Benzmüller, C.: Automated reasoning in non-classical logics in the TPTP world. In: PAAR (2022)
37. Tishkovsky, D., Schmidt, R.A., Khodadadi, M.: MetTeL^2: towards a tableau prover generation platform. In: PAAR, pp. 149–162 (2012). https://doi.org/10.29007/1c73
38. Tishkovsky, D., Schmidt, R.A., Khodadadi, M.: The tableau prover generator MetTeL2. In: JELIA, pp. 492–495 (2012). https://doi.org/10.1007/978-3-642-33353-8_41
39. Tsarkov, D., Horrocks, I.: DL reasoner vs. first-order prover. In: DL (2003)
40. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to reason with OWL. In: ISWC, pp. 471–485 (2004). https://doi.org/10.1007/978-3-540-30475-3_33

DefTab: A Tableaux System for Sceptical Consequence in Default Modal Logics

Carlos Areces[1], Valentin Cassano[1,2], Raul Fervari[1,3](✉), and Guillaume Hoffmann[1,3]

[1] CONICET and Universidad Nacional de Córdoba, Córdoba, Argentina
[2] Universidad Nacional de Río Cuarto, Río Cuarto, Argentina
[3] Guangdong Technion - Israel Institute of Technology, Shantou, China
rfervari@gmail.com

Abstract. We report on an implementation of a tableaux calculus for sceptical consequence in Default Logic built on Hybrid Modal Logic. In turn, our tool offers support for checking default consequence over formulas from Propositional Logic, Basic Modal Logic and Hybrid Logic. We develop a test suite for assessing the correctness, scalability, and efficiency of our system, and inform on the results. Interestingly, our method can be adapted to generate examples for other default provers.

1 Introduction

A tableau method [11] is a standard proof procedure based on 'refutations'. To prove that a certain fact is valid, the procedure begins with a syntactical expression intended to assert the negation of the given fact. Then, successive steps syntactically break down this assertion into cases. Finally, impossibility conditions dictate closing cases. A proof is obtained if all cases are closed. Tableaux are one of the most popular proof calculi for Modal Logics, as they are known to lead to efficient and modular implementations [9].

The tableaux method presented here, called *default tableaux*, operates in the way just described. The novelty is that this tableaux method captures sceptical consequence in Default Logic [17], one of the most prominent approaches for non-monotonic reasoning [1]. Two distinguishing characteristics of a default logic are *defaults* and *alternative extensions*. Briefly, defaults can be understood as defeasible rules of inference, whereas extensions can be understood as sets closed under the application of defaults. Alternative extensions originate from 'consistency checks' on the application of defaults. A formula is called a 'sceptical consequence' if it is a consequence from every alternative extension. Our tableaux method handles sceptical consequence for $\mathscr{D}$HL, a default logic built over Hybrid Logic (HL) [3,4], via *default tableaux*. Default tableaux are introduced as an extension of tableaux for HL. These tableaux build on results presented in [5,7].

Moreover, we report on DefTab, an implementation of the default tableaux mentioned above. DefTab was originally conceived for checking sceptical consequence in Default Intuitionistic Logic [7]. Here, we advance on a modular implementation of a default prover acting over different modal logics. The general implementation of the tool is based on the architecture of HTab [13], a tableaux system for HL (see also [12]). Given the ability of handling formulas from HL, our prover also supports formulas from fragments of HL such as Classical Propositional Logic and Basic Modal Logic. Each fragment is in itself interesting.

We discuss the overall architecture of DefTab, the implementation of default tableaux algorithm, and optimization details. In addition, we present an empirical evaluation of the tool to assess its correctness and efficiency. To this end, we build a test suite for sceptical consequence in $\mathscr{D}$HL by using hGen [2], a random formula generator for HL and the mentioned fragments. We provide a systematic method to convert formulas generated by hGen into interesting test cases for $\mathscr{D}$HL. We posit other provers could benefit from our method in the future.

2 Basic Definitions

Hybrid Logic. The language of HL is defined on an enumerable set $\mathscr{P} = \{ p_i \mid 0 \leq i \}$ of *proposition symbols* and an enumerable set $\mathscr{N} = \{ n_i \mid 0 \leq i \}$ of *nominals*, and is determined by the following BNF:

$$\varphi ::= p_i \mid n_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid @_{n_i}\varphi \mid \mathsf{A}\varphi.$$

Other Boolean connectives are defined as usual. The modal formula $\Diamond\varphi$ is an abbreviation for $\neg\Box\neg\varphi$, whereas $\mathsf{E}\varphi$ abbreviates $\neg\mathsf{A}\neg\varphi$. We will also refer to some fragments of HL: the Basic Hybrid Logic (HL^-) is obtained by removing the constructor $\mathsf{A}\varphi$ from the BNF above. The Basic Modal Logic (BML) is obtained by additionally removing n_i and $@_{n_i}\varphi$ from the BNF. Finally, the Classical Propositional Logic (CPL) is obtained by additionally removing $\Box\varphi$.

A hybrid Kripke model $\mathfrak{M}$ is a tuple $\langle W, R, V \rangle$ where: W is a non-empty set of elements called worlds; $R \subseteq W^2$ is the accessibility relation; and the valuation $V : \mathscr{P} \cup \mathscr{N} \mapsto 2^W$ is a function s.t. for all $n \in \mathscr{N}$, $|V(n)| = 1$.

The notion of satisfiability, written $\mathfrak{M}, w \models \varphi$, is defined inductively as follows, with the Boolean cases defined as usual:

$$\begin{array}{lll}
\mathfrak{M}, w \models p_i & \text{iff} & w \in V(p_i) \\
\mathfrak{M}, w \models n_i & \text{iff} & \{w\} = V(n_i) \\
\mathfrak{M}, w \models \Box\varphi & \text{iff} & \text{for all } w' \in W,\ Rww' \text{ implies } \mathfrak{M}, w' \models \varphi \\
\mathfrak{M}, w \models \mathsf{A}\varphi & \text{iff} & \text{for all } w' \in W,\ \mathfrak{M}, w' \models \varphi \\
\mathfrak{M}, w \models @_{n_i}\varphi & \text{iff} & \mathfrak{M}, w' \models \varphi,\ \text{where } \{w'\} = V(n_i).
\end{array}$$

We write $\mathfrak{M}, w \models \Phi$ to abbreviate: for all $\varphi \in \Phi$, $\mathfrak{M}, w \models \varphi$. We call φ a *(local) semantic consequence* ([3]) of Φ, notation $\Phi \vDash \varphi$, iff for every hybrid Kripke model $\mathfrak{M}$, and world w of $\mathfrak{M}$, if $\mathfrak{M}, w \models \Phi$, then $\mathfrak{M}, w \models \varphi$.

Normal Default Logic. The work on *Default Logic*, initiated in [17], comprises nowadays a wide range of non-monotonic formalisms built on an underlying (typically monotonic) logic. In what follows, we describe a default logic built on HL, and call this default logic Default Hybrid Logic ($\mathscr{D}$HL).

$\mathscr{D}\mathsf{HL}$ is characterized by *normal defaults* and *extensions*. A normal default is a pair (π, χ) of formulas of HL written as π/χ; where π is called the prerequisite of the default, and χ its consequent. A normal default can be understood as a non-admissible rule of inference of HL which is only applied if its application does not yield a contradiction. Normal defaults are common in the literature, since interestingly most existing variants of Default Logic converge in the case of normal defaults (see, e.g., [1]). Extensions are defined with respect to default theories. A default theory is a pair $\Theta = \langle \Phi, \Delta \rangle$ where: Φ is a set of formulas of HL, also indicated by Φ_Θ; and Δ is a set of *normal defaults*, also indicated by Δ_Θ. An extension can be understood as a saturation of a set of facts via the application of defaults. The precise definition of an extension is given in Def. 4.

Definition 1. *Let $\delta = \pi/\chi$ be a default and Δ be a set of defaults; then: $\delta^{\Pi} = \pi$, $\delta^{\mathrm{X}} = \chi$; $\Delta^{\Pi} = \{ \delta^{\Pi} \mid \delta \in \Delta \}$, $\Delta^{\mathrm{X}} = \{ \delta^{\mathrm{X}} \mid \delta \in \Delta \}$ and $\Delta \cup \delta = \Delta \cup \{\delta\}$.*

Definition 2 (Detachment). *Let Θ be a default theory, and $\Delta \cup \delta \subseteq \Delta_\Theta$; we say that δ is* triggered *by Δ (in Θ) iff $(\Phi_\Theta \cup \Delta^{\mathrm{X}}) \vDash \delta^{\Pi}$. We say that δ is* blocked *by Δ iff $(\Phi_\Theta \cup (\Delta \cup \delta)^{\mathrm{X}}) \vDash \bot$. We say that δ is* detached *by Δ if δ is triggered, and not blocked, by Δ.*

If we think of a default π/χ as a rule which enables us to pass from π to χ, the notion of detachment in Def. 2 tells us under which conditions on π we can obtain χ. The definition of detachment is an intermediate step towards the definition of an extension via generating sets.

Definition 3 (Generating Set). *Let Θ be a default theory; we call $\Delta \subseteq \Delta_\Theta$ a* generating set *if there is a total-ordering $\lessdot$ on Δ_Θ s.t. $\Delta = \mathsf{D}^{\lessdot}_{\Theta}(n)$, where $n = |\Delta_\Theta|$, $\mathsf{D}^{\lessdot}_{\Theta}(0) = \emptyset$, and for all $0 < i < n$:*

$$\mathsf{D}^{\lessdot}_{\Theta}(i{+}1) = \begin{cases} \mathsf{D}^{\lessdot}_{\Theta}(i) \cup \delta & \textit{if } \delta \in \Delta_\Theta \backslash \mathsf{D}^{\lessdot}_{\Theta}(i) \textit{ is detached by } \mathsf{D}^{\lessdot}_{\Theta}(i), \textit{ and} \\ & \quad \textit{for all } \eta \neq \delta \in \Delta_\Theta \backslash \mathsf{D}^{\lessdot}_{\Theta}(i), \textit{ if } \eta \textit{ is detached by } \mathsf{D}^{\lessdot}_{\Theta}(i),\ \delta \lessdot \eta \\ \mathsf{D}^{\lessdot}_{\Theta}(i) & \textit{otherwise.} \end{cases}$$

Definition 4 (Extension). *Let Θ be a default theory and $E = \Phi_\Theta \cup \Delta^{X}$; the set E is an extension of Θ iff Δ is a generating subset of Δ_Θ. We use $\mathscr{E}(\Theta)$ to indicate the set of all extensions of Θ.*

As mentioned, intuitively, an extension is a set of formulas that is closed under detachment. We present the definition of default consequence in Def. 5.

Definition 5 (Default Consequence). *We say a formula φ is a* sceptical consequence *of a default theory Θ, notation $\Theta \approx \varphi$, iff for all $E \in \mathscr{E}(\Theta)$, $E \vDash \varphi$.*

The notion of default consequence in Def. 5 is referred to as sceptical in the literature on Default Logic. In Sec. 3 we present a syntactic characterization of sceptical consequence via a default tableaux proof calculus. This proof calculus is the focus of our system description. We illustrate our definitions in Ex. 1.

Example 1. We start by assuming that every world in the model has a successor, and that every world is either a *sink* world (nominal s) or 'sees' the sink world. These assumptions are expressed in a default theory as facts, i.e., by $\Phi = \{\mathsf{A}\Diamond\top, \mathsf{A}(s \vee \Diamond s)\}$. Moreover, we have three defaults: $\delta_1 = \top/@_{n_2}\Diamond n_3$, $\delta_2 = \top/@_{n_3}\neg s$, and $\delta_3 = \top/@_{n_3}\Box n_3$. Thus, we have $\Delta = \{\delta_1, \delta_2, \delta_3\}$, and $\Theta = \langle\Phi, \Delta\rangle$. The default δ_1 expresses that n_2 must 'see' n_3. This default is detached by Φ. Then, we have the defaults δ_2, expressing that n_3 must not be the sink world, and δ_3, expressing that n_3 must only 'see' itself. Both of these defaults are individually detached by δ_1, but they block each other: δ_2 forces n_3 to have a successor different from itself to comply with the facts, while δ_3 forces n_3 to see only itself, i.e., it forces n_3 be the sink. This means that we have two generating sets, $\{\delta_1, \delta_2\}$ and $\{\delta_1, \delta_3\}$, thus there are two extensions: $E_1 = \Phi \cup \{@_{n_2}\Diamond n_3, @_{n_3}\neg s\}$ and $E_2 = \Phi \cup \{@_{n_2}\Diamond n_3, @_{n_3}\Box n_3\}$. In both cases, n_2 sees the sink in two steps, i.e., $\Theta \approx @_{n_2}\Diamond\Diamond s$.

3 Default Tableaux Proof Calculus

We present the default tableaux calculus for sceptical consequence in $\mathscr{D}\mathsf{HL}$ which is the focus of our system description. In what follows, we consider all the formulas from HL in *negation normal form*. The default tableaux calculus for sceptical consequence in $\mathscr{D}\mathsf{HL}$ constructs so-called *default tableaux*. A default tableau is a tree whose nodes are of three different kinds. We write nodes of the first kind as $@_i\varphi$, meaning that φ holds at world i. The second kind of nodes (which is a special case of the first kind) is written as $@_i\Diamond j$, meaning that world j is accessible from world i. Nodes of the third kind are indicated by defaults. This last kind of nodes marks the use of a default in a proof attempt. A default tableau for a formula φ from a default theory Θ, is a default tableau whose root is $@_0\neg\varphi$, and whose construction is carried out using the rules from Fig. 1.

$$\frac{@_i(\varphi \wedge \psi)}{@_i\varphi,\ @_i\psi}\,(\wedge) \qquad \frac{@_i\Diamond\varphi}{@_i\Diamond j,\ @_j\varphi}\,(\Diamond)^1 \qquad \frac{@_i@_a\varphi}{@_a\varphi}\,(@) \qquad \frac{@_i\mathsf{E}\varphi}{@_j\varphi}\,(\mathsf{E})^1$$

$$\frac{@_i(\varphi \vee \psi)}{@_i\varphi \mid @_i\psi}\,(\vee) \qquad \frac{@_i\Box\varphi,\ @_i\Diamond j}{@_j\varphi}\,(\Box) \qquad \frac{@_i\varphi,\ @_ij}{@_j\varphi}\,(nom)^2 \qquad \frac{@_i\mathsf{A}\varphi}{@_j\varphi}\,(\mathsf{A})^2$$

$$\frac{}{@_jj}\,(\mathrm{ref})^2 \qquad \frac{}{@_0\varphi}\,(\mathsf{F})^3 \qquad \frac{}{\begin{array}{ccccc}\delta_1 & & \delta_i & & \delta_n \\ @_0\delta_1^{\mathrm{X}} & \dots & @_0\delta_i^{\mathrm{X}} & \dots & @_0\delta_n^{\mathrm{X}}\end{array}}\,(\mathsf{D})^4$$

1 The nominal j is new to the branch.
2 The nominal j is already in the branch.
3 For $\varphi \in \Phi_\Theta$.
4 For $\{\,\delta_i \mid i \in [1, n]\,\} = \{\,\delta \in \Delta_\Theta \backslash \Delta_B \mid \delta$ is detached by $\Delta_B\,\}$, where Δ_B is the set of defaults in the branch.

Fig. 1. Tableau expansion rules for $\mathscr{D}\mathsf{HL}$.

The rule (F) enables us to incorporate formulas from Φ_Θ into a default tableau, while the rule (D) enables us to incorporate defaults from Δ_Θ. This last rule corresponds to the concept of detachment in Def. 2. The notion of reducibility using default tableaux is made precise in Def. 7.

Definition 6 (Closure). *A branch of a default tableau is* closed ($\blacktriangle$)*, if* $@_i\varphi$ *and* $@_i\neg\varphi$ *occur in the branch. A branch is* open ($\blacktriangledown$) *if it is not closed. A default tableau is* closed *if all of its branches are closed; otherwise it is* open*.*

Definition 7 (Default Deducibility). *We call any closed default tableau for* φ *from* Θ *a* sceptical proof *of* φ *from* Θ*, notation* $\Theta \mathrel{|\!\sim} \varphi$*.*

The expansion rules in Fig. 1 together with Def. 7 yield a sceptical proof calculus which is is *sound* and *complete* (see [7] for details of this claim).

Theorem 1 (Soundess and Completeness.). $\Theta \mathrel{|\!\sim} \varphi$ *iff* $\Theta \mathrel{|\!\approx} \varphi$*.*

In addition, notice that if we forbid the application of the rule (D), we obtain a notion of deducibility $\Phi_\Theta \vdash \varphi$ which yields a sound and complete proof calculus for HL, i.e., $\Phi_\Theta \vdash \varphi$ iff $\Phi_\Theta \models \varphi$ (see [16]). We use $\vdash$ to syntactically check the side condition of the rule (D), and decide whether it can be applied or not.

Definition 8 (Saturation). *A branch of a default tableau is* saturated*, notation* ($\blacklozenge$)*, if the application of any of the expansion rules in Fig. 1 is redundant.*

It can be proven that every branch of a default tableau can be extended to one that is saturated in a finite number of steps. Also, if a default tableau for φ from Θ has a branch that is open and saturated, then $\Theta \mathrel{|\!\not\approx} \varphi$. From these two facts, it follows that default tableaux decide sceptical consequence.

4 Implementation

DefTab is an implementation of the tableaux proof calculus for sceptical default consequence in Sec. 3. The architecture of DefTab is based on the hybrid logic prover HTab [13], and incorporates the specific features for implementing default reasoning. HTab implements a terminating tableaux algorithm for HL and comes ready with some optimizations such as semantic branching and backjumping. All these features, as well as others, are reported in detail in [13]. Given Θ and φ as input, DefTab builds proof attempts of $\Theta \mathrel{|\!\sim} \varphi$ by searching for Kripke models for φ, and subsequently restricting these models with the use of sentences from Φ_Θ and defaults from Δ_Θ. DefTab reports whether a default proof has been found or not. In the latter case, it exhibits an extension of Θ from which the φ does not follow; thus establishing that φ is not a default consequence of Θ. In what follows we discuss some implementation details, including some comments on optimizations. DefTab is available at http://tinyurl.com/deftab0.

Tableaux and Subtableaux. The tableaux algorithm of DefTab follows a standard strategy for proof search, and the novel part is the treatment of the rule (D). In such a case, it selects a default δ from the set Δ_Θ, and checks if δ is detached, according to Def. 2. This relies on subtableaux, that is, tableaux executions that are independent of the main default tableaux. These subtableaux are needed to check whether δ is detached in the branch; i.e., whether it is triggered (i.e., δ^Π is a consequence of the premises and the consequences already obtained in the branch), and not blocked (i.e., if δ^X adds an inconsistency into the branch). If δ is detached, then $@_0\delta^X$ is added to the branch, δ is marked as treated, and the algorithm continues with the expansion of the updated branch. Once no rule can be applied, the algorithm returns TRUE if and only if φ is a default consequence of Θ.

Subtableaux Caching. One of the main optimizations provided in DefTab is *caching*, operating under the following premise. Subtableaux are executed to check which default rules are triggered or blocked in the context of a branch. Many of these checks are redundant, since the results of such subtableaux does not change unless a default rule is applied to a branch. DefTab implements a simple caching system that stores subtableaux results in a dictionary. Each time a subtableaux is about to be executed, the set of initial formulas is checked against the cache. If there is a cache hit, the result is taken from the cache and a tableaux run is saved. Note that subtableaux do not involve the rule (D), that is, they are purely tableaux of the underlying logic.

Default Rules Data Structures. At any given moment, DefTab maintains defaults in two lists: *available* and *triggered*. The available list contains the defaults of the input default theory. When the (D) rule is about to be applied, several steps are performed to handle default rules systematically. First, the *available* list is scanned, and each rule is checked to be triggered. Triggered rules are moved into the *triggered* list, and the rest is left into the *available* list. Note that non-triggered rules, may become triggered in the future after some default is added to the branch. The *triggered* list is also scanned, and each rule is checked to be blocked in the current branch. When a rule is blocked, it is deleted from the *triggered* list and will never come back again in the branch. Once this is done, DefTab uses that list to apply the rule (D). The tableaux branches as many times as there are rules in the (non-blocked) *triggered* list. For each new branch, the procedure removes the corresponding rule from the *triggered* list, and adds it and its consequent formula to the branch.

Backjumping. Backjumping [14] is a standard optimization for the HL calculus that greatly improves performance (see [13]). The overall idea is that, instead of performing a simple backtracking when a branch is found to be closed, backjumping calculates the lowest level to which the execution of the tableaux may directly come back when a clash is found. This requires all formulas in the tableaux to be annotated with a set of *dependencies*. A dependency is the level of a branching rule application. For the specific case of default tableaux, we take special care of tracking dependencies of the formulas introduced by the application of rule (D). To do so, once a default π/χ is triggered, we bookkeep it in the *triggered* list along with the dependencies of the formulas that triggered it, according to Definition 2. Concretely, this is the union of the dependencies of all defaults Δ such that $\Phi_{\ominus} \cup \Delta^{\mathrm{X}} \models \pi$. When (D) rule is applied, the consequent of a default is added to the current branch with these dependencies, plus the dependency of the current tableaux level.

Usage. DefTab takes as input a file following the structure of the following simple example file `hybrid01.dt`.

```
facts:
N0: <> N1;
defaults:
(N0: <>N1) --> (N1:<>N0);

consequence:
N0:<><>N0;
```

- The keyword `facts` indicates the beginning of the set of formulas of the default theory.
- The keyword `defaults` indicates the beginning of the set of defaults. The syntax for a default π/χ is π `-->` χ.
- The keyword `consequence` indicates the formula to be proven.

DefTab is executed from the command line as:

```
$ ./deftab -f hybrid01.dt
--------------------------------
Indeed a sceptical consequence.
Elapsed time: 0.00 seconds
```

The output indicates that `N0:<><>N0` ($@_{n_0}\Diamond\Diamond n_0$) is a sceptical consequence of the default theory.

5 Testing Generation and Methodology

Hybrid and Default Formulas Generation. Another contribution of our work is to provide a systematic way of constructing test cases for $\mathscr{D}$HL provers. To our knowledge, there is no standard test set for automated reasoning with default logic, and less so for default reasoning based on HL.

We build test cases for $\mathscr{D}$HL using the random formula generator hGen [2]. hGen enables us to generate formulas in conjunctive normal form (CNF) from several fragments of HL, such as CPL, BML and HL^{-}. Moreover, hGen also allows us to specify the different parameters of a formula: number of clauses, size of clauses and modal depths of each subformula of a clause, probability of that an operator appears in the clause (e.g. modal, hybrid, universal), and the total number of propositional symbols and nominals.

We adapted hGen to generate normal default theories from random HL formulas. The transformation depends on the satisfiability status of the original HL formulas. The first case applies to satisfiable formulas of HL in CNF. Given $c_1 \ldots c_n$ the clauses of an HL formula, we put each one of them as the consequent of a default $\top/c_i$, and put $\bot$ as the consequence to be proved. As the original set of clauses is satisfiable, and the consequence is never provable, all the defaults will be applied (as putting $\top$ as the prerequisite triggers every rule) in all possible permutations. This is an easy way to stress our tool.

The second case works with unsatisfiable formulas of HL in CNF. Here, we use an intentionally harder transformation. Given $c_1 \ldots c_n$ the clauses of the HL formula, then for all $i<n$, we generate two rules: $\top/c_i \vee c_{i+1}$ and $c_i \vee c_{i+1}/c_i \wedge c_{i+1}$. Finally, we add c_n as consequence. In this case, not all defaults will be applied to a same branch, but a great amount of them. Moreover, the formula c_n may or may not be a sceptical consequence of the default theory; this is another difference with the case of satisfiable formulas. This case not only serves to test the scalability of our tool, but also its correctness.

Test Suite Structure. The Bash script `testsuite.sh` executes four steps: formula generation, renaming, benchmark, and consistency check.

The *formula generation* step uses hGen to generate random sets of formulas from CPL, BML, HL^- and HL, respectively. Initially, each set contains 1000 formulas. Then, the Hybrid Logic prover HTab ([13]) is run to classify each set of formulas into satisfiable (SAT) and unsatisfiable (UNSAT). This way, hGen generates the corresponding default theories, as described in the previous section. The *renaming* step is then performed to organize file names in each folder.

The *benchmark* step enables to specify a list of provers to be run. Currently, it is performed with DefTab with cache disabled (NC) and DefTab with cache enabled (C), but the script can be easily modified to run any new default prover. The provers are executed on all input files of each combination of 4 languages and 2 satisfiability values, and the results (execution time and answer) are stored in log files. The script reports how many formulas could be solved within 10 s, 30 s, and 60 s. This is done by running the provers with the highest timeout value; the other values are deduced from the prover's running time.

Finally, the *consistency check* step looks for inconsistent outputs between provers by comparing the log files generated in the previous step.

Although the preselected option is to run all these steps together, they can also be run separately. This enables to run the benchmark step on a known set of formulas, to reproduce results. Instructions on how to run the tests, the test script and the set of formulas used to generate the following results can be found at http://tinyurl.com/deftab0.

hGen **parameters.** For each language, we tuned hGen's parameters to get a good SAT/UNSAT balance of its output (ideally a 50/50 ratio). We also aimed at getting a balanced difficulty of the translated default theories. That is, the sets of default theories should be hard enough so that many of them make DefTab timeout and we may measure improvements in the future, but not too hard so we can already observe different results according to different timeout values. The parameters for each language are: for CPL, 33 clauses and 10 proposition symbols; for BML, 34 clauses, 10 proposition symbols, one relation and 2 nested modal operators as maximum; for HL^-, 15 clauses, 3 proposition symbols, 3 nominals, one relation and 6 nested modal and hybrid operators as maximum; and for HL, 13 clauses, 2 proposition symbols, 2 nominals, one relation and 6 nested modal, hybrid and universal operators as maximum. Moreover, each language has fine-tuned probabilities of the different logic connectives in order to meet the SAT/UNSAT and timeout balances that the following results show. All parameters can be found in the released test script.

Results. We report below a run of the benchmark script with 1000 formulas per language, performed with DefTab with cache disabled (NC) and DefTab with cache enabled (C). DefTab was compiled with GHC 8.10.7, and the tests were run on the following platform: Ubuntu 22.04 operating system, Linux 5.19 kernel, 12th Gen Intel i7-1260P CPU with 16 cores, 16GB of RAM and SSD storage.

Formulas	Timeout					
	10 secs. (NC)	10 secs. (C)	30 secs. (NC)	30 secs. (C)	60 secs. (NC)	60 secs. (C)
CPL SAT (516)	122	135	133	144	138	146
CPL UNSAT (484)	255	324	309	364	336	384
BML SAT (462)	356	399	384	417	398	425
BML UNSAT (538)	154	193	252	324	295	367
HL^- SAT (534)	401	434	419	444	431	450
HL^- UNSAT (466)	142	153	150	170	158	183
HL SAT (480)	284	321	309	331	320	343
HL UNSAT (520)	145	161	161	183	169	193

Finally, the following table describes the outcome of checking sceptical consequence of those formulas that were originally unsatisfiable. We take therein all the tests cases that finished with timeout of 60 s, solved using caching. The column label by 'Consequence' indicates the number of formulas for which running DefTab returns it is indeed a sceptical consequence in the corresponding default theories; while 'Not Consequence' indicates the number of formulas for which DefTab returns they are not a sceptical consequence.

Formulas	Results		
	Total	Consequence	Not Consequence
CPL UNSAT	384	24	360
BML UNSAT	367	322	45
HL^- UNSAT	183	111	72
HL UNSAT	193	100	93

These results are useful for checking consistency across the execution of different provers, or provers executed with different parameters, as we are currently doing with DefTab's cache option. Moreover, we would like to compare the obtained data with the results of running other provers for the different fragments that are supported by DefTab, to assess both soundness and the performance of our tool. This is part of our future work agenda.

6 Final Remarks

We reported on DefTab, a tableaux-based system to decide sceptical consequence in Default Logic over Hybrid Modal Logic. To the best of our knowledge, DefTab is the first prover combining Modal and Default Logic. This said, other provers do exist for Default Logic. For instance, DeReS is a default logic reasoner with an underlying propositional tableaux calculus [8]. This prover is designed to check default consequence treating reasoning in the underlying logic as a "black box". This contrasts with DefTab which extends tableaux reasoning in the underlying logic with the use of defaults. At present, DefTab only supports sceptical consequence checking, while DeReS also supports credulous consequence checking. We have not been able to find a working implementation of DeReS. However, many of the ideas presented in [8] can be explored in our setting, in particular, the kind of (graph-based) problems that are used to generate test cases.

Although not a default logic reasoner, in [15], a nonmonotonic reasoning plug-in for OWL ontologies is presented. DefTab could approach this tool by implementing multiple relations (roles) and role inclusions to its underlying modal language. In [10] a tool supporting default reasoning over knowledge bases is reported, this time not via a calculus implementation but via a translation into conjunctive query programs in a Description Logic reasoner. After adapting our calculus to handle Description Logic features, it would be interesting to use the above-mentioned tools to perform a comparison with DefTab, both for correctness and performance.

We provided a systematic way of testing our tool, by introducing a test suite generation method based on hGen [2] and HTab [12,13]. This idea can be easily adapted to any kind of default prover working over CPL, BML, HL^- and HL. We tested the performance of our tool using this test suite, and empirically showed that DefTab's *subtableaux caching* optimization positive impacts on performance.

For future work there are several other interesting lines of research. The treatment of defaults in the calculus can be seen as parametric on the underlying logic (modulo some basic properties, e.g., the possibility of using premises, see [6]). DefTab was originally designed to handle Default Logic over Intuitionistic Logic [7]. Herein, the tableaux-based procedure not only handles classical reasoning instead of intuitionistic reasoning, but also it is extended to support a family of Modal Logics (i.e., the fragments we described along the paper). Moreover, our approach allowed us to design test suites that can be used to test DefTab and other nonmonotonic provers. These ideas can be extended to better assess the behaviour of the tools. We believe that our implementation is a first step towards having a modular prover that can be generalized to a wider family of Default Logics.

Acknowledgments. We thank the reviewers for their valuable comments. Our work is partially supported by the projects ANPCyT-PICT-2020-3780, ANPCyT-PICT-2021-00400, CONICET PIP 11220200100812CO, the EU Grant Agreement 101008233 (MISSION), and by the Laboratoire International Associé SINFIN.

References

1. Antoniou, G., Wang, K.: Default logic. In: Gabbay, D., Woods, J. (eds.) The many valued and nonmonotonic turn in logic. vol. 8 of Handbook of the History of Logic, pp. 517–555. North-Holland (2007)
2. Areces, C., Heguiabehere, J.: hGen: a random CNF formula generator for hybrid languages. In: Methods for Modalities 3–M4M-3, Nancy, France, Nancy, France (2003)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge U Press, Cambridge (2001)
4. Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic. Elsevier (2007)
5. Cassano, V., Areces, C., Castro, P.: Reasoning about prescription and description using prioritized default rules. In: Barthe, G., Sutcliffe, G., Veanes, M. (eds.) 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-22). vol. 57 of EPiC Series in Computing, pp. 196–213. EasyChair (2018)
6. Cassano, V., Fervari, R., Areces, C., Castro, P.F.: Interpolation and beth definability in default logics. In: Calimeri, F., Leone, N., Manna, M. (eds.) JELIA 2019. LNCS (LNAI), vol. 11468, pp. 675–691. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19570-0_44
7. Cassano, V., Fervari, R., Hoffmann, G., Areces, C., Castro, P.F.: A tableaux calculus for default intuitionistic logic. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 161–177. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_10
8. Cholewinski, P., Marek, V., Truszczynski, M.: Default reasoning system DeReS. In: 5th International Conference on Principles of Knowledge Representation and Reasoning (KR 1996), pp. 518–528. Morgan Kaufmann (1996)
9. D'Agostino, M., Gabbay, D.M., Hahnle, R., Posegga, J. (eds.) Handbook of Tableau Methods. Springer (1999). https://doi.org/10.1007/978-94-017-1754-0
10. Dao-Tran, M., Eiter, T., Krennwallner, T.: Realizing default logic over description logic knowledge bases. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS (LNAI), vol. 5590, pp. 602–613. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02906-6_52
11. Fitting, M.: Introduction. In: D'Agostino et al. [9], pp. 1–43
12. Hoffmann, G.: Lightweight hybrid tableaux. J. Appl. Logic **8**(4), 397–408 (2010)
13. Hoffmann, G., Areces, C.: HTab: a terminating tableaux system for hybrid logic. In: Areces, C., Demri, S. (eds.) Proceedings of the 5th Workshop on Methods for Modalities, M4M 2007, Cachan, France, 29–30 November 2007. vol. 231 of ENTCS, pp. 3–19. Elsevier (2007)

14. Hustadt, U., Schmidt, R.A.: Simplification and backjumping in modal tableau. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 187–201. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69778-0_22
15. Meyer, T., Moodley, K., Sattler, U.: DIP: a defeasible-inference platform for OWL ontologies. CEUR Workshop Proceedings (2014)
16. Priest, G.: An Introduction to Non-classical Logic: From If to Is. Cambridge U Press, Cambridge (2000)
17. Reiter, R.: A logic for default reasoning. AI **13**(1–2), 81–132 (1980)

Non-distributive Description Logic

Ineke van der Berg[1,2], Andrea De Domenico[1](✉), Giuseppe Greco[1], Krishna B. Manoorkar[1], Alessandra Palmigiano[1,3], and Mattia Panettiere[1]

[1] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
a.de.domenico@vu.nl

[2] Department of Mathematical Sciences, Stellenbosch University, Stellenbosch, South Africa

[3] Department of Mathematics and Applied Mathematics, University of Johannesburg, Johannesburg, South Africa

Abstract. We define LE-$\mathcal{ALC}$, a generalization of the description logic $\mathcal{ALC}$ based on the propositional logic of general (i.e. not necessarily distributive) lattices, and semantically interpreted on relational structures based on formal contexts from Formal Concept Analysis (FCA). The description logic LE-$\mathcal{ALC}$ allows us to formally describe databases with objects, features, and formal concepts, represented according to FCA as Galois-stable sets of objects and features. We describe ABoxes and TBoxes in LE-$\mathcal{ALC}$, provide a tableaux algorithm for checking the consistency of LE-$\mathcal{ALC}$ knowledge bases with acyclic TBoxes, and show its termination, soundness and completeness. Interestingly, consistency checking for LE-$\mathcal{ALC}$ with acyclic TBoxes is in PTIME, while the complexity of the consistency checking of classical $\mathcal{ALC}$ with acyclic TBoxes is PSPACE-complete.

Keywords: Description logic · Tableaux algorithm · Formal Concept Analysis · LE-logics

1 Introduction

Description Logic (DL) [2] is a class of logical formalisms, typically based on classical first-order logic, and widely used in Knowledge Representation and Reasoning to describe and reason about relevant concepts in a given application domain and their relationships. Since certain laws of classical logic fail in certain application domains, in recent years, there has been a growing interest in developing versions of description logics on weaker (non-classical) propositional bases. For instance, in [20], an intuitionistic version of the DL $\mathcal{ALC}$ has been introduced for resolving some inconsistencies arising from the classical law of excluded middle when applying $\mathcal{ALC}$ to legal domains. In [6,19], many-valued

This paper is partially funded by the EU MSCA (grant No. 101007627). The first author is funded by the National Research Foundation of South Africa (grant No. 140841). The third and fourth authors are partially funded by the NWO grant KIVI.2019.001.

(fuzzy) description logics have been introduced to account for uncertainty and imprecision in processing information in the Semantic Web, and recently, frameworks of non-monotonic description logics have been introduced [14,15,18].

One domain of application in which there is no consensus as to how classical logic should be applied is Formal Concept Analysis (FCA). In this setting, formal concepts arise from formal contexts $\mathbb{P} = (A, X, I)$, where A and X are sets (of objects and features respectively), and $I \subseteq A \times X$. Specifically, formal concepts are represented as Galois-stable tuples (B, Y) such that $B \subseteq A$ and $Y \subseteq X$ and $B = \{a \in A \mid \forall y(y \in Y \Rightarrow aIy)\}$ and $Y = \{x \in X \mid \forall b(b \in B \Rightarrow bIx)\}$. The formal concepts arising from a formal context are naturally endowed with a partial order (the sub-concept/super-concept relation) as follows: $(B_1, Y_1) \leq (B_2, Y_2)$ iff $B_1 \subseteq B_2$ iff $Y_2 \subseteq Y_1$. This partial order is a complete lattice, which is in general non-distributive. The failure of distributivity in the lattice of formal concepts introduces a tension between classical logic and the natural logic of formal concepts in FCA. This failure motivated the introduction of lattice-based propositional (modal) logics as the (epistemic) logics of formal concepts [9,10]. Complete relational semantics of these logics is given by *enriched formal contexts* (cf. Sect. 2.2), relational structures $\mathbb{F} = (\mathbb{P}, \mathcal{R}_\Box, \mathcal{R}_\Diamond)$ based on formal contexts.

In this paper, we introduce LE-$\mathcal{ALC}$, a lattice-based version of $\mathcal{ALC}$ which stands in the same relation to the lattice-based modal logic of formal concepts [12] as classical $\mathcal{ALC}$ stands in relation to classical modal logic: the language and semantics of LE-$\mathcal{ALC}$ is based on enriched formal contexts and their associated modal algebras. Thus, just like the language of $\mathcal{ALC}$ can be seen as a hybrid modal logic language interpreted on Kripke frames, the language of LE-$\mathcal{ALC}$ can be regarded as a hybrid modal logic language interpreted on enriched formal contexts.

FCA and DL are different and well known approaches in the formal representation of concepts (or categories). They have been used together for several purposes [1,4,17]. Thus, providing a DL framework which allows us to describe formal contexts (possibly enriched, e.g. with additional relations on them) would be useful in relating these frameworks both at a theoretical and at a practical level. Proposals to connect FCA and DL have been made, in which concept lattices serve as models for DL concepts. Shilov and Han [21] interpret the positive fragment of $\mathcal{ALC}$ concept names over concept lattices and show that this interpretation is compatible with standard Kripke models for $\mathcal{ALC}$. A similar approach is used by Wrum [22] in which complete semantics for the (full) Lambek calculus is defined on concept lattices. The approach of the present paper for defining and interpreting non-distributive description logic and modal logic in relation with concept lattices with operators differs from the approaches mentioned above in that it is based on duality-theoretic insights (cf. [10]). This allows us not only to show that the DL framework introduced in the present paper is consistent with the standard DL setting and its interpretation on Kripke models, but also to show that several properties of these logics and the meaning of their formulas can also be "lifted" from the classical (distributive) to non-distributive settings (cf. [7,8,12] for extended discussions).

The main technical contribution of this paper is a tableaux algorithm for checking the consistency of LE-$\mathcal{ALC}$ ABoxes. We show that the algorithm is

terminating, sound and complete. Interestingly, this algorithm has a polynomial time complexity, compared to the complexity of the consistency checking of classical $\mathcal{ALC}$ ABoxes which is PSPACE-complete. The algorithm also constructs a model for the given ABox which is polynomial in size. Thus, it also implies that the corresponding hybrid modal logic has the finite model property.
Structure of the Paper. In Sect. 2, we give the necessary preliminaries on the DL $\mathcal{ALC}$, lattice-based modal logics and their relational semantics. In Sect. 3, we introduce the syntax and the semantics of LE-$\mathcal{ALC}$. In Sect. 4, we introduce a tableaux algorithm for checking the consistency of LE-$\mathcal{ALC}$ ABoxes and show that it is terminating, sound and complete. In Sect. 5, we conclude and discuss some future research directions.

2 Preliminaries

2.1 Description Logic $\mathcal{ALC}$

Let $\mathcal{C}$ and $\mathcal{R}$ be disjoint sets of primitive or atomic *concept names* and *role names*. The set of *concept descriptions* or compound concept names over $\mathcal{C}$ and $\mathcal{R}$ are defined recursively as follows.

$$C := A \mid \top \mid \bot \mid C \wedge C \mid C \vee C \mid \neg C \mid \exists r.C \mid \forall r.C$$

where $A \in \mathcal{C}$ and $r \in \mathcal{R}$. An *interpretation* is a tuple $\mathrm{I} = (\Delta^{\mathrm{I}}, \cdot^{\mathrm{I}})$ s.t. Δ^{I} is a non-empty set and $\cdot^{\mathrm{I}}$ maps every concept name $A \in \mathcal{C}$ to a set $A^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}}$, and every role name $r \in \mathcal{R}$ to a relation $r^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}} \times \Delta^{\mathrm{I}}$. This mapping extends to all concept descriptions as follows:

$$\begin{aligned} \top^{\mathrm{I}} &= \Delta^{\mathrm{I}} & \bot^{\mathrm{I}} &= \varnothing \\ (C \wedge D)^{\mathrm{I}} &= C^{\mathrm{I}} \cap D^{\mathrm{I}} & (C \vee D)^{\mathrm{I}} &= C^{\mathrm{I}} \cup D^{\mathrm{I}} \\ (\exists r.C)^{\mathrm{I}} &= \{d \in \Delta^{\mathrm{I}} \mid \exists e((d,e) \in r^{\mathrm{I}} \;\&\; e \in C^{\mathrm{I}}\} & (\neg C)^{\mathrm{I}} &= \Delta^{\mathrm{I}} \setminus C^{\mathrm{I}} \\ (\forall r.C)^{\mathrm{I}} &= \{d \in \Delta^{\mathrm{I}} \mid \forall e((d,e) \in r^{\mathrm{I}} \Rightarrow e \in C^{\mathrm{I}}\} \end{aligned}$$

Let $\mathcal{S}$ be a set of individual names disjoint from $\mathcal{C}$ and $\mathcal{R}$, such that for every a in $\mathcal{S}$, $a^{\mathrm{I}} \in \Delta^{\mathrm{I}}$. For any $a, b \in \mathcal{S}$, any $C \in \mathcal{C}$ and $r \in \mathcal{R}$, an expression of the form $a : C$ (resp. $(a, b) : r$) is an $\mathcal{ALC}$ *concept assertion* (resp. *role assertion*). A finite set of $\mathcal{ALC}$ concept and role assertions is an $\mathcal{ALC}$ *ABox*. An assertion $a : C$ (resp. $(a, b) : r$) is *satisfied* in an interpretation I if $a^{\mathrm{I}} \in C^{\mathrm{I}}$ (resp. if $(a^{\mathrm{I}}, b^{\mathrm{I}}) \in r^{\mathrm{I}}$). An $\mathcal{ALC}$ *TBox* is a finite set of expressions of the form $C_1 \equiv C_2$. An interpretation I *satisfies* $C_1 \equiv C_2$ iff $C_1^{\mathrm{I}} = C_2^{\mathrm{I}}$. An $\mathcal{ALC}$ *knowledge base* is a tuple $(\mathcal{A}, \mathcal{T})$, where $\mathcal{A}$ is an $\mathcal{ALC}$ ABox, and $\mathcal{T}$ is an $\mathcal{ALC}$ TBox. An interpretation I is a *model* for a knowledge base $(\mathcal{A}, \mathcal{T})$ iff it satisfies all members of $\mathcal{A}$ and $\mathcal{T}$. A knowledge base $(\mathcal{A}, \mathcal{T})$ is *consistent* if there is a model for it. An ABox $\mathcal{A}$ (resp. TBox $\mathcal{T}$) is *consistent* if the knowledge base $(\mathcal{A}, \varnothing)$ (resp. $(\varnothing, \mathcal{T})$) is consistent.

An $\mathcal{ALC}$ *concept definition* in T is an expression of the form $A \equiv C$ where A is an atomic concept. We say that A *directly uses* B if there is a concept definition $A \equiv C$ in $\mathcal{T}$ such that B occurs in C. We say that A *uses* B if A directly uses B, or if there is a concept name B' such that A uses B' and B' directly uses B. A finite set $\mathcal{T}$ of concept definitions is an *acyclic* TBox if

1. there is no concept name in $\mathcal{T}$ that uses itself,
2. no concept name occurs more than once on the left-hand side of a concept definition in $\mathcal{T}$.

Checking the consistency of a knowledge base is a key problem in description logics, usually solved via tableaux algorithms. In the $\mathcal{ALC}$ case, checking the consistency of any knowledge base is EXPTIME-complete while checking the consistency of a knowledge base with acyclic TBoxes is PSPACE-complete [2].

2.2 Basic Normal Non-distributive Modal Logic and Its Semantics

The logic introduced in this section is part of a family of lattice-based logics, sometimes referred to as *LE-logics* (cf. [11]), which have been studied in the context of a research program on the logical foundations of categorization theory [8–10,12]. Let Prop be a (countable) set of atomic propositions. The language $\mathcal{L}$ is defined as follows:

$$\varphi := \bot \mid \top \mid p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi,$$

where $p \in \mathsf{Prop}$, and $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$ for finite sets $\mathcal{F}$ and $\mathcal{G}$ of unary $\Diamond$-type (resp. $\Box$-type) modal operators. The *basic*, or *minimal normal $\mathcal{L}$-logic* is a set $\mathbf{L}$ of sequents $\varphi \vdash \psi$, with $\varphi, \psi \in \mathcal{L}$, containing the following axioms for every $\Box \in \mathcal{F}$ and $\Diamond \in \mathcal{G}$:

$$\begin{array}{cccccc} p \vdash p & \bot \vdash p & p \vdash p \vee q & p \wedge q \vdash p & \top \vdash \Box\top & \Box p \wedge \Box q \vdash \Box(p \wedge q) \\ & p \vdash \top & q \vdash p \vee q & p \wedge q \vdash q & \Diamond\bot \vdash \bot & \Diamond(p \vee q) \vdash \Diamond p \vee \Diamond q \end{array}$$

and closed under the following inference rules:

$$\frac{\varphi \vdash \chi \quad \chi \vdash \psi}{\varphi \vdash \psi} \quad \frac{\varphi \vdash \psi}{\varphi(\chi/p) \vdash \psi(\chi/p)} \quad \frac{\chi \vdash \varphi \quad \chi \vdash \psi}{\chi \vdash \varphi \wedge \psi} \quad \frac{\varphi \vdash \chi \quad \psi \vdash \chi}{\varphi \vee \psi \vdash \chi} \quad \frac{\varphi \vdash \psi}{\Box\varphi \vdash \Box\psi} \quad \frac{\varphi \vdash \psi}{\Diamond\varphi \vdash \Diamond\psi}$$

Note that unlike in classical modal logic, we cannot assume that $\Box$ and $\Diamond$ are inter-definable in LE-logics, hence we take all connectives as primitive.

Relational Semantics. The following notation, notions and facts are from [8,12]. For any binary relation $T \subseteq U \times V$, and any $U' \subseteq U$ and $V' \subseteq V$, we let T^c denote the set-theoretic complement of T in $U \times V$, and

$$T^{(1)}[U'] := \{v \mid \forall u(u \in U' \Rightarrow uTv)\} \qquad T^{(0)}[V'] := \{u \mid \forall v(v \in V' \Rightarrow uTv)\}. \quad (1)$$

In what follows, we fix two sets A and X, and use a, b (resp. x, y) for elements of A (resp. X), and B, C, A_j (resp. Y, W, X_j) for subsets of A (resp. of X).

A *polarity* or *formal context* (cf. [13]) is a tuple $\mathbb{P} = (A, X, I)$, where A and X are sets, and $I \subseteq A \times X$ is a binary relation. Intuitively, formal contexts can be understood as abstract representations of databases [13], so that A and X represent collections of *objects* and *features*, and for any object a and feature x, the tuple (a, x) belongs to I exactly when object a has feature x.

As is well known, for every formal context $\mathbb{P} = (A, X, I)$, the pair of maps

$$(\cdot)^{\uparrow} : \mathcal{P}(A) \to \mathcal{P}(X) \quad \text{and} \quad (\cdot)^{\downarrow} : \mathcal{P}(X) \to \mathcal{P}(A),$$

defined by the assignments $B^{\uparrow} := I^{(1)}[B]$ and $Y^{\downarrow} := I^{(0)}[Y]$, form a Galois connection, and hence induce the closure operators $(\cdot)^{\uparrow\downarrow}$ and $(\cdot)^{\downarrow\uparrow}$ on $\mathcal{P}(A)$ and on $\mathcal{P}(X)$ respectively. The fixed points of $(\cdot)^{\uparrow\downarrow}$ and $(\cdot)^{\downarrow\uparrow}$ are the *Galois-stable* sets. A *formal concept* of a polarity $\mathbb{P} = (A, X, I)$ is a tuple $c = (B, Y)$ such that $B \subseteq A$ and $Y \subseteq X$, and $B = Y^{\downarrow}$ and $Y = B^{\uparrow}$. The subset B (resp. Y) is the *extension* (resp. the *intension*) of c and is denoted by $\llbracket c \rrbracket$ (resp. $\llparenthesis c \rrparenthesis$). It is well known (cf. [13]) that the sets B and Y are Galois-stable, and that the set of formal concepts of a polarity $\mathbb{P}$, with the order defined by

$$c_1 \leq c_2 \quad \text{iff} \quad \llbracket c_1 \rrbracket \subseteq \llbracket c_2 \rrbracket \quad \text{iff} \quad \llparenthesis c_2 \rrparenthesis \subseteq \llparenthesis c_1 \rrparenthesis,$$

forms a complete lattice $\mathbb{P}^{+}$, namely the *concept lattice* of $\mathbb{P}$.

For the language $\mathcal{L}$ defined above, an *enriched formal $\mathcal{L}$-context* is a tuple $\mathbb{F} = (\mathbb{P}, \mathcal{R}_{\Box}, \mathcal{R}_{\Diamond})$, where $\mathcal{R}_{\Box} = \{R_{\Box} \subseteq A \times X \mid \Box \in \mathcal{G}\}$ and $\mathcal{R}_{\Diamond} = \{R_{\Diamond} \subseteq X \times A \mid \Diamond \in \mathcal{F}\}$ are sets of *I-compatible* relations, that is, for all $\Box \in \mathcal{G}$, $\Diamond \in \mathcal{F}$, $a \in A$, and $x \in X$, the sets $R_{\Box}^{(0)}[x]$, $R_{\Box}^{(1)}[a]$, $R_{\Diamond}^{(0)}[a]$, $R_{\Diamond}^{(1)}[x]$ are Galois-stable in $\mathbb{P}$. For each $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$, their associated relations $R_{\Box}$ and $R_{\Diamond}$ provide their corresponding semantic interpretations as operations $[R_{\Box}]$ and $\langle R_{\Diamond} \rangle$ on the concept lattice $\mathbb{P}^{+}$ defined as follows: For any $c \in \mathbb{P}^{+}$,

$$[R_{\Box}]c = (R_{\Box}^{(0)}[\llparenthesis c \rrparenthesis], I^{(1)}[R_{\Box}^{(0)}[\llparenthesis c \rrparenthesis]]) \quad \text{and} \quad \langle R_{\Diamond} \rangle c = (I^{(0)}[R_{\Diamond}^{(0)}[\llbracket c \rrbracket]], R_{\Diamond}^{(0)}[\llbracket c \rrbracket]).$$

We refer to the algebra $\mathbb{F}^{+} = (\mathbb{P}^{+}, \{[R_{\Box}]\}_{\Box \in \mathcal{G}}, \{\langle R_{\Diamond} \rangle\}_{\Diamond \in \mathcal{F}})$ as the *complex algebra* of $\mathbb{F}$.

A *valuation* on such an $\mathbb{F}$ is a map $V \colon \mathsf{Prop} \to \mathbb{P}^{+}$. For each $p \in \mathsf{Prop}$, we let $\llbracket p \rrbracket := \llbracket V(p) \rrbracket$ (resp. $\llparenthesis p \rrparenthesis := \llparenthesis V(p) \rrparenthesis$) denote the extension (resp. intension) of the interpretation of p under V.

A *model* is a tuple $\mathbb{M} = (\mathbb{F}, V)$ where $\mathbb{F} = (\mathbb{P}, \mathcal{R}_{\Box}, \mathcal{R}_{\Diamond})$ is an enriched formal context and V is a valuation on $\mathbb{F}$. For every $\varphi \in \mathcal{L}$, we let $\llbracket \varphi \rrbracket_{\mathbb{M}} := \llbracket V(\varphi) \rrbracket$ (resp. $\llparenthesis \varphi \rrparenthesis_{\mathbb{M}} := \llparenthesis V(\varphi) \rrparenthesis$) denote the extension (resp. intension) of the interpretation of φ under the homomorphic extension of V. The following 'forcing' relations can be recursively defined as follows:

$$\begin{array}{llll}
\mathbb{M}, a \Vdash p & \text{iff } a \in \llbracket p \rrbracket_{\mathbb{M}} & \mathbb{M}, x \succ p & \text{iff } x \in \llparenthesis p \rrparenthesis_{\mathbb{M}} \\
\mathbb{M}, a \Vdash \top & \text{always} & \mathbb{M}, x \succ \top & \text{iff } aIx \text{ for all } a \in A \\
\mathbb{M}, x \succ \bot & \text{always} & \mathbb{M}, a \Vdash \bot & \text{iff } aIx \text{ for all } x \in X \\
\mathbb{M}, a \Vdash \varphi \wedge \psi & \text{iff } \mathbb{M}, a \Vdash \varphi \text{ and } \mathbb{M}, a \Vdash \psi & \mathbb{M}, x \succ \varphi \wedge \psi & \text{iff } (\forall a \in A)\ (\mathbb{M}, a \Vdash \varphi \wedge \psi \Rightarrow aIx) \\
\mathbb{M}, x \succ \varphi \vee \psi & \text{iff } \mathbb{M}, x \succ \varphi \text{ and } \mathbb{M}, x \succ \psi & \mathbb{M}, a \Vdash \varphi \vee \psi & \text{iff } (\forall x \in X)\ (\mathbb{M}, x \succ \varphi \vee \psi \Rightarrow aIx).
\end{array}$$

As to the interpretation of modal formulas, for every $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$:

$$\begin{array}{llll}
\mathbb{M}, a \Vdash \Box\varphi & \text{iff } (\forall x \in X)(\mathbb{M}, x \succ \varphi \Rightarrow aR_{\Box}x) & \mathbb{M}, x \succ \Box\varphi & \text{iff } (\forall a \in A)(\mathbb{M}, a \Vdash \Box\varphi \Rightarrow aIx) \\
\mathbb{M}, x \succ \Diamond\varphi & \text{iff for all } a \in A, if \mathbb{M}, a \Vdash \varphi \text{ then } xR_{\Diamond}a & \mathbb{M}, a \Vdash \Diamond\varphi & \text{iff } (\forall x \in X)(\mathbb{M}, x \succ \Diamond\varphi \Rightarrow aIx)
\end{array}$$

The definition above ensures that, for any $\mathcal{L}$-formula φ,

$$\mathbb{M}, a \Vdash \varphi \text{ iff } a \in [\![\varphi]\!]_{\mathbb{M}}, \quad \text{and} \quad \mathbb{M}, x \succ \varphi \text{ iff } x \in (\![\varphi]\!)_{\mathbb{M}}.$$

$$\mathbb{M} \models \varphi \vdash \psi \quad \text{iff} \quad [\![\varphi]\!]_{\mathbb{M}} \subseteq [\![\psi]\!]_{\mathbb{M}} \quad \text{iff} \quad (\![\psi]\!)_{\mathbb{M}} \subseteq (\![\varphi]\!)_{\mathbb{M}}.$$

The interpretation of the propositional connectives $\vee$ and $\wedge$ in the framework described above reproduces the standard notion of join and the meet of formal concepts used in FCA. The interpretation of the operators $\Box$ and $\Diamond$ is motivated by algebraic properties and duality theory for modal operators on lattices (cf. [12, Sect. 3] for an expanded discussion). In [8, Proposition 3.7], it is shown that the semantics of LE-logics is compatible with Kripke semantics for classical modal logic, and thus, LE-logics are indeed generalizations of classical modal logic. This interpretation is further justified in [8, Sect. 4] by noticing that, under the interpretations of the relation I as aIx iff "object a has feature x" and $R = R_\Box = R_\Diamond^{-1}$ as aRx iff "there is evidence that object a has feature x", then, for any concept c, the extents of concepts $\Box c$ and $\Diamond c$ can be interpreted as "the set of objects which *certainly* belong to c" (upper approximation), and "the set of objects which *possibly* belong to c" (lower approximation) respectively. Thus, the interpretations of $\Box$ and $\Diamond$ have similar meaning in the LE-logic as in the classical modal logic. A similar justification regarding similarity of epistemic interpretations of $\Box$ in classical and lattice-based modal logics is discussed in [9]. This transfer of meaning of modal axioms from classical modal logic to LE-logics has been investigated as a general phenomenon in [7, Sect. 4.3], [12].

3 LE Description Logic

In this section, we introduce the non-classical DL LE-$\mathcal{ALC}$, so that LE-$\mathcal{ALC}$ will be in same relation with LE-logic as $\mathcal{ALC}$ is with classical modal logic. This similarity extends to the models we will introduce for LE-$\mathcal{ALC}$: in the same way as Kripke models of classical modal logic are used as models of $\mathcal{ALC}$, enriched formal contexts, which provide complete semantics for LE-logic, will serve as models of LE-$\mathcal{ALC}$. In this specific respect, LE-$\mathcal{ALC}$ can be seen as a generalization of the positive fragment (i.e. the fragment with no negations in concept names) of $\mathcal{ALC}$ in which we do not assume distributivity laws to hold for concepts. Consequently, the language of LE-$\mathcal{ALC}$ contains individuals of two types, usually interpreted as the *objects* and *features* of the given database or categorization. Let OBJ and FEAT be disjoint sets of individual names for objects and features.

The set $\mathcal{R}$ of the role names for LE-$\mathcal{ALC}$ is the union of three disjoint sets of relations: (1) the singleton set $\{I \mid I \subseteq \mathsf{OBJ} \times \mathsf{FEAT}\}$; (2) a set $\mathcal{R}_\Box = \{R_\Box \subseteq \mathsf{OBJ} \times \mathsf{FEAT} \mid \Box \in \mathcal{G}\}$; (3) a set $\mathcal{R}_\Diamond = \{R_\Diamond \subseteq \mathsf{FEAT} \times \mathsf{OBJ} \mid \Diamond \in \mathcal{G}\}$. While I is intended to be interpreted as the incidence relation of formal concepts, and encodes information on which objects have which features, the relations in $\mathcal{R}_\Box$

and $\mathcal{R}_\Diamond$ encode additional relationships between objects and features (cf. [8] for an extended discussion).

For any set $\mathcal{C}$ of atomic concept names, the language of LE-$\mathcal{ALC}$ concepts is:

$$C := D \mid C_1 \wedge C_2 \mid C_1 \vee C_2 \mid \top \mid \bot \mid \langle R_\Diamond \rangle C \mid [R_\Box]C$$

where $D \in \mathcal{C}$, $R_\Box \in \mathcal{R}_\Box$ and $R_\Diamond \in \mathcal{R}_\Diamond$. This language matches the language of LE-logic, and has an analogous intended interpretation on the complex algebras of enriched formal contexts (cf. Sect. 2.2). As usual, $\vee$ and $\wedge$ are to be interpreted as the smallest common superconcept and the greatest common subconcept as in FCA. The constants $\top$ and $\bot$ are to be interpreted as the largest and the smallest concept, respectively. We do not include $\neg C$ as a valid concept name in our language, since there is no canonical and natural way to interpret negations in non-distributive settings.

The concept names $\langle R_\Diamond \rangle C$ and $[R_\Box]C$ in LE-$\mathcal{ALC}$ are intended to be interpreted as the operations $\langle R_\Diamond \rangle$ and $[R_\Box]$ defined by the interpretations of their corresponding role names in enriched formal contexts, analogously to the way in which $\exists r$ and $\forall r$ in $\mathcal{ALC}$ are interpreted on Kripke frames. We do not use the symbols $\forall r$ and $\exists r$ in the context of LE-$\mathcal{ALC}$ because, as discussed in Sect. 2.2, the semantic clauses of modal operators in LE-logic use universal quantifiers, and hence using the same notation verbatim would be ambiguous or misleading.

TBox assertions in LE-$\mathcal{ALC}$ are of the shape $C_1 \equiv C_2$, where C_1 and C_2 are concepts defined as above.[1] The ABox assertions are of the form:

$$aR_\Box x, \quad xR_\Diamond a, \quad aIx, \quad a : C, \quad x{::}C, \quad \neg\alpha,$$

where α is any of the first five ABox terms. We refer to the terms of first three types as *relational terms*. The interpretations of the terms $a : C$ and $x{::}C$ are: "object a is a member of concept C", and "feature x is in the description of concept C", respectively.

An *interpretation* for LE-$\mathcal{ALC}$ is a tuple $\mathrm{I} = (\mathbb{F}, \cdot^{\mathrm{I}})$, where $\mathbb{F} = (\mathbb{P}, \mathcal{R}_\Box, \mathcal{R}_\Diamond)$ is an enriched formal context, and $\cdot^{\mathrm{I}}$ maps:

1. individual names $a \in \mathsf{OBJ}$ (resp. $x \in \mathsf{FEAT}$), to some $a^{\mathrm{I}} \in A$ (resp. $x^{\mathrm{I}} \in X$);
2. relation names I, $R_\Box$ and $R_\Diamond$ to relations I^{I}, $R_\Box^{\mathrm{I}}$ and $R_\Diamond^{\mathrm{I}}$ in $\mathbb{F}$;
3. any primitive concept D to $D^{\mathrm{I}} \in \mathbb{F}^+$, and other concepts as follows:

$$\begin{array}{lll}
\bot^{\mathrm{I}} = (X^{\downarrow}, X) & \top^{\mathrm{I}} = (A, A^{\uparrow}) & (C_1 \wedge C_2)^{\mathrm{I}} = C_1^{\mathrm{I}} \wedge C_2^{\mathrm{I}} \\
(C_1 \vee C_2)^{\mathrm{I}} = C_1^{\mathrm{I}} \vee C_2^{\mathrm{I}} & ([R_\Box]C)^{\mathrm{I}} = [R_\Box^{\mathrm{I}}]C^{\mathrm{I}} & (\langle R_\Diamond \rangle C)^{\mathrm{I}} = \langle R_\Diamond^{\mathrm{I}} \rangle C^{\mathrm{I}}
\end{array}$$

where the operators $[R_\Box^{\mathrm{I}}]$ and $\langle R_\Diamond^{\mathrm{I}} \rangle$ are defined as in Sect. 2.2.

The satisfiability relation for an interpretation I is defined as follows:

1. $\mathrm{I} \models C_1 \equiv C_2$ iff $[\![C_1^{\mathrm{I}}]\!] = [\![C_2^{\mathrm{I}}]\!]$ iff $(\![C_2^{\mathrm{I}}]\!) = (\![C_1^{\mathrm{I}}]\!)$.

[1] As is standard in DL (cf. [2] for more details), general concept inclusion of the form $C_1 \sqsubseteq C_2$ can be rewritten as concept definition $C_1 \equiv C_2 \wedge C_3$, where C_3 is a new concept name.

2. $\mathrm{I} \models a : C$ iff $a^{\mathrm{I}} \in [\![C^{\mathrm{I}}]\!]$ and $\mathrm{I} \models x{::}C$ iff $x^{\mathrm{I}} \in (\![C^{\mathrm{I}}]\!)$.
3. $\mathrm{I} \models aIx$ (resp. $aR_\Box x$, $xR_\Diamond a$) iff $a^{\mathrm{I}} I^{\mathrm{I}} x^{\mathrm{I}}$ (resp. $a^{\mathrm{I}} R_\Box^{\mathrm{I}} x^{\mathrm{I}}$, $x^{\mathrm{I}} R_\Diamond^{\mathrm{I}} a^{\mathrm{I}}$).
4. $\mathrm{I} \models \neg\alpha$, where α is any ABox term, iff $\mathrm{I} \not\models \alpha$.

An interpretation I is a *model* for an LE-$\mathcal{ALC}$ knowledge base $(\mathcal{A}, \mathcal{T})$ if $\mathrm{I} \models \mathcal{A}$ and $\mathrm{I} \models \mathcal{T}$.

The framework of LE-$\mathcal{ALC}$ formally brings FCA and DL together in two important ways: (1) the concepts of LE-$\mathcal{ALC}$ are naturally interpreted as formal concepts in FCA; (2) the language of LE-$\mathcal{ALC}$ is designed to represent knowledge and reasoning in the setting of enriched formal contexts.

4 Tableaux Algorithm for ABox of LE-$\mathcal{ALC}$

In this section, we define a tableaux algorithm for checking the consistency of LE-$\mathcal{ALC}$ ABoxes. An LE-$\mathcal{ALC}$ ABox $\mathcal{A}$ contains a *clash* iff it contains both β and $\neg\beta$ for some relational term β. The expansion rules below are designed so that the expansion of $\mathcal{A}$ will contain a clash iff $\mathcal{A}$ is inconsistent. The set $sub(C)$ of sub-formulas of any LE-$\mathcal{ALC}$ concept name C is defined as usual.

A concept name C' *occurs* in $\mathcal{A}$ (in symbols: $C' \in \mathcal{A}$) if $C' \in sub(C)$ for some C such that one of the terms $a : C$, $x{::}C$, $\neg a : C$, or $\neg x{::}C$ is in $\mathcal{A}$. A constant b (resp. y) *occurs* in $\mathcal{A}$ ($b \in \mathcal{A}$, or $y \in \mathcal{A}$), iff some term containing b (resp. y) occurs in it.

The tableaux algorithm below constructs a model $(\mathbb{F}, \cdot^{\mathrm{I}})$ for every consistent $\mathcal{A}$, where $\mathbb{F} = (\mathbb{P}, \mathcal{R}_\Box, \mathcal{R}_\Diamond)$ is such that, for any $C \in \mathcal{A}$, some $a_C \in A$ and $x_C \in X$ exist such that, for any $a \in A$ (resp. any $x \in X$), $a \in [\![C^{\mathrm{I}}]\!]$ (resp. $x \in (\![C]\!)^{\mathrm{I}}$) iff aIx_C (resp. $a_C Ix$). We call a_C and x_C the *classifying object* and the *classifying feature* of C, respectively. To make our notation more easily readable, we will write $a_{\Box C}$, $x_{\Box C}$ (resp. $a_{\Diamond C}$, $x_{\Diamond C}$) instead of $a_{[R_\Box]C}$, $x_{[R_\Box]C}$ (resp. $a_{\langle R_\Diamond \rangle C}$, $x_{\langle R_\Diamond \rangle C}$) Moreover, for every $R_\Box \in \mathcal{R}_\Box$ and $R_\Diamond \in \mathcal{R}_\Diamond$, we will also impose the condition that $a \in [\![[R_\Box]C]\!]$ (resp. $x \in (\![\langle R_\Diamond \rangle C]\!)$) iff $aR_\Box x_C$ (resp. $xR_\Diamond a_C$), where a_C and x_C are the classifying object and the classifying feature of C, respectively. Note that we can always assume w.l.o.g. that any consistent ABox $\mathcal{A}$ is satisfiable in a model with classifying objects and features (cf. Theorem 3).

Algorithm 1. tableaux algorithm for checking LE-$\mathcal{ALC}$ ABox consistency

```
Input: An LE-$\mathcal{ALC}$ ABox $\mathcal{A}$.    Output: whether $\mathcal{A}$ is inconsistent.
1: if there is a clash in $\mathcal{A}$ then return "inconsistent".
2: if no expansion rule is applicable to $\mathcal{A}$ then return "consistent".
3: pick any applicable expansion rule $R$, apply $R$ to $\mathcal{A}$ and proceed recursively.
```

Below, we list the expansion rules. The commas in each rule are metalinguistic conjunctions, hence every tableau is non-branching.

Creation rule **Basic rule** **Rules for $\top$ and $\bot$**

$$\frac{\text{For any } C \in \mathcal{A}}{a_C : C, \quad x_C :: C}\,\text{create} \qquad I\,\frac{b : C, \quad y :: C}{bIy} \qquad \top\,\frac{}{b : \top} \qquad \bot\,\frac{}{y :: \bot}$$

Rules for the logical connectives

$$\vee_A\,\frac{b : C_1 \vee C_2, \quad y :: C_1, \quad y :: C_2}{bIy} \qquad \frac{b : C_1 \wedge C_2}{b : C_1, \quad b : C_2}\,\wedge_A \qquad \vee_X\,\frac{y :: C_1 \vee C_2}{y :: C_1, \quad y :: C_2}$$

$$\frac{y :: C_1 \wedge C_2, \quad b : C_1, \quad b : C_2}{bIy}\,\wedge_X \qquad \Box\,\frac{b : [R_\Box]C, \quad y :: C}{bR_\Box y} \qquad \frac{y :: \langle R_\Diamond\rangle C, \quad b : C}{yR_\Diamond b}\,\Diamond$$

Adjunction rules

$$adj_\Box\,\frac{\blacklozenge b : C}{b : [R_\Box]C} \qquad \frac{\blacksquare y :: C}{\langle R_\Diamond\rangle C :: y}\,adj_\Diamond \qquad R_\Box\,\frac{bR_\Box y}{\blacklozenge bIy, \quad bI\Box y} \qquad \frac{yR_\Diamond b}{\Diamond bIy, \quad bI\blacksquare y}\,R_\Diamond$$

Basic rules for negative assertions **Appending rules**

$$\neg b\,\frac{\neg(b : C)}{\neg(bIx_C)} \qquad \frac{\neg(x :: C)}{\neg(a_C Ix)}\,\neg x \qquad x_C\,\frac{bIx_C}{b : C} \qquad \frac{a_C Iy}{y :: C}\,a_C$$

In rules $\top$ and $\bot$, b and y are any objects or features occurring in the tableau. In the adjunction rules the individuals $\blacklozenge b$, $\Diamond b$, $\Box y$, and $\blacksquare y$ are new and unique for each relation $R_\Box$ and $R_\Diamond$, except for $\Diamond a_C = a_{\Diamond C}$ and $\Box x_C = x_{\Box C}$.

The basic rule and the logical rules for the connectives encode the semantics of the logical connectives in LE-$\mathcal{ALC}$. The creation rule makes sure that, whenever successful, the algorithm outputs models with classifying object a_C and feature x_C for every concept name $C \in \mathcal{A}$. The adjunction rules imply that every $R_\Box \in \mathcal{R}_\Box$ and $R_\Diamond \in \mathcal{R}_\Diamond$ are I-compatible. Appending and negative assertion rules encode the defining property of classifying objects and features of concepts.

Remark 1 (Branching). Note that no expansion rule above involves branching. Thus, unlike tableaux algorithms for $\mathcal{ALC}$, Algorithm 1 does not involve any branching. New elements are added to $\mathcal{A}$ only via adjunction and creation rules.

Example 1. Let $\mathcal{A} = \{b : [R_\Box][R_\Box]C_1, b : [R_\Box][R_\Box]C_2, y{::}[R_\Box](C_1 \wedge C_2), \neg(bR_\Box y)\}$. It is easy to check that $\mathcal{A}$ has no LE-$\mathcal{ALC}$ model. The algorithm applies on $\mathcal{A}$ as follows (We only do the partial expansion to show that the clash exists):

Rule	Premises	Added terms	
Creation		$x_{\Box C_1}{::}[R_\Box]C_1$, $x_{\Box C_2}{::}[R_\Box]C_2$, $x_{C_1\wedge C_2}{::}C_1 \wedge C_2$	
$\Box$	$x_{\Box C_i}{::}[R_\Box]C_i$, $b{:}[R_\Box][R_\Box]C_i$	$bR_\Box x_{\Box C_i}$	$i = 1, 2$
$R_\Box$	$bR_\Box x_{\Box C_i}$	$\blacklozenge bIx_{\Box C_i}$	$i = 1, 2$
Appending	$\blacklozenge bIx_{\Box C_i}$	$\blacklozenge b : [R_\Box]C_i$	$i = 1, 2$

By applying the same process to $\blacklozenge b : [R_\Box]C_1$, $\blacklozenge b : [R_\Box]C_2$ and $x_{\Box C_1}{::}[R_\Box]C_1$, $x_{\Box C_2}{::}[R_\Box]C_2$, we add the terms $\blacklozenge\blacklozenge b : C_1$ and $\blacklozenge\blacklozenge b : C_2$ to the tableau. Then the further tableau expansion is as follows:

Rule	Premises	Added terms
$\wedge_X$	$x_{C_1\wedge C_2}{::}C_1 \wedge C_2$, $\blacklozenge\blacklozenge b : C_1$, $\blacklozenge\blacklozenge b : C_2$, $\blacklozenge\blacklozenge b : C_1$	$\blacklozenge\blacklozenge bIx_{C_1\wedge C_2}$
Appending	$\blacklozenge\blacklozenge bIx_{C_1\wedge C_2}$	$\blacklozenge\blacklozenge b : C_1 \wedge C_2$
$adj_\Box$ (twice)	$\blacklozenge\blacklozenge b : C_1 \wedge C_2$	$b : [R_\Box][R_\Box](C_1 \wedge C_2)$
$\Box$	$b : [R_\Box][R_\Box](C_1 \wedge C_2)$, $y{::}[R_\Box](C_1 \wedge C_2)$	$bR_\Box y$

Thus, there is a clash between $\neg(bR_\Box y)$ and $bR_\Box y$ in the expansion.

Example 2. Let $\mathcal{A} = \{\neg(bIy), y{::}C_1, \neg(b : C_2), b : C_1 \vee C_2, bR_\Box y\}$. The following table shows the tableau expansion for $\mathcal{A}$. Let $\mathcal{W} := \{C_1, C_2, C_1 \vee C_2\}$.

Rule	Premises	Added terms
Initial		$\neg(bIy), y{::}C_1, \neg(b : C_2), b : C_1 \vee C_2, bR_\Box y$
Creation		$a_C : C$, $x_C{::}C$, $C \in \mathcal{W}$
Basic	$a_C : C$, $x_C{::}C$, $C \in \mathcal{W}$	$a_C I x_C$, $C \in \mathcal{W}$
Appending	$a_{C_1} I x_{C_1\vee C_2}$, $a_{C_2} I x_{C_1\vee C_2}$	$a_{C_1} : C_1 \vee C_2$, $a_{C_2} : C_1 \vee C_2$
$\vee_X$	$x_{C_1\vee C_2}{::}C_1 \vee C_2$	$x_{C_1\vee C_2}{::}C_1$, $x_{C_1\vee C_2}{::}C_2$
Basic	$a_{C_1}{::}C_1 \vee C_2$, $x_{C_1\vee C_2}{::}C_1$	$a_{C_1} I x_{C_1\vee C_2}$
Basic	$a_{C_2}{::}C_1 \vee C_2$, $x_{C_1\vee C_2}{::}C_1$	$a_{C_2} I x_{C_1\vee C_2}$
$R_\Box$	$bR_\Box y$	$\blacklozenge bIy$, $bI\Box y$
$\neg_b$	$\neg(b : C_1)$	$\neg(bIx_{C_2})$

Note that no expansion rule is applicable anymore. It is clear that the tableau does not contain any clashes. Thus, this ABox has a model. By the procedure described in Sect. 4.2, this model is given by $\mathcal{R}_\Box = \{R_\Box\}, \mathcal{R}_\Diamond = \{R_\Diamond\}, A = \{a_{C_1}, a_{C_2}, a_{C_1\vee C_2}, b, \blacklozenge b\}$, $X = \{x_{C_1}, x_{C_2}, x_{C_1\vee C_2}, y, \Box y\}$, $I = \{(a_C, x_C)_{C\in\mathcal{W}}, (a_{C_1}, x_{C_1\vee C_2}), (a_{C_2}, x_{C_1\vee C_2}), (\blacklozenge b, y), (b, \Box y)\}$, $R_\Box = \{(b, y)\}$, $R_\Diamond = \varnothing$.

4.1 Termination of the Tableaux Algorithm

In this section, we show that Algorithm 1 always terminates for any finite LE-$\mathcal{ALC}$ ABox $\mathcal{A}$. Since no rule branches out, we only need to check that the number of new individuals added by the expansion rules is finite. Note that the only rules for adding new individuals are the creation and adjunction rules. The creation rules add one new object and feature for every concept C occurring in the expansion of $\mathcal{A}$. Thus, it is enough to show that the number of individuals and new concepts added by applying adjunction rules is finite. To do so, we will show that any individual constant introduced by means of any adjunction rule will contain only finitely many modal operators applied to a constant occurring in $\mathcal{A}$ or added by the creation rule and any new concept name added will contain finitely many $\Box$ and $\Diamond$ operators applied to a concept occurring in $\mathcal{A}$.

Definition 1. *The $\Diamond$-depth $\Diamond_\mathcal{D}$ and $\Box$-depth $\Box_\mathcal{D}$ of C is defined as follows:*

1. *if C is an atomic concept, then $\Diamond_{\mathcal{D}}(C) = \Box_{\mathcal{D}}(C) = 0$;*
2. *$\Diamond_{\mathcal{D}}(\langle R_\Diamond \rangle C) = \Diamond_{\mathcal{D}}(C) + 1$ and $\Box_{\mathcal{D}}(\langle R_\Diamond \rangle C) = \Box_{\mathcal{D}}(C)$;*
3. *$\Diamond_{\mathcal{D}}([R_\Box]C) = \Diamond_{\mathcal{D}}(C)$ and $\Box_{\mathcal{D}}([R_\Box]C) = \Box_{\mathcal{D}}(C) + 1$;*
4. *$\Diamond_{\mathcal{D}}(C_1 \vee C_2) = \max(\Diamond_{\mathcal{D}}(C_1), \Diamond_{\mathcal{D}}(C_2))$ and $\Box_{\mathcal{D}}(C_1 \vee C_2) = \min(\Box_{\mathcal{D}}(C_1), \Box_{\mathcal{D}}(C_2))$;*
5. *$\Diamond_{\mathcal{D}}(C_1 \wedge C_2) = \min(\Diamond_{\mathcal{D}}(C_1), \Diamond_{\mathcal{D}}(C_2))$ and $\Box_{\mathcal{D}}(C_1 \wedge C_2) = \max(\Box_{\mathcal{D}}(C_1), \Box_{\mathcal{D}}(C_2))$.*

Definition 2. *The $\Box$-depth $\Box_{\mathcal{D}}$ and $\Diamond$-depth $\Diamond_{\mathcal{D}}$ of any constants b and y are:*

1. *if $b, y \in \mathcal{A}$, $\Box_{\mathcal{D}}(b) = \Diamond_{\mathcal{D}}(b) = \Box_{\mathcal{D}}(y) = \Diamond_{\mathcal{D}}(y) = 0$;*
2. *$\Box_{\mathcal{D}}(a_C) = \Diamond_{\mathcal{D}}(x_C) = 0$, $\Diamond_{\mathcal{D}}(a_C) = -\Diamond_{\mathcal{D}}(C)$, and $\Box_{\mathcal{D}}(x_C) = -\Box_{\mathcal{D}}(C)$;*
3. *$\Box_{\mathcal{D}}(\blacklozenge b) = \Box_{\mathcal{D}}(b) + 1$, $\Box_{\mathcal{D}}(\Diamond b) = \Box_{\mathcal{D}}(b)$, $\Diamond_{\mathcal{D}}(\blacklozenge b) = \Diamond_{\mathcal{D}}(b)$, $\Diamond_{\mathcal{D}}(\Diamond b) = \Diamond_{\mathcal{D}}(b) - 1$;*
4. *$\Box_{\mathcal{D}}(\Box y) = \Box_{\mathcal{D}}(y) - 1$, $\Diamond_{\mathcal{D}}(\Box y) = \Diamond_{\mathcal{D}}(y)$, $\Box_{\mathcal{D}}(\blacksquare y) = \Box_{\mathcal{D}}(y)$, $\Diamond_{\mathcal{D}}(\blacksquare y) = \Diamond_{\mathcal{D}}(y) + 1$.*

The following lemma is key to give bounds on the $\Box$-depth and $\Diamond$-depth of new concept names added in a tableau expansion.

Lemma 1. *For any individual names b, y and for any $R_\Box \in \mathcal{R}_\Box, R_\Diamond \in \mathcal{R}_\Diamond$,*

1. *If $bR_\Box y$ is added to a tableau expansion, but $bR_\Box y \notin \mathcal{A}$, then $b : [R_\Box]C$ and $y{::}C$ already occur in a previous expansion of $\mathcal{A}$ for some C.*
2. *If $yR_\Diamond b$ is added to a tableau expansion, but $yR_\Diamond b \notin \mathcal{A}$, then $y{::}\langle R_\Diamond \rangle C$ and $b : C$ already occur in a previous expansion of $\mathcal{A}$ for some C.*
3. *If bIy is added to a tableau expansion by any rule other than the adjunction rules $R_\Box$ or $R_\Diamond$ applied to some term occurring in $\mathcal{A}$, then the tableau can (and hence, if $\mathcal{A}$ is consistent, it will at some point) be expanded with the terms $b : C$ and $y{::}C$ (in zero or more steps) for some C.*
4. *If bIy is added to the expansion as described in the previous item, then either:*
   - *(i) The terms $b : C$ and $y{::}C'$ occur in some previous expansion of $\mathcal{A}$ for some C, C' such that $\Diamond_{\mathcal{D}}(C) = \Diamond_{\mathcal{D}}(C')$ and $\Box_{\mathcal{D}}(C) = \Box_{\mathcal{D}}(C')$.*
   - *(ii) $b = \blacklozenge d$ (resp. $b = \Diamond d$) for some d, and the terms $d : [R_\Box]C$ and $y{::}C$ (resp. $y{::}\langle R_\Diamond \rangle C$ and $b : C$) occur in some previous expansion of $\mathcal{A}$ for some C.*
   - *(iii) $y = \blacksquare w$ (resp. $y = \Box w$) for some w, and the terms $w{::}\langle R_\Diamond \rangle C$ and $b : C$ (resp. $b : [R_\Box]C$ and $w{::}C$) occur in some previous expansion of $\mathcal{A}$ for some C.*
5. *If $b : C$ is added to the tableau by some expansion rule, there is $d : C'$ s.t.*
   - *(i) $d : C' \in \mathcal{A}$ or is added by applying the creation rule.*
   - *(ii) b is obtained by applying some finite combination of $\Diamond$ and $\blacklozenge$ to d.*
   - *(iii) $\Diamond_{\mathcal{D}}(C') + \Diamond_{\mathcal{D}}(d) \leq \Diamond_{\mathcal{D}}(C) + \Diamond_{\mathcal{D}}(b)$, and $\Box_{\mathcal{D}}(C) + \Box_{\mathcal{D}}(b) \leq \Box_{\mathcal{D}}(C') + \Box_{\mathcal{D}}(d)$.*
6. *If $y{::}C$ is added to the tableau by some expansion rule, there is $w{::}C'$ s.t.*
   - *(i) $w{::}C' \in \mathcal{A}$ or is added by applying the creation rule.*
   - *(ii) y is obtained by applying some finite combination of $\Box$ and $\blacksquare$ to w.*
   - *(iii) $\Diamond_{\mathcal{D}}(C) + \Diamond_{\mathcal{D}}(y) \leq \Diamond_{\mathcal{D}}(C') + \Diamond_{\mathcal{D}}(w)$, and $\Box_{\mathcal{D}}(C') + \Box_{\mathcal{D}}(w) \leq \Box_{\mathcal{D}}(C) + \Box_{\mathcal{D}}(y)$.*

Proof. Items 1 and 2 follow from the observation that new terms of the type $bR_\Box y$ and $yR_\Diamond b$ are only added through the expansion rules for terms of the forms $b : [R_\Box]C$ and $y{::}\langle R_\Diamond\rangle C$, respectively.

For item 3, the cases where bIy is introduced with the expansion rules for $b : C$ or $y{::}C$ are straightforward. If the expansion rule for $y{::}C_1 \wedge C_2$ is applied, then from the term $x_{C_1\wedge C_2}{::}C_1 \wedge C_2$ we can get $bIx_{C_1\wedge C_2}$ (since both $b : C_1$ and $b : C_2$ must be present), finally obtaining $b : C_1 \wedge C_2$ from the appending rule. The $b : C_1 \vee C_2$ case is analogous. The only other rule that can add bIy is the adjunction rule. However, note that this can only happen if $yR_\Diamond b$ or $bR_\Box y$ is present. By item 1, if the term $bR_\Box y$ is added then $b : [R_\Box]C$ and $y{::}C$ are in the tableau and it also adds the terms $\blacklozenge bIy$ and $bI\Box y$. Note that since $b : [R_\Box]C$ and $y{::}C$ are in the tableau, $\blacklozenge b : C$ and $\Box y{::}[R_\Box]C$ must also be in it. The first term can be obtained from $b : [R_\Box]C$ adding $bR_\Box x_C$ to the tableau and applying the adjunction rule and then the appending rule. Using the fact that $a_{\Box C} : [R_\Box]C$ is in the tableau after applying the creation rule, $\Box y{::}[R_\Box]C$ can be obtained similarly. Therefore, the required condition is satisfied for both $\blacklozenge bIy$ and $bI\Box y$. We can deal with the terms of the form $yR_\Diamond b$ analogously.

For item 4, the only non-trivial case is when $\blacklozenge bIy, bI\Box y$ or $\Diamond bIy, bI\blacksquare y$ are added via an adjunction rule. In the first case, $bR_\Box y$ must be present, meaning that item 1 is applicable and hence for some C, both $b : [R_\Box]C$ and $y{::}C$ appear in the tableau, satisfying the thesis. The other case is treated analogously.

We prove items 5 and 6 by simultaneous induction on the number of expansion rules applied. The rules which can add new terms of the form $b : C$ and $y{::}C$ are the expansion rules for terms of the form $b : C_1 \wedge C_2$, $y{::}C_1 \vee C_2$, the appending rules, and the adjunction rules.

If $b : C$ is obtained from $b : C \wedge C'$, either the latter is present in the original tableau and the thesis follows trivially, or the induction hypothesis applies and it follows by transitivity. The case where $y{::}C$ comes from $y{::}C \vee C'$ is analogous.

If $b : [R_\Box]C$ is obtained from $\blacklozenge b : C$ via an adjunction rule, then it suffices to apply the induction hypothesis to $\blacklozenge b : C$, noticing that no black operators can appear in the starting tableau. The adjunction case for $y{::}\langle R_\Diamond\rangle C$ is similar.

Without loss of generality, we only treat the case where the appending rule is used to add a term of the form $b : C$. Notice that for the appending rule to be applicable we must have bIx_C in the tableau. Then by item 4, either:

(i) There exist terms $b : C_1$ and $x_C{::}C_2$ in the tableau such that $\Diamond_{\mathcal{D}}(C_1) = \Diamond_{\mathcal{D}}(C_2)$ and $\Box_{\mathcal{D}}(C_1) = \Box_{\mathcal{D}}(C_2)$.

(ii) $b = \blacklozenge d$ (resp. $b = \Diamond d$) for some d, and there exist terms $d : [R_\Box]C_2$ and $x_C{::}C_2$ (resp. $x_C{::}\langle R_\Diamond\rangle C_2$ and $b : C_2$) in the tableau for some C_2.

(iii) $x_C = \blacksquare w$ (resp. $x_C = \Box w$) for some w, and there exist terms $w{::}\langle R_\Diamond\rangle C_2$ and $b : C_2$ (resp. $b : [R_\Box]C_2$ and $w{::}C_2$) in the tableau for some C_2.

In case (i), if $C \equiv C_2$, the thesis follows easily, else we apply the induction hypothesis to $x_C{::}C_2$ to find a term $w{::}C_2'$ in the original tableau such that

$$\Diamond_{\mathcal{D}}(C_1) = \Diamond_{\mathcal{D}}(C_2) + \Diamond_{\mathcal{D}}(x_C) \leq \Diamond_{\mathcal{D}}(C_2') + \Diamond_{\mathcal{D}}(w), \tag{2}$$

$$\Box_{\mathcal{D}}(C_2') + \Box_{\mathcal{D}}(w) \leq \Box_{\mathcal{D}}(C_2) + \Box_{\mathcal{D}}(x_C) = \Box_{\mathcal{D}}(C_1) - \Box_{\mathcal{D}}(C), \tag{3}$$

where x_C is obtained by applying n $\Box$-operators to w for some n (note that x_C can not be obtained by application of $\blacksquare$-operators). Thus, we have $w = x_{C_3}$ such that $C = [R_\Box]_1 \cdots [R_\Box]_n C_3$. Since $x_{C_3}::C_2'$ is in the original tableau, it must have been added by a creation rule, meaning that $C_2' \equiv C_3$. Thus, we have $\Box_{\mathcal{D}}(w) = -\Box_{\mathcal{D}}(C_2')$, $\Diamond_{\mathcal{D}}(w) = 0$, $\Diamond_{\mathcal{D}}(C_2') = \Diamond_{\mathcal{D}}(C)$, and $\Box_{\mathcal{D}}(C_2') = \Box_{\mathcal{D}}(C) - n$. Using these equalities in (3) and (2) we obtain

$$\Diamond_{\mathcal{D}}(C_1) + \Diamond_{\mathcal{D}}(b) \leq \Diamond_{\mathcal{D}}(C) + \Diamond_{\mathcal{D}}(b) \quad \text{and} \quad \Box_{\mathcal{D}}(C) + \Box_{\mathcal{D}}(b) \leq \Box_{\mathcal{D}}(C_1) + \Box_{\mathcal{D}}(b).$$

Thus, if $b : C_1 \in \mathcal{A}$, then it is the witness we needed, otherwise it is sufficient to apply the induction hypothesis to $b : C_1$, and the result follows by transitivity.

In case (ii), suppose $d : [R_\Box]C_2$ and $x_C::C_2$ are both in the tableau. If $C \equiv C_2$, then the proof follows easily applying the induction hypothesis once to $b : C_2$ if it is not in the original tableau. Otherwise, we can apply the induction hypothesis to $x_C::\langle R_\Diamond \rangle C_2$, obtaining, by the same argument as in case (i), $\Diamond_{\mathcal{D}}(C_2) \leq \Diamond_{\mathcal{D}}(C)$ and $\Box_{\mathcal{D}}(C) \leq \Box_{\mathcal{D}}(C_2)$. Therefore,
$\Diamond_{\mathcal{D}}([R_\Box]C_2) + \Diamond_{\mathcal{D}}(d) = \Diamond_{\mathcal{D}}(C_2) + \Diamond_{\mathcal{D}}(d) = \Diamond_{\mathcal{D}}(C_2) + \Diamond_{\mathcal{D}}(\blacklozenge d) \leq \Diamond_{\mathcal{D}}(C) + \Diamond_{\mathcal{D}}(b)$,
$\Box_{\mathcal{D}}(C) + \Box_{\mathcal{D}}(b) \leq \Box_{\mathcal{D}}(C_2) + \Diamond_{\mathcal{D}}(\blacklozenge d) = \Box_{\mathcal{D}}(C_2) + \Box_{\mathcal{D}}(d) + 1 = \Box_{\mathcal{D}}([R_\Box]C_2) + \Box_{\mathcal{D}}(d)$.

Thus, if $d : [R_\Box]C_2 \in \mathcal{A}$, then it is the witness we need; otherwise, it is sufficient to apply the induction hypothesis a second time to $d : [R_\Box]C_2$, and the result then follows by transitivity. The proof for the remaining subcase, where $b : C'$ and $x_C::\langle R_\Diamond \rangle C'$ are both present in the tableau, is done similarly.

The proof for case (iii) is analogous to (ii) and therefore omitted.

Definition 3. *The* $\Box$-depth *(resp.* $\Diamond$-depth*) of an ABox* $\mathcal{A}$ *is* $\Box_{\mathcal{D}}(\mathcal{A}) := \max\{\Box_{\mathcal{D}}(C') \mid C' \in \mathcal{A}\}$ *(resp.* $\Diamond_{\mathcal{D}}(\mathcal{A}) := \max\{\Diamond_{\mathcal{D}}(C') \mid C' \in \mathcal{A}\}$*).*

Corollary 1. *Let C be any concept name added to the tableau expansion at some step. Then $\Box_{\mathcal{D}}(C) \leq \Box_{\mathcal{D}}(\mathcal{A})$, and $\Diamond_{\mathcal{D}}(C) \leq \Diamond_{\mathcal{D}}(\mathcal{A})$.*

Proof. By item 5 of Lemma 1, for any $b : C$ added to the tableau we must have another term $d : C'$ in $\mathcal{A}$ or added by a creation rule, such that $\Box_{\mathcal{D}}(C) \leq \Box_{\mathcal{D}}(C) + \Box_{\mathcal{D}}(b) \leq \Box_{\mathcal{D}}(C') + \Box_{\mathcal{D}}(d) = \Box_{\mathcal{D}}(C')$. The first inequality holds because $\Box_{\mathcal{D}}(b)$ is always non-negative, and the equality follows from the fact that, as d is in the original tableau or added by a creation rule, its $\Box$-depth is zero. The proof for the $\Diamond$-depth can be shown in a similar manner using item 6 of Lemma 1.

Definition 4. *For any concept ABox term of the form $t \equiv a : C$ or $t \equiv x::C$, $size(t) = 1 + |sub(C)|$. For any relational term β, $size(\beta) = 2$. For any LE-$\mathcal{ALC}$ ABox $\mathcal{A}$, $size(\mathcal{A}) = \sum_{t \in \mathcal{A}} size(t)$.*

Theorem 1 (Termination). *For any ABox $\mathcal{A}$, the tableaux algorithm 1 terminates in a finite number of steps which is polynomial in $size(\mathcal{A})$.*

Proof. New individuals are added to the tableau only in the following ways:

(1) individuals of the form a_C or x_C can be added by creation rules;
(2) individuals of the form $\Box y$, $\blacksquare y$, $\Diamond b$, and $\blacklozenge b$ can be added through the expansions rules for $bR_\Box x$ and $yR_\Diamond a$.

As to (1), by Corollary 1, the $\Box$-depth (resp. $\Diamond$-depth) of any C appearing in an expansion of $\mathcal{A}$ is bounded by $\Box_\mathcal{D}(\mathcal{A})$ (resp. $\Diamond_\mathcal{D}(\mathcal{A})$). Moreover, no new propositional connective is ever added to create a new concept name in any of the rules. Therefore, the total number of concept names occurring in an expansion of $\mathcal{A}$ is bounded by $size(\mathcal{A}) * (\Box_\mathcal{D}(\mathcal{A}) + \Diamond_\mathcal{D}(\mathcal{A}))$. Thus, only finitely many constants of type (1) can be added.
For (2), for any individual name b added by some expansion rule, b occurs in $b : C$ for some C. By Lemma 1 (5), there is a term $d : C' \in \mathcal{A}$ s.t.

$$\Box_\mathcal{D}(b) + \Box_\mathcal{D}(C) \leq \Box_\mathcal{D}(d) + \Box_\mathcal{D}(C') = \Box_\mathcal{D}(C').$$

Therefore, $\Box_\mathcal{D}(b)$ is bounded by $\Box_\mathcal{D}(\mathcal{A})$. On the other hand, by item 6 of the same lemma we also have $0 \leq \Diamond_\mathcal{D}(C') + \Diamond_\mathcal{D}(d) \leq \Diamond_\mathcal{D}(C) + \Diamond_\mathcal{D}(b)$.

The first inequality follows from the fact that $d \in \mathcal{A}$, and thus $\Diamond_\mathcal{D}(d) = 0$ or $d = a_{C'}$, and thus $\Diamond_\mathcal{D}(d) = -\Diamond_\mathcal{D}(C')$. Therefore, we must have $-\Diamond_\mathcal{D}(C) \leq \Diamond_\mathcal{D}(b)$, meaning that $\Diamond_\mathcal{D}(b)$ is bounded below by $-\Diamond_\mathcal{D}(\mathcal{A})$. Thus, the number of connectives $\Diamond$ and $\blacklozenge$ in b is bounded by $\Box_\mathcal{D}(\mathcal{A}) + \Diamond_\mathcal{D}(\mathcal{A})$. Repeating the same argument for the individual names of type y, the total number of new constant names occurring in an expansion of $\mathcal{A}$ is bounded by $size(\mathcal{A}) * (\Box_\mathcal{D}(\mathcal{A}) + \Diamond_\mathcal{D}(\mathcal{A}))$. Thus, only finitely many constants of type (2) are added. Overall, the size of the tableau expansion (and hence the model) is $O((size(\mathcal{A}) * (\Box_\mathcal{D}(\mathcal{A}) + \Diamond_\mathcal{D}(\mathcal{A}))^2 * (|\mathcal{R}_\Box| + |\mathcal{R}_\Diamond|))$. Since the tableaux algorithm for LE-$\mathcal{ALC}$ does not involve any branching, the above theorem implies that the time complexity of checking the consistency of an LE-$\mathcal{ALC}$ ABox $\mathcal{A}$ using the tableaux algorithm is $Poly(size(\mathcal{A}))$.

4.2 Soundness of the Tableau Algorithm

For any consistent ABox $\mathcal{A}$, we let its *completion* $\overline{\mathcal{A}}$ be its maximal expansion (which exists due to termination). If there is no clash in $\overline{\mathcal{A}}$, we construct a model $(\mathbb{F}, \cdot^{\mathrm{I}})$ where A and X are the sets of names of objects and features occurring in the expansion, and for any $a \in A$, $x \in X$, and any role names $R_\Box \in \mathcal{R}_\Box$, $R_\Diamond \in \mathcal{R}_\Diamond$ we have aIx, $aR_\Box x$, $xR_\Diamond a$ iff such relational terms explicitly occur in $\overline{\mathcal{A}}$. Let $\mathbb{F} = (A, X, I, \mathcal{R}_\Box, \mathcal{R}_\Diamond)$ be the relational structure obtained in this manner. We define an interpretation I on it as follows. For any object name a, and feature name x, we let $a^{\mathrm{I}} := a$ and $x^{\mathrm{I}} := x$. For any atomic concept D, we define $D^{\mathrm{I}} = (x_D{}^{\downarrow}, a_D{}^{\uparrow})$. Next, we show that I is a valid interpretation for LE-$\mathcal{ALC}$. To this end, we need to show that $\mathbb{F}$ is an enriched formal context, i.e. that all $R_\Box$ and $R_\Diamond$ are I-compatible, and that D^{I} is a concept in the concept lattice $\mathbb{P}^+$ of $\mathbb{P} = (A, X, I)$. The latter condition is shown in the next lemma, and the former in the subsequent one.

Lemma 2. *$x_D^{\downarrow\uparrow} = a_D^{\uparrow}$ and $a_D^{\uparrow\downarrow} = x_D^{\downarrow}$ for any $D \in \mathcal{C}$.*

Proof. By the creation rules, we always have $a_D : D$ and $x_D{::}D$ in $\overline{\mathcal{A}}$, meaning that the tableau can be expanded with $a_D I x_D$. Therefore, we always have $x_D^{\downarrow\uparrow} \subseteq a_D^{\uparrow}$. Suppose $a_D I y$ and $b I x_D$ for some $y \in X$, $b \in A$. Then by the appending rules we have $y{::}D \in \overline{\mathcal{A}}$. This along with $bIx_D \in \overline{\mathcal{A}}$ immediately implies $bIy \in \overline{\mathcal{A}}$. Thus, we also have $a_D^{\uparrow} \subseteq x_D^{\downarrow\uparrow}$. We can prove the other equality analogously.

Lemma 3. *All the relations $R_\Box \in \mathcal{R}_\Box$ and $R_\Diamond \in \mathcal{R}_\Diamond$ in $\mathbb{F} = (\mathbb{P}, \mathcal{R}_\Box, \mathcal{R}_\Diamond)$ are I-compatible.*

Proof. We need to show that for any $b \in A$ and $y \in X$, and any $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$, (1) $R_\Box^{(0)}[y] = (\Box y)^{\downarrow}$, (2) $R_\Box^{(1)}[b] = (\blacklozenge b)^{\uparrow}$, (3) $R_\Diamond^{(0)}[b] = (\Diamond b)^{\uparrow}$, and (4) $R_\Diamond^{(1)}[y] = (\blacksquare y)^{\downarrow}$. We prove only (1) and (2). The proofs for (3) and (4) are analogous.

1. For any $b \in A$, if $bR_\Box y \in \mathcal{A}$, then $bI\Box y$ can be added by the adjunction rule, and thus $R_\Box^{(0)}[y] \subseteq (\Box y)^{\downarrow}$. If $bR_\Box y \notin \mathcal{A}$, then $bI\Box y$ is not added by applying adjunction rule to some $bR_\Box y$ in the original tableau. Thus, by item 1 of Lemma 1, $b : C, \Box y{::}C \in \overline{\mathcal{A}}$. Since $\Box y{::}C$ can only be added by the appending rule if $a_C I \Box y \in \overline{\mathcal{A}}$, and since this term can only be introduced by applying the adjunction rule to the term $\blacklozenge a_C I y$, some concept C' exists such that $\blacklozenge a_C : C', y{::}C' \in \overline{\mathcal{A}}$ (again by item 3 of Lemma 1). Then by the adjunction rule we have $a_C : [R_\Box]C' \in \overline{\mathcal{A}}$. Since $b : C$, $x_{\Box C'}{::}C$, and $y{::}C'$ are all in $\overline{\mathcal{A}}$, $bIx_{\Box C'}$ and $b : [R_\Box]C'$ must be in it as well. This, along with $y{::}C' \in \overline{\mathcal{A}}$, ensures that $bR_\Box y$ is added to the tableau expansion at some step, and we can conclude that $(\Box y)^{\downarrow} \subseteq R_\Box^{(0)}[y]$, as desired.
2. For every $b \in A$, if $bR_\Box y \in \mathcal{A}$, then by the adjunction rule we add $\blacklozenge bIy$. Thus, $R_\Box^{(1)}[b] \subseteq (\blacklozenge b)^{\uparrow}$. If $bR_\Box y \notin \mathcal{A}$, then by item 1 of Lemma 1, some terms $\blacklozenge b : C$ and $y{::}C$ must occur in $\overline{\mathcal{A}}$ for some C. So we have $y{::}C$ and (by an adjunction rule) $b : [R_\Box]C$, and hence $bR_\Box y$ must occur in $\overline{\mathcal{A}}$. So $\blacklozenge bIy \in \overline{\mathcal{A}}$ implies $bR_\Box y \in \overline{\mathcal{A}}$. Thus, $(\blacklozenge b)^{\uparrow} \subseteq R_\Box^{(1)}[b]$, as desired.

From the lemmas above, it immediately follows that the tuple $M = (\mathbb{F}, \cdot^{\mathrm{I}})$, with $\mathbb{F}$ and $\cdot^{\mathrm{I}}$ defined at the beginning of the present section, is a model for LE-$\mathcal{ALC}$. The following lemma states that the interpretation of any concept C in the model M is completely determined by the terms of the form bIx_C and $a_C Iy$ occurring in the tableau expansion.

Lemma 4. *Let $M = (\mathbb{F}, \cdot^{\mathrm{I}})$ be the model defined by the construction above. Then for any concept C and individuals b, x occurring in $\overline{\mathcal{A}}$,*

(1) $b \in [\![C]\!]_M$ iff $bIx_C \in \overline{\mathcal{A}}$ (2) $x \in (\![C]\!)_M$ iff $a_C I x \in \overline{\mathcal{A}}$.

Proof. By induction on the complexity of C. The base case (when C is atomic) is immediate by the construction of the model. For $C = \top$, by rule $\top$, and $x_\top{::}\top$ from the creation rule, $bIx_T \in \overline{\mathcal{A}}$ for any $b \in A$. Therefore, $x_\top^{\downarrow} = A = [\![\top]\!]$. For item 2, for any y, and if $a_T I y \in \overline{\mathcal{A}}$, then by the appending rule $y{::}\top \in \overline{\mathcal{A}}$. Then by $\top$ and the basic rule $bIy \in \overline{\mathcal{A}}$ for all b. Thus, $(\![\top]\!) = A^{\uparrow} \subseteq a_\top^{\uparrow}$. Moreover, if $y \in (\![\top]\!)$, then $bIy \in \overline{\mathcal{A}}$ for any b. In particular $a_T I y \in \overline{\mathcal{A}}$. Thus, $(\![\top]\!) = a_\top^{\uparrow}$. The proof for $\bot$ is analogous. For the induction step, we have four cases.

1. Suppose $C = C_1 \vee C_2$. For the first claim, notice that $b \in [\![C_1 \vee C_2]\!]$ iff $\forall y(y \in (\![C_1]\!) \cap (\![C_2]\!) \Rightarrow bIy)$. By the induction hypothesis, this is equivalent to
$$\forall y(y{::}C_1 \in \overline{\mathcal{A}} \;\&\; y{::}C_2 \in \overline{\mathcal{A}} \implies bIy \in \overline{\mathcal{A}}).$$
By the creation rule for $C_1 \vee C_2$, we have $x_{C_1 \vee C_2}{::}C_1 \vee C_2$, and consequently both $x_{C_1 \vee C_2}{::}C_1$ and $x_{C_1 \vee C_2}{::}C_2$ are added to the tableau. Thus, if the condition $y{::}C_1$ & $y{::}C_2 \Rightarrow bIy$ is satisfied for any y in $\overline{\mathcal{A}}$, then $bIx_{C_1 \vee C_2} \in \overline{\mathcal{A}}$. So $b \in [\![C_1 \vee C_2]\!]$ implies that $bIx_{C_1 \vee C_2} \in \overline{\mathcal{A}}$. Conversely, if $bIx_{C_1 \vee C_2} \in \overline{\mathcal{A}}$, then by the appending rule $b : C_1 \vee C_2 \in \overline{\mathcal{A}}$. Thus, for any $y{::}C_1$ and $y{::}C_2 \in \overline{\mathcal{A}}$, $bIy \in \overline{\mathcal{A}}$ due to rule $\vee_A$. Hence, $bIx_{C_1 \vee C_2} \in \overline{\mathcal{A}}$ implies
$$\forall y(y{::}C_1 \in \overline{\mathcal{A}} \;\&\; y{::}C_2 \in \overline{\mathcal{A}} \implies bIy \in \overline{\mathcal{A}}).$$
As observed before, this is equivalent to $y \in (\![C_1 \vee C_2]\!)$, as desired.

   For the second claim, notice that $x \in (\![C_1 \vee C_2]\!)$ iff $x \in (\![C_1]\!)$ and $x \in (\![C_2]\!)$. By induction hypothesis, this is equivalent to $x{::}C_1$ and $x{::}C_2$ occurring in $\overline{\mathcal{A}}$. By the creation rule for $C_1 \vee C_2$, $a_{C_1 \vee C_2} : C_1 \vee C_2 \in \overline{\mathcal{A}}$. Since $x{::}C_1, x{::}C_2 \in \overline{\mathcal{A}}$, we have $a_{C_1 \vee C_2}Ix \in \overline{\mathcal{A}}$ by the rule $\vee_X$. Conversely, if $a_{C_1 \vee C_2}Ix \in \overline{\mathcal{A}}$, then $x{::}C_1 \vee C_2 \in \overline{\mathcal{A}}$ by the appending rules, which implies $x{::}C_1, x{::}C_2 \in \overline{\mathcal{A}}$, or equivalently, $x \in (\![C_1 \vee C_2]\!)$.
2. The proof for $C = C_1 \wedge C_2$ is similar to the previous one.
3. Suppose $C = [R_\Box]C_1$. For the first claim, note that $b \in [\![[R_\Box]C_1]\!]$ iff $\forall y(y \in (\![C_1]\!) \Rightarrow bR_\Box y)$. By induction hypothesis, this is equivalent to $\forall y(y{::}C_1 \in \overline{\mathcal{A}} \Rightarrow bR_\Box y \in \overline{\mathcal{A}})$. Since $x_{C_1}{::}C_1 \in \overline{\mathcal{A}}$, by the creation rule for C_1, it follows that $bR_\Box x_{C_1} \in \overline{\mathcal{A}}$. By the adjunction rule, this implies $bI\Box x_{C_1} = bIx_{\Box C_1} \in \overline{\mathcal{A}}$. Conversely, if $bIx_{\Box C_1} \in \overline{\mathcal{A}}$, then by the appending rule also $b : [R_\Box]C_1 \in \overline{\mathcal{A}}$. That is, for any y, if $y{::}C_1 \in \overline{\mathcal{A}}$, then $bR_\Box y \in \overline{\mathcal{A}}$ by the expansion rule for $\Box$. As observed before, this implication is equivalent to $b \in [\![[R_\Box]C_1]\!]$, as desired.

   For the second claim, notice that $y \in (\![[R_\Box]C_1]\!)$ iff $\forall b(b \in [R_\Box]C_1 \Rightarrow bIy)$. Equivalently (as proved previously), for all b, if $b : [R_\Box]C_1 \in \overline{\mathcal{A}}$, implies $bIy \in \overline{\mathcal{A}}$. Combining this with the fact that the creation rule for $[R_\Box]C_1$ implies $a_{\Box C_1}{::}[R_\Box]C_1 \in \overline{\mathcal{A}}$, this implies that $a_{\Box C_1}Iy \in \overline{\mathcal{A}}$ as well. Conversely, suppose $a_{\Box C_1}Iy \in \overline{\mathcal{A}}$. Then for any b, if $b : [R_\Box]C_1 \in \overline{\mathcal{A}}$, then $bIy \in \overline{\mathcal{A}}$. This is equivalent to $y \in (\![[R_\Box]C_1]\!)$.
4. The proof for $C = \langle R_\Diamond \rangle C_1$ is similar to the previous one.

Theorem 2 (Soundness). *The model $M = (\mathbb{F}, \cdot^I)$ defined above satisfies the ABox $\mathcal{A}$.*

Proof. We proceed by cases.

1. By construction, M satisfies all terms of the form $bR_\Box y$, bIy, or $yR_\Diamond b$ in $\mathcal{A}$.
2. By construction, any relational term is satisfied by M iff it explicitly occurs in $\overline{\mathcal{A}}$. Thus, either M satisfies all terms of the form $\neg(bR_\Box y)$, $\neg(bIy)$, or $\neg(yR_\Diamond b)$ occurring in $\mathcal{A}$, or some expansion of $\mathcal{A}$ contains a clash.

3. For the terms of the form $b : C$, y::C, $\neg(b : C)$, or $\neg(y$::$C)$, we have $b \in [\![C]\!]$ iff $bIx_C \in \overline{\mathcal{A}}$, and $y \in (\![C]\!)$ iff $a_C Iy \in \overline{\mathcal{A}}$ (Lemma 4). For any $b : C$, y::C, $\neg(b : C)$, or $\neg(y$::$C)$ occurring in $\mathcal{A}$, we respectively add bIx_C, $a_C Iy$, $\neg(bIx_C)$, or $\neg(a_C Iy)$ to $\overline{\mathcal{A}}$ via the expansion rules, and thus M satisfies the constraints.

The following corollary is an immediate consequence of the termination and soundness of the tableau procedure.

Corollary 2 (Finite Model Property). *For any consistent LE-$\mathcal{ALC}$ ABox $\mathcal{A}$, some model of $\mathcal{A}$ exists the size of which is polynomial in size$(\mathcal{A})$.*

Proof. The model M of Theorem 2 is the required witness. The polynomial bound on the size of M follows from the proof of Theorem 1.

4.3 Completeness of the Tableau Algorithm

In this section, we prove the completeness of the tableau algorithm. The following lemma is key to this end, since it shows that every model for an LE-$\mathcal{ALC}$ ABox can be extended to a model with classifying object and features.

Lemma 5. *For any ABox $\mathcal{A}$, any model $M = (\mathbb{F}, \cdot^{\mathrm{I}})$ of $\mathcal{A}$ can be extended to a model $M' = (\mathbb{F}', \cdot^{\mathrm{I}'})$ such that $\mathbb{F}' = (A', X', I', \{R'_\Box\}_{\Box\in\mathcal{G}}, \{R'_\Diamond\}_{\Diamond\in\mathcal{F}})$, $A \subseteq A'$ and $X \subseteq X'$, and moreover for every $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$:*

1. There exists $a_C \in A'$ and $x_C \in X'$ such that:

$$C^{\mathrm{I}'} = (I'^{(0)}[x_C^{\mathrm{I}'}], I'^{(1)}[a_C^{\mathrm{I}'}]), \quad a_C^{\mathrm{I}'} \in [\![C^{\mathrm{I}'}]\!], \quad x_C^{\mathrm{I}'} \in (\![C^{\mathrm{I}'}]\!), \tag{4}$$

2. For every individual b in A there exist $\Diamond b$ and $\blacklozenge b$ in A' such that:

$$I'^{(1)}[\blacklozenge b] = R'^{(1)}_\Box[b^{\mathrm{I}'}] \quad \textit{and} \quad I'^{(1)}[\Diamond b] = R'^{(0)}_\Diamond[b^{\mathrm{I}'}], \tag{5}$$

3. For every individual y in X there exist $\Box y$ and $\blacksquare y$ in X' such that:

$$I'^{(0)}[\blacksquare y] = R'^{(1)}_\Diamond[y^{\mathrm{I}'}] \quad \textit{and} \quad I'^{(0)}[\Box y] = R'^{(0)}_\Box[y^{\mathrm{I}'}]. \tag{6}$$

4. For any C, $[\![C^{\mathrm{I}}]\!] = [\![C^{\mathrm{I}'}]\!] \cap A$ and $(\![C^{\mathrm{I}}]\!) = (\![C^{\mathrm{I}'}]\!) \cap X$.

Proof. Fix $\Box \in \mathcal{G}$ and $\Diamond \in \mathcal{F}$. Let M' be defined as follows. For every concept C, we add new elements a_C and x_C to A and X (respectively) to obtain the sets A' and X'. For any $J \in \{I, R_\Box\}$, any $a \in A'$ and $x \in X'$, we set $aJ'x$ iff one of the following holds:

1. $a \in A$, $x \in X$, and aJx;
2. $x \in X$, and $a = a_C$ for some concept C, and bJx for all $b \in [\![C^{\mathrm{I}}]\!]$;
3. $a \in A$, and $x = x_C$ for some concept C, and aJy for all $y \in (\![C^{\mathrm{I}}]\!)$;
4. $a = a_{C_1}$ and $x = x_{C_2}$ for some C_1, C_2, and bJy for all $b \in [\![C_1^{\mathrm{I}}]\!]$, and $y \in (\![C_2^{\mathrm{I}}]\!)$.

We set $xR'_\Diamond a$ iff one of the following holds:

1. $a \in A$, $x \in X$, and $xR_\Diamond a$;
2. $x \in X$, and $a = a_C$ for some concept C, and $xR_\Diamond b$ for all $b \in [\![C^{\mathrm{I}}]\!]$;
3. $a \in A$, and $x = x_C$ for some concept C, and $yR_\Diamond a$ for all $y \in (\![C^{\mathrm{I}}]\!)$;
4. $a = a_{C_1}$ and $x = x_{C_2}$ for some C_1, C_2, and $yR_\Diamond b$ for all $b \in [\![C_1^{\mathrm{I}}]\!]$, $y \in (\![C_2^{\mathrm{I}}]\!)$.

For any $b \in A$, $y \in X$, let $\blacklozenge b = a_{\Box(cl(b))}$, $\Diamond b = a_{\Diamond(cl(b))}$, $\blacksquare y = x_{\Diamond(cl(y))}$, and $\Box y = x_{\Box(cl(y))}$, where $cl(b)$ (resp. $cl(y)$) is the smallest concept generated by b (resp. y). For any C, let $C^{\mathrm{I}'} = (I'^{(0)}[x_C], I'^{(1)}[a_C])$. Then M' is as required.

Theorem 3 (Completeness). *Let $\mathcal{A}$ be a consistent ABox and $\mathcal{A}'$ be obtained via the application of any expansion rule applied to $\mathcal{A}$. Then $\mathcal{A}'$ is also consistent.*

Proof. If $\mathcal{A}$ is consistent, by Lemma 5, a model M' of $\mathcal{A}$ exists which satisfies (4), (5) and (6). The statement follows from the fact that any term added by any expansion rule is satisfied by M' where we interpret a_C, x_C, $\blacklozenge b$, $\Diamond b$, $\Box y$, $\blacksquare y$ as in Lemma 5.

Remark 2. The algorithm can easily be extended to acyclic TBoxes, via the unravelling technique (cf. [3] for details).

5 Conclusion and Future Work

In this paper, we define a two-sorted non-distributive description logic LE-$\mathcal{ALC}$ to describe and reason about formal concepts arising from (enriched) formal contexts from FCA. We describe ABox and TBox terms for the logic and define a tableaux algorithm for it. This tableaux algorithm decides the consistency of ABoxes and acyclic TBoxes, and provides a procedure to construct a model when the input is consistent. We show that this algorithm is computationally more efficient than the tableaux algorithm for $\mathcal{ALC}$.

This work can be extended in several interesting directions.

Dealing with Cyclic TBoxes and RBox Axioms. In this paper, we introduced a tableaux algorithm only for knowledge bases with acyclic TBoxes. We conjecture that the following statement holds of general (i.e. possibly cyclic) TBoxes.
Conjecture. The tableaux algorithm introduced in this paper can be extended to check the consistency of any knowledge base $(\mathcal{A}, \mathcal{T})$ (with possibly cyclic TBox axioms) in time polynomial in $size(\mathcal{A} \cup \mathcal{T})$.

Developing such an algorithm is a research direction we are currently pursuing. Another aspect we intend to develop in future work concerns giving a complete axiomatization for LE-$\mathcal{ALC}$. RBox axioms are used in description logics to describe the relationship between different relations in knowledge bases and the properties of these relations such as reflexivity, symmetry, and transitivity. It would be interesting to see if it is possible to obtain necessary and/or sufficient conditions on the shape of RBox axioms for which a tableaux algorithm can be obtained. This has an interesting relationship with the problem in LE-logic of providing computationally efficient proof systems for various extensions of LE-logic in a modular manner [5,16].

Generalizing to Other Semantic Frameworks. The non-distributive DL introduced in this paper is semantically motivated by a relational semantics for LE-logics which establishes a link with FCA. A different semantics for the same logic, referred to as graph-based semantics [12], provides another interpretation of the same logic as a logic suitable for evidential and hyper-constructivist reasoning. In the future, we intend to develop description logics for reasoning in the framework of graph-based semantics, to appropriately model evidential and hyper-constructivist settings.

Generalizing to More Expressive Description Logics. The DL LE-$\mathcal{ALC}$ is the non-distributive counterpart of $\mathcal{ALC}$. A natural direction for further research is to explore the non-distributive counterparts of extensions of $\mathcal{ALC}$ such as $\mathcal{ALCI}$ and $\mathcal{ALCIN}$.

Description Logic and Formal Concept Analysis. The relationship between FCA and DL has been studied and used in several applications [1,4,17]. The framework of LE-$\mathcal{ALC}$ formally brings FCA and DL together, both because its concepts are naturally interpreted as formal concepts in FCA, and because its language is designed to represent knowledge and reasoning in enriched formal contexts. Thus, these results pave the way to the possibility of establishing a closer and more formally explicit connection between FCA and DL, and of using this connection in theory and applications.

References

1. Atif, J., Hudelot, C., Bloch, I.: Explanatory reasoning for image understanding using formal concept analysis and description logics. IEEE Trans. Syst. Man Cybern. Syst. **44**(5), 552–570 (2014)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
3. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press, Cambridge (2017)
4. Baader, F., Sertkaya, B.: Applying formal concept analysis to description logics. In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 261–286. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24651-0_24
5. van der Berg, I., De Domenico, A., Greco, G., Manoorkar, K.B., Palmigiano, A., Panettiere, M.: Labelled calculi for the logics of rough concepts. In: Banerjee, M., Sreejith, A.V. (eds.) Logic and Its Applications, ICLA 2023. LNCS, vol. 13963, pp. 172–188. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-26689-8_13
6. Borgwardt, S., Peñaloza, R.: Fuzzy description logics – a survey. In: Moral, S., Pivert, O., Sánchez, D., Marín, N. (eds.) SUM 2017. LNCS (LNAI), vol. 10564, pp. 31–45. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67582-4_3
7. Conradie, W., et al.: Modal reduction principles across relational semantics. arXiv preprint arXiv:2202.00899 (2022)
8. Conradie, W., et al.: Rough concepts. Inf. Sci. **561**, 371–413 (2021)

9. Conradie, W., Frittella, S., Palmigiano, A., Piazzai, M., Tzimoulis, A., Wijnberg, N.M.: Toward an epistemic-logical theory of categorization. In: Electronic Proceedings in Theoretical Computer Science, EPTCS 251 (2017)
10. Conradie, W., Frittella, S., Palmigiano, A., Piazzai, M., Tzimoulis, A., Wijnberg, N.M.: Categories: how i learned to stop worrying and love two sorts. In: Väänänen, J., Hirvonen, Å., de Queiroz, R. (eds.) WoLLIC 2016. LNCS, vol. 9803, pp. 145–164. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52921-8_10
11. Conradie, W., Palmigiano, A.: Algorithmic correspondence and canonicity for non-distributive logics. Ann. Pure Appl. Logic **170**(9), 923–974 (2019)
12. Conradie, W., Palmigiano, A., Robinson, C., Wijnberg, N.: Non-distributive logics: from semantics to meaning. In: Rezus, A. (ed.) Contemporary Logic and Computing, Landscapes in Logic, vol. 1, pp. 38–86. College Publications (2020)
13. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-59830-2
14. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Semantic characterization of rational closure: from propositional logic to description logics. Artif. Intell. **226**, 1–33 (2015)
15. Giordano, L., Gliozzi, V., Theseider Dupré, D.: A conditional, a fuzzy and a probabilistic interpretation of self-organizing maps. J. Log. Comput. **32**(2), 178–205 (2022)
16. Greco, G., Ma, M., Palmigiano, A., Tzimoulis, A., Zhao, Z.: Unified correspondence as a proof-theoretic tool. J. Log. Comput. **28**(7), 1367–1442 (2016)
17. Jiang, Y.: Semantifying formal concept analysis using description logics. Knowl. Based Syst. **186**, 104967 (2019)
18. Lieto, A., Pozzato, G.L.: A description logic framework for commonsense conceptual combination integrating typicality, probabilities and cognitive heuristics. J. Exp. Theoret. Artif. Intell. **32**(5), 769–804 (2020)
19. Ma, Z.M., Zhang, F., Wang, H., Yan, L.: An overview of fuzzy description logics for the semantic web. Knowl. Eng. Rev. **28**(1), 1–34 (2013)
20. de Paiva, V., Haeusler, E.H., Rademaker, A.: Constructive description logics hybrid-style. Electron. Notes Theoret. Comput. Sci. **273**, 21–31 (2011)
21. Shilov, N.V., Han, S.Y.: A proposal of description logic on concept lattices. In: Proceedings of the Fifth International Conference on Concept Lattices and their Applications, pp. 165–176 (2007)
22. Wurm, C.: Language-theoretic and finite relation models for the (full) Lambek calculus. J. Logic Lang. Inform. **26**(2), 179–214 (2017). https://doi.org/10.1007/s10849-017-9249-z

Sequent Calculi

A New Calculus for Intuitionistic Strong Löb Logic: Strong Termination and Cut-Elimination, Formalised

Ian Shillito[1(✉)], Iris van der Giessen[2], Rajeev Goré[3,4], and Rosalie Iemhoff[5]

[1] Australian National University, Canberra, Australia
ian.shillito@anu.edu.au
[2] University of Birmingham, Birmingham, UK
[3] Technical University of Vienna, Vienna, Austria
[4] Polish Academy of Science, Warsaw, Poland
[5] Utrecht University, Utrecht, The Netherlands

Abstract. We provide a new sequent calculus that enjoys syntactic cut-elimination and strongly terminating backward proof search for the intuitionistic Strong Löb logic iSL, an intuitionistic modal logic with a provability interpretation. A novel measure on sequents is used to prove both the termination of the naive backward proof search strategy, and the admissibility of cut in a syntactic and direct way, leading to a straightforward cut-elimination procedure. All proofs have been formalised in the interactive theorem prover Coq.

Keywords: Intuitionistic provability logic · Cut-elimination · Backward proof search · Interactive theorem proving · Proof theory

1 Introduction

Gödel-Löb logic GL extends classical modal logic K with the Gödel-Löb axiom $\Box(\Box\varphi \to \varphi) \to \Box\varphi$. GL is the provability logic of Peano Arithmetic PA, i.e. it consists of all modal formulas that are true under any arithmetical interpretation where $\Box\varphi$ means "φ is provable in PA" (expressed in the language of PA).

An intuitionistic version of GL is iGL and the intuitionistic counterpart of PA is Heyting Arithmetic HA. For a long time, the provability logic of HA was an open problem and was only known to be an extension of iGL. However, Mojtahedi claims to have found a solution in a preprint [34] currently under review.

Several other logics also have provability interpretations, such as modalised Heyting calculus mHC, Kuznetsov-Muravitsky logic KM, and intuitionistic Strong Löb logic iSL [14,30,32,35]. All these intuitionistic modal logics except mHC include the Gödel-Löb axiom and all except iGL contain the so-called completeness axiom $\varphi \to \Box\varphi$.

Important to note is that these logics are defined over the language with only the $\Box$-modality and without $\Diamond$. In classical modal logic, $\Diamond$ is dual to $\Box$ and reads as consistency in the provability interpretation. However, for intuitionistic modal logics, in general, $\Diamond$ and $\Box$ are not interdefinable and several choices can be made. Interestingly, intuitionistic modal logics defined over the language with only the $\Box$ already reveal intrinsic intuitionistic characters. Important for us is the aforementioned completeness principle, also known as the coreflection principle. It trivializes in a classical setting, but has interesting intuitionistic readings. Indeed, in our setting of provability, $\varphi \to \Box\varphi$ reads as completeness: "if φ is true then φ is provable" (see [45] for a discussion on the completeness principle in extensions of Heyting Arithmetic). The coreflection principle also appears in intuitionistic epistemic logic and lax logic (for overviews see, e.g., [18,32]).

Here, we consider iSL, the minimal intuitionistic modal logic with both the Gödel-Löb axiom and the completeness axiom, which can also be axiomatised over intuitionistic modal logic iK by the Strong Löb axiom $(\Box\varphi \to \varphi) \to \varphi$. The logic iSL is the provability logic of an extension of Heyting Arithmetic with respect to so-called slow provability [46] and plays an important role in the Σ_1-provability logic of HA [3].

The Gödel-Löb axiom characterises transitive converse well-founded Kripke frames for GL and also for the birelational frames for iGL, iSL, and KM. Interestingly, for iSL, mHC, and KM, the modal relation is a part of the intuitionistic relation. This semantics plays an important role in the study of iSL, e.g. in the characterisation of its admissible rules [19]. A natural deduction system for iSL can be found in [7]. The proof systems that we focus on here are sequent calculi.

From a proof-theoretic perspective, the "diagonal formula" $\Box\varphi$ in the modal (GLR) rule for GL causes difficulties for direct cut-elimination because the standard induction on the size of the cut-formula and the height fail. Cut-elimination is highly nontrivial as witnessed by decades of unsuccessful attempts and controversies before the proof by Valentini [44] was finally shown to be correct [23].

$$\frac{\Gamma, \Box\Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi, \Delta}\,(\mathrm{GLR}) \qquad \frac{\Gamma, \varphi \to \psi \Rightarrow \varphi \qquad \Gamma, \psi \Rightarrow \varphi}{\Gamma, \varphi \to \psi \Rightarrow \varphi}\,(\to\mathrm{L_i})$$

In backward proof search, the (GLR) rule causes loops because $\Box\Gamma$ is preserved upwards from conclusion to premise. For (GLR), a simple terminating and complete strategy consists in applying (GLR) only if $\Box\varphi \notin \Box\Gamma$. In sequent calculi for intuitionistic logic, the traditional $(\to \mathrm{L_i})$ rule, shown above right, can cause backward proof search to go into loops. For termination without loop check, various authors have independently discovered the sequent calculus $\mathsf{G4ip}$ which replaces the $(\to \mathrm{L_i})$ rule with multiple rules, depending on the form of φ [12]. Iemhoff [29] developed $\mathsf{G4}$-like calculi for several intuitionistic modal logics.

Thus, in a sequent calculus for an intuitionistic provability logic, both the modal rule and left implication rule have the potential to cause loops *and* the modal rule can complicate direct cut-elimination! For logic iGL, van der Giessen and Iemhoff have developed $\mathsf{G3iGL}$ and $\mathsf{G4iGL}$ [20], providing a direct cut-elimination procedure for the former. The initial proof of cut-elimination for $\mathsf{G4iGL}$ was indirect, via $\mathsf{G3iGL}$, but Goré and Shillito later formalised direct cut-elimination using the maximal height of derivations as induction parameter [26].

Recently, van der Giessen and Iemhoff [21] developed two sequent calculi, G3iSL and G4iSL, for iSL for which they provided the analogue results compared to G3iGL and G4iGL mentioned above. In particular, they show that backward proof search in G4iSL *weakly* terminates: *there exists* a terminating (and complete) backward proof search strategy, namely one similar to the above-described for logic GL. However, *not all* strategies terminate on this calculus: the naive backward proof search strategy, apply any rule in any order, does not.

Here, we present G4iSLt which replaces the G4iSL rules of the top row below, by the rules in the bottom row. As suggested by van der Giessen and Iemhoff [21], the new modal rule drops the explicit embedding of transitivity. But crucially, the new left-implication rule drops both transitivity and contraction on $\Box\varphi \to \psi$ in the left premise. The right premise $S = \Phi, \Box\Gamma, \psi \Rightarrow \chi$ is kept untouched:

$$\frac{\Phi, \Gamma, \Box\Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi} \qquad \frac{\Phi, \Gamma, \Box\Gamma, \Box\varphi \to \psi, \Box\varphi \Rightarrow \varphi \qquad S}{\Phi, \Box\Gamma, \Box\varphi \to \psi \Rightarrow \chi}$$

$$\frac{\Phi, \Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi} \qquad \frac{\Phi, \Gamma, \psi, \Box\varphi \Rightarrow \varphi \qquad S}{\Phi, \Box\Gamma, \Box\varphi \to \psi \Rightarrow \chi}$$

Our results improve on the work of van der Giessen and Iemhoff [21]. First, our new measure ensures that the naive backward proof search strategy for our new calculus terminates. This is unusual for sequent calculi for provability logics, and especially for intuitionistic provability logics. Second, we prove direct cut-elimination for G4iSLt using a proof technique similar to the *mhd proof technique* [6,24]. Third, all our results are formalised in Coq and can be found here: https://ianshil.github.io/G4iSLT. We consequently contribute to the rapidly growing literature of formalised proof theory [1,8,9,15,17,24,26,39]. We also think that our work sheds light on what one might call proof-theoretic meta considerations. Namely, it shows the subtle consequences of rule choices on termination and cut-elimination.

In Sect. 2, we introduce the preliminaries of iSL, including our calculus G4iSLt. Section 3 presents the admissibility of structural rules in G4iSLt. In Sect. 4, we prove that backward proof search in G4iSLt strongly terminates. Finally, in Sect. 5, we directly prove cut-admissibility for G4iSL using a proof technique similar to the *mhd proof technique* [6,24].

2 Preliminaries

In this section we successively present the syntax, axiomatic system, Kripke semantics and sequent calculus for the logic iSL.

2.1 Syntax

Let $\mathbb{V} = \{p, q, r \ldots\}$ be a countably infinite set of propositional variables on which equality is decidable, that is $\forall p, q \in \mathbb{V}$, we can decide whether $p = q$ or-else $p \neq q$. Modal formulae are defined using BNF notation as below:

$$\varphi ::= p \in \mathbb{V} \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \Box\varphi$$

We use the greek letters $\varphi, \psi, \chi, \delta, \ldots$ for formulae and $\Gamma, \Delta, \Phi, \Psi \ldots$ for multisets of formulae. We say that φ is a *boxed formula* if $\Box$ is its main connective. For a multiset Γ, we define the multiset $\Box\Gamma := \{\Box\varphi : \varphi \in \Gamma\}$. By the unboxing of a multiset $\Box\Gamma$ we mean the multiset Γ.

Following Goré et al. [24,26], we encode formulae as an inductive type `MPropF` whose base case encodes $\mathbb{V}$ as the type `nat` of natural numbers because `nat` is countably infinite and equality is decidable on it. A list of such formulae then has the type `list MPropF`. The usual operations on lists "append" and "cons" are respectively represented by `++` and `::` but Coq also allows us to write lists in infix notation using `;`. Thus the terms `φ1 :: φ2 :: φ3 :: nil` and `[φ1] ++ [φ2] ++ [φ3]` and `[φ1 ; φ2 ; φ3]` all encode the list $\varphi_1, \varphi_2, \varphi_3$.

We straightforwardly extend Dyckhoff's notion of weight of a formula [11], defined for the intuitionistic language, to the modal language.

Definition 1. *The weight $w(\varphi)$ of a formula φ is defined as follows:*

$$\begin{aligned} w(\bot) = w(p) &= 1 \\ w(\psi \lor \chi) = w(\psi \to \chi) &= w(\psi) + w(\chi) + 1 \\ w(\psi \land \chi) &= w(\psi) + w(\chi) + 2 \\ w(\Box\psi) &= w(\psi) + 1 \end{aligned}$$

The main motivation behind this weight is to ensure that $w(\varphi \to (\psi \to \chi)) < w((\varphi \land \psi) \to \chi)$, which is crucial to show termination of naive backward proof search on the sequent calculus G4ip for intuitionistic logic.

2.2 Axiomatic Systems as Consequence Relations

Traditional Hilbert calculi are designed to capture logics as sets of theorems, that is sets of the form $\{\varphi : \vdash \varphi\}$. However, when considering logics as consequence relations these systems are inadequate, and notably lead to historical confusions about properties such as the deduction theorem [25,27].

Generalised Hilbert calculi manipulate expressions $\Gamma \vdash \varphi$, where Γ is a set of formulae. They clearly distinguish between the notion of deducibility from a set of assumptions, versus theoremhood. They are particularly useful for identifying the appropriate form of deduction theorem holding for a logic [25]. Still, they correspond to traditional Hilbert calculi when restricted to consecutions of the shape $\emptyset \vdash \varphi$, as we do here. Thus, we can connect the generalised Hilbert calculus here to the traditional Hilbert calculus considered by Ardeshir and Mojtahedi [3].

The generalised Hilbert calculus iSLH for iSL, shown in Fig. 1, extends the one for intuitionistic modal logic iK with the Strong Löb axiom $(\Box\varphi \to \varphi) \to \varphi$. We write $\Gamma \vdash_{\mathsf{iSLH}} \varphi$ if $\Gamma \vdash \varphi$ is provable in iSLH.

Note that if we replace the premise of the rule (Nec) by $\Gamma \vdash \varphi$ we obtain an equivalent calculus. This is implied by the completeness axiom $\varphi \to \Box\varphi$ and the holding of the deduction theorem in iSLH [18].

Axioms

A_1 $\varphi \to (\psi \to \varphi)$	A_7 $(\varphi \wedge \psi) \to \psi$
A_2 $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$	A_8 $(\varphi \to \psi) \to ((\varphi \to \chi) \to (\varphi \to (\psi \wedge \chi)))$
A_3 $\varphi \to (\varphi \vee \psi)$	A_9 $\bot \to \varphi$
A_4 $\psi \to (\varphi \vee \psi)$	A_{10} $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$
A_5 $(\varphi \to \chi) \to ((\psi \to \chi) \to ((\varphi \vee \psi) \to \chi))$	A_{11} $(\Box\varphi \to \varphi) \to \varphi$
A_6 $(\varphi \wedge \psi) \to \varphi$	

Rules of Inference

$$\frac{\varphi \text{ is an instance of an axiom}}{\Gamma \vdash \varphi}\ (\mathrm{Ax}) \qquad \frac{\varphi \in \Gamma}{\Gamma \vdash \varphi}\ (\mathrm{El})$$

$$\frac{\emptyset \vdash \varphi}{\Gamma \vdash \Box\varphi}\ (\mathrm{Nec}) \qquad \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \varphi \to \psi}{\Gamma \vdash \psi}\ (\mathrm{MP})$$

Fig. 1. Generalised Hilbert calculus iSLH for iSL

2.3 Kripke Semantics

We now present the Kripke semantics for iSL [3,32] to notably prove soundness of our sequent calculus G4iSLt, and explain its rules (SLtR) and ($\Box\!\to$L).

The Kripke semantics of iSL is a restriction of the Kripke semantics for intuitionistic modal logics. More precisely, the semantic interpretation of connectives is preserved, but the class of models is restricted. The models for this logic are defined below, where for a set W, we write $\mathcal{P}(W)$ for the set of all subsets of W.

Definition 2. *A* Kripke model $\mathcal{M}$ *for* iSL *is a tuple* $(W, \leq, R, I)$*, where* W *is a non-empty set (of possible worlds), both* $\leq$ *(the intuitionistic relation) and* R *(the modal relation) are subsets of* $W \times W$*, and* $I : \mathbb{V} \to \mathcal{P}(W)$*, which satisfies the following:* $\leq$ *is reflexive and transitive;* R *is transitive and converse well-founded;* $(\leq \circ R) \subseteq R$ *where "*$\circ$*" is relational composition;* $R \subseteq \leq$*; and for all* $p \in \mathbb{V}$ *and* $w, v \in W$*, if* $w \leq v$ *and* $w \in I(p)$ *then* $v \in I(p)$*.*

Note the peculiarity of the models for iSL: $R \subseteq \leq$, that is the modal relation is a subset of the intuitionistic relation. We recall the standard definition of forcing for intuitionistic modal logics, and show that persistence holds.

Definition 3. *Given a Kripke model* $\mathcal{M} = (W, \leq, R, I)$*, we define the forcing relation as follows, where* $v \geq w$ *is just* $w \leq v$*:*

$$\begin{array}{lll}
\mathcal{M}, w \Vdash p & \textit{if} & w \in I(p) \\
\mathcal{M}, w \Vdash \bot & & \textit{never} \\
\mathcal{M}, w \Vdash \varphi \wedge \psi & \textit{if} & \mathcal{M}, w \Vdash \varphi \textit{ and } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \vee \psi & \textit{if} & \mathcal{M}, w \Vdash \varphi \textit{ or } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \to \psi & \textit{if} & \forall v \geq w.\ \mathcal{M}, v \Vdash \varphi \textit{ implies } \mathcal{M}, v \Vdash \psi \\
\mathcal{M}, w \Vdash \Box\varphi & \textit{if} & \forall v \in W.\ wRv \textit{ implies } \mathcal{M}, v \Vdash \varphi
\end{array}$$

Local consequence is as below where $\mathcal{M}, w \Vdash \Gamma$ *means* $\forall \varphi \in \Gamma, \mathcal{M}, w \Vdash \varphi$*:*

$$\Gamma \models \varphi \qquad \textit{iff} \qquad \forall \mathcal{M}. \forall w. (\mathcal{M}, w \Vdash \Gamma \quad \textit{implies} \quad \mathcal{M}, w \Vdash \varphi)$$

Lemma 1 (Persistence). *For any model $\mathcal{M} = (W, \leq, R, I)$, formula φ and points $w, v \in W$, if $w \leq v$ and $\mathcal{M}, w \Vdash \varphi$ then $\mathcal{M}, v \Vdash \varphi$.*

Interestingly, as iSL satisfies the finite model property [46] it can also be characterised by the class of *finite* frames where R is transitive and *irreflexive*.

2.4 Sequent Calculus

A *sequent* is a pair of a finite multiset Γ of formulae and a formula φ, denoted $\Gamma \Rightarrow \varphi$. For a sequent $\Gamma \Rightarrow \varphi$ we call Γ the *antecedent* of the sequent and φ the *consequent* of the sequent. For multisets Γ and Δ, the multiset sum $\Gamma \uplus \Delta$ is the multiset whose multiplicity (at each formula) is a sum of the multiplicities of Γ and Δ. We write Γ, Δ to mean $\Gamma \uplus \Delta$. For a formula φ, we write φ, Γ and Γ, φ to mean $\{\varphi\} \uplus \Gamma$. From the formalisation perspective, a pair of a list of formulae `(list MPropF)` and a formula `MPropF` has type `(list MPropF) * MPropF`, using the Coq notation `*` for forming pairs. The latter is the type we give to sequents in our formalisation, for which we use the macro `Seq`. Thus the sequent $\varphi_1, \varphi_2, \varphi_3 \Rightarrow \psi$ is encoded by the term `[φ1 ; φ2 ; φ3] * ψ`, which itself can also be written as the pair `([φ1 ; φ2 ; φ3], ψ)`. Note that `[φ1 ; φ2 ; φ3] * ψ` is different from `[φ2 ; φ1 ; φ3] * ψ` since the order of the elements is crucial, so our lists do not capture multisets (yet).

A *sequent calculus* consists of a finite set of *sequent rule schemas*. Each rule schema consists of a conclusion sequent schema and some number of premise sequent schemas. A rule schema with zero premise schemas is called an initial rule. The conclusion and premises are built in the usual way from propositional-variables, formula-variables and multiset-variables. A *rule instance* is obtained by uniformly instantiating every variable in the rule schema with a concrete object of that type. This is the standard definition from structural proof theory.

Definition 4 (Derivation/Proof). *A* derivation *of a sequent S in the sequent calculus* C *is a finite tree of sequents such that (i) the root node is S; and (ii) each interior node and its direct children are the conclusion and premise(s) of a rule instance in* C*. A* proof *is a derivation where every leaf is the conclusion of an instance of an initial rule.*

Note that we explicitly define the notion of a derivation as an object rather than define the notion of derivability, as is done in some papers. We do so as we want to create a "deep" embedding of such derivations into Coq [9].

In what follows, it should be clear from context whether the word "proof" refers to the object defined in Definition 4, or to the meta-level notion. We say that a sequent is *provable* in G4iSLt if it has a proof in G4iSLt. We elide the details of the encodings of sequent rules and derivations, as these can be found elsewhere [1,39]. We define a predicate `G4iSLt_prv` on sequents to encode *provability* in G4iSLt. Our encodings rely on the type `Type`, which bears computational content, unlike `Prop`, and is crucially compatible with the extraction function of Coq.

Before presenting our calculus, we recall standard notions from proof theory.

Definition 5 (Height). *For any derivation δ, its* height $h(\delta)$ *is the maximum number of nodes on a path from root to leaf.*

Definition 6 (Admissibility, Invertibility, Height-Preservation). *Let* R *be a rule schema with premises $S_0, \dots, S_n$ and conclusion S. We say that* R *is:*

admissible: *if for every instance of* R*, the instance of S is provable whenever the instances of $S_1, \dots, S_n$ are all provable;*
invertible: *if for every instance of* R*, the instances of $S_1, \dots, S_n$ are all provable whenever the instance of S is provable;*
height-preserving admissible: *if for every instance of* R*, if there are proofs $\pi_0, \dots, \pi_n$ of the instances of $S_0, \dots, S_n$ then there is a proof π of the instance of S such that $h(\pi) \leq h(\pi_i)$ for some $0 \leq i \leq n$;*
height-preserving invertible: *if for every instance of* R*, if π is a proof of the instance of S then there are proofs $\pi_0, \dots, \pi_n$ of the instances of $S_0, \dots, S_n$ such that $h(\pi_i) \leq h(\pi)$ for all $0 \leq i \leq n$.*

The sequent calculus $\mathsf{G4iSLt}$ is given in Fig. 2. When defining rules we put the label naming of the rule on the left of the horizontal line, while the label appears on the right of the line in *instances* of rules.

$$(\bot\mathrm{L})\ \frac{}{\bot, \Gamma \Rightarrow \chi} \qquad (\mathrm{IdP})\ \frac{}{\Gamma, p \Rightarrow p}$$

$$(\wedge\mathrm{L})\ \frac{\Gamma, \varphi, \psi \Rightarrow \chi}{\Gamma, \varphi \wedge \psi \Rightarrow \chi} \qquad (\wedge\mathrm{R})\ \frac{\Gamma \Rightarrow \varphi \qquad \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \varphi \wedge \psi}$$

$$(\vee\mathrm{L})\ \frac{\Gamma, \varphi \Rightarrow \chi \qquad \Gamma, \psi \Rightarrow \chi}{\Gamma, \varphi \vee \psi \Rightarrow \chi} \qquad (\vee\mathrm{R}_i)\ \frac{\Gamma \Rightarrow \varphi_i}{\Gamma \Rightarrow \varphi_1 \vee \varphi_2}\ (i \in \{1, 2\})$$

$$(p\to\mathrm{L})\ \frac{\Gamma, p, \varphi \Rightarrow \chi}{\Gamma, p, p \to \varphi \Rightarrow \chi} \qquad (\to\mathrm{R})\ \frac{\Gamma, \varphi \Rightarrow \psi}{\Gamma \Rightarrow \varphi \to \psi}$$

$$(\Box\to\mathrm{L})\ \frac{\Phi, \Gamma, \psi, \Box\varphi \Rightarrow \varphi \qquad \Phi, \Box\Gamma, \psi \Rightarrow \chi}{\Phi, \Box\Gamma, \Box\varphi \to \psi \Rightarrow \chi} \qquad (\mathrm{SLtR})\ \frac{\Phi, \Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi}$$

$$(\wedge\to\mathrm{L})\ \frac{\Gamma, \varphi \to (\psi \to \chi) \Rightarrow \delta}{\Gamma, (\varphi \wedge \psi) \to \chi \Rightarrow \delta} \qquad (\vee\to\mathrm{L})\ \frac{\Gamma, \varphi \to \chi, \psi \to \chi \Rightarrow \delta}{\Gamma, (\varphi \vee \psi) \to \chi \Rightarrow \delta}$$

$$(\to\to\mathrm{L})\ \frac{\Gamma, \psi \to \chi \Rightarrow \varphi \to \psi \qquad \Gamma, \chi \Rightarrow \delta}{\Gamma, (\varphi \to \psi) \to \chi \Rightarrow \delta}$$

Fig. 2. The sequent calculus $\mathsf{G4iSLt}$, where Φ contains no boxed formula.

In (IdP), a propositional variable instantiating the featured occurrences of p is principal. In a rule instance of ($\wedge$R), ($\wedge$L), ($\vee$R$_i$), ($\vee$L) or ($\to$R), the *principal* *formula* of that instance is defined as usual. In a rule instance of ($p \to$L), both a propositional variable instantiating p and the formula instantiating the featured $p \to \varphi$ are principal formulae of that instance. In a rule instance of ($\wedge\to$L), ($\vee\to$L), ($\to\to$L) or ($\Box\to$L), the formula instantiating respectively $(\varphi \wedge \psi) \to \chi$, $(\varphi \vee \psi) \to \chi$, $(\varphi \to \psi) \to \chi$ or $\Box\varphi \to \psi$ is the principal formula of that instance. In a rule instance of (SLtR) or ($\Box\to$L), $\Box\varphi$ is called the *diagonal formula* [38].

The non-modal rules are taken from the calculus for IPC for which backward proof search strongly terminates [11]. Keypoint is that the usual intuitionistic left implication rule is replaced by four implication rules depending on the main connective in the antecedent of the principal formula, in such a way that each premise is less complex than the conclusion. In particular, when considering the rule ($\to\to$ L), an application of the "regular" left implication rule yields the more complex left premise $\Gamma, (\varphi \to \psi) \to \chi \Rightarrow \varphi \to \psi$, which is (semantically) equivalent to the simpler left premise stated in rule ($\to\to$ L).

We proceed to give semantic intuitions for the rules (SLtR) and ($\Box\to$L).

The (SLtR) rule has similarities with the rule (GLR) (shown below) from sequent calculi for provability logics such as GL, but with two major differences: (1) the non-boxed formulae Φ in the antecedent of the sequent are preserved from conclusion to premise in (SLtR), while they are deleted in (GLR); and (2) the formulae in $\Box\Gamma$ are not preserved upwards in (SLtR), while they are in (GLR).

$$\frac{\Phi, \Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi}\ \text{(SLtR)} \qquad\qquad \frac{\Gamma, \Box\Gamma, \Box\varphi \Rightarrow \varphi}{\Phi, \Box\Gamma \Rightarrow \Box\varphi}\ \text{(GLR)}$$

From a backward proof search perspective, both rules correspond, semantically, to a "modal jump" from a point w which falsifies the conclusion $\Phi, \Box\Gamma \Rightarrow \Box\varphi$ to a modal successor v which forces Γ but falsifies the succedent φ of the premise. The underlying relation R in both logics is transitive and converse well-founded. Using converse well-foundedness we can assume that v is the last modal successor making φ false, thus v forces $\Box\varphi$ in both logics. Transitivity implies that v forces $\Box\Gamma$ in both logics, so all its successors force Γ. But, in iSL, the underlying relation R is also persistent so v also forces Φ in iSL, but not in GL, thus explaining difference (1). Thanks to persistence, v forcing Γ implies that all its successors force Γ, meaning that v forces $\Box\Gamma$ already, thus explaining difference (2).

The two premises of ($\Box\to$L) capture how $\Box\varphi \to \psi$ in the antecedent of the conclusion can be true. The simple case is when ψ is true, which corresponds to the right premise. The more complicated case is when ψ is not true, implying that $\Box\varphi$ must also be not true. Now, $\Box\varphi$ true semantically means that φ is true in all modal successors, hence $\Box\varphi$ not true means that φ is not true in a modal successor. But converse well-foundedness implies the existence of a last modal successor where φ is not true, with all its modal successors making φ true. The left premise corresponds to this last modal successor, as it encodes that φ is not true but $\Box\varphi$ is true. Moreover, this last modal successor is also an intuitionistic successor as $R \subseteq \leq$. By persistence, this last successor must also make $\Box\varphi \to \psi$ true. But then, a simple modus ponens on $\Box\varphi$ and $\Box\varphi \to \psi$ gives us ψ.

Finally, we show that G4iSLt indeed captures the set of theorems of iSL.

Theorem 1. *For all φ we have: $\emptyset \vdash_{\mathsf{iSLH}} \varphi$ iff $\Rightarrow \varphi$ is provable in* $\mathsf{G4iSLt}$*.*

Proof. We proved in Coq the two following results.

(1)	$\Gamma \vdash_{\mathsf{iSLH}} \varphi$	implies	there exists a finite $\Gamma' \subseteq \Gamma$ s.t. $\Gamma' \Rightarrow \varphi$ is provable in $\mathsf{G4iSLt}$
(2)	$\Gamma \Rightarrow \varphi$ is provable in $\mathsf{G4iSLt}$	implies	$\Gamma \models \varphi$

The result (1), which relies on the admissibility of cut (Theorem 2), shows that $\mathsf{G4iSLt}$ is (strongly) complete with respect to iSLH and gives us the left-to-right direction of our theorem. The other direction involves the soundness of $\mathsf{G4iSLt}$ w.r.t. the local consequence shown in (2), as well as the (non-formalised) result of (weak) completeness of iSLH w.r.t. the local consequence obtained by Ardeshir and Mojtahedi [3]. ■

3 Admissible Rules in G4iSLt

This section aims at showing that the contraction rule is admissible. To do so, it follows the work developed by Goré and Shillito [26] on the sequent calculus $\mathsf{GL4ip}$ for the intuitionistic provability logic iGL, which extends itself on the work of Dyckhoff and Negri [13] on $\mathsf{G4ip}$. Most of the overall structure of the argument is the same as for the case of $\mathsf{GL4ip}$, except for the crucial and typical *left-unboxing rule* ($\boxtimes$), shown to be height-preserving admissible.

Most of the results of this section are proven by inductions on the weight of formulae and/or height of derivations. We omit the Coq encodings for brevity.

Lemma 2 (Height-preserving invertibility of rules). *The rules* $(\wedge R)$, $(\wedge L)$*,* $(\vee L)$, $(\rightarrow R)$, $(p\!\rightarrow L)$, $(\wedge\!\rightarrow L)$, $(\vee\!\rightarrow L)$ *are height-preserving invertible.*

We present height-preserving admissible and admissible rules in Fig. 3.

The structural rules of weakening (Wkn), contraction (Ctr) and exchange (Exc), are all (at least) admissible. The presence of the latter may be surprising, as the sequents we use are based on multisets. However, as mentioned earlier, our formalisation encodes sequents using lists and not multisets. So, the formal proof of the height-preserving admissibility of (Exc) shows that list-sequents of our formalisation mimic multiset-sequents of the pen-and-paper definition. In fact, we designed the formalisation of $\mathsf{G4iSLt}$ so that it admits exchange [26].

The rule ($\boxtimes$) is quite typical of the logic iSL, as it reflects one of its theorems: the completeness axiom $\varphi \rightarrow \Box\varphi$. Indeed, this axiom implies that Γ entails $\Box\Gamma$, allowing the replacement of $\Box\Gamma$ by Γ in the antecedent of a provable sequent while preserving provability. The height-preserving admissibility of ($\boxtimes$) is crucially used in many places, notably Lemma 2 and the admissibility of cut.

The height-preserving admissibility of ($\Box\rightarrow$LIR) and ($\rightarrow\rightarrow$LIR) shows height-preserving invertibility in the right premise of the rules ($\Box\rightarrow$L) and ($\rightarrow\rightarrow$L).

The admissible rule ($\rightarrow$L) is the traditional left-implication rule. We use this rule to prove the admissibility of ($\rightarrow\rightarrow$LIL), resembling the invertibility in the left premise of ($\rightarrow\rightarrow$L). In turn, ($\rightarrow\rightarrow$LIL) is crucial in the admissibility of (Ctr).

Height-preserving admissible rules

$$(\mathrm{Exc})\ \frac{\Gamma_0, \Gamma_3, \Gamma_2, \Gamma_1, \Gamma_4 \Rightarrow \chi}{\Gamma_0, \Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4 \Rightarrow \chi} \qquad (\mathrm{Wkn})\ \frac{\Gamma \Rightarrow \chi}{\Gamma, \varphi \Rightarrow \chi} \qquad (\boxtimes)\ \frac{\Phi, \Box\Gamma \Rightarrow \chi}{\Phi, \Gamma \Rightarrow \chi}$$

$$(\Box{\to}\mathrm{LIR})\ \frac{\Phi, \Box\Gamma, \Box\varphi \to \psi \Rightarrow \chi}{\Phi, \Box\Gamma, \psi \Rightarrow \chi} \qquad ({\to}{\to}\mathrm{LIR})\ \frac{\Gamma, (\varphi \to \psi) \to \chi \Rightarrow \delta}{\Gamma, \chi \Rightarrow \delta}$$

Admissible rules

$$(\mathrm{Id})\ \frac{}{\varphi, \Gamma \Rightarrow \varphi} \qquad ({\to}\mathrm{L})\ \frac{\Gamma \Rightarrow \varphi \qquad \Gamma, \psi \Rightarrow \chi}{\Gamma, \varphi \to \psi \Rightarrow \chi}$$

$$({\to}{\to}\mathrm{LIL})\ \frac{\Gamma, (\varphi \to \psi) \to \delta \Rightarrow \chi}{\Gamma, \varphi, \psi \to \delta, \psi \to \delta \Rightarrow \chi} \qquad (\mathrm{Ctr})\ \frac{\varphi, \varphi, \Gamma \Rightarrow \chi}{\varphi, \Gamma \Rightarrow \chi}$$

Fig. 3. Height-preserving admissible and admissible rules in G4iSLt.

In the following section we introduce a measure on sequents which we use to show that the naive backward proof search strategy for G4iSLt terminates. This measure could thus be used to derive the notion of maximum height of derivations (mhd) for a sequent, as was done in previous works [24,26]. There, the mhd measure was used as secondary induction measure in the proof of admissibility of cut. Here, we simply use the termination measure instead.

4 Naive Backward Proof Search Terminates

Sequent calculi enjoying cut-elimination can often be used to decide whether a given formula φ is deducible from a given set of assumptions Γ by strategically applying the rules "backwards" from the end-sequent $\Gamma \Rightarrow \varphi$. To obtain a decision procedure, we require a backward proof search strategy which terminates and is complete, i.e. which provides a proof for any sequent provable in the calculus.

But often, terminating complete strategies necessitate a "loop check" mechanism, that stops the search if the same sequent appears twice on a branch. For example, the sequent calculus LJ, for propositional intuitionistic logic, only has a strategy with loop check as terminating complete strategy. The termination of these strategies is messy to reason about, as in most cases their unguarded version is not terminating and results in proof trees with infinite branches.

While some calculi have terminating complete strategies without loop checks, like GLS for GL [24] and GL4ip for iGL [20], we consider a stronger kind of calculus: calculi with *strongly terminating* backward proof search, such as G4ip for intuitionistic propositional logic [12]. Backward proof search for a sequent calculus is strongly terminating if and only if *all* backward proof search strategies for this calculus, complete or not, terminate. This characterisation has other equivalent forms: (1) the naive backward proof search strategy terminates, and (2) there is a well-founded ordering on sequents decreasing upwards in all the rules of the calculus. In contrast, backward proof search is *weakly* terminating if and only if *there is* a terminating complete strategy for this calculus.

In this section we show that backward proof search for G4iSLt is strongly terminating. More precisely, we show that the naive strategy terminates. To do this, we need two ingredients: (1) a locally defined measure on sequents, and (2) a well-founded order making this measure decrease upwards in the rules of G4iSLt.

4.1 Shortlex: A Well-Founded Order on `list` $\mathbb{N}$

We define the shortlex order, which is a well-founded order on `list` $\mathbb{N}$, i.e. the set of all lists of natural numbers.

In the following, we use $<$ to mean the usual ordering on natural numbers. Let us recall the definition of the lexicographic order on lists of natural numbers.

Definition 7 (Lexicographic order). *Let $n \in \mathbb{N}$. We define the lexicographic order $<^n_{lex}$ on lists of natural numbers of length n. For two lists of natural numbers $[m_1; \cdots; m_n]$ and $[k_1; \cdots; k_n]$, we write $[m_1; \cdots; m_n] <^n_{lex} [k_1; \cdots; k_n]$ if there is a $1 \leq j \leq n$ such that: (1) $m_p = k_p$ for all $1 \leq p < j$, and (2) $m_j < k_j$.*

Note that as $<$ is a well-founded order, then $<^n_{lex}$ is also well-founded [36]. Finally, we define the shortlex order, also called *breadth-first* [31] or *length-lexicographic* order, over lists of natural numbers (viewed as n-tuples).

Definition 8 (Shortlex order). *The shortlex order over lists of natural numbers, noted $\ll$, is defined as follows. For two lists l_0 and l_1 of natural numbers, we say that $l_0 \ll l_1$ whenever one of the following conditions is satisfied:*

1. *$length(l_0) < length(l_1)$;*
2. *$length(l_0) = length(l_1) = n$ and $l_0 <^n_{lex} l_1$.*

Intuitively, the shortlex order is ordering lists according to their length and follows the lexicographic order whenever length does not discriminate. Note that on top of being well-founded, $\ll$ is obviously transitive.

4.2 A (`list` $\mathbb{N}$)-Measure on Sequents

We proceed to attach to each sequent $\Gamma \Rightarrow \chi$ a "measure" $\Theta(\Gamma \Rightarrow \chi)$ which is a (finite) list of natural numbers, i.e. of type `list` $\mathbb{N}$. For simplicity, in the following we consider a fixed sequent $\Gamma \Rightarrow \chi$ for which we define the measure.

To introduce our measure, we first wish to explain why the measure used for GL4ip [26], acting as a substitute of the Dershowitz-Manna order [10] considered in Dyckhoff's article on G4ip [11], does not work for our purpose. The explanation of this failure justifies the modification we made to obtain the measure for G4iSLt.

The intuition behind the measure for GL4ip and G4ip is the following: for a multiset we create an ordered list of counters for each weight of occurrences of formulae of this weight. For more details, take a finite multiset of formulae Δ. As it is finite, it contains a *topmost* formula of maximal weight n. We can create a list of length n such that at each position m in the list (counting from right to left) for $1 \leq m \leq n$, we find the number of occurrences in Δ of *topmost* formulae of weight m. Such a list gives the count of occurrences in Δ of formulae of weight n in its leftmost (i.e. n-th) component, then of occurrences of formulae of weight $n-1$ in the next (i.e. $(n-1)$-th) component, and so on until we reach 1.

The measure for GL4ip and G4ip consisted in attaching to $\Gamma \Rightarrow \chi$ the list obtained by applying the above procedure on the multiset $\Gamma \uplus \{\chi\}$. Call this function Θ_{fail}. This measure fails to show termination of the naive strategy for G4iSLt, as it does not decrease upwards in the following application of (SLtR).

$$\frac{\Box p \Rightarrow p}{\Rightarrow \Box p} \text{ (SLtR)}$$

We have that $\Theta_{fail}(\Rightarrow \Box p) = [1, 0]$ because $\Box p$ is the formula of maximum weight 2, and it is the only formula with this weight occurring in the list, while no formula of weight 1 appears in $\Rightarrow \Box p$. In addition to that, we have that $\Theta_{fail}(\Box p \Rightarrow p) = [1, 1]$. Consequently, we obtain $\Theta_{fail}(\Rightarrow \Box p) \ll \Theta_{fail}(\Box p \Rightarrow p)$: the measure increased upwards. So, the measure used for GL4ip and G4ip cannot be used here. We need to define another one.

With enough scrutinising, one can notice that in G4iSLt the principal box of a boxed formula in the antecedent of a sequent is a "deadweight". More precisely, once a formula $\Box\varphi$ is in the antecedent of a sequent, only two things can happen to its outermost box: it is either deleted (via the modal rule (SLtR) or $(\Box\!\rightarrow\!\text{L})$), or else it is preserved (through all other rules). Intuitively, this observation suggests that boxed formulae in the antecedent are destined to be unboxed eventually in the upward application of rules, without having any other effect.

Consequently, as the top-level boxes in the antecedent of a sequent are deadweights, we can think about unboxing the antecedent of $\Gamma \Rightarrow \chi$ before applying the procedure described above. This is precisely what we do: if Γ is of the shape $\Gamma_0, \Box\Gamma_1$ with no boxed formula in Γ_0, we define $\Theta(\Gamma \Rightarrow \chi)$ to be the list of natural numbers obtained via the above machinery applied on the multiset $\Gamma_0 \uplus \Gamma_1 \uplus \{\chi\}$.

For example, to compute $\Theta(\Box(p \wedge q), p \vee q \Rightarrow q \rightarrow p)$, we first unbox the antecedent of this sequent by transforming $\Box(p \wedge q)$ into $p \wedge q$ to obtain the multiset $\{p \wedge q, p \vee q, q \rightarrow p\}$. Because $p \wedge q$ is the only formula of maximum weight four, our list of length four begins with 1. Since both $p \vee q$ and $q \rightarrow p$ are of weight three, the second element is 2. Finally, since there are no formulae of weights two and one, we obtain $\Theta(\Box(p \wedge q), p \vee q \Rightarrow q \rightarrow p) = [1, 2, 0, 0]$. Following this explanation, observe that the issue we faced with $\Rightarrow \Box p$ and $\Box p \Rightarrow p$ is now fixed: we first unbox $\Box p$ in $\Box p \Rightarrow p$, hence $\Theta(\Box p \Rightarrow p) = [2] \ll [1, 0] = \Theta(\Rightarrow \Box p)$.

Two things need to be noted about such lists. First, if no topmost occurrence of a formula is of weight $1 \leq k \leq n$, then a 0 appears in position k in the list. This is the case for the weight 2 in the last example above. Second, as no formula is of weight 0 we do not dedicate a position for this particular weight in our list.

4.3 Every Rule of G4iSLt Reduces Θ Upwards

We obtain the sought after result about our measure Θ: it decreases upwards through the rules of G4iSLt on the $\ll$ ordering.

Lemma 3. *For all sequents $S_0, S_1, ..., S_n$ and for all $1 \leq i \leq n$, if there is an instance of a rule r of* G4iSLt *of the form below, then $\Theta(S_i) \ll \Theta(S_0)$:*

$$\frac{S_1 \quad \dots \quad S_n}{S_0}\ r$$

Clearly, this result implies that the naive strategy for G4iSLt terminates: any rule application makes the measure decrease on $\ll$, ensuring termination via well-foundedness of $\ll$. Thus, backward proof search is strongly terminating.

Moreover, this lemma is quite crucial in the proof of admissibility of cut: as we use $\Theta(\Gamma \Rightarrow \chi)$ as secondary induction measure (through well-foundedness of $\ll$) there, we know that we can apply the secondary induction hypothesis on any sequent S which is a premise of $\Gamma \Rightarrow \chi$ through a rule, as $\Theta(S) \ll \Theta(\Gamma \Rightarrow \chi)$.

5 Cut-Elimination for G4iSLt

To reach cut-elimination, our main theorem, we first state and prove the admissibility of the cut rule in a direct and purely syntactic way. More precisely, we prove that the *additive*-cut rule, with *cut formula* φ, is admissible. This statement and its formalisation are given below, where Γ is encoded as Γ0++Γ1.

Theorem 2 (Admissibility of additive-cut). *The additive cut rule below is admissible in* G4iSLt*.*

$$\frac{\Gamma \Rightarrow \varphi \quad \varphi, \Gamma \Rightarrow \psi}{\Gamma \Rightarrow \psi}\ (Cut)$$

```
Theorem G4iSLt_cut_adm : forall φ Γ0 Γ1 χ,
 (G4iSLt_prv (Γ0++Γ1,φ) * G4iSLt_prv (Γ0++φ::Γ1,χ)) ->
                   G4iSLt_prv (Γ0++Γ1,χ).
```

Proof. Let d_1 (with last rule r_1) and d_2 (with last rule r_2) be proofs in G4iSLt of $\Gamma \Rightarrow \varphi$ and $\varphi, \Gamma \Rightarrow \chi$ respectively, as shown below.

$$\frac{d_1}{\Gamma \Rightarrow \varphi}\ r_1 \qquad \frac{d_2}{\varphi, \Gamma \Rightarrow \chi}\ r_2$$

We show that there is a proof in G4iSLt of $\Gamma \Rightarrow \chi$. We reason by strong primary induction (PI) on the weight of the cut-formula φ, giving the primary inductive hypothesis (PIH). We also use a strong secondary induction (SI) on $\Theta(\Gamma \Rightarrow \chi)$ of the conclusion of a cut, giving the secondary inductive hypothesis (SIH). Crucially, by using SIH we avoid the issues caused by the diagonal formula [23,44].

We consider r_1. In total, there are thirteen cases for r_1: one for each rule in G4iSLt. However, we can reduce the number of cases to eight. We separate them by using Roman numerals and showcase the most interesting ones.

(V) $\mathbf{r_1} = (\rightarrow\mathbf{R})$: Then r_1 has the following form where $\varphi = \varphi_0 \rightarrow \varphi_1$:

$$\dfrac{\varphi_0, \Gamma \Rightarrow \varphi_1}{\Gamma \Rightarrow \varphi_0 \to \varphi_1}\ (\to\text{R})$$

For the cases where $\varphi_0 \to \varphi_1$ is principal in r_2 and $r_2 \neq (\Box\!\to \text{L})$, or where $r_2 \in \{(\text{IdP}), (\bot\text{L})\}$, we refer to Dyckhoff and Negri's proof [13] as the cuts produced in these cases involve the traditional induction hypothesis PIH. We are left with seven sub-cases, but here again focus on the most interesting ones.

(V-d) If r_2 is $(\to\to \text{L})$ where the cut formula is not principal in r_2, then it must have the following form where $(\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 = \Gamma$.

$$\dfrac{\varphi_0 \to \varphi_1, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_0 \to \gamma_1 \qquad \varphi_0 \to \varphi_1, \gamma_2, \Gamma_0 \Rightarrow \chi}{\varphi_0 \to \varphi_1, (\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \chi}\ (\to\to\text{L})$$

Thus, $\Gamma \Rightarrow \chi$ is of the form $(\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \chi$ and $\Gamma \Rightarrow \varphi_0 \to \varphi_1$ is of the form $(\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \varphi_0 \to \varphi_1$. Using the admissible rule $(\to\to\text{LIR})$ on the latter we obtain a proof of the sequent $\gamma_2, \Gamma_0 \Rightarrow \varphi_0 \to \varphi_1$. Then consider the following proof of the sequent $\gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_0 \to \gamma_1$, where the rule $(\to\to\text{LIL})$ deconstructs the implication $(\gamma_0 \to \gamma_1) \to \gamma_2$, rule (Ctr) contracts $\gamma_1 \to \gamma_2$ and Lemma 2 is the invertibility of the rule $(\to\text{R})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{(\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \varphi_0 \to \varphi_1}{\gamma_0, \gamma_1 \to \gamma_2, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \varphi_0 \to \varphi_1}\ (\to\to\text{LIL})}{\gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \varphi_0 \to \varphi_1}\ (Ctr) \qquad \dfrac{\varphi_0 \to \varphi_1, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_0 \to \gamma_1}{\varphi_0 \to \varphi_1, \gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_1}\ \text{Lem.2}}{\gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_1}\ \text{SIH}}{\gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_0 \to \gamma_1}\ (\to\text{R})$$

The crucial point here is to see that the use of SIH is justified, in other words, that $\Theta(\gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_1) \ll \Theta((\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \chi)$. This is the case as the rule applications $(\to\to\text{L})$ and $(\to\text{R})$ entail $\Theta(\gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_1) \ll \Theta(\gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_0 \to \gamma_1) \ll \Theta((\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \chi)$ by Lemma 3, hence $\Theta(\gamma_0, \gamma_1 \to \gamma_2, \Gamma_0 \Rightarrow \gamma_1) \ll \Theta((\gamma_0 \to \gamma_1) \to \gamma_2, \Gamma_0 \Rightarrow \chi)$ by transitivity of $\ll$. So, we are done. Note that the created cut could not be justified by usual induction on height, as the admissibility of $(\to\to\text{LIL})$ is not height-preserving.

(V-f) If r_2 is $(\Box\!\to\!\text{L})$ with a principal formula different from the cut formula, then it must have the following form where $\Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 = \Gamma$.

$$\dfrac{\varphi_0 \to \varphi_1, \gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \gamma_0 \qquad \gamma_1, \varphi_0 \to \varphi_1, \Phi, \Box\Gamma_0 \Rightarrow \chi}{\varphi_0 \to \varphi_1, \Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \chi}\ (\Box\!\to\!\text{L})$$

Thus, we have that $\Gamma \Rightarrow \chi$ and $\Gamma \Rightarrow \varphi_0 \to \varphi_1$ are respectively of the form $\Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \chi$ and $\Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \varphi_0 \to \varphi_1$. Using the admissible rule $(\Box\!\to\!\text{LIR})$ on the latter we obtain a proof of $\gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \varphi_0 \to \varphi_1$. Then, we proceed as follows by combining the proof π second-below with the first one.

$$\dfrac{\begin{matrix}\pi\\ \gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \gamma_0\end{matrix} \qquad \dfrac{\gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \varphi_0 \to \varphi_1 \qquad \gamma_1, \varphi_0 \to \varphi_1, \Phi, \Box\Gamma_0 \Rightarrow \chi}{\gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \chi}\ \text{SIH}}{\Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \chi}\ (\Box\!\to\!\text{L})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\varphi_0, \Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0 \Rightarrow \varphi_1}{\varphi_0, \Box\gamma_0 \to \gamma_1, \Phi, \Box\Gamma_0, \Box\gamma_0 \Rightarrow \varphi_1}\ (\text{Wkn})}{\varphi_0, \gamma_1, \Phi, \Box\Gamma_0, \Box\gamma_0 \Rightarrow \varphi_1}\ (\Box\!\to\!\text{LIR})}{\varphi_0, \gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \varphi_1}\ (\boxtimes)}{\gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \varphi_0 \to \varphi_1}\ (\to\text{R}) \qquad \varphi_0 \to \varphi_1, \gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \gamma_0}{\gamma_1, \Phi, \Gamma_0, \Box\gamma_0 \Rightarrow \gamma_0}\ \text{SIH}$$

Note that both uses of SIH are justified here, as the last rule in the first proof is an instance of ($\Box\to$L) hence $\Theta(\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi) \ll \Theta(\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi)$ and $\Theta(\gamma_1,\Phi,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0) \ll \Theta(\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi)$ by Lemma 3.

(VII) $\mathbf{r_1}$ =($\Box\to$L): Then r_1 is as follows, where $\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 = \Gamma$.

$$\begin{prooftree}
\AxiomC{$\gamma_1,\Phi,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0$}
\AxiomC{$\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \varphi$}
\RightLabel{($\Box\to$L)}
\BinaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \varphi$}
\end{prooftree}$$

Thus, the sequents $\Gamma \Rightarrow \chi$ and $\varphi,\Gamma \Rightarrow \chi$ are of the form $\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$ and $\varphi,\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$, respectively. Then, we proceed as follows.

$$\begin{prooftree}
\AxiomC{$\gamma_1,\Phi,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0$}
\AxiomC{$\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \varphi$}
\AxiomC{$\varphi,\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$}
\dashedLine
\RightLabel{($\Box\to$LIR)}
\UnaryInfC{$\varphi,\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$}
\dashedLine
\RightLabel{SIH}
\BinaryInfC{$\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$}
\RightLabel{($\Box\to$L)}
\BinaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi$}
\end{prooftree}$$

Note that the use of SIH is justified, as the last rule in this proof gives us $\Theta(\gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi) \ll \Theta(\Box\gamma_0 \to \gamma_1,\Phi,\Box\Gamma_0 \Rightarrow \chi)$ by Lemma 3.

(VIII) $\mathbf{r_1}$ =(SLtR): Then φ is the diagonal formula in r_1:

$$\begin{prooftree}
\AxiomC{$\Phi,\Gamma_0,\Box\varphi_0 \Rightarrow \varphi_0$}
\RightLabel{(SLtR)}
\UnaryInfC{$\Phi,\Box\Gamma_0 \Rightarrow \Box\varphi_0$}
\end{prooftree}$$

where $\varphi = \Box\varphi_0$ and $\Phi,\Box\Gamma_0 = \Gamma$. Thus, we have that $\Gamma \Rightarrow \chi$ and $\varphi,\Gamma \Rightarrow \chi$ are respectively of the form $\Phi,\Box\Gamma_0 \Rightarrow \chi$ and $\Box\varphi_0,\Phi,\Box\Gamma_0 \Rightarrow \chi$. We now consider r_2.

(VIII-b) If r_2 is ($\Box\to$L) it is of the following form, where $\Phi = \Box\gamma_0 \to \gamma_1,\Phi_0$.

$$\begin{prooftree}
\AxiomC{$\gamma_1,\Phi_0,\Box\gamma_0,\varphi_0,\Gamma_0 \Rightarrow \gamma_0$}
\AxiomC{$\gamma_1,\Phi_0,\Box\varphi_0,\Box\Gamma_0 \Rightarrow \chi$}
\RightLabel{($\Box\to$L)}
\BinaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\varphi_0,\Box\Gamma_0 \Rightarrow \chi$}
\end{prooftree}$$

We proceed as follows.

$$\begin{prooftree}
\AxiomC{π_0}
\noLine
\UnaryInfC{$\gamma_1,\Phi_0,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0$}
\AxiomC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{($\Box\to$LIR)}
\UnaryInfC{$\gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \Box\varphi_0$}
\AxiomC{$\gamma_1,\Phi_0,\Box\varphi_0,\Box\Gamma_0 \Rightarrow \chi$}
\dashedLine
\RightLabel{SIH}
\BinaryInfC{$\gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \chi$}
\RightLabel{($\Box\to$L)}
\BinaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \chi$}
\end{prooftree}$$

where π_0 is the first proof given below, which depends π_1, the second one:

$$\begin{prooftree}
\AxiomC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{($\boxtimes$)}
\UnaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Gamma_0,\Box\gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{($\Box\to$LIR)}
\UnaryInfC{$\gamma_1,\Phi_0,\Gamma_0,\Box\gamma_0 \Rightarrow \Box\varphi_0$}
\AxiomC{π_1}
\noLine
\UnaryInfC{$\gamma_1,\Phi_0,\Box\gamma_0,\Box\varphi_0,\Gamma_0 \Rightarrow \gamma_0$}
\dashedLine
\RightLabel{SIH}
\BinaryInfC{$\gamma_1,\Phi_0,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0$}
\end{prooftree}$$

$$\begin{prooftree}
\AxiomC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\varphi_0,\Gamma_0 \Rightarrow \varphi_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\gamma_0,\Box\varphi_0,\Gamma_0 \Rightarrow \varphi_0$}
\dashedLine
\RightLabel{($\Box\to$LIR)}
\UnaryInfC{$\gamma_1,\Phi_0,\Box\gamma_0,\Box\varphi_0,\Gamma_0 \Rightarrow \varphi_0$}
\AxiomC{$\varphi_0,\gamma_1,\Phi_0,\Box\gamma_0,\Gamma_0 \Rightarrow \gamma_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\varphi_0,\gamma_1,\Phi_0,\Box\gamma_0,\Box\varphi_0,\Gamma_0 \Rightarrow \gamma_0$}
\dashedLine
\RightLabel{PIH}
\BinaryInfC{$\gamma_1,\Phi_0,\Box\gamma_0,\Box\varphi_0,\Gamma_0 \Rightarrow \gamma_0$}
\end{prooftree}$$

Note that both uses of SIH are justified here as the rule application ($\Box\to$L) entails $\Theta(\gamma_1,\Phi_0,\Gamma_0,\Box\gamma_0 \Rightarrow \gamma_0) \ll \Theta(\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \chi)$ and we have $\Theta(\gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \chi) \ll \Theta(\Box\gamma_0 \to \gamma_1,\Phi_0,\Box\Gamma_0 \Rightarrow \chi)$ by Lemma 3.

(VIII-c) If r_2 is (SLtR), then it is of the following form where $\chi = \Box\chi_0$.

$$\frac{\Phi, \varphi_0, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0}{\Phi, \Box\varphi_0, \Box\Gamma_0 \Rightarrow \Box\chi_0}\ (\mathrm{SLtR})$$

We proceed as follows.

$$\begin{prooftree}
\AxiomC{$\Phi, \Gamma_0, \Box\varphi_0 \Rightarrow \varphi_0$}
\RightLabel{(SLtR)}
\UnaryInfC{$\Phi, \Box\Gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{($\boxtimes$)}
\UnaryInfC{$\Phi, \Gamma_0 \Rightarrow \Box\varphi_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\Phi, \Gamma_0, \Box\chi_0 \Rightarrow \Box\varphi_0$}
\AxiomC{$\Box\varphi_0, \Phi, \Gamma_0 \Rightarrow \varphi_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\Box\varphi_0, \Phi, \Gamma_0, \Box\chi_0 \Rightarrow \varphi_0$}
\AxiomC{$\varphi_0, \Phi, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0$}
\dashedLine
\RightLabel{(Wkn)}
\UnaryInfC{$\varphi_0, \Box\varphi_0, \Phi, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0$}
\dashedLine
\RightLabel{PIH}
\BinaryInfC{$\Box\varphi_0, \Phi, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0$}
\dashedLine
\RightLabel{SIH}
\BinaryInfC{$\Phi, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0$}
\RightLabel{(SLtR)}
\UnaryInfC{$\Phi, \Box\Gamma_0 \Rightarrow \Box\chi_0$}
\end{prooftree}$$

The use of SIH is justified because the last rule in this proof ensures that $\Theta(\Phi, \Gamma_0, \Box\chi_0 \Rightarrow \chi_0) \ll \Theta(\Phi, \Box\Gamma_0 \Rightarrow \Box\chi_0)$ by Lemma 3. ■

The attentive reader may have noticed that our proof technique requires the use of additive, and not multiplicative, cuts. Indeed, the use of SIH relies on the decrease of the measure Θ, which is notably ensured by the upward application of any rule of the calculus. More generally, in the proof of admissibility if the cut we initially consider has $\Gamma \Rightarrow \chi$ as conclusion, then we can justify a cut with conclusion $\Gamma' \Rightarrow \chi'$ using SIH as long as we have a chain $r_0, \ldots, r_n$ of application of rules of G4iSLt of the following form.

$$\begin{prooftree}
\AxiomC{$\ldots \quad \Gamma' \Rightarrow \chi' \quad \ldots$}
\RightLabel{r_n}
\UnaryInfC{$\vdots$}
\AxiomC{$\ldots$}
\AxiomC{$\ldots$}
\RightLabel{r_0}
\TrinaryInfC{$\Gamma \Rightarrow \chi$}
\end{prooftree}$$

However, the contraction rule does not ensure the decrease of the measure Θ from conclusion to premise: it is not the case that $\Theta(\Gamma, \varphi, \varphi \Rightarrow \chi) \ll \Theta(\Gamma, \varphi \Rightarrow \chi)$. So, this prevents us from allowing one of $r_0, \ldots, r_n$ above to be (Ctr). This is where multiplicative cuts are problematic: they most often use the contraction rule as follows, where $\Gamma \Rightarrow \chi$ is the conclusion of the initial cut and $\Gamma', \Gamma'' \Rightarrow \chi'$ is the conclusion of the cut we want to justify through SIH.

$$\begin{prooftree}
\AxiomC{$\Gamma' \Rightarrow \varphi$}
\AxiomC{$\varphi, \Gamma'' \Rightarrow \chi'$}
\dashedLine
\RightLabel{SIH}
\BinaryInfC{$\Gamma', \Gamma'' \Rightarrow \chi'$}
\noLine
\UnaryInfC{$\vdots$}
\dashedLine
\RightLabel{(Ctr)*}
\UnaryInfC{$\Gamma \Rightarrow \chi$}
\end{prooftree}$$

Unfortunately, the presence of the contraction rule above $\Gamma \Rightarrow \chi$ disallows us from using SIH on $\Gamma', \Gamma'' \Rightarrow \chi'$, as we are not ensured that the measure decreased between the two sequents. So, our proof technique prohibited us from using multiplicative cuts, forcing us to use additive ones. This observation was already made by Goré and Shillito [26].

Using our purely syntactic proof of cut-admissibility above, we easily obtain a cut-elimination procedure for the calculus G4iSLt extended with (cut), by simply repetitively eliminating topmost cuts first. To effectively prove this statement in Coq we explicitly encode the additive cut rule as follows:

$$\frac{(\Gamma 0\texttt{++}\Gamma 1 \texttt{ * } \varphi) \qquad (\Gamma 0\texttt{++}\varphi\texttt{::}\Gamma 1 \texttt{ * } \chi)}{(\Gamma 0\texttt{++}\Gamma 1 \texttt{ * } \chi)}$$

We encode the calculus G4iSLt + (*cut*) as `G4iSLt_cut_rules`, i.e. `G4iSLt_rules` enhanced with (cut). Finally, we turn to the elimination of additive cuts:

Theorem 3. *The additive cut rule is eliminable from* G4iSLt + (*cut*).

```
Theorem G4iSLt_cut_elimination : forall s,
 (G4iSLt_cut_prv s) -> (G4iSLt_prv s).
```

The above theorem shows that any proof in G4iSLt + (*cut*) of a sequent, i.e. `G4iSLt_cut_prv s`, can be transformed into a proof in G4iSLt of the same sequent. As this theorem is in fact a constructive function based on `Type`, we can use the extraction feature of Coq and obtain a cut-eliminating Haskell program.

6 Conclusion

This paper introduces a sequent calculus for iSL, denoted G4iSLt. It is an improvement over the sequent calculus G4iSL from [21], because backward proof search for G4iSLt is strongly terminating (instead of weakly terminating) shown via a new well-founded measure, and cut-elimination is proved directly (instead of indirectly via an equivalent calculus based on G3i [21]). All our results are formalised in Coq in a constructive way. In turn, Coq's extraction mechanism can generate a Haskell program for the cut-elimination procedure for G4iSLt.

One of the reasons to develop G4iSLt is to use its strongly terminating proof search to investigate uniform interpolation, a strengthening of Craig interpolation, in the setting of intuitionistic provability logics. Typically, calculi with good (weakly or strongly) terminating proof search form good grounds for constructive proofs of uniform interpolation (see e.g. [2,5,22,28,37,41–43]).

We also suggest to develop a countermodel construction for G4iSLt similarly to the one for G4iSL in [21]. Furthermore, as iSL is an intuitionistic modal logic only defined with $\Box$, there is the question how it can be extended by $\Diamond$ operators. It is clear from the literature of intuitionistic modal logics that several choices can be made (e.g. [4,16,33,40,47]), so we leave this for future work.

Acknowledgements. Iris van der Giessen would like to thank Sonia Marin and Marianna Girlando for an interesting discussion on the subtle choice of rules in proof systems. We would like to thank the anonymous reviewers for their helpful comments and suggestions. Van der Giessen is supported by a UKRI Future Leaders Fellowship, 'Structure vs Invariants in Proofs', project reference MR/S035540/1. Rosalie Iemhoff is supported by the Netherlands Organisation for Scientific Research under grant 639.073.807 and by the EU H2020-MSCA-RISE-2020 Project 101007627. Rajeev Goré is supported by FWF project P 33548 and the National Centre for Research and Development, Poland (NCBR), and the Luxembourg National Research Fund (FNR), under the PolLux/FNR-CORE project STV (POLLUX-VII/1/2019).

References

1. D'Abrera, C., Dawson, J., Goré, R.: A formally verified cut-elimination procedure for linear nested sequents for tense logic. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 281–298. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_17
2. Afshari, B., Leigh, G.E., Menéndez Turata, G.: Uniform interpolation from cyclic proofs: the case of modal mu-calculus. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 335–353. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_20
3. Ardeshir, M., Mojtahedi, M.: The Σ_1-provability logic of HA. Ann. Pure Appl. Logic **169**(10), 997–1043 (2018). https://doi.org/10.1016/j.apal.2018.05.001
4. Bellin, G., de Paiva, V., Ritter, E.: Extended Curry-Howard correspondence for a basic constructive modal logic. In: Proceedings of Methods for Modalities, vol. 2 (2001). https://profs.sci.univr.it/~bellin/m4m.ps
5. Bílková, M.: Interpolation in modal logics. Ph.D. thesis, Univerzita Karlova, Prague (2006). https://dspace.cuni.cz/handle/20.500.11956/15732
6. Brighton, J.: Cut elimination for GLS using the terminability of its regress process. J. Philos. Logic **45**(2), 147–153 (2016). https://doi.org/10.1007/s10992-015-9368-4
7. Perini Brogi, C.: Investigations of Proof Theory and Automated Reasoning for Non-classical Logics. Ph.D. thesis, Università degli Studi di Genova, Genova (2022)
8. Chaudhuri, K., Lima, L., Reis, G.: Formalized meta-theory of sequent calculi for linear logics. Theor. Comput. Sci. **781**, 24–38 (2019). https://doi.org/10.1016/j.tcs.2019.02.023
9. Dawson, J.E., Goré, R.: Generic methods for formalising sequent calculi applied to provability logic. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR 2010. LNCS, vol. 6397, pp. 263–277. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16242-8_19
10. Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM **22**(8), 465–476 (1979). https://doi.org/10.1145/359138.359142
11. Dyckhoff, R.: Contraction-free sequent calculi for intuitionistic logic. J. Symbolic Logic **57**(3), 795–807 (1992). https://doi.org/10.2307/2275431
12. Dyckhoff, R.: Intuitionistic decision procedures since Gentzen. In: Kahle, R., Strahm, T., Studer, T. (eds.) Advances in Proof Theory. PCSAL, vol. 28, pp. 245–267. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29198-7_6
13. Dyckhoff, R., Negri, S.: Admissibility of structural rules for contraction-free systems of intuitionistic logic. J. Symbolic Logic **65**(4), 1499–1518 (2000). https://doi.org/10.2307/2695061
14. Esakia, L.: The modalized Heyting calculus: a conservative modal extension of the intuitionistic logic. J. Appl. Non-Class. Logics **16**, 349–366 (2006). https://doi.org/10.3166/jancl.16.349-366
15. Férée, H., van Gool, S.: Formalizing and computing propositional quantifiers. In: Krebbers, R., Traytel, D., Pientka, B., Zdancewic, S. (eds.) Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2023, Boston, MA, USA, 16–17 January 2023, pp. 148–158. ACM (2023). https://doi.org/10.1145/3573105.3575668
16. Fischer-Servi, G.: On modal logic with an intuitionistic base. Stud. Logica. **36**, 141–149 (1977). https://doi.org/10.1007/bf02121259

17. Gattinger, M.: A Verified Proof of Craig Interpolation for Basic Modal Logic via Tableaux in Lean (2022). https://malv.in/2022/AiML2022-basic-modal-interpolation-lean.pdf
18. van der Giessen, I.: Uniform Interpolation and Admissible Rules: Proof-theoretic investigations into (intuitionistic) modal logics. Ph.D. thesis, Utrecht University, Utrecht (2022). https://dspace.library.uu.nl/handle/1874/423244
19. van der Giessen, I.: Admissible rules for six intuitionistic modal logics. Ann. Pure Appl. Logic **174**(4), 103233 (2023). https://doi.org/10.1016/j.apal.2022.103233
20. van der Giessen, I., Iemhoff, R.: Sequent calculi for intuitionistic gödel-Löb logic. Notre Dame J. Formal Logic **62**(2), 221–246 (2021). https://doi.org/10.1215/00294527-2021-0011
21. van der Giessen, I., Iemhoff, R.: Proof theory for intuitionistic strong Löb logic (2023). https://doi.org/10.48550/arXiv.2011.10383, (To appear in Special Volume of the Workshop Proofs!, Paris 2017)
22. van der Giessen, I., Jalali, R., Kuznets, R.: Uniform interpolation via nested sequents. In: Silva, A., Wassermann, R., de Queiroz, R. (eds.) WoLLIC 2021. LNCS, vol. 13038, pp. 337–354. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88853-4_21
23. Goré, R., Ramanayake, R.: Valentini's cut-elimination for provability logic resolved. Rev. Symb. Log. **5**(2), 212–238 (2012). https://doi.org/10.1017/S1755020311000323
24. Goré, R., Ramanayake, R., Shillito, I.: Cut-elimination for provability logic by terminating proof-search: formalised and deconstructed using coq. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 299–313. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_18
25. Goré, R., Shillito, I.: Bi-intuitionistic logics: a new instance of an old problem. In: Proceedings of the Thirteenth Conference on "Advances in Modal Logic" 24–28 August 2020, pp. 269–288 (2020). https://www.aiml.net/volumes/volume13/Gore-Shillito.pdf
26. Goré, R., Shillito, I.: Direct elimination of additive-cuts in GL4ip: verified and extracted. In: Proceedings of the Fourteenth Conference on "Advances in Modal Logic", 22–26 August 2022 (2022)
27. Hakli, R., Negri, S.: Does the deduction theorem fail for modal logic? Synthese **187**(3), 849–867 (2012). https://doi.org/10.1007/s11229-011-9905-9
28. Iemhoff, R.: Uniform interpolation and the existence of sequent calculi. Ann. Pure Appl. Logic **170**(11), 1–37 (2019). https://doi.org/10.1016/j.apal.2019.05.008
29. Iemhoff, R.: The G4i analogue of a G3i calculus. Stud. Log. **110**, 1493–1506 (2022). https://doi.org/10.1007/s11225-022-10008-3
30. Kuznetsov, A.V., Muravitsky, A.Y.: On superintuitionistic logics as fragments of proof logic extensions. Studia Logica **45**(1), 77–99 (1986). https://www.jstor.org/stable/20015249
31. Larchey-Wendling, D., Matthes, R.: Certification of breadth-first algorithms by extraction. In: Hutton, G. (ed.) MPC 2019. LNCS, vol. 11825, pp. 45–75. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33636-3_3
32. Litak, T.: Constructive modalities with provability smack. In: Bezhanishvili, G. (ed.) Leo Esakia on Duality in Modal and Intuitionistic Logics. OCL, vol. 4, pp. 187–216. Springer, Dordrecht (2014). https://doi.org/10.1007/978-94-017-8860-1_8
33. Mendler, M., de Paiva, V.: Constructive CK for contexts. Context Representation and Reasoning (CRR-2005) **13** (2005). https://www.cs.bham.ac.uk/~vdp/publications/ck-paper2.pdf

34. Mojtahedi, M.: On provability logic of HA (2022). https://doi.org/10.48550/arXiv.2206.00445
35. Muravitsky, A.: Logic **KM**: a biography. In: Bezhanishvili, G. (ed.) Leo Esakia on Duality in Modal and Intuitionistic Logics. OCL, vol. 4, pp. 155–185. Springer, Dordrecht (2014). https://doi.org/10.1007/978-94-017-8860-1_7
36. Paulson, L.C.: Constructing recursion operators in intuitionistic type theory. J. Symbol. Comput. **2**(4), 325–355 (1986). https://doi.org/10.1016/S0747-7171(86)80002-5
37. Pitts, A.M.: On an interpretation of second order quantification in first order intuitionistic propositional logic. J. Symbol. Logic **57**(1), 33–52 (1992). https://doi.org/10.2307/2275175
38. Sambin, G., Valentini, S.: The modal logic of provability: the sequential approach. J. Philos. Logic **11**, 311–342 (1982). https://doi.org/10.1007/BF00293433
39. Shillito, I.: New Foundations for the Proof Theory of Bi-Intuitionistic and Provability Logics Mechanized in Coq. Ph.D. thesis, Australian National University, Canberra (2023). https://www.proquest.com/docview/2812065824?pq-origsite=gscholar&fromopenview=true
40. Simpson, A.K.: The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh (1994). https://era.ed.ac.uk/handle/1842/407
41. Akbar Tabatabai, A., Iemhoff, R., Jalali, R.: Uniform lyndon interpolation for basic non-normal modal logics. In: Silva, A., Wassermann, R., de Queiroz, R. (eds.) WoLLIC 2021. LNCS, vol. 13038, pp. 287–301. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88853-4_18
42. Akbar Tabatabai, A., Iemhoff, R., Jalali, R.: Uniform lyndon interpolation for intuitionistic monotone modal logic. In: Advances in Modal Logic 14, Papers from the Fourteenth Conference on "Advances in Modal Logic", 22–26 August 2022. College Publications (2022). https://doi.org/10.48550/arXiv.2208.04607
43. Akbar Tabatabai, A., Jalali, R.: Universal proof theory: semi-analytic rules and uniform interpolation. CoRR (2018). http://arxiv.org/abs/1808.06258
44. Valentini, S.: The modal logic of provability: cut-elimination. J. Philos. Logic **12**, 471–476 (1983). https://doi.org/10.1007/BF00249262
45. Visser, A.: On the completeness principle: a study of provability in Heyting's arithmetic and extensions. Ann. Math. Logic **22**(3), 263–295 (1982). https://doi.org/10.1016/0003-4843(82)90024-9
46. Visser, A., Zoethout, J.: Provability logic and the completeness principle. Ann. Pure Appl. Logic **170**(6), 718–753 (2019). https://doi.org/10.1016/j.apal.2019.02.001
47. Wolter, F., Zakharyaschev, M.: On the relation between intuitionistic and classical modal logics. Algebra Logic **36**, 73–92 (1997). https://doi.org/10.1007/BF02672476

Some Analytic Systems of Rules

Timo Lang(✉)

University College London, London, UK
timo.lang@ucl.ac.uk

Abstract. We define two simple systems of rules, i.e. calculi with a global condition on the order of rule instances in a proof, for the modal logics of shift-reflexive and Euclidean frames respectively. Cut-elimination, and therefore the subformula property, can be derived directly from the cut-elimination property of adjacent logics. We compare our system to the calculus of grafted hypersequents, which has previously been used to capture both logics.

We then discuss an attempt to obtain similar 'modular' cut-elimination proofs in other systems of rules. This general attempt is carried out for two more logics, namely the modal logic of serial frames and the intermediate logic axiomatised by the law of the weak excluded middle.

1 Introduction

Among the various proof frameworks used in the investigation of nonclassical logics, *systems of rules* as introduced by Negri [16] remain relatively little studied. Broadly speaking, a system of rules is a sequent-type calculus with a global correctness condition on the order in which rules may be applied; they form an instance of *higher-level rules* [20]. In [16], for example, it is shown that extending the sequent calculus for intuitionistic logic with the system of rules

$$\dfrac{\begin{array}{c}\dfrac{A, B, \Gamma_1 \Rightarrow \Pi_1}{A, \Gamma_1 \Rightarrow \Pi_1}\,(A,B)_L \\ \vdots \\ \Gamma \Rightarrow \Pi\end{array} \qquad \begin{array}{c}\dfrac{A, B, \Gamma_2 \Rightarrow \Pi_2}{B, \Gamma_2 \Rightarrow \Pi_2}\,(A,B)_R \\ \vdots \\ \Gamma \Rightarrow \Pi\end{array}}{\Gamma \Rightarrow \Pi}\,(Lin)$$

yields a calculus for *Gödel Logic*, i.e. the extension of intuitionistic logic by the linearity axiom $(A \to B) \vee (B \to A)$. The schematic representation of the system above is understood as follows: Both rules $(A, B)_L$ and $(A, B)_R$ can be used in branches of the proof tree as long as those branches meet below in an instance of (Lin). By using such global conditions it is possible to capture analytically various logics that do not have a cutfree sequent calculus. For example, [16] develops systems of rules based on the labelled sequent calculus for all normal modal logics axiomatised by (generalised) Sahlqvist formulas. In [9] it is shown that proofs in the hypersequent calculus can be rewritten as particular systems of sequent rules, called *2-systems* (and vice versa). A different use of global conditions is shown in [1]: By replacing the (local) eigenvariable condition in

R. Ramanayake and J. Urban (Eds.): TABLEAUX 2023, LNAI 14278, pp. 94–111, 2023.
https://doi.org/10.1007/978-3-031-43513-3_6

first-order **LK**-proofs by a global condition, one obtains sound but potentially much shorter proofs.

The study of cut-elimination in systems of rules is in a rather unsatisfying stage. In [9] the analyticity of the systems of rules is obtained, but only indirectly via cut-elimination in the hypersequent calculus. [16] argues that a standard cut reduction argument goes through in the system of rules and illustrates one reduction step. As already remarked in [9], the argument seems to apply only to rules handling atomic formulas. This restriction is possible in the labelled sequent calculus but is too strong in an unlabelled system.

In the first part of this article we develop *grounded proofs*, a simple system of rules for the modal logics $\mathbf{KT}^{\Box}$ and **K5** of shift-reflexive and Euclidean frames respectively. These logics are of interest because their proof theory is less straightforward than that of other modal logics. In particular, neither shift-reflexivity nor Euclideanness is a *simple frame property* [13] which would guarantee the existence of a cutfree hypersequent calculus. The most elementary proof system for $\mathbf{KT}^{\Box}$ and **K5** seems to be the *grafted hypersequent calculus* of Lellmann and Kuznets [12]. Nested [7], prefixed tableaux [14] and labelled sequent calculi [15] are also available.

Our systems can be succinctly described as follows. For $\mathbf{KT}^{\Box}$, grounded proofs can make use of all rules of a sequent calculus for **KT**, with the proviso that every unsound modal rule has an instance of the rule (K) below it. For **K5**, grounded proofs can make use of all rules of a hypersequent calculus for **S5**, with the proviso that every unsound modal rule has an instance of the rule (MM) below it:

$$\frac{\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\,(K) \qquad \frac{\Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n}\,(MM)$$

It is a remarkable feature of both systems that their cutfree completeness can be proved *directly*, using only the deduction theorem and the cutfreeness of the (hyper)sequent calculi for **K**, **KT** and **S5**. With these ingredients the proof is almost trivial for $\mathbf{KT}^{\Box}$; for **K5** we additionally have to prove a combinatorial lemma about hypersequent derivations. In retrospect, grounded proofs can be seen as proofs in the grafted hypersequent calculus that satisfy a normal form. We make this observation precise by defining a translation from our system into the grafted hypersequent calculus, thereby obtaining a new (and arguably much simpler) proof of cut-elimination for the latter calculus.

In the second part of this article we explore the theme of *strongly modular proofs of cut-elimination*, i.e.: Proofs of cut-elimination that build on the cut-elimination property of adjacent logics (**K**, **KT** and **S5** in our example) but do not require knowledge about *how* cut-elimination for these systems was obtained. In other words, a proof of cut-elimination is strongly modular if it uses other cut-elimination theorems as 'blackboxes'. What is the scope of strongly modular proofs? We show that for many logics, strongly modular proofs of cut-elimination are possible in a simple sequent system with a global correctness condition called *revivability*. This condition however is defined only abstractly, and so the usefulness of said result depends on finding a simpler equivalent characterisation of revivability. We conclude by showing two examples where such a simple characterisation is possible: The modal logic **KD** of serial frames and the intermediate logic **LQ** axiomatised by the law of the weak excluded middle.

2 Preliminaries

Modal Logics. By a *modal logic* we mean any set of formulas in the language $\{\bot, \neg, \wedge, \vee, \rightarrow, \Box\}$ that contains all propositional tautologies, the normality axiom $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$, and is closed under uniform substitution, *Modus Ponens* (from A and $A \rightarrow B$ infer B) and *Necessitation* (from A infer $\Box A$).

The smallest modal logic (with respect to $\subseteq$) is $\mathbf{K}$. For any modal logic $\mathbf{L}$ and formula C, $\mathbf{L}+C$ denotes the smallest extension of $\mathbf{L}$ to a modal logic containing all instances of C. The table below lists some modal logics relevant to this paper, together with their corresponding frame condition (for proofs, see e.g. [5]).

modal logic	frame condition	first-order formula
$\mathbf{KT} := \mathbf{K} + \Box p \rightarrow p$	reflexive	$\forall x\, xRx$
$\mathbf{KT}^{\Box} := \mathbf{K} + \Box(\Box p \rightarrow p)$	shift-reflexive	$\forall x \forall y.\, xRy \rightarrow yRy$
$\mathbf{K5} := \mathbf{K} + \neg\Box p \rightarrow \Box\neg\Box p$	Euclidean	$\forall x \forall y \forall z.\, xRy \wedge xRz \rightarrow yRz$
$\mathbf{S5} := \mathbf{K5} + \Box p \rightarrow p$	totally connected	$\forall x \forall y.\, xRy$

The deduction theorem has to be slightly adapted for modal logics. We define $\Box^k A := \Box \ldots \Box A$ (k boxes) for $k > 0$ and $\Box^0 A := A$. A *modalized instance of* C is any formula of the form $\Box^k C_0$ where C_0 is an instance of C and $k \geq 0$. Then:

Theorem 1 (essentially [10, Theorem 2]). *$A \in \mathbf{K} + C$ iff $(\wedge \Omega) \rightarrow A \in \mathbf{K}$ for some finite set Ω of modalized instances of C.*

Sequent Calculi. A *sequent* is a pair of finite multisets of formulas written $\Gamma \Rightarrow \Delta$. Its *formula interpretation* is $\wedge\Gamma \rightarrow \vee\Delta$ where $\wedge\emptyset := \neg\bot$ and $\vee\emptyset := \bot$. We say that a sequent is valid in a logic if its formula interpretation is.

The propositional rules in Fig. 1 constitute a calculus **LK** for classical propositional logic.[1] We obtain sequent calculi

- $\mathcal{C}_{\mathbf{K}}$ by adding the modal rule (K);
- $\mathcal{C}_{\mathbf{KT}}$ by adding both modal rules (K) and (T).

[1] The metavariables in Fig. 1 are chosen such that by enforcing $|\Pi| = 0$ and $|\Delta| \leq 1$ one obtains a calculus for intuitionistic logic. This will be used in Sect. 4.3.

$$\frac{}{p \Rightarrow p}\,(id) \quad \frac{\Gamma \Rightarrow \Pi}{\Sigma, \Gamma \Rightarrow \Pi, \Delta}\,(w) \quad \frac{\Sigma, \Sigma, \Gamma \Rightarrow \Delta, \Pi, \Pi}{\Sigma, \Gamma \Rightarrow \Delta, \Pi}\,(c) \quad \frac{\Gamma \Rightarrow A, \Pi \qquad \Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \Pi}\,(cut)$$

$$\frac{}{\Gamma, \bot \Rightarrow \Delta}\,(\bot_L) \quad \frac{\Gamma \Rightarrow A, \Pi}{\Gamma, \neg A \Rightarrow \Pi}\,(\neg_L) \quad \frac{\Gamma, A \Rightarrow \Pi}{\Gamma \Rightarrow \neg A, \Pi}\,(\neg_R) \quad \frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta}\,(\wedge_L)$$

$$\frac{\Gamma \Rightarrow A, \Pi \qquad \Gamma \Rightarrow B, \Pi}{\Gamma \Rightarrow A \wedge B, \Pi}\,(\wedge_R) \quad \frac{\Gamma, A \Rightarrow \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta}\,(\vee_L) \quad \frac{\Gamma \Rightarrow A_i, \Pi}{\Gamma \Rightarrow A_1 \wedge A_2, \Pi}\,(\vee_R)$$

$$\frac{\Gamma, B \Rightarrow \Delta \qquad \Gamma \Rightarrow A, \Pi}{\Gamma, A \to B \Rightarrow \Delta, \Pi}\,(\to_L) \quad \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow A \to B, \Delta}\,(\to_R)$$

$$\frac{\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\,(K) \quad \frac{\Gamma, A \Rightarrow \Delta}{\Gamma, \Box A \Rightarrow \Delta}\,(T)$$

$$\frac{\mathcal{H} \mid \Gamma, A \Rightarrow \Delta}{\mathcal{H} \mid \Box A \Rightarrow \mid \Gamma \Rightarrow \Delta}\,(\Box^5_L) \quad \frac{\mathcal{H} \mid \Rightarrow A}{\mathcal{H} \mid \Rightarrow \Box A}\,(\Box^5_R) \quad \frac{\Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n}\,(MM)$$

$$\frac{\mathcal{H}}{\mathcal{H} \mid \Gamma \Rightarrow \Delta}\,(ew) \quad \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma \Rightarrow \Delta}\,(ec)$$

Fig. 1. Propositional, modal and structural hypersequent rules.

Derivations in sequent calculi will be denoted by letters α, β. The formula A is said to be derivable in a sequent calculus if the sequent $\Rightarrow A$ is. A sequent calculus is called *adequate for a logic* if the formulas it derives are exactly the theorems of the logic. Finally, a proof in a sequent calculus is *cutfree* if it does not use the rule (*cut*), and a sequent calculus *admits cut-elimination* if every sequent provable in it has a cutfree proof. The following is folklore:

Theorem 2. *The calculi $\mathcal{C}_{\mathbf{K}}$ and $\mathcal{C}_{\mathbf{KT}}$ are adequate for the modal logics* **K** *and* **KT** *respectively and admit cut-elimination.*

3 Two Systems of Rules

The similarity of the modal logics $\mathbf{KT}^{\Box}$ and **K5** lies in the fact that they are both 'one step away' from their companion logics **KT** and **S5** respectively. That is, in any shift-reflexive (Euclidean) frame the subframe induced by all worlds reachable from some fixed world is reflexive (totally connected), and therefore adequate for **KT** (**S5**). We formalize this observation for later reference.

Theorem 3. *Let M be a Kripke model containing a world w, and let M_w be obtained from M by restricting M's frame to worlds that are reachable from w (using one or more steps) via the accessibility relation. Then:*

1. *$M, v \models \varphi \iff M_w, v \models \varphi$ for all worlds v in M_w and modal formulas φ;*
2. *If M is shift-reflexive, then M_w is reflexive;*
3. *If M is Euclidean, then M_w is totally connected.*

From this one can easily deduce the following known equivalences:

Theorem 4. $\Box A \in \mathbf{KT}^{\Box} \iff A \in \mathbf{KT}$ *and* $\Box A \in \mathbf{K5} \iff A \in \mathbf{S5}$.

Theorem 4 implies that we can use the sequent calculus $\mathcal{C}_{\mathbf{KT}}$ and the hypersequent calculus **HS5** (see Sect. 3.2) to derive formulas in the boxed fragment of $\mathbf{KT}^{\Box}$ and **K5**. But it is not immediate what Theorem 4 tells us about the proofs of theorems in $\mathbf{KT}^{\Box}$ and **K5** that are not prefixed with $\Box$, e.g. $\neg\Box p \to \Box\neg\Box p \in \mathbf{K5}$ or $\Box\Box p \to \Box p \in \mathbf{KT}^{\Box}$.

3.1 $\mathbf{KT}^{\Box}$

We start by describing a simple system of rules for $\mathbf{KT}^{\Box}$, which is obtained by imposing a global constraint on $\mathcal{C}_{\mathbf{KT}}$-proofs. The crucial notion is the following:

Definition 1 (grounded $\mathcal{C}_{\mathbf{KT}}$-proof). *A proof in $\mathcal{C}_{\mathbf{KT}}$ is* grounded *if any lowermost modal inference in it is (K).*

In other words, only those instances of (T) are admitted in a grounded $\mathcal{C}_{\mathbf{KT}}$-proof that have an instance of (K) below. No exact pairing is required, i.e. the same instance of (K) can 'ground' multiple instances of (T) above it. Figure 2 (left and middle) shows two grounded $\mathcal{C}_{\mathbf{KT}}$-proofs with the modal rules highlighted.

$$\dfrac{\dfrac{\dfrac{p \Rightarrow p}{\Box p \Rightarrow p}\,(T)}{\Rightarrow \Box p \to p}}{\Rightarrow \Box(\Box p \to p)}\,(K) \qquad \dfrac{\dfrac{\dfrac{\dfrac{p \Rightarrow p}{\Box p \Rightarrow p}\,(T) \qquad \dfrac{p \Rightarrow p}{\Box p \Rightarrow p}\,(T)}{\Box p \vee \Box p \Rightarrow p}}{\Box(\Box p \vee \Box p) \Rightarrow \Box p}\,(K)}{\Rightarrow \Box(\Box p \vee \Box p) \to \Box p} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \Rightarrow p}{\Rightarrow p \mid \Box p \Rightarrow}\,(\Box^5_L)}{\Rightarrow p \mid \Rightarrow \neg\Box p}}{\Rightarrow \Box p, \Box\neg\Box p}\,(MM)}{\neg\Box p \Rightarrow \Box\neg\Box p}}{\Rightarrow \neg\Box p \to \Box\neg\Box p}$$

Fig. 2. Grounded proofs in **KT** (left and middle) and in **HS5** (right)

Theorem 5 (Soundness of grounded $\mathcal{C}_{\mathbf{KT}}$-proofs). *If there is a grounded $\mathcal{C}_{\mathbf{KT}}$-proof of $\Gamma \Rightarrow \Delta$, then $\Gamma \Rightarrow \Delta$ is valid in $\mathbf{KT}^{\Box}$.*

Proof. It suffices to show that the conclusion of an instance of (K) in a $\mathcal{C}_{\mathbf{KT}}$-proof is valid in $\mathbf{KT}^{\Box}$. Indeed, as the endsequent of a grounded $\mathcal{C}_{\mathbf{KT}}$-proof is derivable from the conclusions of its lowermost instances of (K) using only propositional rules, it then follows that the endsequent is valid in $\mathbf{KT}^{\Box}$ as well.[2] So let

$$\dfrac{\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A}\,(K)$$

[2] Note that if a grounded proof has no instances of (K) at all, then it is essentially a propositional proof, and so the statement is trivial.

be such an instance. As its premise $\Gamma \Rightarrow A$ is valid in $\mathbf{KT}$, we can use the deduction theorem (Theorem 1) to obtain a finite set Ω of modalized instances of the reflexivity axiom $\Box p \to p$ such that the sequent $\Omega, \Gamma \Rightarrow A$ is valid in $\mathbf{K}$. Then, by (K), also $\Box\Omega, \Box\Gamma \Rightarrow \Box A$ is valid in $\mathbf{K}$. As all formulas in $\Box\Omega$ are modalized instances of the axiom of shift-reflexivity and therefore valid in $\mathbf{KT}^{\Box}$, it follows that the reduced sequent $\Box\Gamma \Rightarrow \Box A$ is valid in $\mathbf{KT}^{\Box}$. □

Theorem 6 (Cutfree completeness of grounded $\mathcal{C}_{\mathbf{KT}}$-proofs). *If $\Gamma \Rightarrow \Delta$ is valid in $\mathbf{KT}^{\Box}$, then there is a grounded cutfree $\mathcal{C}_{\mathbf{KT}}$-proof of it.*

Proof. Let $\Gamma \Rightarrow \Delta$ be valid in $\mathbf{KT}^{\Box}$. By the deduction theorem there is a finite set Ω of modalized instances of $\Box(\Box p \to p)$ such that $\Omega, \Gamma \Rightarrow \Delta$ is valid in $\mathbf{K}$. We may write Ω as $\Box\Omega'$, where Ω' is now a set of modalized instances of $\Box p \to p$.

Consider a lowermost instance of (K) in a cutfree $\mathcal{C}_{\mathbf{K}}$-proof α of $\Omega, \Gamma \Rightarrow \Delta$:

$$\frac{\Omega', \Sigma \Rightarrow A}{\Box\Omega', \Box\Sigma \Rightarrow \Box A}\ (K)$$

Here we assume harmlessly that $\Box\Omega'$ in the conclusion of (K) contains exactly the antecessors of $\Omega = \Box\Omega'$ in the endsequent, i.e. no contraction or weakening has been applied to a formula in $\Box\Omega'$ between this instance of (K) and the endsequent. We now construct a cutfree grounded proof as follows. In α, replace the proof of the premise (for all lowermost (K) simultaneously) with a cutfree $\mathcal{C}_{\mathbf{KT}}$-proof of $\Sigma \Rightarrow A$; this is possible as every formula in Ω' is valid in $\mathbf{KT}$, and moreover $\mathbf{KT}$ admits cut-elimination. Apply (K) to obtain the sequent $\Box\Sigma \Rightarrow \Box A$, and now follow the original proof downwards while removing antecessors of $\Box\Omega'$ to eventually obtain $\Gamma \Rightarrow \Delta$. □

3.2 K5

The system of rules for $\mathbf{K5}$ will involve a hypersequent calculus for $\mathbf{S5}$, so we first introduce some notation. A *hypersequent* is a multiset of sequents written $\Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_n \Rightarrow \Delta_n$ and its (modal) formula interpretation is $\Box(\wedge\Gamma_1 \to \vee\Delta_1) \vee \ldots \vee \Box(\wedge\Gamma_1 \to \vee\Delta_1)$. We say that a hypersequent is valid in a logic if its formula interpretation is.

There are now two ways of assigning a formula to $\Gamma \Rightarrow \Delta$, namely $\Box(\wedge\Gamma \to \vee\Delta)$ "boxed" or $\wedge\Gamma \to \vee\Delta$ "flat", depending on whether we treat $\Gamma \Rightarrow \Delta$ as a one-component hypersequent or as a sequent. To avoid any ambiguity, we will explicitly say in this section that $\Gamma \Rightarrow \Delta$ is *flat-valid in a logic* $\mathbf{L}$ if $\wedge\Gamma \to \vee\Delta \in \mathbf{L}$. Otherwise, by validity of a hypersequent (possibly with only one component) we always mean the boxed interpretation above. In any modal logic $\mathbf{L} \supseteq \mathbf{KT}$ (so in particular, $\mathbf{S5}$) we have the equivalence $A \in \mathbf{L} \iff \Box A \in \mathbf{L}$ and so the notions of valid and flat-valid coincide on sequents. However, we will work in $\mathbf{K5}$ where such an equivalence does not apply.

Definition 2. *The rules of the hypersequent calculus* $\mathbf{HS5}$ *are as follows:*

- *Any rule of* **LK***, applied componentwise in a hypersequent;*
- *Additionally, we have rules* (ew) *and* (ec)*, the modal rules* $(\Box^5_L), (\Box^5_R)$ *(see Fig. 1) and the* modal merging rule (MM)*:*

$$\frac{\Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n}\ (MM)$$

There are a number of slightly different hypersequent calculi for **S5** (see the survey [3]) and any of these would be suitable for the system of rules we define below. We use a variant due to Restall [18] as this calculus underlies the grafted hypersequent calculus in [12] to which we later relate.

The only change from [18] is that we include the rule (MM). While being redundant—(MM) is derivable from $(\Box^5_L)$ and $(\Box^5_R)$—it will be useful to formulate the system of rules. Note that (MM) has no hypersequent context and so its conclusion is always a sequent. For $n = 1$ the rule coincides with (K).

Theorem 7 ([18]). HS5 *is adequate for* **S5** *and admits cut-elimination.*

Definition 3. *A proof in* **HS5** *is* grounded *if every lowermost modal rule in it is* (MM)*.*

Figure 2 (right) shows a grounded **HS5**-proof of the characteristic **K5**-axiom. While it is formally possible due to (ew) and (ec) that hypersequents with more than one component appear in the lower part of a grounded **HS5**-proof, it is easy to see that this is never necessary. We will therefore tacitly assume that Definition 3 is extended by the clause: *...and every hypersequent that is not above an instance of* (MM) *has exactly one component.* The following Lemma will give us the soundness of grounded **HS5**-proofs.

Lemma 1. *If the premise of an instance of* (MM) *is valid in* **S5***, then its conclusion is flat-valid in* **K5***.*

Proof. Assume contrapositively the conclusion $\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n$ is not flat-valid in **K5**. Then $(\wedge_{i\leq n}\Box\Gamma_i) \rightarrow (\vee_{i\leq n}\Box A_i)$ fails at a world w of an Euclidean model M. In particular, there are worlds $v_1, \ldots, v_n$ accessible from w such that v_i satisfies every formula in Γ_i but falsifies A_i. Now we use Theorem 3. Pick an arbitrary world v in M_w (say, v_1). As M_w is totally connected, every world $v_1, \ldots, v_n$ is accessible from v. Hence $\Box(\wedge\Gamma_i \rightarrow A_i)$ fails at v for every $i \leq n$, and consequently so does $\vee_{i\leq n}\Box(\wedge\Gamma_i \rightarrow A_i)$, which is the (boxed) interpretation of the premise $\Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n$ of (MM). Since M_w is totally connected, it follows that this hypersequent is not valid in **S5**. □

Theorem 8 (soundness of grounded HS5-proofs). *If there is a grounded* **HS5***-proof of* $\Gamma \Rightarrow \Delta$*, then* $\Gamma \Rightarrow \Delta$ *is flat-valid in* **K5***.*

Proof. Similar to the proof of Theorem 5. The endsequent $\Gamma \Rightarrow \Delta$ of a grounded proof is derivable from the conclusions of instances of (MM) using only propositional inferences. As these conclusions are flat-valid in **S5** by Lemma 1, the same follows[3] for $\Gamma \Rightarrow \Delta$. □

[3] Note that propositional rules preserve both validity and flat-validity.

We now turn to the cutfree completeness of grounded **HS5**-proofs. This will again be derived from the deduction theorem and cut-elimination for $\mathcal{C}_{\mathbf{K}}$ and **HS5**. The situation in **K5** is more complicated than in $\mathbf{KT}^{\Box}$ for the following reason: The outermost connective of the axiom $\Box(\Box p \to p)$ is a $\Box$, and thus the first (read bottom-up) rule that will be applied to it when used as an assumption in a $\mathcal{C}_{\mathbf{K}}$-proof is (K), i.e. the very rule that separates the top from the bottom part in our system of rules. In contrast, the outermost connective of $\neg\Box p \to \Box\neg\Box p$ is $\to$. So if we follow an occurrence of the axiom upwards in the proof, it will first be split into two different parts $\Box p$ and $\Box\neg\Box p$ via $(\to_L)$ and $(\neg_R)$ that only later encounter a modal rule. Thus at the part of the proof where we want to introduce the rule (MM) to obtain a system of rules, the constituent formulas of the axiom instances have been scattered among the branches of the $\mathcal{C}_{\mathbf{K}}$-proof. In a first step, we use the hypersequent structure to bring these scattered axiom parts back together.

Lemma 2. *The following rule is admissible in* **S5***:*

$$\frac{\mathcal{H} \mid C, \Gamma_1 \Rightarrow \Delta_1 \qquad \mathcal{H} \mid \neg\Box C, \Gamma_2 \Rightarrow \Delta_2}{\mathcal{H} \mid \Gamma_1 \Rightarrow \Delta_1 \mid \Gamma_2 \Rightarrow \Delta_2}$$

Proof. The rule can easily shown to be sound using the Kripke semantics of **S5**. It can also be derived from the generalised rule for cuts on boxed formulas that Avron uses in his proof [2] of cut-elimination for **S5**. $\Box$

$$\dfrac{\begin{array}{c}\dfrac{C,\Gamma_1\Rightarrow A_1}{\Box C,\Box\Gamma_1\Rightarrow\Box A_1}(K)\\ \vdots\,\alpha_1\\ \Box C,\Gamma\Rightarrow\Delta\end{array} \qquad \begin{array}{c}\dfrac{\neg\Box C,\Gamma_2\Rightarrow A_2}{\Box\neg\Box C,\Box\Gamma_2\Rightarrow\Box A_2}(K)\\ \vdots\,\alpha_2\\ \Box\neg\Box C,\Gamma\Rightarrow\Delta\end{array}}{\neg\Box C\to\Box\neg\Box C,\Gamma\Rightarrow\Delta}(\to_L),(\neg_R) \quad\rightsquigarrow\quad \begin{array}{c} \text{Lemma 2}\\ \dfrac{\Gamma_1\Rightarrow A_1 \mid \Gamma_2\Rightarrow A_2}{\Box\Gamma_1,\Box\Gamma_2\Rightarrow\Box A_1,\Box A_2}(MM)\\ \vdots\,\alpha_1\\ \Gamma,\Box\Gamma_2\Rightarrow\Delta,\Box A_2\\ \vdots\,\alpha_2\\ \dfrac{\Gamma,\Gamma\Rightarrow\Delta,\Delta}{\Gamma\Rightarrow\Delta}\end{array}$$

Fig. 3. Constructing a grounded **HS5**-proof

At this point we can already illustrate how the grounded **HS5**-proof will be constructed in a very simple case—see Fig. 3. Here we start from a cutfree $\mathcal{C}_{\mathbf{K}}$-proof using only a single non-modalized axiom instance $\neg\Box C \to \Box\neg\Box C$. After breaking up the axiom into two parts $\Box C$ and $\Box\neg\Box C$ using invertible rules, both parts are traced upwards in their respective branch α_1 and α_2 until they are principal in an inference of (K). Then both premises of (K) are rejoined using Lemma 2 into a single hypersequent, thereby eliminating the axiom parts. Below this hypersequent we can simulate both proofs α_1, α_2 (this time omitting the axiom parts) to arrive at the desired $\Gamma \Rightarrow \Delta$.

To deal with the general case, we need to extend Lemma 2. For this we introduce some notation: Given an index set $I = \{1, \dots, n\}$ we write $\Gamma, \{C_i\}_{i \in I} \Rightarrow \Delta$ for the sequent $\Gamma, C_1, \dots, C_n \Rightarrow \Delta$, and $\mathcal{H} \mid [\Gamma_i \Rightarrow \Delta_i]_{i \in I}$ for the hypersequent $\mathcal{H} \mid \Gamma_1 \Rightarrow \Delta_1 \mid \dots \mid \Gamma_n \Rightarrow \Delta_n$.

Lemma 3. *Let $\{C_i \mid i \in I\}$ be a set of formulas. If the hypersequent*

$$\mathcal{H} \mid \{C_j\}_{j \in J}, \{\neg\Box C_k\}_{k \in I \setminus J}, \Gamma_J \Rightarrow \Delta_J$$

is valid in **S5** *for all $J \subseteq I$, then so is $\mathcal{H} \mid [\Gamma_J \Rightarrow \Delta_J]_{J \subseteq I}$.*

Proof. By induction on $|I|$. For $I = \emptyset$ the statement is trivial. Thus let $i_0 \in I$. For $J \subseteq I$ we call S_J the hypersequent

$$\mathcal{H} \mid \{C_j\}_{j \in J}, \{\neg\Box C_k\}_{k \in I \setminus J}, \Gamma_J \Rightarrow \Delta_J.$$

For any $J \subseteq I$ with $i_0 \in J$ and $L \subseteq (I \setminus \{i_0\})$ we apply Lemma 2 (with $C := C_{i_0}$) to S_J and S_L obtaining

$$\mathcal{H} \mid \{C_j\}_{j \in J \setminus \{i_0\}}, \{\neg\Box C_k\}_{k \in I \setminus J}, \Gamma_J \Rightarrow \Delta_J \mid \{C_l\}_{l \in L}, \{\neg\Box C_m\}_{m \in (I \setminus \{i_0\}) \setminus L}, \Gamma_L \Rightarrow \Delta_L$$

Call S^*_J the component with right hand side Δ_J. Keeping J fixed while letting $L \subseteq (I \setminus \{i_0\})$ vary, we can use the induction hypothesis to obtain the hypersequent

$$\mathcal{H} \mid S^*_J \mid [\Gamma_L \Rightarrow \Delta_L]_{L \subseteq I \setminus \{i_0\}}.$$

By another application of the induction hypothesis, now letting J vary across subsets of I containing i_0 (in other words: letting J' vary across subsets of $I \setminus \{i_0\}$ and setting $J := J' \cup \{i_0\}$), we obtain

$$\mathcal{H} \mid [\Gamma_J \Rightarrow \Delta_J]_{J \subseteq I, i_0 \in J} \mid [\Gamma_L \Rightarrow \Delta_L]_{L \subseteq I \setminus \{i_0\}}$$

i.e. $\mathcal{H} \mid [\Gamma_J \Rightarrow \Delta_J]_{J \subseteq I}$. □

Note that Lemma 2 is the instance of Lemma 3 where $|I| = 1$. We can now prove the completeness theorem.

Theorem 9 (Cutfree completeness of grounded HS5-proofs). *If $\Gamma \Rightarrow \Delta$ is flat-valid in* **K5**, *then there is a cutfree grounded* **HS5**-*proof of it.*

Proof. Let $\Gamma \Rightarrow \Delta$ by flat-valid in **K5**. By the deduction theorem, there is a set Ω of modalized instances of $\neg\Box p \to \Box\neg\Box p$ such that $\Omega, \Gamma \Rightarrow \Delta$ is flat-valid in **K**, and therefore has a cutfree $\mathcal{C}_{\mathbf{K}}$-proof α. We can write Ω as $\Box\Omega_1 \cup \{\Box C_i \to \Box\neg\Box C_i\}_{i \in I}$ where $\Box\Omega_1$ contains modalized instances of the axiom with at least one box. By standard invertibility results in $\mathcal{C}_{\mathbf{K}}$, we may assume that the lowermost inferences in α are $(\to_L)$ and $(\neg_R)$ applied to all axioms $\neg\Box C_i \to \Box\neg\Box C_i$. In this way, we obtain $2^{|I|}$-many premises, which can succinctly be described as follows: For every $J \subseteq I$, we have a premise T_J containing the (negated) antecedents of all axioms with index $j \in J$ and the consequents of all other axioms, i.e.

$$T_J := \Box\Omega_1, \{\Box C_j\}_{j \in J}, \{\Box\neg\Box C_k\}_{k \in I \setminus J}, \Gamma \Rightarrow \Delta.$$

We now fix cutfree $\mathcal{C}_{\mathbf{K}}$-proofs α_J of T_J for every $J \subseteq I$. Letting P_J denote the number of lowermost inferences of (K) in α_J, we enumerate them as

$$\frac{\Omega_1, \{C_j\}_{j\in J}, \{\neg\Box C_k\}_{k\in I\setminus J}, \Gamma_J^p \Rightarrow A_J^p}{\Box\Omega_1, \{\Box C_j\}_{j\in J}, \{\Box\neg\Box C_k\}_{k\in I\setminus J}, \Box\Gamma_J^p \Rightarrow \Box A_J^p}\ (K)_J^p$$

where $0 < p \leq P_J$. Once again we assume harmlessly that the modalized axiom instances and their parts in the antecedent have not been subject to contraction or weakening. Let us assume moreover that $P_J \neq 0$ for all $J \subseteq I$, i.e. there is at least one instance of (K) in every α_J, as the other case is very simple.[4]

As the premise of $(K)_J^p$ is flat-valid in **K** and every formula in Ω_1 is valid in **S5**, it follows that the sequent

$$S_J^p := \{C_j\}_{j\in J}, \{\neg\Box C_k\}_{k\in I\setminus J}, \Gamma_J^p \Rightarrow A_J^p$$

is flat-valid, and therefore also valid, in **S5**. Define $\mathcal{F} := \{f : \mathcal{P}(I) \to \mathbb{N} \mid 0 < f(J) \leq P_J\}$ and fix one $f \in \mathcal{F}$. We think of f as choosing one specific lowermost instances $(K)_J^{f(J)}$ in every α_J. The family $\{S_J^{f(J)}\}_{J\subseteq I}$ is such that Lemma 3 is applicable to it, and therefore the following hypersequent is valid in **S5**:

$$\mathcal{H}^f := [\Gamma_J^{f(J)} \Rightarrow A_J^{f(J)}]_{J\subseteq I}$$

We now construct the grounded **HS5**-proof. Fix cutfree **HS5**-proofs β^f of $\mathcal{H}^f$ for every $f \in \mathcal{F}$. Below each β^f apply (MM) to obtain the sequent

$$\{\Box\Gamma_J^{f(J)}\}_{J\subseteq I} \Rightarrow \{\Box A_J^{f(J)}\}_{J\subseteq I}.$$

Letting $J_1, J_2, \ldots$ be an enumeration of $\mathcal{P}(I)$, we focus on the subfamily of sequents

$$\{\Box\Gamma_J^{f(J)}\}_{J\subseteq I, J\neq J_1}, \Box\Gamma_{J_1}^p \Rightarrow \Box A_{J_1}^p, \{\Box A_J^{f(J)}\}_{J\subseteq I, J\neq J_1}$$

for fixed $f \in \mathcal{F}$ and varying $0 < p \leq P_{J_1}$. In other words, we consider all possible values of f on J_1 while keeping the other values fixed. Now observe that these P_{J_1}-many sequents look similar to the conclusions of the instances $(K)_{J_1}^p$ where $0 < p \leq P_{J_1}$, only that the axiom parts have been replaced. We can therefore simulate[5] the proof α_{J_1} below these sequents obtaining

$$\{\Box\Gamma_J^{f(J)}\}_{J\subseteq I, J\neq J_1}, \Gamma \Rightarrow \Delta, \{\Box A_J^{f(J)}\}_{J\subseteq I, J\neq J_1}$$

instead of the original endsequent T_J of α_{J_1}. Starting from this new family of sequents (for all $f \in \mathcal{F}$), we can repeat the above steps, simulating the proofs α_{J_2}, α_{J_3}, $\alpha_{J_4} \ldots$ until we eventually arrive at the sequent $\Gamma, \ldots, \Gamma \Rightarrow \Delta, \ldots, \Delta$ from which we then obtain $\Gamma \Rightarrow \Delta$ by contraction. $\Box$

[4] Assume (K) is never applied in α_J. Then no modal formula is ever principal in α_J (note here that modal formulas do not appear in initial sequents, which we require to be atomic). It is then easy to see that the modal formulas in the conclusion of α_J can simply be removed to obtain a (still cutfree) $\mathcal{C}_{\mathbf{K}}$-proof of $\Gamma \Rightarrow \Delta$. This proves the theorem, as a cutfree $\mathcal{C}_{\mathbf{K}}$-proof is also a cutfree grounded **HS5**-proof.

[5] Note that α_{J_1} has only propositional inferences below $(K)_{J_1}^p$, so we do not have to worry about the changed contexts breaking some instance of (K).

3.3 Grounded Proofs and Grafted Hypersequents

In [12] calculi for the logics $\mathbf{KT}^\Box$ and **K5** are defined. These build on the notion of a *grafted hypersequent* $\Gamma \Rightarrow \Delta \mid\mid \Sigma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Sigma_n \Rightarrow \Delta_n$ consisting of a sequent $\Gamma \Rightarrow \Delta$ called the *trunk* and a hypersequent $\Sigma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Sigma_n \Rightarrow \Delta_n$ called the *crown*. If the crown is empty, we write $\Gamma \Rightarrow \Delta$ instead of $\Gamma \Rightarrow \Delta \mid\mid$. A grafted hypersequent corresponds to the modal formula $(\wedge\Gamma \rightarrow \vee\Delta) \vee \vee_{i=1}^{n}\Box(\wedge\Sigma_i \rightarrow \vee\Delta_i)$, i.e. one combines the flat interpretation of the trunk with the boxed interpretation of the crown. As pointed out in [12], grafted hypersequents are a restricted form of *nested sequents*.

We can now compare our systems of grounded proofs with the calculi in [12]. Let us first consider the grafted hypersequent calculus $\mathcal{R}_{\mathbf{K5}}$ for **K5**. We refer to [12, Figs. 1 and 2] for a complete list of the rules. The following presentation should suffice for our purposes:

- The *trunk rules* are the rules of **LK** applied to the trunk, the crown remaining unchanged;
- The *crown rules* are the rules of $\mathbf{HS5} \setminus \{(MM)\}$ applied to the crown, where it is required that the trunk is the empty sequent $\Rightarrow$;
- Two *transfer rules* mediate between the trunk and the crown:

$$\frac{\Gamma \Rightarrow \Delta \mid\mid \mathcal{H} \mid \Rightarrow A}{\Gamma \Rightarrow \Delta, \Box A \mid\mid \mathcal{H}}\ (\Box_R) \qquad \frac{\Gamma \Rightarrow \Delta \mid\mid \mathcal{H} \mid \Sigma, A \Rightarrow \Pi}{\Gamma, \Box A \Rightarrow \Delta \mid\mid \mathcal{H} \mid \Sigma \Rightarrow \Pi}\ (\Box_L)$$

A grounded **HS5**-proof can be translated into a proof in $\mathcal{R}_{\mathbf{K5}}$ as follows:

1. Replace every non-lowermost (MM) by its derivation via $(\Box_L^5)$ and $(\Box_R^5)$.
2. Replace every hypersequent $\mathcal{H}$ above some instance of (MM) by $\Rightarrow\mid\mid \mathcal{H}$.
3. Replace every lowermost (MM)-inference by transfer rules as shown below:

$$\frac{\Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n} \rightsquigarrow \cfrac{\cfrac{\Rightarrow\mid\mid \Gamma_1 \Rightarrow A_1 \mid \ldots \mid \Gamma_n \Rightarrow A_n}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow\mid\mid\Rightarrow A_1 \mid \ldots \mid\Rightarrow A_n}\ \text{some } (\Box_L)\text{'s}}{\Box\Gamma_1, \ldots, \Box\Gamma_n \Rightarrow \Box A_1, \ldots, \Box A_n}\ \text{some } (\Box_R)\text{'s}$$

The grafted hypersequent calculus $\mathcal{R}_{\mathbf{KT}^\Box}$ for the logic of shift-reflexive frames is defined similarly; here it is only componentwise applications of $\mathcal{C}_{\mathbf{KT}}$-rules that are admitted in the crown (it follows that one only needs crowns with one component). An analogous translation from grounded $\mathcal{C}_{\mathbf{KT}}$-proofs to $\mathcal{R}_{\mathbf{KT}^\Box}$ can be defined. The translated proofs satisfy a normal form that already appears in [12, see Def. 4.3].

As the translation described above does not introduce cuts, and as there are cutfree grounded proofs for all theorems of $\mathbf{KT}^\Box$ (Theorem 6) and **K5** (Theorem 8), we immediately obtain a new proof of the following (first established in [12] via a syntactic reduction procedure):

Theorem 10. *$\mathcal{R}_{\mathbf{K5}}$ and $\mathcal{R}_{\mathbf{KT}^\Box}$ admit cut elimination.*

4 Strongly Modular Proofs of Cut-Elimination

The method of the previous section can be summarized as follows: Aiming to show $\Gamma \Rightarrow \Delta$ in an extended system ($\mathbf{KT}^{\Box}$ or $\mathbf{S5}$), we start from a cutfree $\mathcal{C}_{\mathbf{K}}$-proof α of $\Omega, \Gamma \Rightarrow \Delta$ for some (modularized) axiom instances Ω of the extended logic. Then we inspect α and replace some parts of it with cutfree proofs in $\mathcal{C}_{\mathbf{KT}}$ or **HS5**, this way getting rid of the axiom instance in Ω and thereby obtaining a cutfree 'grounded' proof of $\Gamma \Rightarrow \Delta$.

We emphasize the following: *At no point in the argument one needed to understand how cut-elimination for $\mathcal{C}_{\mathbf{K}}$, $\mathcal{C}_{\mathbf{KT}}$ and* **HS5** *is established.* In other words, these cut-elimination results are used as 'blackboxes' in the proof. Let us introduce the following informal terminology: A proof of cut-elimination is

- *weakly modular* if it is obtained by modifying or extending the cut-elimination proof of some other logic;
- *strongly modular* if it is obtained by using the cut-elimination property of some other logic, irrespective of how this property was obtained.

Our proofs of Theorem 6 and Theorem 9 are strongly modular in this sense. We are not aware of other such proofs in the literature.[6] On the other hand, weakly modular proofs are numerous: One might for example argue for cut-elimination in $\mathcal{C}_{\mathbf{KT}}$ by describing how the reduction steps in the cut-elimination algorithm for $\mathcal{C}_{\mathbf{K}}$ have to be extended to accommodate the additional rule (T).[7] The disadvantage of this approach is of course that the reader has to know the algorithm for $\mathcal{C}_{\mathbf{K}}$. If such a proof were to be formalised, one would have to copy and extend the complete formalisation of the proof for $\mathcal{C}_{\mathbf{K}}$, instead of using $\mathcal{C}_{\mathbf{K}}$'s already established cut-elimination as a lemma in the formalised proof for $\mathcal{C}_{\mathbf{KT}}$. The most successful attempts at modularity in cut-elimination have been proofs that are parametrized over a specific class of axioms or rules (e.g. [4,8,13,17]).

We believe strongly modular proofs of cut-elimination are interesting and deserve further study. They have the potential of being both shorter[8] and more reliable through the reuse of already established theorems. Moreover, given the general significance of cut-elimination, any method for obtaining it is important.

Of course, with only two[9] examples at hand there is the possibility that we have encountered a 'happy coincidence' rather than a general idea. Indeed the situation of $\mathbf{KT}^{\Box}$ and $\mathbf{K5}$ is quite special in that they are sandwiched between logics with cutfree calculi, i.e. $\mathbf{K} \subseteq \mathbf{KT}^{\Box} \subseteq \mathbf{KT}$ and $\mathbf{K} \subseteq \mathbf{K5} \subseteq \mathbf{S5}$, and the gap to the 'upper logic' **KT** or **S5** is very small in a precise sense (Theorem 4).

In the remainder of this article we sketch an idea that could be useful for obtaining strongly modular proofs of cut-elimination for other logics. We conduct

[6] We do not count proofs using cutfreeness of another calculus for the *same* logic, or a conservative extension thereof.

[7] Also, a *weakly modular* proof of cut-elimination for grounded **KT**-proofs is obtained by observing that all reduction steps in $\mathcal{C}_{\mathbf{KT}}$'s cut-elimination preserve groundedness.

[8] E.g., compare our proof for **K5** with the one in the grafted hypersequent calculus [12].

[9] Side remark: The result for $\mathbf{KT}^{\Box}$ also applies to all modal logics $\mathbf{K} + \Box C$ where $\mathbf{K} + C$ has a cutfree calculus.

the discussion in a semi-formal style. While there will not be enough evidence for a ‘general method’, we do present two further examples where a strongly modular proof is possible: The modal logic **KD** (using cut-elimination in **K**) and the intermediate logic **LQ** (using cut-elimination in intuitionistic logic).

4.1 Calculi with Ghost Rules

We start from the general situation that $\mathbf{L} \subsetneq \mathbf{M}$ where $\mathbf{L}$ is some logic with a cutfree sequent calculus $\mathcal{C}_{\mathbf{L}}$. We seek a calculus for $\mathbf{M}$ that admits a strongly modular proof of cut-elimination, relative to cut-elimination in $\mathcal{C}_{\mathbf{L}}$. We additionally assume that a deduction theorem holds between $\mathbf{L}$ and $\mathbf{M}$. That is, a sequent $\Gamma \Rightarrow \Delta$ is valid in $\mathbf{M}$ iff $\Omega, \Gamma \Rightarrow \Delta$ is valid (and therefore cutfree provable) in $\mathbf{L}$ for a suitable set of formulas Ω.

Our proofs of the completeness theorems (Theorems 6 and 9) suggest that we should attempt to construct a cutfree $\mathbf{M}$-proof of $\Gamma \Rightarrow \Delta$ by somehow transforming a cutfree $\mathcal{C}_{\mathbf{L}}$-proof α of $\Omega, \Gamma \Rightarrow \Delta$. Now one naive transformation might immediately spring to mind: Can we simply take α and remove all occurrences of Ω and its ancestors in α to obtain a cutfree proof $\alpha^\dagger$ of $\Gamma \Rightarrow \Delta$?

The first question then is, in what system does $\alpha^\dagger$ qualify as a proof? Clearly removing formulas from inferences in $\mathcal{C}_{\mathbf{L}}$ creates unsound rules. In a first step, we therefore extend $\mathcal{C}_{\mathbf{L}}$ with *‘ghost rules’*: These are rules in which the principal formula in the conclusion and its ancestors in the premises have been removed. For examples, the ghost rules corresponding to $(\wedge_R)$ and (K) are

$$\frac{\Gamma \Rightarrow \Delta \qquad \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\,(\wedge_R)^\dagger \quad \text{and} \quad \frac{\Gamma \Rightarrow}{\Box\Gamma \Rightarrow}\,(K)^\dagger.$$

Different rules can have the same ghost rules, e.g. $(\wedge_R)^\dagger = (\vee_L)^\dagger$. Some ghost rules, e.g. $(\wedge_L)^\dagger$, are ‘dummy inferences’ $\Gamma \Rightarrow \Delta / \Gamma \Rightarrow \Delta$ that we do not add to the system. If $\mathcal{C}_{\mathbf{L}}$ has initial sequents $p \Rightarrow p$ then one or both occurrences of p can be ancestors of Ω, and thus we need different ghost initial sequents:

$$\frac{}{\Rightarrow p}\,(* \Rightarrow)^\dagger \qquad \frac{}{p \Rightarrow}\,(\Rightarrow *)^\dagger \qquad \frac{}{\Rightarrow}\,(* \Rightarrow *)^\dagger$$

Letting $\mathcal{C}_{\mathbf{L}}^\dagger$ denote the calculus extended by such ghost inferences we see that $\alpha^\dagger$ is (up to dummy inferences) a cutfree $\mathcal{C}_{\mathbf{L}}^\dagger$-proof of $\Gamma \Rightarrow \Delta$. More generally we infer from the deduction theorem that every sequent valid in $\mathbf{M}$ has a cutfree proof in $\mathcal{C}_{\mathbf{L}}^\dagger$. But of course, $\mathcal{C}_{\mathbf{L}}^\dagger$ also has many derivations which do not correspond to proofs in $\mathbf{M}$.

Definition 4. *A class $\mathbb{P}$ of $\mathcal{C}_{\mathbf{L}}^\dagger$-proofs is* cutfree-adequate for $\mathbf{M}$ *if the endsequent of every $\mathbb{P}$-proof is valid in $\mathbf{M}$ (‘soundness’) and there is a cutfree $\mathbb{P}$-proof of every $\mathbf{M}$-valid sequent (‘completeness’).*

Let us informally call $\mathbf{M}$-*revivable* a $\mathcal{C}_{\mathbf{L}}^\dagger$-proof of $\Gamma \Rightarrow \Delta$ if we can insert formulas and inferences into it to obtain a $\mathcal{C}_{\mathbf{L}}$-proof of $\Omega, \Gamma \Rightarrow \Delta$, where Ω is a set of $\mathbf{M}$-valid formulas. The proof $\alpha^\dagger$ from the above discussion is the typical example of an $\mathbf{M}$-revivable proof.

By the deduction theorem and cut-elimination in $\mathcal{C}_{\mathbf{L}}$ it follows that the **M**-revivable proofs in $\mathcal{C}^{\dagger}_{\mathbf{L}}$ form a cutfree-adequate class for **M**.[10] So what we have obtained is indeed a strongly modular proof of cut-elimination for the system of **M**-revivable $\mathcal{C}^{\dagger}_{\mathbf{L}}$-proofs. The property of being **M**-revivable can be seen as a global correcteness condition on $\mathcal{C}^{\dagger}_{\mathbf{L}}$-proofs, and therefore constitutes—in its broadest interpretation—a system of rules for $\mathcal{C}^{\dagger}_{\mathbf{L}}$. But of course this observation is rather[11] useless in practice unless we can express the property of being revivable in simpler terms, say via a condition on the order of rules being applied.

To conclude this article, we now discuss two logics—**KD** and **LQ**—where this is the case. Their similarity lies in the fact that they admit a very strong version of the deduction theorem, and this will allow us to express their notions of 'revivability' in fairly simple terms. In doing so, we obtain both a system of rules and a strongly modular proof of cut-elimination.

4.2 $\mathbf{K} \subseteq \mathbf{KD}$

The modal logic **KD** is the extension of **K** by the seriality axiom $\neg\Box\bot$; in terms of the Kripke semantics, $\neg\Box\bot$ enforces that every world has at least one successor. It is well-known (see, e.g., [13]) that extending $\mathcal{C}_{\mathbf{K}}$ with the rule

$$\frac{\Gamma \Rightarrow}{\Box\Gamma \Rightarrow}\ (D)$$

yields a sequent calculus $\mathcal{C}_{\mathbf{KD}}$ for **KD** admitting cut-elimination. We now present a new proof of cut-elimination for **KD** that is strongly modular.

As the seriality axiom has no variables, the modalized instances of it are exactly the formulas $\Box^k\neg\Box\bot$ for $k \geq 0$. Following the methodology sketched in the previous section, we now extend $\mathcal{C}_{\mathbf{K}}$ to a calculus $\mathcal{C}^{\dagger}_{\mathbf{K}}$ with ghost rules. Crucially, the ghost rule $(K)^{\dagger}$ coincides with the rule (D) above.

Theorem 11. *Those proofs in $\mathcal{C}^{\dagger}_{\mathbf{K}}$ whose only ghost rule is $(K)^{\dagger}$ form a cutfree-adequate class for* **KD***.*

Proof. Let us first deal with completeness. If $\Gamma \Rightarrow \Delta$ is valid in **KD**, then there is a set of modalized instances of $\neg\Box\bot$ such that $\Omega, \Gamma \Rightarrow \Delta$ has a $\mathcal{C}_{\mathbf{K}}$-proof α. Using cut-elimination in $\mathcal{C}_{\mathbf{K}}$, we may assume that α is cutfree. As there is no right rule for $\bot$, the $\mathcal{C}_{\mathbf{K}}$-rules that can be applied in α to an ancestor of a modalized instance of $\neg\Box\bot$ in Ω are only $(\neg_L)$ and (K). Now obtain $\alpha^{\dagger}$ by removing Ω and all its ancestors from the proof. As $(\neg_L)^{\dagger}$ is a dummy rule, the only ghost rule we need to create is $(K)^{\dagger}$. Thus $\alpha^{\dagger}$ is as desired.

[10] The idea of systematically replacing systems of rules with axiom instances in order to prove *soundness* already appears in [16].

[11] One could maybe make the following remark: When looking for a simple cut-free sequent calculus that endowed with *some* global correctness criterion captures the logic **M**, one does not have to look further than $\mathcal{C}^{\dagger}_{\mathbf{L}}$.

We now turn to soundness. For this we have to 'revive' a $\mathcal{C}^{\dagger}_{\mathbf{K}}$-proof β of $\Gamma \Rightarrow \Delta$ whose only ghost rule is $(K)^{\dagger}$. This is done as follows:

$$\frac{\Gamma \Rightarrow}{\Box\Gamma \Rightarrow}\,(K)^{\dagger} \quad \rightsquigarrow \quad \cfrac{\cfrac{\cfrac{\Gamma \Rightarrow}{\Gamma \Rightarrow \bot}\,(w)}{\Box\Gamma \Rightarrow \Box\bot}\,(K)}{\Box\Gamma, \neg\Box\bot \Rightarrow}\,(\neg_L)$$

Now propagate the newly added $\neg\Box\bot$ downwards in the proof. We will have to add $\Box$'s in front of it whenever we encounter the rule (K). Doing so for all instances of $(K)^{\dagger}$ we eventually obtain a $\mathcal{C}_{\mathbf{K}}$-proof of $\Omega, \Gamma \Rightarrow \Delta$ where Ω contains modalized instances of $\neg\Box\bot$. Thus $\Gamma \Rightarrow \Delta$ is valid in **KD**. $\Box$

As restricting the ghost inferences in $\mathcal{C}^{\dagger}_{\mathbf{K}}$ to $(K)^{\dagger}$ yields exactly $\mathcal{C}_{\mathbf{KD}}$, we have obtained a new (and strongly modular) proof of cut-elimination for $\mathcal{C}_{\mathbf{KD}}$.

4.3 IL $\subseteq$ LQ

For our final example, we leave the realm of modal logics and consider an intermediate logic instead. **LQ** extends **IL** by the law of *weak excluded middle* $\neg p \vee \neg\neg p$; it is known [11] that the following deduction theorem holds: $A \in \mathbf{LQ} \iff (\wedge_{i \leq n} \neg p_i \vee \neg\neg p_i) \rightarrow A \in \mathbf{IL}$ where $p_1, \ldots, p_n$ are the variables occurring in A. Let $\mathcal{C}_{\mathbf{IL}}$ be the single-conclusion calculus obtained from the first group of rules in Fig. 1 by stipulating that $|\Pi| = 0$ and $|\Delta| \leq 1$. $\mathcal{C}_{\mathbf{IL}}$ is adequate for **IL** and admits cut-elimination.

Definition 5. *A proof in* $\mathcal{C}^{\dagger}_{\mathbf{IL}}$ *is* **LQ***-grounded if the following holds:*

1. *The only ghost rules in it are* $(\vee_L)^{\dagger}$ *and ghost initial sequents* $\Rightarrow p$, $p \Rightarrow$, $\Rightarrow$.
2. *Letting* $(\vee_L)^{\dagger}_1, \ldots, (\vee_L)^{\dagger}_n$ *denote all instance of* $(\vee_L)^{\dagger}$ *in the proof, there are sets* $L_1, R_1, \ldots, L_n, R_n$ *of ghost initial sequent occurrences such that*
   - *every ghost initial sequent* $p \Rightarrow$ *(resp.* $\Rightarrow p$*, resp.* $\Rightarrow$*) appears in exactly one* L_i *(resp. exactly one* R_i*, resp. exactly one* R_i *and exactly one* L_j*);*
   - *No two distinct variables appear in connected components, where being connected is the reflexive, transitive and symmetric closure of the relation* $L_i \sim R_j \iff i = j \vee L_i \cap R_j \neq \emptyset$
   - *Every branch of the proof containing a sequent in* L_i *(*R_i*) goes through the left (right) premise of* $(\vee_L)^{\dagger}_i$*. If it goes through the right premise, it contains a sequent with empty right hand side above* $(\vee_L)^{\dagger}_i$*.*

Figure 4 (middle) shows a simple **LQ**-grounded proof where $n = 1$.

Theorem 12. *The class of* **LQ***-grounded* $\mathcal{C}^{\dagger}_{\mathbf{IL}}$*-proofs is cutfree-adequate for* **LQ**.

Proof. (Sketch). Completeness is similar to Theorem 11; **LQ**'s special deduction theorem restricts the necessary ghost inferences to initial sequents and $(\vee_L)^{\dagger}$.

We now show soundness by 'reviving' an **LQ**-grounded proof of $\Gamma \Rightarrow \Delta$. Start by adding variables and $(\neg_R)$-inferences to the ghost initial sequents as follows:

$$(p \Rightarrow) \in L_i \rightsquigarrow \left(\frac{p \Rightarrow p^{L_i}}{p, \neg p^{L_i} \Rightarrow}\right) \quad (\Rightarrow p) \in R_i \rightsquigarrow (p^{R_i} \Rightarrow p) \quad (\Rightarrow) \in L_i \cap R_j \rightsquigarrow \left(\frac{p^{R_j} \Rightarrow p^{L_i}}{p^{R_j}, \neg p^{L_i} \Rightarrow}\right)$$

The superscripts act only as markers, i.e. p, p^{R_i}, p^{L_i} denote the same variable. In replacing $(\Rightarrow) \in L_i \cap R_j$ we add the variable p from a component connected to L_i or R_j (unique if it exists) and an arbitrary variable otherwise; in the other cases the choice of the added variable is forced by the preexisting p. The $\neg p^{L_i}$'s are then propagated downwards until the left premise of $(\vee_L)_i^\dagger$. The p^{R_i}'s are propagated downwards until we encounter the first sequent $\Sigma \Rightarrow$ with empty right hand side, at which point we introduce double negations:

$$(\Sigma \Rightarrow) \rightsquigarrow \left(\frac{\dfrac{\Sigma, p^{R_i} \Rightarrow}{\Sigma \Rightarrow \neg p^{R_i}}}{\Sigma, \neg\neg p^{R_i} \Rightarrow}\right)$$

Propagate the $\neg\neg p^{R_i}$'s down to the right premise of $(\vee_L)_i^\dagger$ and rewrite as follows:

$$\frac{\Sigma \Rightarrow \Pi \qquad \Sigma \Rightarrow \Pi}{\Sigma \Rightarrow \Pi}\,(\vee_L)_i^\dagger \quad \rightsquigarrow \quad \frac{\Sigma, \neg p^{L_i} \Rightarrow \Pi \qquad \Sigma, \neg\neg p^{R_i} \Rightarrow \Pi}{\Sigma, \neg p \vee \neg\neg p \Rightarrow \Pi}\,(\vee_L)$$

Propagate the new formula $\neg p \vee \neg\neg p$ to the endsequent. Doing so for all $i \leq n$, we obtain a $\mathcal{C}_{\mathbf{IL}}$-proof of $\Omega, \Gamma \Rightarrow \Delta$ where Ω contains instances of the weak excluded middle axiom. Thus $\Gamma \Rightarrow \Delta$ is valid in **LQ**. □

It is instructive to compare **LQ**-grounded proofs to other calculi in the literature. For example, a hypersequent calculus for **LQ** [8] is obtained by adding the rule (lq) (below left) to a hypersequent calculus for intuitionistic logic.[12] The corresponding 2-system of rules [9] is pictured on the right:

$$\frac{\Sigma, \Sigma' \Rightarrow}{\Sigma \Rightarrow \mid \Sigma' \Rightarrow}\,(lq) \qquad\qquad \frac{\begin{array}{cc} \dfrac{}{\Sigma \Rightarrow} & \dfrac{\Sigma, \Sigma' \Rightarrow}{\Sigma' \Rightarrow} \\ \vdots & \vdots \\ \Gamma \Rightarrow \Delta & \Gamma \Rightarrow \Delta \end{array}}{\Gamma \Rightarrow \Delta}\,(bot)$$

Figure 4 hints at the translation of **LQ**-grounded proofs into both calculi.

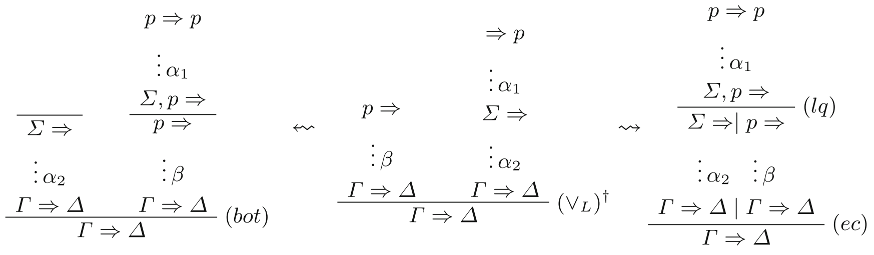


Fig. 4. From **LQ**-grounded proofs to 2-systems (left) and hypersequents (right)

[12] An interesting sequent calculus for **LQ** is presented in [6].

5 Conclusion and Future Work

We have defined *grounded proofs*, a system of rules for $\mathbf{KT}^{\Box}$ and **K5**, and proved the cut-elimination theorem. We showed how grounded proofs relate to grafted hypersequents, thereby recovering and simplifying the cut-elimination theorem for the latter calculus. We then elaborated on *strongly modular proofs of cut-elimination*, providing two more examples through the logics **KD** and **LQ**.

Future work. Strongly modular proofs do not directly yield an algorithm for eliminating cuts. We would like to know whether the arguments given here can be used to write an algorithm that, e.g., eliminates cuts in grounded **K5**-proofs by calling the cut-elimination algorithms for **K** and **S5** as subroutines.

The method of obtaining strongly modular proofs through calculi with ghost rules is in a very early stage and so much remains to be explored. As a first step, one could try to extend the argument for **LQ** to all intermediate logics with a similar deduction theorem, i.e. logics with the *simple substitution property* [19].

Acknowledgements. The author is indebted to the anonymous reviewers for many corrections and helpful suggestions.

References

1. Aguilera, J.P., Baaz, M.: Unsound inferences make proofs shorter. J. Symb. Logic **84**(1), 102–122 (2019)
2. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In: Hodges, W. (ed.) Logic: Foundations to Applications, pp. 1–32 (1996)
3. Bednarska, K., Indrzejczak, A.: Hypersequent calculi for S5: the methods of cut elimination. Logic Log. Philos. **24**, 08 (2015)
4. Belnap, N.D.: Display logic. J. Philos. Logic 375–417 (1982)
5. Blackburn, P., van Benthem, J., Wolter, F.: Handbook of Modal Logic. Elsevier (2006)
6. Boričić, B.R.: A cut-free Gentzen-type system for the logic of the weak law of excluded middle. Stud. Logica. **45**, 39–53 (1986)
7. Brünnler, K.: Deep sequent systems for modal logic. Arch. Math. Logic **48**(6), 551–577 (2009)
8. Ciabattoni, A., Galatos, N., Terui, K.: From axioms to analytic rules in nonclassical logics. In: 2008 23rd Annual IEEE Symposium on Logic in Computer Science, pp. 229–240. IEEE (2008)
9. Ciabattoni, A., Genco, F.A.: Hypersequents and systems of rules: embeddings and applications. ACM Trans. Comput. Logic (TOCL) **19**(2), 1–27 (2018)
10. Fitting, M.: Modal proof theory. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic. Studies in Logic and Practical Reasoning, vol. 3, pp. 85–138. Elsevier (2007)
11. Hosoi, T.: Pseudo two-valued evaluation method for intermediate logics. Stud. Logica. **45**, 3–8 (1986)
12. Kuznets, R., Lellmann, B.: Grafting hypersequents onto nested sequents. Logic J. IGPL **24**(3), 375–423 (2016)

13. Lahav, O.: From frame properties to hypersequent rules in modal logics. In: 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 408–417. IEEE (2013)
14. Massacci, F.: Strongly analytic tableaux for normal modal logics. In: Bundy, A. (ed.) CADE 1994. LNCS, vol. 814, pp. 723–737. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58156-1_52
15. Negri, S.: Proof analysis in modal logic. J. Philos. Log. **34**, 507–544 (2005)
16. Negri, S.: Proof analysis beyond geometric theories: from rule systems to systems of rules. J. Log. Comput. **26**(2), 513–537 (2014)
17. Pattinson, D., Schröder, L: Generic modal cut elimination applied to conditional logics. Logical Methods Comput. Sci. **7** (2011)
18. Restall, G.: Proofnets for S5: sequents and circuits for modal logic. In: Dimitracopoulos, C., Newelski, L., Normann, D. (eds.) Logic Colloquium 2005, pp. 151–172. Cambridge University Press, Cambridge (2007)
19. Sasaki, K.: The simple substitution property of the intermediate propositional logics. Bull. Section Logic **18**(3) (1989)
20. Schroeder-Heister, P.: The calculus of higher-level rules, propositional quantification, and the foundational approach to proof-theoretic harmony. Stud. Logica. **102**, 1185–1216 (2014)

A Cut-Free, Sound and Complete Russellian Theory of Definite Descriptions

Andrzej Indrzejczak and Nils Kürbis[✉]

Department of Logic, University of Lodz, Lodz, Poland
{andrzej.indrzejczak,nils.kurbis}@filhist.uni.lodz.pl

Abstract. We present a sequent calculus for first-order logic with lambda terms and definite descriptions. The theory formalised by this calculus is essentially Russellian, but avoids some of its well known drawbacks and treats definite description as genuine terms. A constructive proof of the cut elimination theorem and a Henkin-style proof of completeness are the main results of this contribution.

Keywords: Definite Descriptions · Predicate abstracts · Sequent Calculus · Cut Elimination

1 Introduction

Definite descriptions (DD) are complex terms commonly applied not only in natural languages but also in mathematics and computer science. In formal languages they are usually expressed by means of the iota operator, which forms terms from formulas. Thus $\imath x\varphi$ means 'the (only) x satisfying φ'. A DD aims to denote a unique object by virtue of a property that only it has. Sometimes a DD fails, because nothing or more than one thing has the property. A DD that succeeds to denote only one object is *proper*; otherwise it is *improper*.

Definite descriptions, proper and improper, are ubiquitous not only in natural languages but also in mathematics and science (like the proper 'the sum of 7 and 5' or the improper 'the square root of n'). In formal languages the application of functional terms is the prevailing way of representing complex names. However, applying DD can outrun functional terms in many ways, since they are more expressive than functional terms, in the sense that an arbitrary functional term $f^n(t_1, \ldots, t_n)$ can be represented as a description $\imath xF^{n+1}(x, t_1, \ldots, t_n)$, where F is a predicate corresponding to the function f. On the other hand, not every definite description, even if proper, can be expressed using functional terms; it is possible only in the case of predicates expressing functional relations, whereas every sentence can be used to form a DD. For example, both 'the father of Ben'

Funded by the European Union (ERC, ExtenDD, project number: 101054714). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

and ‘the daughter of Mary’ may be represented as terms using the iota operator, but only the first may be represented as a functional term. Moreover, even if we can use functional terms instead of DD we enrich a language with another sort of functors in addition to predicates. This has an impact on the formalisation of valid arguments in which very often the conclusion follows on the basis of the content expressed by functional terms which is directly expressed by predicates. For example: ‘Adam has children’ follows from ‘Adam is the father of Ben’. However to prove its validity, its formal representation $a = f(b) \vdash \exists x(Cxa)$ requires two enthymematic premisses: $\forall xy(Mxy \vee Fxy \leftrightarrow Cyx)$ and $\forall xy(x = f(y) \leftrightarrow Fxy)$. Let us call the latter premiss a bridge principle allowing us to transfer information conveyed by predicates to related functions and vice versa. In general they have a form: $\forall x_1, \dots, x_n, y(y = f^n(x_1, \dots, x_n) \leftrightarrow F^{n+1}(y, x_1, \dots, x_n)$ and show how the information encoded by the functional predicates is represented by predicates. In the case of using DD instead of functional terms we do not need such extra bridge principles, whereas in languages with functional terms they are necessary in an analysis of obviously valid arguments.[1]

The usefulness of formal devices like the iota operator and other term-forming operators has recently been better recognised (cf. Tennant’s [32] or Scott and Benzmüller’s implementation of free logic using proof assistant *Isabelle/HOL* [3]) also in the fields connected with computer science, like differential dynamic logic used for verification of hybrid systems [5] or description logics (see [1] or [25]). Logics with DD are often implemented to enable formalisation of deep philosophical problems. e.g. Anselm’s ontological argument (see the work by Oppenheimer and Zalta using the automated reasoning tool *PROVER9* [26] or its encoding by Blumson [4]).

Since several rival theories of DD were formulated, the applicability and potential usefulness of DD was underestimated so far. It leads to a question which approach is the best one, at least for some specific kind of applications. In this paper we focus on the Russellian approach to definite descriptions ([28] and [35]) which plays a central role in this area. Although Russell’s theory of DD has some controversial points, it became a standard point of reference of almost all works devoted to the analysis of definite descriptions. Moreover, it is still widely accepted by formal logicians as a proper way of handling descriptions; the scores of textbooks that use it as their official theory of definite descriptions count as witnesses for this claim. Russell’s theory has also strong affinities to logics closely connected with applications in constructive mathematics and computer science like the logic of the existence predicate by Scott [30] or the definedness logic (or the logic of partial terms) of Beeson [2] and Feferman [8]. These connections were elaborated in [14].

Russell treated DD as incomplete signs and defined their use by contextual definitions of the form:

$$\psi[x/\imath y\varphi] \ := \ \exists x(\forall y(\varphi \leftrightarrow y = x) \wedge \psi)$$

[1] Some other advantages of using DD instead of functional terms are discussed in more detail in [17].

but this solution leads to scoping difficulties if ψ is not elementary. $\neg\psi[x/\imath y\varphi]$, e.g., is ambiguous: is the whole formula negated or only the predicate ψ? The method which Russell introduced in [35] to draw scope distinctions is rather clumsy. Fortunately, it is possible to develop a logic which treats DD as genuine terms and yet retains desirable features of the Russellian approach. Such a logic was formalised as a natural deduction system by Kalish, Montague, and Mar [18] and by Francez and Więckowski [11]. These systems involve complex rules and axioms, but recently Indrzejczak [16] provided an analytic and cut-free sequent calculus equivalent to the Russellian logic as formalised in [18]. However, in all these systems the formal counterpart of the Russellian policy of eliminating DD from sentences must be restricted to predicate letters, which is connected with the scoping difficulties of the Russellian approach just mentioned.

Can we offer any improvement on the state of the art? A possible strategy of avoiding these problems is to treat DD by means of a binary quantifier; this approach was formally developed by Kürbis (cf. [19–23]). However, if we want to treat DD as terms, then the introduction of the lambda operator to construct complex predicate abstracts from formulas offers a good solution. $\lambda x\varphi$ means 'the property of being φ' and applied to some term, in particular to a DD, forms a formula called a lambda atom. This device was introduced into studies of modal predicate logic by Thomason and Stalnaker [31], and the idea was further developed by Bressan [6] and Fitting [9], in particular, to distinguish between *de dicto* and *de re* reading of modal operators. Independently, this technique was used by Scales [29] in his formulation of attributional logic, where Aristotle's distinction between the negation of a sentence and of a predicate is formally expressible. In fact, Scales seems to be the first one to apply predicate abstraction to formalise a theory of DD which relates closely to Russell's. Predicate abstracts were also successfully applied by Fitting and Mendelsohn [10] to obtain a theory of DD in a modal setting. This approach, with slight modifications, was further developed independently by Orlandelli [27] and Indrzejczak [12] to obtain cut-free sequent calculi for modal logics with DD and predicate abstracts.

In this article we focus on a different logic **RL**, first introduced in [17], which also combines the iota and lambda operators. It avoids the shortcomings of the Russellian approach while saving all its plausible features. Predicate abstracts permit us to draw scope distinctions rather more elegantly than with the Russellian scope markers and their application is more general. **RL** is essentially Russellian but with DD treated as genuine terms. Nonetheless, the reductionist aspect of Russell's approach is retained in several ways. On the level of syntax the occurrences of DD are restricted to arguments of predicate abstracts to form lambda atoms. On the level of semantics DD are not defined by an interpretation function but by satisfaction clauses for lambda atoms. Eventually, on the level of calculus DD cannot be instantiated for variables in quantifier rules but are subject to special rules for lambda atoms. This strict connection of DD with predicate abstracts avoids disadvantages of the Russellian approach connected with scoping difficulties, and, at the same time, simplifies proofs of metalogical properties.

RL was originally characterised semantically and formalised as an analytic tableau calculus in [17], where it was also applied for proving the Craig interpolation theorem. Here we are completing the research on **RL** by providing an adequate sequent calculus for which the cut elimination theorem is proved constructively. We characterise the language, semantics and axiomatisation of **RL** in Sect. 2. Then we present the sequent calculus GRL for **RL** and show its equivalence with an axiomatic Hilbert style system HRL. Section 4 contains a proof of the cut elimination theorem, and Sect. 5 a Henkin-style proof of completeness. The paper finishes with some comparative remarks.

2 Preliminaries

The language $\mathcal{L}$ of **RL** is standard, except that it contains the operators $\imath$ and λ. Following the remarks on the functional terms from the Introduction, as well as the original Russellian attitude towards terms, the 'official' language has neither constant nor function symbols; in the completeness proof we add constants solely for the purpose of constructing models from consistent sets. As is customary in proof theoretic investigations since Gentzen, we distinguish free and bound variables graphically in deductions. It is not customary to make this distinction in semantics, and so there we won't make it either. This blend of two customs should not lead to confusion, and we are following Fitting and Mendelsohn [10] in this respect. There are two disjoint sets VAR of variables and PAR of parameters. The former plays the role of the bound, the latter of the free variables in the presentation of the proof theory of **RL**; in the presentation of the semantics, this restriction is relaxed and members of VAR are permitted as free variables. The *terms* of the language in the strict sense are the variables and parameters. Expressions formed by $\imath$ are admitted as terms in a more general sense: their application is restricted to predicate abstracts and they are called quasi-terms. We mention only the following formation rules for the more general notion of a formula used in the semantics:

- If P^n is a predicate symbol (including $=$) and $t_1 \dots t_n \in VAR \cup PAR$, then $P^n(t_1, ..., t_n)$ is a formula (atomic formula).
- If φ is a formula, then $(\lambda x \varphi)$ is a predicate abstract.
- If φ is a formula, then $\imath x \varphi$ is a quasi-term.
- If φ is a predicate abstract and t a term or quasi-term, then φt is a formula (lambda atom).

$\varphi[x/t]$ denotes the result of replacing x by t in φ. To save space, we'll often write φ^x_t instead of $\varphi[x/t]$. If t is a variable y, it is assumed that y is free for x in φ, that is, no occurrence of y becomes bound in φ in the replacement. To save space and simplify things in the statement of semantics and in the completeness proof in Sect. 4, we treat $\vee, \rightarrow, \exists$ as defined notions.

A *model* is a structure $M = \langle D, I \rangle$, where for each n-argument predicate P^n, $I(P^n) \subseteq D^n$. An *assignment* v is a function $v : VAR \cup PAR \longrightarrow D$. An *x-variant* v' of v agrees with v on all arguments, save possibly x. We write v^x_o to denote the x-variant of v with $v^x_o(x) = o$. The notion of *satisfaction* of a formula φ *with* v, in symbols $M, v \models \varphi$, is defined as follows, where $t \in VAR \cup PAR$:

$$\begin{array}{lll}
M, v \models P^n(t_1, ..., t_n) & \text{iff} & \langle v(t_1), \ldots, v(t_n)\rangle \in I(P^n) \\
M, v \models t_1 = t_2 & \text{iff} & v(t_1) = v(t_2) \\
M, v \models (\lambda x\psi)t & \text{iff} & M, v^x_o \models \psi, \text{ where } o = v(t) \\
M, v \models (\lambda x\psi)\imath y\varphi & \text{iff} & \text{there is an } o \in D \text{ such that } M, v^x_o \models \psi, \text{ and} \\
 & & M, v^x_o \models \varphi[y/x], \text{ and for any } y\text{-variant } v' \text{ of } v^x_o, \\
 & & \text{if } M, v' \models \varphi, \text{ then } v'(y) = o \\
M, v \models \neg\varphi & \text{iff} & M, v \not\models \varphi, \\
M, v \models \varphi \wedge \psi & \text{iff} & M, v \models \varphi \text{ and } M, v \models \psi, \\
M, v \models \forall x\varphi & \text{iff} & M, v^x_o \models \varphi, \text{ for all } o \in D
\end{array}$$

A formula φ is *satisfiable* if there are a model M and an assignment v such that $M, v \models \varphi$. A formula is *valid* if, for all models M and assignments v, $M, v \models \varphi$. Semantically, HRL is identified with the set of valid formulas, **RL** with the set of valid sequents. A set of formulas Γ is *satisfiable* iff there is some structure M and an assignment v such that M satisfies every member of Γ with v. A sequent $\Gamma \Rightarrow \Delta$ is satisfied by a structure M with an assignment v if and only if, if for all $\varphi \in \Gamma$, $M, v \models \varphi$, then for some $\psi \in \Delta$, $M, v \models \psi$. We symbolise this by $M, v \models \Gamma \Rightarrow \Delta$. A sequent $\Gamma \Rightarrow \Delta$ is *valid* iff it is satisfied by every structure with every assignment v. In this case we write $\models \Gamma \Rightarrow \Delta$.

Note that we do not characterise DD semantically by means of interpretation function I as it is usually done (for example in [10,27])). The syntactic restriction making DD only arguments in lambda atoms allows us to define them together as a separate satisfaction clause instead. It is closer to the original Russellian treatment of descriptions and simplifies the completeness proof.

Before presenting the sequent calculus, we briefly give the Hilbert system HRL. As we noted Russell treated DD as incomplete symbols and eliminated them by means of contextual definitions. Adopting the following axiom corresponding to his definitions would be too simplistic:

$$R \qquad \psi(\imath y\varphi) \leftrightarrow \exists x(\forall y(\varphi \leftrightarrow y = x) \wedge \psi)$$

R must be restricted to atomic ψ or it is necessary to add means for marking scope distinctions. Whitehead and Russell chose the latter part, but their method is far from ideal. It is possible to avoid the problem in more elegant fashion with the help of a λ operator. In particular, we can use it to distinguish the application of the negated predicate $\neg\psi$ to $\imath y\varphi$ from negating the application of ψ to it. In the present context scoping difficulties arise only in relation to DD, and the problem is solved by restricting predication on DD to predicate abstracts. Accordingly, atomic formulas are built from predicate symbols and variables/parameters only. This is in full accordance with Russell, since the language of *Principia* contains no primitive constant and function symbols: they are introduced by contextual definitions by means of DD. We modify R to reflect the restriction that $\imath$ terms require λ abstracts:

$$R_\lambda \qquad (\lambda x\psi)\imath y\varphi \leftrightarrow \exists x(\forall y(\varphi \leftrightarrow y = x) \wedge \psi)$$

This way we avoid problems with scope while permitting complex as well as primitive predicates to be applied to DD. The axiomatic system HRL for our logic **RL** results from a standard axiomatization of pure first-order logic with identity and quantifier rules restricted to parameters by adding the axiom R_λ and β-conversion for λ but restricted again to parameters: $(\lambda x\psi)t \leftrightarrow \psi[x/t]$, where t is a parameter. The adequacy of HRL will be demonstrated below.

3 Sequent Calculus

We now formalise the Russellian logic **RL** as a sequent calculus GRL. Sequents $\Gamma \Rightarrow \Delta$ are ordered pairs of finite multisets of formulas, called the antecedent and the succedent, respectively. GRL is essentially the calculus G1c of Troelstra and Schwichtenberg [34] with rules for identity and lambda atoms: see Fig. 1.

Let us recall that formulas displayed in the schemata are active, whereas the remaining ones are parametric, or form a context. In particular, all active formulas in the premisses are called side formulas, and the one in the conclusion is the principal formula of the respective rule application. Proofs are defined in the standard way as finite trees with nodes labelled by sequents. The height of a proof $\mathcal{D}$ of $\Gamma \Rightarrow \Delta$ is defined as the number of nodes of the longest branch in $\mathcal{D}$. $\vdash_k \Gamma \Rightarrow \Delta$ means that $\Gamma \Rightarrow \Delta$ has a proof with height at most k. $\vdash$ means that there is a proof of the expression standing to its right, be it a formula (in the case of HRL) or a sequent (in the case of GRL).

We need some auxiliary results. In particular, since $(= -)$ is Leibniz' Principle restricted to atomic formulas, we must prove its unrestricted form.

Lemma 1. *1. $\vdash b_1 = b_2, \varphi[x/b_1] \Rightarrow \varphi[x/b_2]$, for any formula φ.*
2. If $\vdash_k \Gamma \Rightarrow \Delta$, then $\vdash_k \Gamma[b_1/b_2] \Rightarrow \Delta[b_1/b_2]$, where k is the height of a proof.

Proof. 1. follows by induction over the complexity of formulas, which is standard for all cases except those concerning lambda atoms with DD. We note that $\varphi^{z\,y}_{b\,c}$ is the same as $\varphi^{y\,z}_{c\,b}$, etc. We write $[(\lambda x\psi)\imath y\varphi]^{z}_{b_1}$ to denote substitutions in lambda atoms in more readable fashion. To simplify proofs applications of weakening and contraction rules to derive shared contexts are omitted from now on. Let $\mathcal{D}$ be the following deduction, where the leaves are axioms and c a fresh parameter:

$$(\imath_2 \Rightarrow)\ \frac{\varphi^{y\,z}_{c\,b_1} \Rightarrow \varphi^{y\,z}_{c\,b_1} \qquad \varphi^{y\,z}_{a\,b_1} \Rightarrow \varphi^{y\,z}_{a\,b_1} \qquad c = a \Rightarrow c = a}{[(\lambda x\psi)\imath y\varphi]^{z}_{b_1}, \varphi^{y\,z}_{a\,b_1}, \varphi^{y\,z}_{c\,b_1} \Rightarrow c = a}$$

Then we derive $\vdash b_1 = b_2, [(\lambda x\psi)\imath y\varphi]]^{z}_{b_1} \Rightarrow [(\lambda x\psi)\imath y\varphi]^{z}_{b_2}$:

$$(Cut)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma} \qquad (AX)\ \varphi \Rightarrow \varphi$$

$$(W\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow W)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(C\Rightarrow)\ \frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow C)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(\neg\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\neg)\ \frac{\varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi}$$

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

$$(\vee\Rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\vee)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi}$$

$$(\leftrightarrow\Rightarrow)\ \frac{\Gamma\Rightarrow \Delta, \varphi, \psi \qquad \varphi, \psi, \Gamma\Rightarrow \Delta}{\varphi \leftrightarrow \psi, \Gamma\Rightarrow \Delta} \qquad (\forall\Rightarrow)\ \frac{\varphi[x/b], \Gamma\Rightarrow \Delta}{\forall x\varphi, \Gamma\Rightarrow \Delta}$$

$$(\Rightarrow\leftrightarrow)\ \frac{\varphi, \Gamma\Rightarrow \Delta, \psi \qquad \psi, \Gamma \Rightarrow \Delta, \varphi}{\Gamma\Rightarrow \Delta, \varphi \leftrightarrow \psi} \qquad (\Rightarrow\forall)\ \frac{\Gamma\Rightarrow \Delta, \varphi[x/a]}{\Gamma\Rightarrow \Delta, \forall x\varphi}$$

$$(\exists\Rightarrow)\ \frac{\varphi[x/a], \Gamma\Rightarrow \Delta}{\exists x\varphi, \Gamma\Rightarrow \Delta} \qquad (\Rightarrow\exists)\ \frac{\Gamma\Rightarrow \Delta, \varphi[x/b]}{\Gamma\Rightarrow \Delta, \exists x\varphi}$$

$$(=-)\ \frac{\varphi[x/b_2], \Gamma\Rightarrow \Delta}{b_1 = b_2, \varphi[x/b_1], \Gamma\Rightarrow \Delta} \qquad (=+)\ \frac{b = b, \Gamma\Rightarrow \Delta}{\Gamma\Rightarrow \Delta}$$

$$(\lambda \Rightarrow)\ \frac{\psi[x/b], \Gamma\Rightarrow \Delta}{(\lambda x\psi)b, \Gamma\Rightarrow \Delta} \qquad (\Rightarrow \lambda)\ \frac{\Gamma\Rightarrow \Delta, \psi[x/b]}{\Gamma\Rightarrow \Delta, (\lambda x\psi)b}$$

$$(\imath_1 \Rightarrow)\ \frac{\varphi[y/a], \psi[x/a], \Gamma\Rightarrow \Delta}{(\lambda x\psi)\imath y\varphi, \Gamma\Rightarrow \Delta}$$

$$(\imath_2 \Rightarrow)\ \frac{\Gamma\Rightarrow \Delta, \varphi[y/b_1] \qquad \Gamma\Rightarrow \Delta, \varphi[y/b_2] \qquad b_1 = b_2, \Gamma \Rightarrow \Delta}{(\lambda x\psi)\imath y\varphi, \Gamma \Rightarrow \Delta}$$

$$(\Rightarrow \imath)\ \frac{\Gamma\Rightarrow \Delta, \varphi[y/b] \qquad \Gamma\Rightarrow \Delta, \psi[x/b] \qquad \varphi[y/a], \Gamma \Rightarrow \Delta, a = b}{\Gamma \Rightarrow \Delta, (\lambda x\psi)\imath y\varphi}$$

where a is a fresh parameter (Eigenvariable), not present in Γ, Δ and φ, whereas b, b_1, b_2 are arbitrary parameters. φ in $(=-)$ is an atomic formula.

Fig. 1. Calculus GRL

$$(C\Rightarrow)\,\dfrac{(\imath_1\Rightarrow)\,\dfrac{(\Rightarrow\imath)\,\dfrac{b_1=b_2,\varphi^{y\,z}_{a\,b_1}\Rightarrow\varphi^{y\,z}_{a\,b_2}\qquad b_1=b_2,\psi^{x\,z}_{a\,b_1}\Rightarrow\psi^{x\,z}_{a\,b_2}\qquad\mathcal{D}}{b_1=b_2,\varphi^{y\,z}_{a\,b_1},\psi^{x\,z}_{a\,b_1},[(\lambda x\psi)\imath y\varphi]^z_{b_1}\Rightarrow[(\lambda x\psi)\imath y\varphi]^z_{b_2}}}{b_1=b_2,[(\lambda x\psi)\imath y\varphi]^z_{b_1},[(\lambda x\psi)\imath y\varphi]^z_{b_1}\Rightarrow[(\lambda x\psi)\imath y\varphi]^z_{b_2}}}{b_1=b_2,[(\lambda x\psi)\imath y\varphi]^z_{b_1}\Rightarrow[(\lambda x\psi)\imath y\varphi]^z_{b_2}}$$

The two left leaves are provable by the induction hypothesis (if b_1, b_2 are not present in ψ or φ, we have an axiomatic sequent).

The proof of 2 is by a standard induction on the height of proofs; the rules for lambda atoms with DD are treated similarly to the rules for quantifiers. □

Let us now show that the Russellian axiom R_λ is provable in GRL. We will provide proofs for two sequents corresponding to two implications. Let $\mathcal{D}$ be:

$$(\imath_2\Rightarrow)\,\dfrac{\varphi^y_a\Rightarrow\varphi^y_a\qquad\varphi^y_{a_1}\Rightarrow\varphi^y_{a_1}\qquad a_1=a\Rightarrow a_1=a}{(\lambda x\psi)\imath y\varphi,\varphi^y_a,\varphi^y_{a_1}\Rightarrow a_1=a}$$

The following establishes one half of R_λ:

$$(C\Rightarrow)\,\dfrac{(\imath_1\Rightarrow)\,\dfrac{(\Rightarrow\exists)\,\dfrac{(\Rightarrow\wedge)\,\dfrac{(\Rightarrow\forall)\,\dfrac{(\Rightarrow\leftrightarrow)\,\dfrac{\mathcal{D}\qquad\varphi^y_a,a_1=a\Rightarrow\varphi^y_{a_1}}{(\lambda x\psi)\imath y\varphi,\varphi^y_a\Rightarrow\varphi^y_{a_1}\leftrightarrow a_1=a}}{(\lambda x\psi)\imath y\varphi,\varphi^y_a\Rightarrow\forall y(\varphi\leftrightarrow y=a)}\qquad\psi^x_a\Rightarrow\psi^x_a}{(\lambda x\psi)\imath y\varphi,\psi^x_a,\varphi^y_a\Rightarrow\forall y(\varphi\leftrightarrow y=a)\wedge\psi^x_a}}{(\lambda x\psi)\imath y\varphi,\psi^x_a,\varphi^y_a\Rightarrow\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)}}{(\lambda x\psi)\imath y\varphi,(\lambda x\psi)\imath y\varphi\Rightarrow\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)}}{(\lambda x\psi)\imath y\varphi\Rightarrow\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)}$$

where the only nonaxiomatic sequent is provable by lemma 1.1. Next, where $\mathcal{D}$ is:

$$(\forall\Rightarrow)\,\dfrac{(\leftrightarrow\Rightarrow)\,\dfrac{\varphi^y_b\Rightarrow\varphi^y_b\qquad b=a\Rightarrow b=a}{\varphi^y_b\leftrightarrow b=a,\varphi^y_b\Rightarrow b=a}}{\forall y(\varphi\leftrightarrow y=a),\varphi^y_b\Rightarrow b=a}$$

the following establishes the other half of R_λ:

$$(\exists\Rightarrow)\,\dfrac{(\wedge\Rightarrow)\,\dfrac{(\Rightarrow\imath)\,\dfrac{\psi^x_a\Rightarrow\psi^x_a\qquad(\forall\Rightarrow)\,\dfrac{(\leftrightarrow\Rightarrow)\,\dfrac{(=+)\,\dfrac{a=a\Rightarrow a=a}{\Rightarrow a=a}\qquad\varphi^y_a\Rightarrow\varphi^y_a}{\varphi^y_a\leftrightarrow a=a\Rightarrow\varphi^y_a}}{\forall y(\varphi\leftrightarrow y=a)\Rightarrow\varphi^y_a}\qquad\mathcal{D}}{\forall y(\varphi\leftrightarrow y=a),\psi^x_a\Rightarrow(\lambda x\psi)\imath y\varphi}}{\forall y(\varphi\leftrightarrow y=a)\wedge\psi^x_a\Rightarrow(\lambda x\psi)\imath y\varphi}}{\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)\Rightarrow(\lambda x\psi)\imath y\varphi}$$

Conversely, the three rules for lambda atoms with DD are derivable in G1 with R_λ added in the form of two axiomatic sequents. To derive $(\imath_1\Rightarrow)$, let $R^{\Rightarrow}_{\lambda}$ be $(\lambda x\psi)\imath y\varphi\Rightarrow\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)$:

$$(Cut)\,\dfrac{R_\lambda^{\Rightarrow} \qquad (\exists\Rightarrow)\,\dfrac{(\wedge\Rightarrow)\,\dfrac{(\forall\Rightarrow)\,\dfrac{(\leftrightarrow\Rightarrow)\,\dfrac{(=+)\,\dfrac{a=a\Rightarrow a=a}{\Rightarrow a=a} \qquad \varphi^y_a,\psi^x_a,\varGamma\Rightarrow\varDelta}{\varphi^y_a\leftrightarrow a=a,\psi^x_a,\varGamma\Rightarrow\varDelta}}{\forall y(\varphi\leftrightarrow y=a),\psi^x_a,\varGamma\Rightarrow\varDelta}}{\forall y(\varphi\leftrightarrow y=a)\wedge\psi^x_a,\varGamma\Rightarrow\varDelta}}{\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi),\varGamma\Rightarrow\varDelta}}{(\lambda x\psi)\imath y\varphi,\varGamma\Rightarrow\varDelta}$$

To derive $(\imath_2\Rightarrow)$, use (Cut) with $(\lambda x\psi)\imath y\varphi\Rightarrow\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)$ and:

$$(\exists\Rightarrow)\,\dfrac{(\wedge\Rightarrow)\,\dfrac{(C\Rightarrow)\,\dfrac{(\forall\Rightarrow)\,\dfrac{(\leftrightarrow\Rightarrow)\,\dfrac{\varGamma\Rightarrow\varDelta,\varphi^y_{b_1} \qquad (\leftrightarrow\Rightarrow)\,\dfrac{\varGamma\Rightarrow\varDelta,\varphi^y_{b_2} \qquad (=-)\,\dfrac{b_1=b_2,\varGamma\Rightarrow\varDelta}{b_1=a,b_2=a,\varGamma\Rightarrow\varDelta}}{b_1=a,\varphi^y_{b_2}\leftrightarrow b_2=a,\varGamma\Rightarrow\varDelta}}{\varphi^y_{b_1}\leftrightarrow b_1=a,\varphi^y_{b_2}\leftrightarrow b_2=a,\varGamma\Rightarrow\varDelta}}{\forall y(\varphi\leftrightarrow y=a),\forall y(\varphi\leftrightarrow y=a),\varGamma\Rightarrow\varDelta}}{\forall y(\varphi\leftrightarrow y=a),\psi^x_a,\varGamma\Rightarrow\varDelta}}{\forall y(\varphi\leftrightarrow y=a)\wedge\psi^x_a,\varGamma\Rightarrow\varDelta}}{\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi),\varGamma\Rightarrow\varDelta}$$

The following derives $(\Rightarrow\imath)$:

$$(\Rightarrow\exists)\,\dfrac{(\Rightarrow\wedge)\,\dfrac{(\Rightarrow\forall)\,\dfrac{(\Rightarrow\leftrightarrow)\,\dfrac{\varphi^y_a,\varGamma\Rightarrow\varDelta,a=b \qquad (Cut)\,\dfrac{\varGamma\Rightarrow\varDelta,\varphi^y_b \qquad a=b,\varphi^y_b\Rightarrow\varphi^y_a}{a=b,\varGamma\Rightarrow\varDelta,\varphi^y_a}}{\varGamma\Rightarrow\varDelta,\varphi^y_a\leftrightarrow a=b}}{\varGamma\Rightarrow\varDelta,\forall y(\varphi\leftrightarrow y=b)} \qquad \varGamma\Rightarrow\varDelta,\psi^x_b}{\varGamma\Rightarrow\varDelta,\forall y(\varphi\leftrightarrow y=b)\wedge\psi^x_b}}{\varGamma\Rightarrow\varDelta,\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)}$$

where the right premiss of (Cut) is provable by lemma 1.1, and the conclusion of the rule follows by (Cut) with $\exists x(\forall y(\varphi\leftrightarrow y=x)\wedge\psi)\Rightarrow(\lambda x\psi)\imath y\varphi$.

Since the proofs of the interderivability of the axiom of λ conversion and $(\lambda\Rightarrow),(\Rightarrow\lambda)$ are trivial we are done and conclude with:

Theorem 1. $\vdash_{HRL}\varphi$ *iff* $\vdash_{GRL}\Rightarrow\varphi$

4 Cut Elimination

We will show that (Cut) is eliminable from every proof in GRL using the general strategy of cut elimination proofs applied originally for hypersequent calculi in Metcalfe, Olivetti and Gabbay [24], which works well also in the context of standard sequent calculi (see [15]). Such a proof has a particularly simple structure and allows us to avoid many complexities inherent in other methods of proving cut elimination. In particular, we avoid well known problems with contraction, since two auxiliary lemmata deal with this problem in advance. We assume that all proofs are regular in the sense that every parameter a which is fresh by the side condition of the respective rule must be fresh in the entire proof, not only on the branch where the application of this rule takes place. There is no loss of generality since every proof may be systematically transformed into a regular one by lemma 1.2. The following notions are crucial for the proof:

1. The cut-degree is the complexity of the cut-formula φ, i.e. the number of logical constants (connectives, quantifiers and operators) occurring in φ; it is denoted by $d\varphi$.
2. The proof-degree ($d\mathcal{D}$) is the maximal cut-degree in $\mathcal{D}$.

The proof of the cut elimination theorem is based on two lemmata which successively make a reduction: first of the height of the right, and then of the height of the left premiss of cut. φ^k, Γ^k denote $k > 0$ occurrences of φ, Γ, respectively.

Lemma 2 (Right reduction). *Let $\mathcal{D}_1 \vdash \Gamma \Rightarrow \Delta, \varphi$ and $\mathcal{D}_2 \vdash \varphi^k, \Pi \Rightarrow \Sigma$ with $d\mathcal{D}_1, d\mathcal{D}_2 < d\varphi$, and φ principal in $\Gamma \Rightarrow \Delta, \varphi$, then we can construct a proof $\mathcal{D}$ such that $\mathcal{D} \vdash \Gamma^k, \Pi \Rightarrow \Delta^k, \Sigma$ and $d\mathcal{D} < d\varphi$.*

Proof. By induction on the height of $\mathcal{D}_2$. The basis is trivial, since $\Gamma \Rightarrow \Delta, \varphi$ is identical with $\Gamma^k, \Pi \Rightarrow \Delta^k, \Sigma$. The induction step requires examination of all cases of possible derivations of $\varphi^k, \Pi \Rightarrow \Sigma$, and the role of the cut-formula in the transition. In cases where all occurrences of φ are parametric we simply apply the induction hypothesis to the premisses of $\varphi^k, \Pi \Rightarrow \Sigma$ and then apply the respective rule – it is essentially due to the context independence of almost all rules and the regularity of proofs, which together prevent violation of side conditions on eigenvariables. If one of the occurrences of φ in the premiss(es) is a side formula of the last rule we must additionally apply weakening to restore the missing formula before the application of the relevant rule.

In cases where one occurrence of φ in $\varphi^k, \Pi \Rightarrow \Sigma$ is principal we make use of the fact that φ in the left premiss is also principal; for the cases of contraction and weakening this is trivial. We consider the cases of lambda atoms with DD. Hence $\mathcal{D}_1$ finishes with:

$$\frac{\Gamma \Rightarrow \Delta, \varphi[y/b] \qquad \Gamma \Rightarrow \Delta, \psi[x/b] \qquad \varphi[y/a], \Gamma \Rightarrow \Delta, a = b}{\Gamma \Rightarrow \Delta, (\lambda x \psi)\imath y \varphi}$$

and $\mathcal{D}_2$ finishes with:

$$\frac{\varphi[y/a'], \psi[x/a'], (\lambda x \psi)\imath y \varphi^{k-1}, \Pi \Rightarrow \Sigma}{(\lambda x \psi)\imath y \varphi^{k}, \Pi \Rightarrow \Sigma}$$

or

$$\frac{(\lambda x \psi)\imath y \varphi^{k-1}, \Pi \Rightarrow \Sigma, \varphi[y/b_1] \qquad (\lambda x \psi)\imath y \varphi^{k-1}, \Pi \Rightarrow \Sigma, \varphi[y/b_2] \qquad b_1 = b_2, (\lambda x \psi)\imath y \varphi^{k-1}, \Pi \Rightarrow \Sigma}{(\lambda x \psi)\imath y \varphi^{k}, \Pi \Rightarrow \Sigma}$$

In the first case, by the induction hypothesis and lemma 1.2 we obtain $\varphi[y/b], \psi[x/b], \Gamma^{k-1}, \Pi \Rightarrow \Delta^{k-1}, \Sigma$ and by two cuts with the leftmost and central premiss of ($\Rightarrow \imath$) in $\mathcal{D}_1$ we obtain $\Gamma^{k+1}, \Pi \Rightarrow \Delta^{k+1}, \Sigma$, which by contraction yields the result.

In the second case note first that by lemma 1.2 from the rightmost premiss of $(\Rightarrow \imath)$ in $\mathcal{D}_1$ we obtain

a. $\varphi[y/b_1], \Gamma \Rightarrow \Delta, b_1 = b$ and
b. $\varphi[y/b_2], \Gamma \Rightarrow \Delta, b_2 = b$.

Again by the induction hypothesis from the three premisses we get:

1. $\Gamma^{k-1}, \Pi \Rightarrow \Delta^{k-1}, \Sigma, \varphi[y/b_1]$
2. $\Gamma^{k-1}, \Pi \Rightarrow \Delta^{k-1}, \Sigma, \varphi[y/b_2]$
3. $b_1 = b_2, \Gamma^{k-1}, \Pi \Rightarrow \Delta^{k-1}, \Sigma$

We proceed as follows with a series of the applications of cut, followed by contractions, using the provable sequent $b_1 = b, b_2 = b \Rightarrow b_1 = b_2$:

$$\dfrac{\dfrac{\dfrac{2 \qquad b}{\Gamma^k, \Pi \Rightarrow \Delta^k, \Sigma, b_2 = b} \qquad \dfrac{\dfrac{1 \qquad a}{\Gamma^k, \Pi \Rightarrow \Delta^k, \Sigma, b_1 = b} \qquad \dfrac{b_1 = b, b_2 = b \Rightarrow b_1 = b_2 \qquad 3}{b_1 = b, b_2 = b, \Gamma^{k-1}, \Pi \Rightarrow \Delta^{k-1}, \Sigma}}{b_2 = b, \Gamma^{2k-1}, \Pi^2 \Rightarrow \Delta^{2k-1}, \Sigma^2}}{\Gamma^{3k-1}, \Pi^3 \Rightarrow \Delta^{3k-1}, \Sigma^3}}{\Gamma^k, \Pi \Rightarrow \Delta^k, \Sigma}$$

□

Lemma 3 (Left reduction). *Let $\mathcal{D}_1 \vdash \Gamma \Rightarrow \Delta, \varphi^k$ and $\mathcal{D}_2 \vdash \varphi, \Pi \Rightarrow \Sigma$ with $d\mathcal{D}_1, d\mathcal{D}_2 < d\varphi$, then we can construct a proof $\mathcal{D}$ such that $\mathcal{D} \vdash \Gamma, \Pi^k \Rightarrow \Delta, \Sigma^k$ and $d\mathcal{D} < d\varphi$.*

Proof. By induction on the height of $\mathcal{D}_1$ but with some important differences to the proof of the right reduction lemma. First note that we do not require φ to be principal in $\varphi, \Pi \Rightarrow \Sigma$, so it includes the case where φ is atomic. In all these cases we just apply the induction hypothesis. This guarantees that even if an atomic cut formula was introduced in the right premiss by $(= -)$ the reduction of the height is achieved only on the left premiss, and we always obtain the expected result. Now, in cases where one occurrence of φ in $\Gamma \Rightarrow \Delta, \varphi^k$ is principal, we first apply the induction hypothesis to eliminate all other $k-1$ occurrences of φ in the premisses and then we apply the respective rule. Since the only new occurrence of φ is principal, we can make use of the right reduction lemma again and obtain the result, possibly after some applications of structural rules. □

Now we are ready to prove the cut elimination theorem:

Theorem 2. *Every proof in GRL can be transformed into cut-free proof.*

Proof. By double induction: primary on $d\mathcal{D}$ and subsidiary on the number of maximal cuts (in the basis and in the inductive step of the primary induction). We always take the topmost maximal cut and apply lemma 3 to it. By successive repetition of this procedure we reduce either the degree of a proof or the number of cuts in it until we obtain a cut-free proof. □

5 Adequacy

In this section, we'll make use of the fact that for every set there is a corresponding multiset, so if $\mathit{\Gamma}$, $\mathit{\Delta}$ are sets of formulas, we may write $\mathit{\Gamma} \Rightarrow \mathit{\Delta}$. We recall that we treat $\vee, \rightarrow, \exists$ as defined notions. For the completeness proof we assume that a denumerable set of individual constants may be added to the language. I assigns objects in the domain D of the model $\langle D, I\rangle$ to these constants. For brevity we introduce the notation I_v, where if t is a variable or parameter, $I_v(t) = v(t)$ and where t is a constant, $I_v(t) = I(t)$.

Recall the distinction between terms and pseudo-terms, the former variables and parameters and now also constants, the latter iota terms. In the following lemma, t denotes a variable, parameter or constant, not a DD, hence the proof is standard, with the case of lambda atoms similar to the case of quantifiers. In the rest of this section, too, t will refer to terms only. In particular, there is no need to consider pseudo-terms in the Lindenbaum-Henkin construction (theorem 4), because in substitution in the formulas concerned only terms can be used. Pseudo-terms are treated, just as they are in the semantics, as occurring in lambda atoms, and thus like the logical constants by the consideration of the consistent addition of formulas to a set in the construction of its maximally consistent extension.

Lemma 4 (The Substitution Lemma.). *$M, v \models \varphi^x_t$ iff $M, v^x_{I_v(t)} \models \varphi$, if t is free for x in φ.*

Proof. See e.g. [7, 133f] and adjust. □

Next, the soundness of GRL.

Theorem 3 (Soundness of GRL). *If $\vdash \mathit{\Gamma} \Rightarrow \mathit{\Delta}$, then $\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}$*

Proof. By induction on the height of the proof. Since it is well-known that the rules of G1 are validity preserving, and it is obvious for both lambda rules, we show this property only for $(\imath_2 \Rightarrow)$ and $(\Rightarrow \imath)$, leaving $(\imath_1 \Rightarrow)$ as an exercise.

$(\imath_2 \Rightarrow)$. Suppose (1) $\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}, \varphi^y_{b_1}$, (2) $\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}, \varphi^y_{b_2}$, (3) $\models b_1 = b_2, \mathit{\Gamma} \Rightarrow \mathit{\Delta}$, and $\not\models (\lambda x\psi)\imath y\varphi, \mathit{\Gamma} \Rightarrow \mathit{\Delta}$. By the last, there are a structure $M = \langle D, I\rangle$ and assignment v, such that $M, v \models (\lambda x\psi)\imath y\varphi$, for all $\gamma \in \mathit{\Gamma}$, $M, v \models \gamma$ and for all $\delta \in \mathit{\Delta}$, $M, v \not\models \delta$. Thus by (1), (2) and (3): (4) $M, v \models \varphi^y_{b_1}$, (5) $M, v \models \varphi^y_{b_2}$ and (6) $M, v \not\models b_1 = b_2$. And there is an $o \in D$ such that $M, v^x_o \models \psi$, and $M, v^x_o \models \varphi[y/x]$, and (7) for any y-variant v' of v^x_o, if $M, v' \models \varphi$, then $v'(y) = o$. By the conventions on the use of free and bound variables in sequents, x is not free in $\varphi^y_{b_1}$ or $\varphi^y_{b_2}$, so v and v^x_o agree on them, and so by (4) and (5) $M, v^x_o \models \varphi^y_{b_1}$ and $M, v^x_o \models \varphi^y_{b_2}$. By the substitution lemma, $M, v^{x\,y}_{o\,I_v(b_1)} \models \varphi$ and $M, v^{x\,y}_{o\,I_v(b_2)} \models \varphi$. So the y-variants v' and v'' of v^x_o that assign $I_{v^x_o}(b_1)$ and $I_{v^x_o}(b_2)$ to y satisfy φ with M, so by (7) $I_{v'}(b_1) = I_{v''}(b_2) = o$. But v' and v'' differ from v only in what they assign to x and y, and by (6) $I_v(b_1) \neq I_v(b_2)$. Contradiction.

$(\Rightarrow \imath)$. Suppose (1) $\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}, \varphi^y_b$, (2) $\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}, \psi^x_b$, (3) $\models \varphi^y_a, \mathit{\Gamma} \Rightarrow \mathit{\Delta}, a = b$, but $\not\models \mathit{\Gamma} \Rightarrow \mathit{\Delta}, (\lambda x\psi)\imath y\varphi$, a not free in any formulas in $\mathit{\Gamma}$ and $\mathit{\Delta}$ nor in φ. Then there are a structure $M = \langle D, I\rangle$ and assignment v such that for all $\gamma \in \Gamma$, $M, v \models \gamma$, for all $\delta \in \Delta$, $M, v \not\models \delta$ and (4) $M, v \not\models (\lambda x\psi)\imath y\varphi$. So by (1), $M, v \models \varphi^y_b$, by (2), $M, v \models \psi^x_b$, and by (4), it is not the case that there is an $o \in D$ such that $M, v^x_o \models \psi$, and $M, v^x_o \models \varphi^y_x$, and for any y-variant v' of v^x_o, if $M, v' \models \varphi$, then $v'(y) = o$, i.e. for every $o \in D$, either $M, v^x_o \not\models \psi$, or $M, v^x_o \not\models \varphi^y_x$, or for some y-variant v' of v^x_o, $M, v' \models \varphi$ and $v'(y) \neq o$. Consider $I_v(b)$. We have either (5) $M, v^x_{I_v(b)} \not\models \psi$, or (6) $M, v^x_{I_v(b)} \not\models \varphi^y_x$, or (7) for some y-variant v' of $v^x_{I_v(b)}$, $M, v' \models \varphi$ and $v'(y) \neq I_v(b)$. By the substitution lemma from (5) and (6) we have $M, v \not\models \psi^x_b$ and $M, v \not\models \varphi^{xy}_{yb}$, and as φ^{yx}_{xb} is the same as φ^y_b, this contradicts consequences of (1) and (2). By conventions on the use of free and bound variables in sequents, x and y are not free in any of their formulas, so $v^x_{I_v(b)}$ agrees with v on all formulas in Γ, Δ, so for all $\gamma \in \Gamma$, $M, v^x_{I_v(b)} \models \gamma$, and for all $\delta \in \Delta$, $M, v^x_{I_v(b)} \not\models \delta$. So by (3), if $M, v^x_{I_v(b)} \models \varphi^y_a$, then $M, v^x_{I_v(b)} \models a = b$. By the substitution lemma and the semantic clause for identity, if $M, v^{x\,y}_{I_v(b)I_v(a)} \models \varphi$, then $I_v(a) = I_v(b)$. Now evidently $v^{x\,y}_{I_v(b)I_v(a)}(y) = I_v(a)$, so $v^{x\,y}_{I_v(b)I_v(a)}(y) = I_v(b)$. But $v^{x\,y}_{I_v(b)I_v(a)}$ is a y-variant of $v^x_{I_v(b)}$, and the reasoning holds for any such y-variant, contradicting (7). □

Let $\bot$ represent an arbitrary contradiction. A set of formulas Γ is *inconsistent* iff $\Gamma \vdash \bot$. Γ is *consistent* iff it is not inconsistent. A set of formulas Γ is *maximal* iff for any formula A, either $A \in \Gamma$ or $\neg A \in \Gamma$. A set of formulas Γ is *deductively closed* iff, if $\Gamma \vdash A$, then $A \in \Gamma$. We state without proof this standard result:

Lemma 5. *Any maximally consistent set is deductively closed.*

Extend $\mathcal{L}$ to a language $\mathcal{L}^+$ by adding countably new constants ordered by a list $\mathcal{C} = c_1, c_2 \ldots$. We will say that such a constant occurs *parametrically* if its occurrence satisfies the restrictions imposed on parameters in $(\Rightarrow \forall)$ and $(\imath_1 \Rightarrow)$.

Theorem 4. *Any consistent set of formulas Δ can be extended to a maximally consistent set Δ^+ such that:*
(a) for any formula φ and variable x, if $\neg\forall x\varphi \in \Delta^+$, then for some constant c, $\varphi^x_c \notin \Delta^+$;
(b) for any formulas φ, ψ and variables x, y, if $(\lambda x\psi)\imath y\varphi \in \Delta^+$, then for some constant c, $\varphi^y_c, \psi^x_c \in \Delta^+$ and for all terms t, if $\varphi^y_t \in \Delta^+$, then $t = c \in \Delta^+$;
(c) for any formulas φ, ψ and variables x, y, if $\neg(\lambda x\psi)\imath y\varphi \in \Delta^+$, then for all terms t, either $\varphi^y_t \notin \Delta^+$, or for some constant c, $\varphi^y_c \in \Delta^+$ and $c = t \notin \Delta^+$, or $\psi^x_t \notin \Delta^+$.

Proof. Extend Δ by following an enumeration $\phi_1, \phi_2 \ldots$ of the formulas of $\mathcal{L}^+$ on which every formula occurs infinitely many times as follows:

$\Delta_0 = \Delta$
If Δ_n, ϕ_n is inconsistent, then
$\Delta_{n+1} = \Delta_n$.
If Δ_n, ϕ_n is consistent, then:

(i) If ϕ_n has neither the form $\neg\forall x\varphi$ nor $(\lambda x\psi)\imath y\varphi$ nor $\neg(\lambda x\psi)\imath y\varphi$, then
$\Delta_{n+1} = \Delta_n, \phi_n$.
(ii) If ϕ_n has the form $\neg\forall x\varphi$, then
$\Delta_{n+1} = \Delta_n, \neg\forall x\varphi, \neg\varphi^x_c$
where c is the first constant of $\mathcal{C}$ that does not occur in Δ_n or ϕ_n.
(iii) If ϕ_n has the form $(\lambda x\psi)\imath y\varphi$, then
$\Delta_{n+1} = \Delta_n, (\lambda x\psi)\imath y\varphi, \varphi^y_c, \psi^x_c$
where c is the first constant of $\mathcal{C}$ that does not occur in Δ_n or ϕ_n.
(iv) If ϕ_n has the form $\neg(\lambda x\psi)\imath y\varphi$, then
$\Delta_{n+1} = \Delta_n, \neg(\lambda x\psi)\imath y\varphi, \Sigma_n$

where Σ_n is constructed in the following way. Take a sequence of formulas $\sigma_1, \sigma_2 \ldots$ of the form $\varphi^y_t \rightarrow (\psi^x_t \rightarrow \neg(\varphi^y_c \rightarrow c = t))$, where t is a term in Δ_n, ϕ_n, and c is a constant of $\mathcal{C}$ not in Δ_n, ϕ_n or any previous formulas in the sequence. Let $\mathcal{T} = t_1, t_2, \ldots$ be an enumeration of all terms occurring in Δ_n, ϕ_n. In case Δ_0 contains infinitely many formulas, it must be ensured that $\mathcal{C}$ is not depleted of constants needed later. So pick constants from $\mathcal{C}$ by a method that ensures some constants are always left over for later use. The following will do. Let σ_1 be $\varphi^y_{t_1} \rightarrow (\psi^x_{t_1} \rightarrow \neg(\varphi^y_{c_1} \rightarrow c_1 = t_1))$, where t_1 is the first term of $\mathcal{T}$ and c_1 is the first constant of $\mathcal{C}$ not in Δ_n, ϕ_n; let σ_2 be $\varphi^y_{t_2} \rightarrow (\psi^x_{t_2} \rightarrow \neg(\varphi^y_{c_2} \rightarrow c_2 = t_2))$, where t_2 is the second term on $\mathcal{T}$ and c_2 is the $2^2 = 4$th constant of $\mathcal{C}$ not in $\Delta_n, \phi_n, \sigma_1$. In general, let σ_n be $\varphi^y_{t_n} \rightarrow (\psi^x_{t_n} \rightarrow \neg(\varphi^y_{c_n} \rightarrow c_n = t_n))$, where t_n is the nth term of $\mathcal{T}$ and c_n is the 2^nth constant of $\mathcal{C}$ not in Δ_n, ϕ_n nor any σ_i, $i < n$. The entire collection of σ_is is Σ_n.

Δ_{n+1} is consistent if Δ_n, ϕ_n is:

Case (i). Trivial.

Case (ii). Suppose $\Delta_{n+1} = \Delta_n, \neg\forall x\varphi, \neg\varphi^x_c$ is inconsistent. Then for some finite $\Delta'_n \subseteq \Delta_n$: $\vdash \Delta'_n, \neg\forall x\varphi, \neg\varphi^x_c \Rightarrow \bot$. Hence $\vdash \Delta'_n, \neg\forall x\varphi \Rightarrow \varphi^x_c$ by deductive properties of negation. c does not occur in any formula in Δ'_n nor in $\neg\forall x\varphi$, so it occurs parametrically, and so by $(\Rightarrow \forall)$, $\vdash \Delta'_n, \neg\forall x\varphi \Rightarrow \forall x\varphi$. Hence $\vdash \Delta'_n \Rightarrow \forall x\varphi$, again by deductive properties of negation. But then $\Delta'_n, \neg\forall x\varphi$ is inconsistent, and hence so is $\Delta_n, \neg\forall x\varphi$.

Case (iii). Suppose $\Delta_{n+1} = \Delta_n, (\lambda x\psi)\imath y\varphi, \varphi^y_c, \psi^x_c$ is inconsistent. Then for some finite $\Delta'_n \subseteq \Delta_n$, $\vdash \Delta'_n, (\lambda x\psi)\imath y\varphi, \varphi^y_c, \psi^x_c \Rightarrow \bot$. c does not occur in $\Delta'_n, (\lambda x\psi)\imath y\varphi$, so it occurs parametrically, and hence by $(\imath_1 \Rightarrow)$, $\vdash \Delta'_n, (\lambda x\psi)\imath y\varphi \Rightarrow \bot$, that is to say $\Delta'_n, (\lambda x\psi)\imath y\varphi$ is inconsistent, and so is $\Delta_n, (\lambda x\psi)\imath y\varphi$.

Case (iv). Suppose $\Delta_{n+1} = \Delta_n, \neg(\lambda x\psi)\imath y\varphi, \Sigma_n$ is inconsistent. Then for some finite $\Delta'_n \subseteq \Delta_n$ and a finite $\{\sigma_j \ldots \sigma_k\} \subseteq \Sigma_n$, $\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \ldots \sigma_k \Rightarrow \bot$. Let σ_k be $\varphi^y_{t_k} \rightarrow (\psi^x_{t_k} \rightarrow \neg(\varphi^y_{c_k} \rightarrow c_k = t_k))$. Then by the deductive properties of implication and negation:

$\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \ldots \sigma_{k-1} \Rightarrow \varphi^y_{t_k}$
$\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \ldots \sigma_{k-1} \Rightarrow \psi^x_{t_k}$
$\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \ldots \sigma_{k-1}, \varphi^y_{c_k} \Rightarrow c_k = t_k$

c_k was chosen so as not to occur in any previous σ_i, $i < k$, nor in Δ_n, ϕ_n. Hence it occurs parametrically and the conditions for $(\Rightarrow \imath)$ are fulfilled. Thus $\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \dots \sigma_{k-1} \Rightarrow (\lambda x\psi)\imath y\varphi$. But $\vdash \Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \dots \sigma_{k-1} \Rightarrow \neg(\lambda x\psi)\imath y\varphi$. So $\Delta'_n, \neg(\lambda x\psi)\imath y\varphi, \sigma_j \dots \sigma_{k-1}$ is inconsistent. Repeat this process from σ_{k-1} all the way down to σ_j, showing that $\Delta'_n, \neg(\lambda x\psi)\imath y\varphi$ is inconsistent. Hence so is $\Delta_n, \neg(\lambda x\psi)\imath y\varphi$.

Let Δ^+ be the union of all Δ_i. Δ^+ is maximal, for if neither φ not $\neg\varphi$ are in Δ^+, then there is a $\Delta_k \subseteq \Delta^+$ such that $\Delta_k, \varphi \vdash \bot$ and $\Delta_k, \neg\varphi \vdash \bot$, but then Δ_k is inconsistent, contradicting the method of construction of Δ_k. Δ^+ is consistent, because otherwise some Δ_i would have to be inconsistent, but they are not.

Δ^+ satisfies (a) by construction.

To see that it satisfies (b), suppose $(\lambda x\psi)\imath y\varphi \in \Delta^+$. Then there is a $\Delta_{n+1} = \Delta_n, (\lambda x\psi)\imath y\varphi, \varphi^y_c, \psi^x_c$, and so $\varphi^y_c, \psi^x_c \in \Delta^+$. Suppose $\varphi^y_t \in \Delta^+$. Then there is a $\Delta' \subseteq \Delta^+$ such that $\vdash \Delta' \Rightarrow \varphi^y_c$, $\vdash \Delta' \Rightarrow \varphi^y_t$ and by properties of identity $\vdash t = c \Rightarrow t = c$. But then by $(\imath_2 \Rightarrow)$, $\vdash \Delta', (\lambda x\psi)\imath y\varphi \Rightarrow t = c$, hence $t = c \in \Delta^+$ by the deductive closure of Δ^+.

To see that it satisfies (c), suppose $\neg(\lambda x\psi)\imath y\varphi \in \Delta^+$, but for some term t, $\varphi^y_t \in \Delta^+$, (1) for all constants c, if $\varphi^y_c \in \Delta^+$, then $c = t \in \Delta^+$, and $\psi^x_t \in \Delta^+$. As every formula occurs infinitely many times on the enumeration of formulas of $\mathcal{L}^+$, there is a Δ_n that contains φ^y_t and ψ^x_t and $\Delta_{n+1} = \Delta_n, \neg(\lambda x\psi)\imath y\varphi, \Sigma_n$. Thus $\varphi^y_t \to (\psi^x_t \to \neg(\varphi^y_b \to b = t)) \in \Sigma_n$, for some constant b of $\mathcal{C}$. Consequently, this formula is in Δ^+, too. By the deductive properties of implication and negation and the deductive closure and consistency of Δ^+, (2) $\varphi^y_b \in \Delta^+$ and $b = t \notin \Delta^+$. But by (1) and (2), $b = t \in \Delta^+$. Contradiction.

This completes the proof of Theorem 4. □

Theorem 5. *If Δ is a consistent set of formulas, then Δ is satisfiable.*

Proof. Extend Δ to a maximally consistent set Δ^+ as per Theorem 4. We construct a structure $M = \langle D, I\rangle$ and function $v \colon VAR \cup PAR \to D$ from Δ^+ which will satisfy Δ. D is the set of equivalence classes of terms under identities $t_1 = t_2 \in \Delta^+$. Denote the equivalence class to which t belongs by $[t]$. For all predicate letters P, $\langle [t_1], ..., [t_n]\rangle \in I(P^n)$ iff $P^n(t_1, ..., t_n) \in \Delta^+$. For all variables $v(x) = [x]$, and for all parameters $v(a) = [a]$. In these latter cases $I_v = v$, and for all new constants of $\mathcal{C}$, $I_v(c) = [c]$. We'll show by induction over the number of logical constants (connectives, quantifiers, $\imath$ and λ symbols) in formula φ that $M, v \models \varphi$ if and only if $\varphi \in \Delta^+$.

Suppose φ is an atomic formula. (a) φ is $P^n(t_1, ..., t_n)$. Then $M, v \models P^n(t_1, ..., t_n)$ iff $\langle I_v(t_1), ..., I_v(t_n)\rangle \in I(P^n)$, iff $\langle [t_1] \dots [t_n]\rangle \in I(P^n)$, iff $P^n(t_1, ..., t_n) \in \Delta^+$. (b) φ is $t_1 = t_2$. Then $M, v \models t_1 = t_2$ iff $I_v(t_1) = I_v(t_2)$, iff $[t_1] = [t_2]$, and as these are equivalence classes under identities in Δ^+, iff $t_1 = t_2 \in \Delta^+$.

For the rest of the proof suppose $M, v \models \varphi$ if and only if $\varphi \in \Delta$, where φ has fewer than n connectives. We skip the standard cases of $\neg, \wedge, \forall$ (see e.g. [7]).

Case 4. φ is $(\lambda x\psi)t$.

$(\lambda x\psi)t \in \Delta^+$ iff $\psi^x_t \in \Delta^+$ by deductive closure of Δ^+, iff $M, v \models \psi^x_t$ by induction hypothesis. t must be free for x in ψ, hence by the substitution lemma, $M, v \models \psi^x_t$ iff $M, v^x_{I_v(t)} \models \psi$, iff $M, v^x_{[t]} \models \psi$ and $I_v(t) = [t]$, as the latter holds by construction of M, and this in turn is the case iff $M, v \models (\lambda x\psi)t$ by the first semantic clause for lambda atoms.

Case 5. φ is $(\lambda x\psi)\imath y\chi$.
(a) If $(\lambda x\psi)\imath y\chi \notin \Delta^+$, then by deductive closure $\neg(\lambda x\psi)\imath y\chi \in \Delta^+$, and so for all terms t, either $\chi^y_t \notin \Delta^+$, or for some constant c, $\chi^y_c \in \Delta^+$ and $c = t \notin \Delta^+$, or $\psi^x_t \notin \Delta^+$. $[t] \in D$ iff t is a term, so by induction hypothesis, for all $[t] \in D$, either $M, v \not\models \chi^y_t$, or there is a $[c] \in D$ such that $M, v \models \chi^y_c$ and $M, v \not\models c = t$, or $M, v \not\models \psi^x_t$. χ^y_t is the same formula as $\chi^{y\,x}_{x\,t}$, so $M, v \not\models \chi^{y\,x}_{x\,t}$. Furthermore, x and y are not free in χ^y_c, so for any $o \in D$, $M, v \models \chi^y_c$ iff $M, v^x_o \models \chi^y_c$. By the substitution lemma, either $M, v^x_{I_v(t)} \not\models \chi^y_x$, or $M, v^x_{I_v(t)} \not\models \psi$, or there is a $[c] \in D$ such that $M, v^{x\ \ \ \ y}_{I_v(t)\,I_v(c)} \models \chi$ and $M, v^{x\ \ \ \ y}_{I_v(t)\,I_v(c)} \not\models y = x$. $I_v(t) = [t]$ and $I_v(c) = [c]$, so either $M, v^x_{[t]} \not\models \chi^y_x$, or $M, v^x_{[t]} \not\models \psi$, or there is a $[c] \in D$ such that $M, v^{x\ y}_{[t]\,[c]} \models \chi$ and $M, v^{x\ y}_{[t]\,[c]} \not\models y = x$, i.e. $v^{x\ y}_{[t]\,[c]}(y) \neq [t]$. $v^{x\ y}_{[t]\,[c]}$ is a y-variant of $v^x_{[t]}$, hence $M, v \not\models (\lambda x\psi)\imath y\chi$.
(b) If $(\lambda x\psi)\imath y\chi \in \Delta^+$, then for some constant c, $\psi^x_c, \chi^y_c \in \Delta^+$ and for all terms t, if $\chi^y_t \in \Delta^+$, then $c = t \in \Delta^+$. By induction hypothesis, $M, v \models \psi^x_c$ and $M, v \models \chi^y_c$. As y is either identical to x or x is not free in χ, χ^y_c is the same formula as $\chi^{y\,x}_{x\,c}$ and $I_v(c) = [c]$, so by the substitution lemma $M, v^x_{[c]} \models \psi$ and $M, v^x_{[c]} \models \chi^y_x$. Furthermore, for all $[t] \in D$, if $M, v \models \chi^y_t$, then $M, v \models c = t$, i.e. $I_v(t) = I_v(c)$, i.e. $I_v(t) = [c]$. Let v' be a y-variant of $v^x_{[c]}$, i.e. $v' = v^{x\ y}_{[c]\,[s]}$, for some $[s] \in D$. Either y is identical to x or x is not free in χ, so $v^{x\ y}_{[c]\,[s]}$ and v agree on the assignments of elements of D to all variables in χ except possibly y, and so $M, v^{x\ y}_{[c]\,[s]} \models \chi$ iff $M, v^y_{[s]} \models \chi$. So suppose now $M, v' \models \chi$ and $v'(y) \neq [c]$. $v'(y) = [s]$, so $[c] \neq [s]$. Then $M, v^y_{[s]} \models \chi$, and also if $M, v \models \chi^y_s$, then $M, v \models c = s$, i.e. $I_v(s) = I_v(c)$, i.e. $I_v(s) = [c]$. But $I_v(s) = [s]$, so $I_v(s) \neq [c]$. Hence $M, v \not\models \chi^y_s$, and so by the substitution lemma, $M, v^y_{[s]} \not\models \chi$. Contradiction.

Finally, restrict the language again to the language of Δ: structure M constructed from Δ^+ satisfies Δ. This completes the proof of Theorem 5. □

Theorem 6 (Completeness for Sequents). *If* $\models \Gamma \Rightarrow \Delta$, *then* $\vdash \Gamma \Rightarrow \Delta$.

Proof. Let $\neg\Delta$ be the negation of all formulas in Δ. If $\models \Gamma \Rightarrow \Delta$, then $\Gamma, \neg\Delta$ is not satisfiable. Hence by Theorem 5 it is inconsistent, and as they are both finite, $\vdash \Gamma, \neg\Delta \Rightarrow \bot$. Hence by the properties of negation $\vdash \Gamma \Rightarrow \Delta$. □

Theorem 7 (Completeness for Sets). *If* $\Gamma \models A$, *then* $\Gamma \vdash A$.

Proof. Suppose $\Gamma \models A$. Then $\Gamma, \neg A$ is not satisfiable, hence by Theorem 5 it is inconsistent and $\Gamma, \neg A \vdash \bot$. So for some finite $\Sigma \subseteq \Gamma, \neg A$, $\Sigma \Rightarrow \bot$. If $\neg A \in \Sigma$, then by the deductive properties of negation, $\Sigma - \{\neg A\} \Rightarrow A$, and as $\Sigma - \{\neg A\}$ is certain to be a subset of Γ, $\Gamma \vdash A$. If $\neg A \notin \Sigma$, then $\Sigma \Rightarrow A$ by the properties of negation, and again $\Gamma \vdash A$. □

By theorem 1 and 7 we also obtain the (strong) completeness of HRL.

6 Conclusion

Summing up, **RL** saves the essential features of the Russellian approach to definite descriptions. It avoids problems like the arbitrary restriction of axiom R to predicate symbols and scoping difficulties. In the semantics it retains the reductionist Russellian flavour in the sense that DD are not characterised by an interpretation function, but instead they are treated as a case in the clauses of the forcing definition for lambda atoms. In this respect **RL** is different from the approach provided by Fitting and Mendelsohn [10] which is closer to the Fregean tradition.

The rules of GRL are in principle direct counterparts of the tableau rules from [17] but with two important exceptions. The tableau rule corresponding to $(= -)$ is not restricted to atomic formulas and the tableau rule corresponding to $(\imath_2 \Rightarrow)$ is not branching. Its counterpart in sequent calculus would be:

$$(\imath_2 \Rightarrow') \quad \frac{b_1 = b_2, \Gamma \Rightarrow \Delta}{(\lambda x \psi)\imath y \varphi, \varphi[y/b_1], \varphi[y/b_2], \Gamma \Rightarrow \Delta}$$

Such a non-branching rule is certainly much better for proof search, but it is not possible to prove the cut elimination theorem in its presence. The same applies to $(= -)$ without restriction to atomic formulas. In both cases the occurrences of arbitrary formulas φ in the antecedent of the conclusion can be cut formulas and, in case the cut formula in the left premiss of the cut application is principal, it is not possible to make a reduction of the complexity of the cut formulas.

There is an interesting advantage of introducing the sequent characterisation of **RL** over tableau formalisation from [17]. Since no rule specific to GRL has more than one active formula in the succedent they are also correct in the setting of intuitionistic logic as characterised by G1i [34]. It is sufficient to change the background calculus for the intuitionistic version (with $(\leftrightarrow\Rightarrow)$, $(\Rightarrow \vee)$ split into two rules, and $(\Rightarrow C), (\Rightarrow W)$ deleted) and check that all proofs from Sect. 3, 4 hold also for a (syntactically characterised) intuitionistic version of **RL**. By comparison, the changes in the tableau setting would be rather more involved and connected with the introduction of labels for naming the states of knowledge in the constructed model.

The approach provided here may be modified also to cover some more expressive logics (like modal ones) and some other theories of DD like those proposed in the context of free logics. Some preliminary work in this direction is found in [12] and [13]. On the other hand the problems briefly mentioned in Sect. 1 need serious examination and this may be carried out only after the implementation of the presented formal systems. This is one of the most important future tasks.

Acknowledgements. We would like to thank Michał Zawidzki for his comments and suggestions.

References

1. Artale, A., Mazzullo, A., Ozaki, A., Wolter, F.: On free description logics with definite descriptions. In: Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, pp. 63–73. IJCAI Organization (2021)
2. Beeson, M.: Foundations of Constructive Mathematics. Springer (1985). https://doi.org/10.1007/978-3-642-68952-9
3. Benzmüller, C., Scott, D.S.: Automating free logic in HOL, with an experimental application in category theory. J. Autom. Reason. **64**, 53–72 (2020)
4. Blumson, B.: Anselm's God in Isabelle/HOL URL (2020). https://www.isa-afp.org/browser_info/current/AFP/AnselmGod/document.pdf
5. Bohrer, B., Fernández, M., Platzer, A.: dL_ι: definite descriptions in differential dynamic logic. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 94–110. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_6
6. Bressan, A.: A General Interpreted Modal Calculus. Yale University Press, Yale (1972)
7. Enderton, H.B.: A Mathematical Introduction to Logic. Harcourt Academic Press, San Diego (2000)
8. Feferman, S.: Definedness. Erkenntnis **43**, 295–320 (1995)
9. Fitting, M.: A modal logic epsilon-calculus. Notre Dame J. Formal Logic **16**(1), 1–16 (1975)
10. Fitting, M., Mendelsohn, R. L.: First-Order Modal Logic. Synthese Library, vol. 277. Springer, Dordrecht (1998). https://doi.org/10.1007/978-94-011-5292-1
11. Francez, N., Więckowski, B.: A proof-theoretic semantics for contextual definiteness. In: Moriconi, E, Tesconi, L. (eds.) Proceedings of the Second Pisa Colloquium in Logic, Language and Epistemology. Edizioni ETS, Pisa, pp. 181–212 (2014)
12. Indrzejczak, A.: Existence, definedness and definite descriptions in hybrid modal logic. In: Olivetti, N., Verbrugge, R., Negri, S., Sandu, G. (eds.) Advances in Modal Logic, vol. 13, pp. 349–368. College Publications, Rickmansworth (2020)
13. Indrzejczak, A.: Free definite description theory - sequent calculi and cut elimination. Logic Log. Philos. **29**(4), 505–539 (2020)
14. Indrzejczak, A.: Free logics are cut-free. Stud. Logica. **109**, 859–886 (2021)
15. Indrzejczak, A.: Sequents and Trees. An Introduction to the Theory and Applications of Propositional Sequent Calculi, Birkhäuser (2021)
16. Indrzejczak, A.: Russellian definite description theory–a proof-theoretic approach. Rev. Symbolic Logic **16**(2), 624–649 (2023)
17. Indrzejczak, A., Zawidzki, M.: When Iota meets Lambda. Synthese **201**(2), 1–33 (2023)
18. Kalish, D., Montague, R., Mar, G.: Logic. Techniques of Formal Reasoning, 2 ed. Oxford University Press, New York, Oxford (1980)
19. Kürbis, N.: A binary quantifier for definite descriptions in intuitionist negative free logic: natural deduction and normalization. Bull. Sect. Logic **48**(2), 81–97 (2019)
20. Kürbis, N.: Two treatments of definite descriptions in intuitionist negative free logic. Bull. Sect. Logic **48**(4), 299–317 (2019)
21. Kürbis, N.: Definite descriptions in intuitionist positive free logic. Logic Log. Philos. **20**(2), 327–358 (2021)
22. Kürbis, N.: Proof-theory and semantics for a theory of definite descriptions. In: Das, A., Negri, S. (eds.) Autom. Reason. Analytic Tableaux Related Methods. Springer, Berlin, Heidelberg (2021)

23. Kürbis, N.: A binary quantifier for definite descriptions for cut free free logics. Stud. Logica. **110**(1), 219–239 (2022)
24. Metcalfe, G., Olivetti, N., Gabbay, D.: Proof Theory for Fuzzy Logics. Springer (2008). https://doi.org/10.1007/978-1-4020-9409-5
25. Neuhaus, F., Kutz, O., Righetti, G.: Free description logic for ontologists. In: Hammar, K. et al. (eds.), Proceedings of the Joint Ontology Workshops co-located with the Bolzano Summer of Knowledge (BOSK 2020). vol. 2708, Bozen-Bolzano (2020)
26. Oppenheimer, P.E., Zalta, E.N.: A computationally-discovered simplification of the ontological argument. Australas. J. Philos. **89**, 333–349 (2011)
27. Orlandelli, E.: Labelled calculi for quantified modal logics with definite descriptions. J. Log. Comput. **31**(3), 923–946 (2021)
28. Russell, B.: On Denoting. Mind **XIV**, 479–494 (1905)
29. Scales, R.: Attribution and Existence. Ph.D. Dissertation. University of California, Irvine (1969)
30. Scott, D.: Identity and existence in intuitionistic logic. In: Fourman, M., Mulvey, C., Scott, D. (eds.) Applications of Sheaves. LNM, vol. 753, pp. 660–696. Springer, Heidelberg (1979). https://doi.org/10.1007/BFb0061839
31. Stalnaker, R.C., Thomason, R.H.: Abstraction in first-order modal logic. Theoria **34**(3), 203–207 (1968)
32. Tennant, N.: A general theory of abstraction operators. Philos. Q. **54**(214), 105–133 (2004)
33. Thomason, R. H.: Some completeness results for modal predicate calculi. In: Lambert, K. (ed.) Philosophical Problems in Logic, Reidel, pp. 56–76 (1970)
34. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Oxford University Press, Oxford (1996)
35. Whitehead, A.N., Russell, B.: Principia Mathematica, vol. I. Cambridge University Press, Cambridge (1910)

Towards Proof-Theoretic Formulation of the General Theory of Term-Forming Operators

Andrzej Indrzejczak(✉)

Department of Logic, University of Lodz, Lodz, Poland
andrzej.indrzejczak@filhist.uni.lodz.pl

Abstract. Term-forming operators (tfos), like iota- or epsilon-operator, are technical devices applied to build complex terms in formal languages. Although they are very useful in practice their theory is not well developed. In the paper we provide a proof-theoretic formulation of the general approach to tfos provided independently by several authors like Scott, Hatcher, Corcoran, and compare it with an approach proposed later by Tennant. Eventually it is shown how the general theory can be applied to specific areas like Quine's set theory NF.

Keywords: Term-Forming Operators · Abstraction Operator · Definite Descriptions · Sequent Calculus · Quine

1 Introduction

In formal languages terms are usually treated as these elements of language which only refer to the objects in the domain of discourse. In particular, this way of treating terms is prevailing in proof theory and automated deduction where usually only functional terms are approved. In contrast, in natural languages, naming expressions are used very often not only for referring to objects but also for conveying information about them. In the earlier stages of development of mathematical logic several formal devices were introduced for this aim which currently are rather neglected. These term-forming operators, also called shortly tfos or vbtos (variable binding term operators), include, among others:

- iota-operator (Peano): $\imath x\varphi$ - the (only) x such that φ;
- epsilon-operator (Hilbert): $\epsilon x\varphi$ - a(n) x such that φ;
- abstraction-operator: $\{x : \varphi\}$ - the set of (all) x satisfying φ;
- counting-operator (Frege): $\sharp x\varphi$ - the number of x such that φ;
- lambda-operator (Church): $\lambda x\varphi$ - the property of being φ.

Funded by the European Union (ERC, ExtenDD, project number: 101054714). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

It seems that currently only the lambda-operator is treated as an important tool and found diverse applications in recursion theory, type theory and proof theory. Abstraction-operator, although commonly used in practice, is rather not treated seriously in the formal development of set theories. The remaining ones are sadly treated as formal tools having only some historical value. Since the role of complex terms as information conveying tools is crucial in communication it is important to fill this gap.

Recently, some more attention was paid to proof theory of definite descriptions. In particular, cut-free sequent calculi were provided for Fregean [11], Russellian [17] and free description theories [13]. The latter theories were also characterised in terms of tableau systems [18] and tableau calculus was also used to develop a Russelian theory in the language enriched with lambda-operator [19]. Some modal logics of definite descriptions were also developed in terms of cut-free sequent calculus [10], in particular, the logic of Fitting and Mendelsohn [5] was independently formalised as a labelled sequent calculus [28] and as a hybrid system [12]. Alternatively, interesting natural deduction and sequent calculi were proposed for free and intuitionistic logics of definite descriptions characterised in terms of binary quantifier [21–25].

Since definite descriptions are amenable to proof theoretic treatment it is tempting to suspect that for other tfos we can obtain equally interesting results. Perhaps one should start with posing a question whether a general theory of such operators is possible? In fact at least two different attempts to develop such a theory were proposed. The earlier approach was independently introduced by several authors, including: Scott [32], Da Costa [3,4], Hatcher [7,8], Corcoran and Herring [1,2]. It was formulated semantically and as an axiomatic theory. In what follows it will be called simply S-theory (after Scott). The second approach was introduced by Neil Tennant [33], and then developed in [35] as a general theory of abstraction operators (see also [34,36]). This T-theory was formulated in terms of natural deduction system and with adequate semantical characterisation. In what follows we will examine these two approaches and show how they can be formulated as well-behaved sequent calculi in Sect. 3. Then, in Sect. 4 we consider their specification with respect to set-abstraction operator. For this aim we focus on Quine's version of set theory NF (New Foundations) [29] (see also [30]) but the proposed systems may be modified to apply to other formulations of set theory as well.

2 Preliminaries

We will be using standard first-order predicate languages with quantifiers $\forall, \exists$, identity predicate $=$ and arbitrary term-forming operator τ making complex terms from formulae of the language. The definition of a term and formula is standard, by simultaneous recursion on both categories. In the presented system the only terms are variables and complex terms constructed by means of arbitrary unary tfo τ. The complex terms are written as $\tau x\varphi$ where φ is a formula in the scope of respective operator.

In accordance with Gentzen's custom we divide individual variables into bound $VAR = \{x, y, z, \ldots\}$ and free variables (parameters) $PAR = \{a, b, c, \ldots\}$. It makes easier an elaboration of some technical issues concerning substitution and proof transformations. In the metalanguage φ, ψ, χ denote any formulae and $\Gamma, \Delta, \Pi, \Sigma$ their multisets. Metavariables $t, t_1, \ldots$ denote arbitrary terms. $\varphi[t_1/t_2]$ is officially used for the operation of correct substitution of a term t_2 for all occurrences of a term t_1 (a variable or parameter) in φ, and similarly $\Gamma[t_1/t_2]$ for a uniform substitution in all formulae in Γ. Ocassionally, we will use simplified notation $\varphi(t)$ to denote the result of correct substitution.

First-order logic in general will be abbreviated as FOL or FOLI if identity is primitive. CFOL(I), PFFOL(I), NFFOL(I) denote the classical, positive free and negative free versions. The basic system GC for CFOL consists of the following rules:

$$(Cut)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma} \qquad (AX)\ \varphi, \Gamma \Rightarrow \Delta, \varphi$$

$$(\neg\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\neg)\ \frac{\varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi} \qquad (W\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}$$

$$(\Rightarrow\wedge)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi} \qquad (\wedge\Rightarrow)\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow W)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(\vee\Rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\vee)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \qquad (C\Rightarrow)\ \frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}$$

$$(\rightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi} \qquad (\Rightarrow C)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(\leftrightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi \qquad \varphi, \psi, \Gamma \Rightarrow \Delta}{\varphi \leftrightarrow \psi, \Gamma \Rightarrow \Delta} \qquad (\forall\Rightarrow)\ \frac{\varphi[x/t], \Gamma \Rightarrow \Delta}{\forall x\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\exists)\ \frac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{\Gamma \Rightarrow \Delta, \exists x\varphi}$$

$$(\Rightarrow\leftrightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi \qquad \psi, \Gamma \Rightarrow \Delta, \varphi}{\Gamma \Rightarrow \Delta, \varphi \leftrightarrow \psi} \qquad (\Rightarrow\forall)\ \frac{\Gamma \Rightarrow \Delta, \varphi[x/a]}{\Gamma \Rightarrow \Delta, \forall x\varphi} \qquad (\exists\Rightarrow)\ \frac{\varphi[x/a], \Gamma \Rightarrow \Delta}{\exists x\varphi, \Gamma \Rightarrow \Delta}$$

where a is a fresh parameter (eigenvariable), not present in Γ, Δ and φ.

If instead of $(\forall\Rightarrow)$ and $(\Rightarrow\exists)$ we introduce:

$$(\forall\Rightarrow)\ \frac{\varphi[x/b], \Gamma \Rightarrow \Delta}{\forall x\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\exists)\ \frac{\Gamma \Rightarrow \Delta, \varphi[x/b]}{\Gamma \Rightarrow \Delta, \exists x\varphi}$$

we obtain a pure variant GPC which is adequate for CFOL with variables as the only terms but in general incomplete for extensions with some tfos.

The variant GF for PFFOL can be obtained by changing all quantifier rules into:

$$(\forall\Rightarrow)^F\ \frac{\varphi[x/t], \Gamma \Rightarrow \Delta}{Et, \forall x\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\forall)^F\ \frac{Ea, \Gamma \Rightarrow \Delta, \varphi[x/a]}{\Gamma \Rightarrow \Delta, \forall x\varphi}$$

$$(\exists\Rightarrow)^F\ \frac{Ea, \varphi[x/a], \Gamma \Rightarrow \Delta}{\exists x\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\exists)^F\ \frac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{Et, \Gamma \Rightarrow \Delta, \exists x\varphi}$$

where E is the existence predicate, which is usually defined as $Et := \exists x(x = t)$. This form of rules follows from the fact that in free logics terms may designate nonexistent objects whereas quantifiers have existential import. For pure version GPF again we use b instead of t in $(\forall\Rightarrow)^F$ and $(\Rightarrow\exists)^F$.

Moreover, in negative free logic atomic formulae with such terms are false which implies that $Et \rightarrow t = t$ and $\varphi(t) \rightarrow Et$, for any atomic formula φ. Hence to obtain GNF (or GPNF) for NFFOL we have to add to GF (or GPF) the rule requiring all predicates to be strict in the sense that they are satisfied only by denoting terms:

$$(Str)\ \frac{Et, \Gamma \Rightarrow \Delta}{\varphi(t), \Gamma \Rightarrow \Delta} \qquad \text{where } \varphi \text{ is atomic.}$$

Identity can be characterised in GC (GPC) and GF (GPF) in several ways (see [16]). For our purposes we use the following rules:

$$(Ref)\ \frac{t = t, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \quad (2LL)\ \frac{\Gamma \Rightarrow \Delta, t_1 = t_2 \qquad \Gamma \Rightarrow \Delta, \varphi[x/t_1]}{\Gamma \Rightarrow \Delta, \varphi[x/t_2]}$$

where φ is atomic.

GCI, GPCI, GFI, GPFI will denote the respective calculi with the rules for identity added. In case of NFFOLI, due to strictness condition, reflexivity does not hold unconditionally and we must weaken the first rule, using instead:

$$(Ref)^N\ \frac{t=t, \Gamma \Rightarrow \Delta}{Et, \Gamma \Rightarrow \Delta}$$

GNFI, GPNFI will denote the respective calculi for NFFOLI with the rules for identity having $(Ref)^N$.

Proofs are defined in the standard way as finite trees with nodes labelled by sequents. The height of a proof $\mathcal{D}$ of $\Gamma \Rightarrow \Delta$ is defined as the number of nodes of the longest branch in $\mathcal{D}$. $\vdash_k \Gamma \Rightarrow \Delta$ means that $\Gamma \Rightarrow \Delta$ has a proof with height at most k. Let us recall that formulae displayed in the schemata are active, whereas the remaining ones are parametric, or form a context. In particular, all active formulae in the premisses are called side formulae, and the one in the conclusion is the principal formula of the respective rule application.

Note that the Cut-elimination theorem holds for all above mentioned calculi (see e.g. [15]) and the full Leibniz' Law LL: $t_1 = t_2, \varphi[x/t_1] \Rightarrow \varphi[x/t_2]$ (for arbitrary formula φ) is also provable.

3 The General Theory

The S-theory of tfos is expressed by two general principles:

EXT: $\forall x(\varphi(x) \leftrightarrow \psi(x)) \rightarrow \tau x\varphi(x) = \tau x\psi(x)$
AV: $\tau x\varphi(x) = \tau y\varphi(y)$

or, equivalently, by one principle:

EXTAV: $\forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y))) \rightarrow \tau x\varphi(x) = \tau y\psi(y)$

Such a general theory was first developed on the basis of positive free first-order logic with identity by Scott [32]. However, the remaining authors used the classical first-order logic with identity as the basis. In both cases the general completeness theorem was provided and several important model theoretic results which hold for CFOLI (see in particular Da Costa [4]). In what follows, we will pay more attention to classical case since for several kinds of tfos, in particular for descriptions, it is rather difficult to find reasonable theories, in contrast to the situation in free logic (see [26]).

Several possible objections can be raised against such a theory. In a sense it is too general and too weak, on the other hand, for specific kind of operators it may be too strong, in particular in the setting of classical logic. Let us illustrate these remarks with some examples. For example, for $\imath$-operator Rosser [30] is enforced to add (in CFOLI) to EXT and AV the following axiom:

$$\exists_1 x\varphi(x) \rightarrow \forall x(x = \imath x\varphi(x) \leftrightarrow \varphi(x))$$

which still gives incomplete logic as noticed by Hailperin [6]. Da Costa [4] adds:

$$\exists_1 x\varphi(x) \rightarrow \forall x(x = \imath x\varphi(x) \rightarrow \varphi(x)) \text{ and}$$

$$\neg\exists_1 x\varphi(x) \rightarrow \imath x\varphi(x) = \imath x(x \neq x)$$

In fact, the theory of descriptions axiomatised by the addition of these two axioms to EXT and AV is redundant, since the latter principles can be proven with their help. This theory is in fact equivalent to Fregean/Carnapian theory of descriptions (often called the chosen object theory), in particular in the formulation of Kalish and Montague [20]. However, we call an S-theory every theory of arbitrary tfo where EXT and AV hold either as axioms or as derived theses.

On the other hand, for some theories of definite descriptions these two principles are too strong. For example, in the Russellian theory [31,37] both principles do not hold. Instead we have their weaker versions:

wEXT: $E\imath x\varphi(x) \rightarrow ((\varphi(x) \leftrightarrow \psi(x)) \rightarrow \imath x\varphi(x) = \imath x\psi(x))$

wAV: $E\imath x\varphi(x) \rightarrow \imath x\varphi(x) = \imath y\varphi(y)$.

In other cases of tfos, like set-abstraction operator or counting operator, EXT may be even more disastrous, since for the latter it yields one half of the Fregean ill-famed V law, in fact this half which is sufficient for deriving contradiction. Similar problems with set-abstraction will be discussed below.

3.1 The Formalisation of S-Theory

To obtain an adequate sequent calculus for S-theory we add to GCI the following two rules:

$$(Ext)\ \frac{\varphi(a), \Gamma \Rightarrow \Delta, \psi(a) \qquad \psi(a), \Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, \tau x\varphi(x) = \tau x\psi(x)} \quad (AV)\ \frac{\tau x\varphi(x) = \tau y\varphi(y), \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

where a is a fresh parameter.

Alternatively, we can add just one rule corresponding to EXTAV:

$$(ExtAV)\ \frac{a = b, \varphi(a), \Gamma \Rightarrow \Delta, \psi(b) \qquad a = b, \psi(b), \Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, \tau x\varphi(x) = \tau y\psi(y)}$$

where both a, b are fresh parameters.

Theorem 1. *GCI+$\{(Ext), (AV)\}$ and GCI+$\{(ExtAV)\}$ are equivalent to axiomatic formulations of S-theory of tfos.*

Proof. It is sufficient to prove respective axioms in GCI+$\{(Ext), (AV)\}$ or in GCI+$\{(ExtAV)\}$ and to show that the above rules are derivable in GCI with added axioms EXT, AV or EXTAV. We will show this for the more compact version with $(ExtAV)$ and EXTAV; proofs for the remaining rules and axioms are similar and simpler. Provability of EXTAV:

$$(ExtAV)\ \dfrac{(\forall\Rightarrow)\ \dfrac{(\rightarrow\Rightarrow)\ \dfrac{a = b \Rightarrow a = b \qquad \varphi(a) \leftrightarrow \psi(b), \varphi(a) \Rightarrow \psi(b)}{a = b \rightarrow (\varphi(a) \leftrightarrow \psi(b)), a = b, \varphi(a) \Rightarrow \psi(b)}}{\forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y))), a = b, \varphi(a) \Rightarrow \psi(b)} \qquad \mathcal{D}}{\forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y))) \Rightarrow \tau x\varphi(x) = \tau y\psi(y)}$$

where the rightmost leaf is provable and $\mathcal{D}$ is an analogous proof of $\forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y))), a = b, \psi(b) \Rightarrow \varphi(a)$.

Derivability of $(ExtAV)$:

$$(Cut)\ \dfrac{(\Rightarrow\forall)\ \dfrac{(\Rightarrow\rightarrow)\ \dfrac{(\Rightarrow\leftrightarrow)\ \dfrac{a = b, \varphi(a), \Gamma \Rightarrow \Delta, \psi(b) \qquad a = b, \psi(b), \Gamma \Rightarrow \Delta, \varphi(a)}{a = b, \Gamma \Rightarrow \Delta, \varphi(a) \leftrightarrow \psi(b)}}{\Gamma \Rightarrow \Delta, a = b \rightarrow (\varphi(a) \leftrightarrow \psi(b))}}{\Gamma \Rightarrow \Delta, \forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y)))} \qquad \mathcal{D}}{\Gamma \Rightarrow \Delta, \tau x\varphi(x) = \tau y\psi(y)}$$

where both leaves are premisses and $\mathcal{D}$ is a proof of $\forall xy(x = y \rightarrow (\varphi(x) \leftrightarrow \psi(y))) \Rightarrow \tau x\varphi(x) = \tau y\psi(y)$ from the axiom $\Rightarrow EXTAV$. □

Let us consider the question of cut elimination for either of the two formalisations of S-theory. We can observe that the choice of the rule $(2LL)$ for representation of LL was connected with the shape of (Ext) or $(ExtAV)$. In both calculi identities can appear as the principal formulae of some rule application only in the succedent. This makes it safe for proving cut elimination since identities in antecedents can only appear either as parametric formulae or as formulae introduced by weakening. In both cases if identity is a cut formula under consideration it is eliminable either by induction on the height of cut or directly.

Still there is a problem connected with the application of $(\forall\Rightarrow)$ and $(\Rightarrow\exists)$ to complex terms. If for example $\forall x\varphi$ is a cut formula which was in both premisses of cut introduced as the principal formula, and in the right premiss x was instantiated with $\tau y\psi$, then the formula $\varphi[x/\tau y\psi]$ may have higher complexity than $\forall x\varphi$ and the induction on the complexity of cut formulae fails. This problem may be overcome either by introduction of more complex way of measuring the complexity of formulae (see e.g. [11]) or by replacing the basic calculus GCI with its pure version GPCI. Of course, the restriction of all quantifier rules to parameters makes the calculus with complex terms incomplete. However, to avoid the loss of generality we can add to GPCI the rule:

$$(a\Rightarrow)\ \frac{a=\tau x\varphi(x), \Gamma\Rightarrow\Delta}{\Gamma\Rightarrow\Delta}$$

where a is a fresh parameter.

Theorem 2. *The calculus GPCI+$\{(Ext),(AV)\}$ (or GPCI+$\{(ExtAV)\}$) with added $(a\Rightarrow)$ is equivalent to GCI+$\{(Ext),(AV)\}$ (or GCI+$\{(ExtAV)\}$)*

Proof. It is enough to show that $(a\Rightarrow)$ is derivable in GCI:

$$(Cut)\ \dfrac{(\Rightarrow\exists)\ \dfrac{\Rightarrow\tau x\varphi(x)=\tau x\varphi(x)}{\Rightarrow\exists y(y=\tau x\varphi(x))} \qquad \dfrac{a=\tau x\varphi(x),\Gamma\Rightarrow\Delta}{\exists y(y=\tau x\varphi(x)),\Gamma\Rightarrow\Delta}\ (\exists\Rightarrow)}{\Gamma\Rightarrow\Delta}$$

and that unrestricted $(\forall\Rightarrow),(\Rightarrow\exists)$ are derivable in GPC with $(a\Rightarrow)$:

$$(Cut)\ \dfrac{\Gamma\Rightarrow\Delta,\varphi(\tau x\psi(x)) \qquad \varphi(\tau x\psi(x)), a=\tau x\psi(x)\Rightarrow\varphi(a)}{(a\Rightarrow)\ \dfrac{(\Rightarrow\exists)\ \dfrac{a=\tau x\psi(x),\Gamma\Rightarrow\Delta,\varphi(a)}{a=\tau x\psi(x),\Gamma\Rightarrow\Delta,\exists x\varphi}}{\Gamma\Rightarrow\Delta,\exists x\varphi}}$$

where the rightmost sequent being an instance of LL is provable. Similar proof works for $(\forall\Rightarrow)$. □

Let us call GPCI+$\{(Ext),(AV)\}$ (or GPCI+$\{(ExtAV)\}$) with added $(a\Rightarrow)$ simply GS (GS'). Note that for both systems the following lemma holds:

Lemma 1. *1. $\vdash t_1=t_2,\varphi[x/t_1]\Rightarrow\varphi[x/t_2]$, for any formula φ.*
2. If $\vdash_k\Gamma\Rightarrow\Delta$, then $\vdash_k\Gamma[b_1/b_2]\Rightarrow\Delta[b_1/b_2]$, where k is the height of a proof.

Proof. 1. follows by induction on the complexity of φ and is standard for all cases. The proof of 2 is by induction on the height of proofs. □

The first result is Leibniz' Law (LL) stated in full generality, i.e. covering also complex terms. Since $(2LL)$ yields only LL restricted to atomic formulae, we need its unrestricted form for completeness. The second result is a substitution lemma which is necessary for unifying terms while proving the cut elimination theorem. Note that it is restricted to parameters only but in the case of GS (GS'), which is an extension of GPCI, it is sufficient since only parameters are instantiated for bound variables in all applications of quantifier rules.

Theorem 3. *The cut elimination theorem holds for GS and GS'.*

Proof. The proof is standard and essentially requires two inductions: on the complexity of cut formula and on the height of the derivations of both premisses of cut. In general we can follow the strategy applied for example in [15]; here we focus only on the crucial points connected with the new rules which could lead to troubles.

Consider the situation where the cut formula in the left premiss is the principal formula of the application of $(2LL)$. It is an atomic formula, possibly an identity. Since in no logical rule atomic formula in the antecedent can be a principal formula, so in the right premiss a suitable cut formula is either introduced by weakening or is just a parametric formula. In the first case it is directly eliminated, in the second it is eliminated by induction on the height of the proof. The case where the right premiss is axiomatic is also directly eliminable.

The cases where in the left premiss the cut formula is the principal formula of the application of (Ext) or $(ExtAV)$ are treated in a similar way. Eventually, rules like (AV) or $(a \Rightarrow)$ have no impact on the elimination of cuts since there are no principal formulae in the conclusion. □

Although we cannot totally avoid the loss of the subformula property in GS and GS', the introduction of complex terms is separated from quantifier rules and technically it is more desirable. In fact, from the semantic point of view we are not really in need of introducing an arbitrary complex term in the premiss while doing a proof-search. The rule is required only for these terms which either occur already in Γ, Δ, or have in their scope the formulae from Γ, Δ. It can be shown by providing Hintikka-style completeness proof for this system which is possible since Henkin-style proofs were provided by the mentioned authors; we omit the details because of space restrictions.

In fact, for the needs of proof-search we could simplify GS (GS') a little bit. In particular we could use a more convenient one-premiss rule of Negri and von Plato [27] for LL of the form:

$$(1LL)\ \frac{\varphi(t_2), \Gamma \Rightarrow \Delta}{t_1 = t_2, \varphi(t_1), \Gamma \Rightarrow \Delta}$$

for all cases where at least one of t_1, t_2 is a parameter and $\varphi(t_1)$ is not an identity with both arguments being complex terms. In fact, the only troublesome cases of LL which could make a clash in the proof of cut elimination are three:

1. $b = t, t = t' \Rightarrow b = t'$
2. $t = t', \varphi(t) \Rightarrow \varphi(t')$
3. $t = t', t' = t'' \Rightarrow t = t''$

where t, t' are complex terms, and only for these cases a two-premiss rule $(2LL)$ is necessary.

Also note that instead of (Ref) we can use more restricted version:

$$(Ref')\ \frac{b = b, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

since $\tau x \varphi(x) = \tau x \varphi(x)$ is derivable by (Ext) or $(ExtAV)$.

3.2 The Formalisation of T-Theory

The theory of abstraction-operators developed by Tennant, which we call here a T-theory of tfos, is generally much stronger than S-theory. But we must emphasize that it is formulated in the setting of much weaker logic, namely NFFOLI (negative free FOLI), where not only quantifier rules are weaker but also the identity is not (unconditionally) reflexive.

Tennant's theory of tfo is based on the following natural deduction rules:

(τI) If $\varphi(a), Ea \vdash aRt$ and $aRt \vdash \varphi(a)$ and Et, then $t = \tau x \varphi(x)$;
$(\tau E1)$ If $t = \tau x \varphi(x)$ and $\varphi(b)$ and Eb, then bRt
$(\tau E2)$ If $t = \tau x \varphi(x)$, then Et
$(\tau E3)$ If $t = \tau x \varphi(x)$ and bRt, then $\varphi(b)$

where a is an eigenvariable, and R is a specific relation involved in the characterisation of τ. For example, R is $=$ for the case of $\imath$, and $\in$ for set-abstraction operator. The corresponding sequent rules are:

$$(\Rightarrow \tau)\ \frac{\Gamma \Rightarrow \Delta, Et \qquad Ea, \varphi(a), \Gamma \Rightarrow \Delta, aRt \qquad aRt, \Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, t = \tau x \varphi(x)}$$

where a is not in Γ, Δ, φ

$$(\Rightarrow \tau E1)\ \frac{\Gamma \Rightarrow \Delta, Eb \qquad \Gamma \Rightarrow \Delta, \varphi(b) \qquad \Gamma \Rightarrow \Delta, t = \tau x \varphi(x)}{\Gamma \Rightarrow \Delta, bRt}$$

$$(\Rightarrow \tau E2)\ \frac{\Gamma \Rightarrow \Delta, t = \tau x \varphi(x)}{\Gamma \Rightarrow \Delta, Et}$$

$$(\Rightarrow \tau E3)\ \frac{\Gamma \Rightarrow \Delta, bRt \qquad \Gamma \Rightarrow \Delta, t = \tau x \varphi(x)}{\Gamma \Rightarrow \Delta, \varphi(b)}$$

To get more standard SC we can apply the rule-generation theorem (see e.g. [14]) and obtain left introduction rules for τ:

$$(\tau \Rightarrow 1)\ \frac{\Gamma \Rightarrow \Delta, Eb \qquad \Gamma \Rightarrow \Delta, \varphi(b) \qquad bRt, \Gamma \Rightarrow \Delta}{t = \tau x \varphi(x), \Gamma \Rightarrow \Delta}$$

$$(\tau \Rightarrow 2)\ \frac{Et, \Gamma \Rightarrow \Delta}{t = \tau x \varphi(x), \Gamma \Rightarrow \Delta}$$

$$(\tau \Rightarrow 3)\ \frac{\Gamma \Rightarrow \Delta, bRt \qquad \varphi(b), \Gamma \Rightarrow \Delta}{t = \tau x\varphi(x), \Gamma \Rightarrow \Delta}$$

Note that if we transfer these rules to the setting of CFOLI we do not need formulae of the form Et, and the rule $(\tau \Rightarrow 2)$, being specific to negative free logic, is superfluous. As a result we obtain the following three rules:

$$(\Rightarrow \tau)\ \frac{\varphi(a), \Gamma \Rightarrow \Delta, aRt \qquad aRt, \Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, t = \tau x\varphi(x)}$$

where a is not in Γ, Δ, φ

$$(\tau \Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi(b) \qquad bRt, \Gamma \Rightarrow \Delta}{t = \tau x\varphi(x), \Gamma \Rightarrow \Delta}$$

$$(\tau \Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, bRt \qquad \varphi(b), \Gamma \Rightarrow \Delta}{t = \tau x\varphi(x), \Gamma \Rightarrow \Delta}$$

In general what we obtain with these rules is equivalent to the following principle, often called Lambert axiom:

LA: $\forall y(y = \tau x\varphi(x) \leftrightarrow \forall x(\varphi(x) \leftrightarrow xRy))$

which is derivable also in the setting of NFFOLI. In the setting of CFOLI it is equivalent to Hintikka axiom:

HA: $t = \tau x\varphi(x) \leftrightarrow \forall x(\varphi(x) \leftrightarrow xRt)$

for which we demonstrate syntactically the equivalence with the stated rules. In one direction we have:

$$(\Rightarrow \forall)\ \dfrac{(\Rightarrow\leftrightarrow)\ \dfrac{(\tau \Rightarrow)\ \dfrac{\varphi[x/a] \Rightarrow \varphi[x/a] \qquad aRt \Rightarrow aRt}{t = \tau x\varphi(x), \varphi[x/a] \Rightarrow aRt} \qquad \dfrac{aRt \Rightarrow aRt \qquad \varphi[x/a] \Rightarrow \varphi[x/a]}{t = \tau x\varphi(x), aRt \Rightarrow \varphi[x/a]}}{t = \tau x\varphi(x) \Rightarrow \varphi[x/a] \leftrightarrow aRt}}{t = \tau x\varphi(x) \Rightarrow \forall x(\varphi(x) \leftrightarrow xRt)}$$

In the second direction:

$$(\Rightarrow \tau)\ \dfrac{(\forall \Rightarrow)\ \dfrac{(\leftrightarrow\Rightarrow)\ \dfrac{aRt \Rightarrow aRt \qquad \varphi[x/a] \Rightarrow \varphi[x/a]}{\varphi[x/a] \leftrightarrow aRt, aRt \Rightarrow \varphi[x/a]}}{\forall x(\varphi(x) \leftrightarrow xRt), aRt \Rightarrow \varphi[x/a]} \qquad \dfrac{\dfrac{\varphi[x/a] \Rightarrow \varphi[x/a] \qquad aRt \Rightarrow aRt}{\varphi[x/a] \leftrightarrow aRt, \varphi[x/a] \Rightarrow aRt}}{\forall x(\varphi(x) \leftrightarrow xRt), \varphi[x/a] \Rightarrow aRt}}{\forall x(\varphi(x) \leftrightarrow xRt) \Rightarrow t = \tau x\varphi(x)}$$

Derivability of the specific rules is straightforward. Notice that from HA as additional axioms we obtain:

(a) $t = \tau x\varphi(x) \Rightarrow \forall x(\varphi(x) \leftrightarrow xRt)$ and
(b) $\forall x(\varphi(x) \leftrightarrow xRt) \Rightarrow t = \tau x\varphi(x)$.

From the premisses of any variant of $(\tau \Rightarrow)$, applying weakening we deduce:

$$(\leftrightarrow\Rightarrow)\ \dfrac{\Gamma \Rightarrow \Delta, bRt, \varphi[x/b] \qquad bRt, \varphi[x/b], \Gamma \Rightarrow \Delta}{(\forall\Rightarrow)\ \dfrac{\varphi[x/b] \leftrightarrow bRt, \Gamma \Rightarrow \Delta}{\forall x(\varphi(x) \leftrightarrow xRt), \Gamma \Rightarrow \Delta}}$$

which, by cut with (a) yields the conclusion of $(\tau \Rightarrow)$. In a similar way we deduce $\Gamma \Rightarrow \Delta, \forall x(\varphi(x) \leftrightarrow xRt)$ from premisses of $(\Rightarrow \tau)$, and by cut with (b) we obtain the conclusion of this rule.

One should note that T-theory is much stronger than S-theory; both central principles EXT and AV are provable (in fact even in the setting of NFFOLI by means of the weaker rules).

$$(\Rightarrow\tau)\ \dfrac{(\forall\Rightarrow)\ \dfrac{(Ref)\ \dfrac{(\tau\Rightarrow)\ \dfrac{aR\tau x\varphi(x) \Rightarrow aR\tau x\varphi(x) \qquad \varphi[x/a], \varphi[x/a] \leftrightarrow \psi[x/a] \Rightarrow \psi[x/a]}{\tau x\varphi(x) = \tau x\varphi(x), \varphi[x/a] \leftrightarrow \psi[x/a], aR\tau x\varphi(x) \Rightarrow \psi[x/a]}}{\varphi[x/a] \leftrightarrow \psi[x/a], aR\tau x\varphi(x) \Rightarrow \psi[x/a]}}{\forall x(\varphi(x) \leftrightarrow \psi(x)), aR\tau x\varphi(x) \Rightarrow \psi[x/a]} \qquad \mathcal{D}}{\forall x(\varphi(x) \leftrightarrow \psi(x)) \Rightarrow \tau x\varphi(x) = \tau x\psi(x)}$$

where the second leaf is directly provable and $\mathcal{D}$ is an analogous proof of $\forall x(\varphi(x) \leftrightarrow \psi(x)), \psi[x/a] \Rightarrow aR\tau x\varphi(x)$.

$$(\Rightarrow\tau)\ \dfrac{(Ref)\ \dfrac{(\tau\Rightarrow)\ \dfrac{aR\tau x\varphi(x) \Rightarrow aR\tau x\varphi(x) \qquad \varphi[x/a] \Rightarrow \varphi[y/a]}{\tau x\varphi(x) = \tau x\varphi(x), aR\tau x\varphi(x) \Rightarrow \varphi[y/a]}}{aR\tau x\varphi(x) \Rightarrow \varphi[y/a]} \qquad \dfrac{\dfrac{\varphi[y/a] \Rightarrow \varphi[x/a] \qquad aR\tau x\varphi(x) \Rightarrow aR\tau x\varphi(x)}{\tau x\varphi(x) = \tau x\varphi(x), \varphi[y/a] \Rightarrow aR\tau x\varphi(x)}}{\varphi[y/a] \Rightarrow aR\tau x\varphi(x)}}{\Rightarrow \tau x\varphi(x) = \tau y\varphi(y)}$$

Note that $\varphi[x/a]$ and $\varphi[y/a]$ are identical since $\varphi(x)$ and $\varphi(y)$ are alphabetic variants.

One may even prove the converse of EXT:

$$(\Rightarrow\forall)\ \dfrac{\dfrac{(\tau\Rightarrow)\ \dfrac{(Ref)\ \dfrac{(\tau\Rightarrow)\ \dfrac{\varphi[x/a] \Rightarrow \varphi[x/a] \qquad aR\tau x\varphi(x) \Rightarrow aR\tau x\varphi(x)}{\tau x\varphi(x) = \tau x\varphi(x), \varphi[x/a] \Rightarrow aR\tau x\varphi(x)}}{\varphi[x/a] \Rightarrow aR\tau x\varphi(x)} \qquad \psi[x/a] \Rightarrow \psi[x/a]}{\tau x\varphi(x) = \tau x\psi(x), \varphi[x/a] \Rightarrow \psi[x/a]} \qquad \mathcal{D}}{\tau x\varphi(x) = \tau x\psi(x) \Rightarrow \varphi[x/a] \leftrightarrow \psi[x/a]}}{\tau x\varphi(x) = \tau x\psi(x) \Rightarrow \forall x(\varphi(x) \leftrightarrow \psi(x))}$$

where $\mathcal{D}$ is a similar proof of $\tau x\varphi(x) = \tau x\psi(x), \psi[x/a] \Rightarrow \varphi[x/a]$.

To realise how strong is this principle on the ground of CFOLI notice that when t is instantiated with $\tau x\varphi(x)$ we obtain:

$$\tau x\varphi(x) = \tau x\varphi(x) \leftrightarrow \forall x(\varphi(x) \leftrightarrow xR\tau x\varphi(x)).$$

which by (unrestricted) reflexivity of $=$ yields:

$$\forall x(\varphi(x) \leftrightarrow xR\tau x\varphi(x)).$$

For several term-forming operators, at least on the ground of CFOLI, it is too strong. For example if we instantiate this principle with iota-operator (where R is $=$) we run into contradiction:

1. $\imath x(Ax \wedge \neg Ax) = \imath x(Ax \wedge \neg Ax) \rightarrow \forall x(Ax \wedge \neg Ax \leftrightarrow x = \imath x(Ax \wedge \neg Ax))$
2. $\imath x(Ax \wedge \neg Ax) = \imath x(Ax \wedge \neg Ax)$
3. $\forall x(Ax \wedge \neg Ax \leftrightarrow x = \imath x(Ax \wedge \neg Ax))$ 1, 2
4. $A(\imath x(Ax \wedge \neg Ax)) \wedge \neg A(\imath x(Ax \wedge \neg Ax)) \leftrightarrow \imath x(Ax \wedge \neg Ax) = \imath x(Ax \wedge \neg Ax))$ 3
5. $A(\imath x(Ax \wedge \neg Ax)) \wedge \neg A(\imath x(Ax \wedge \neg Ax))$ 4, 2

Similarly in the case of set-abstraction operator (where R is $\in$) we obtain just unrestricted axiom of comprehension which immediately leads to Russell's paradox. Hence it is crucial to establish what is R for the specific tfo to decide if Tennant's rules may be safely added to GCI or GPCI. Therefore, we do not attempt here to state T-theory as a general calculus GT. Instead we will consider in the next section the application of his theory to set-abstraction operator, since even in this context one may introduce restrictions which can prevent us against troubles.

4 Application to Set-Abstracts

Several kinds of set theory with set-abstraction operator as primitive can be rather easily developed on the basis of S- or T-theory as formalised in the preceding section. In fact, both Scott [32] and Tennant [33] applied their theories to set-abstract operators but in the context of free logic the unrestricted axiom of comprehension does not lead to Russell's paradox. However we work here in the setting of CFOL so the rules responsible for its derivation must be somehow restricted. For these reasons we decided to examine the possible formalisations of Quine's NF (New Foundations) as developed in [30], where the comprehension axiom is suitably restricted by means of the outer syntactic side condition which is independent of the structure of rules. In fact, NF is not very popular formalisation of set theory due to some peculiarities. However, it has also several advantages which we are not going to discuss here because of the space restrictions[1]. In particular, the syntactic simplicity of NF make it a very convenient theory for proof-theoretic investigations.

Before we focus on sequent calculi for NF let us start with some general preliminaries concerning arbitrary formalisation of set theory. It often goes unnoticed that it may be developed in the language where only $\in$ is a primitive predicate or in the language with $=$ primitive, which is rather more popular choice. In the latter case we assume that we have already some axioms/rules for $=$, so the only specific axiom we need for sets is:

$$ExtAx : \forall xy(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y)$$

since the converse is already provable by LL.

If we start with CFOL (only $\in$ primitive), $=$ may be defined either in the Leibnizian spirit:

$$=^L\text{:}\ \ t = t' := \forall z(t \in z \leftrightarrow t' \in z)$$

[1] See in particular its presentation in [30] and discussion in [8,9].

or in the way Quine prefers:

$=^Q$: $t = t' := \forall z(z \in t \leftrightarrow z \in t')$

The first choice leads to the standard characterisation of = and the axiom *ExtAx* is still required. The second one is different since *ExtAx* is provable but still we cannot obtain the full characterisation of identity. Therefore we must add a special form of LL as an extensionality axiom:

ExtAx': $\forall xyz(x = y \to (x \in z \to y \in z))$

and this is the way Quine proceeded with the development of NF. The second axiom is the axiom of abstraction:

ABS: $\forall x(x \in \{y : \varphi(y)\} \leftrightarrow \varphi[y/x])$

where φ is stratified. Assuming that the only predicate is $\in$ this condition may be defined roughly as follows: it is possible to define a mapping from variables of φ into integers in a way that for each atom we have $i \in i + 1$. In case we admit $=$, a mapping should yield $i = i$. In what follows we will admit both kinds of formulae as atomic, briefly called $\in$-atoms and $=$-atoms.

We will consider two approaches to construction of cut-free sequent calculus for NF. Although the rules $(Ext), (AV)$ will be not primitive but derivable in both, the first one, following closely Quinean formulation, is closer to the general GS, whereas the second starts with Tennant's rules suitably restricted.

4.1 The S-Approach to NF

There is no sense to take the instances of (Ext) and (AV) as primitive rules since it will not save us from addition of most of the specific rules for set-abstraction operators and =. So it is better to follow quite closely the original Quinean axiomatisation of NF. A difference with the latter is connected with the treatment of identity, since we take it as a primitive predicate characterised by rules. However, we do not take the primitive rules of GPCI for identity as primitive but rather provide new rules based on $=^Q$. Hence we take GPC as the basis and add:

$$(\Rightarrow=)\ \frac{a \in t, \Gamma \Rightarrow \Delta, a \in t' \qquad a \in t', \Gamma \Rightarrow \Delta, a \in t}{\Gamma \Rightarrow \Delta, t = t'}$$

$$(=\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, b \in t, b \in t' \qquad b \in t, b \in t', \Gamma \Rightarrow \Delta}{t = t', \Gamma \Rightarrow \Delta}$$

These rules correspond to $=^Q$. Moreover, we add two rules corresponding to the axiom ABS:

$$(Abs \Rightarrow)\ \frac{\varphi[x/t], \Gamma \Rightarrow \Delta}{t \in \{x : \varphi(x)\}, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow Abs)\ \frac{\Gamma \Rightarrow \Delta, \varphi[x/t]}{\Gamma \Rightarrow \Delta, t \in \{x : \varphi(x)\}}$$

with φ stratified.

We omit easy proofs of the equivalence of stated rules with respective axioms: ABS and the object language counterpart of $=^Q$. Proofs of these axioms, as well as derivability of our rules in GPC enriched with axiomatic sequents expressing ABS and $=^Q$ are straightforward and similar to proofs from Theorem 1. Instead we will show that although we have neither (Ext) nor (AV) as primitive rules they are derivable in such a system for stratified φ.

Lemma 2. *Derivability of* (Ext) *and* (AV)

Proof. :

$$\dfrac{\dfrac{\varphi(a), \Gamma \Rightarrow \Delta, \psi(a)}{a \in \{x : \varphi(x)\}, \Gamma \Rightarrow \Delta, a \in \{x : \psi(x)\}}\,(Abs \Rightarrow Abs) \qquad \dfrac{\psi(a), \Gamma \Rightarrow \Delta, \varphi(a)}{a \in \{x : \psi(x)\}, \Gamma \Rightarrow \Delta, a \in \{x : \varphi(x)\}}}{\Gamma \Rightarrow \Delta, \{x : \varphi(x)\} = \{x : \psi(x)\}}\,(\Rightarrow =)$$

The proof of (AV) or alternatively, of $(ExtAV)$ is similar. □

But the rules $(\Rightarrow =)$ and $(= \Rightarrow)$ are not sufficient for obtaining the complete characterisation of identity in NF. In particular they are not sufficient for the case corresponding to the specific instance of LL expressed by the axiom $ExtAx'$ Note that in general we must be able to prove:

1. $t = t', t'' = t' \Rightarrow t = t''$
2. $t = t', t'' \in t \Rightarrow t'' \in t'$
3. $t = t', t \in t'' \Rightarrow t' \in t''$

With case 1 there is no problem; it is derivable by $(\Rightarrow =), (= \Rightarrow)$, similarly as other properties of $=$, including reflexivity and symmetry. The case 2 would be provable by $(= \Rightarrow)$ provided instead of b we are allowed to use any term t''. So this case is problematic and needs reformulation of the rules which in general destroys the subformula property and may be troublesome in proving the cut elimination theorem. The case 3 corresponds exactly to $ExtAx'$ and requires a separate rule which possibly covers also the case 2. To avoid troubles we might follow the general solution introduced for GS and use the rule $(2LL)$ as two-premiss right-sided rule but it does not work since $(Abs \Rightarrow)$ introduces an $\in$-atom as a principal formula in the antecedent. As a result while proving cut elimination we cannot make a reduction of the following cut instance:

$$(2LL)\,\dfrac{\dfrac{\Gamma \Rightarrow \Delta, t = t' \qquad \Gamma \Rightarrow \Delta, t' \in \{x : \varphi\}}{\Gamma \Rightarrow \Delta, t \in \{x : \varphi\}} \qquad \dfrac{\varphi(t), \Pi \Rightarrow \Sigma}{t \in \{x : \varphi\}, \Pi \Rightarrow \Sigma}\,(Abs \Rightarrow)}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\,(Cut)$$

It seems that in the presence of $(Abs \Rightarrow)$ and $(\Rightarrow Abs)$ the only solution is to add a 3-premiss version of LL:

$$(3LL)\ \frac{\Gamma \Rightarrow \Delta, t = t' \qquad \Gamma \Rightarrow \Delta, \varphi(t) \qquad \varphi(t'), \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

where $\varphi(t)$ and $\varphi(t')$ are either $t'' \in t$ and $t'' \in t'$ or $t \in t''$ and $t' \in t''$.

Summing up we obtain a system GSNF which adds to GPC the following rules: $(=\Rightarrow), (\Rightarrow=), (Abs \Rightarrow), (\Rightarrow Abs)$ and $(3LL)$ ((Ref) is derivable).

Theorem 4. *GSNF is an adequate formalisation of NF.*

Moreover the cut elimination theorem can be proved for GSNF in a similar fashion as in [13] where similar solution was provided for sequent calculi for free description theories. Note however that the situation with the subformula property is even worse than in GS (GS') due to the presence of $(3LL)$. Is it possible to obtain a better formalisation of NF by means of Tennant's rules?

4.2 The T-Approach to NF

If we want to apply the approach of Tennant to NF we have = as a primitive predicate not only present in the language but already characterised by specific rules so we start with GPCI and add the following Tennant's-style rules:

$$(\Rightarrow:)\ \frac{\varphi(a), \Gamma \Rightarrow \Delta, a \in t \qquad a \in t, \Gamma \Rightarrow \Delta, \varphi(a)}{\Gamma \Rightarrow \Delta, t = \{x : \varphi(x)\}}$$

$$(:\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi(b) \qquad b \in t, \Gamma \Rightarrow \Delta}{t = \{x : \varphi(x)\}, \Gamma \Rightarrow \Delta}$$

$$(:\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, b \in t \qquad \varphi(b), \Gamma \Rightarrow \Delta}{t = \{x : \varphi(x)\}, \Gamma \Rightarrow \Delta}$$

where a is not in Γ, Δ, φ, t is any term and φ is stratified.

Note that (Ext) and (AV) are derivable which follows from the proofs of EXT and AV presented in Sect. 3.2. As we noticed there, also the axiom ABS is provable, so we do not need special rules $(Abs \Rightarrow), (\Rightarrow Abs)$ too. We do not need to care even about the axiom $ExtAx$ since it is provable:

$$(\Rightarrow\forall)\ \dfrac{(\Rightarrow\to)\ \dfrac{(2LL)\ \dfrac{(\Rightarrow:)\ \dfrac{(\forall\Rightarrow)\ \dfrac{c \in a \leftrightarrow c \in b, c \in a \Rightarrow \Delta, c \in b}{\forall z(z \in a \leftrightarrow z \in b), c \in a \Rightarrow c \in b} \qquad \dfrac{c \in a \leftrightarrow c \in b, c \in b \Rightarrow c \in a}{\forall z(z \in a \leftrightarrow z \in b), c \in b \Rightarrow c \in a}}{\forall z(z \in a \leftrightarrow z \in b) \Rightarrow a = \{x : x \in b\}} \qquad \dfrac{c \in b \Rightarrow c \in b}{\Rightarrow b = \{x : x \in b\}}\ (\Rightarrow:)}{\forall z(z \in a \leftrightarrow z \in b) \Rightarrow a = b}}{\Rightarrow \forall z(z \in a \leftrightarrow z \in b) \to a = b}}{\Rightarrow \forall xy(\forall z(z \in x \leftrightarrow z \in y) \to x = y)}$$

It seems that T-approach is better than S-approach to NF since it is more economical. However, if we think about cut elimination we must consider carefully the problem of primitive rules for identity. Although we first stated that we add the special Tennant's-style rules to GPCI and we used $(2LL)$ in the above proof it seems that we cannot keep $(2LL)$ since in general we face the same problem with cut elimination as in the case of S-system illustrated in Subsect. 4.1. To prove the cut elimination theorem we must again either generally replace $(2LL)$ with $(3LL)$ or to follow the strategy introduced in [17] and separate the rules for LL dealing with special cases of atomic formulae. One possibility is to keep:

$$(2LL')\ \frac{\Gamma \Rightarrow \Delta, t = t' \qquad \Gamma \Rightarrow \Delta, \varphi(t)}{\Gamma \Rightarrow \Delta, \varphi(t')}$$

for φ being $\in$-atom and restrict $(3LL)$ only to $=$-atoms:

$$(3LL')\ \frac{\Gamma \Rightarrow \Delta, t = t' \qquad \Gamma \Rightarrow \Delta, t = t'' \qquad t' = t'', \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

This way we obtain a system GTNF which adds to GPC the rules: $(:\Rightarrow), (\Rightarrow:), (2LL'), (3LL'), (Ref)$. $(2LL')$ deals only with $\in$-atoms and all properties of identity are derivable by (Ref) and $(3LL)$.

Theorem 5. *GTNF is an adequate formalisation of NF.*

The cut elimination theorem is provable for GTNF as well. Unfortunately, the situation with the subformula property is similar to that in the system GSNF from the preceding subsection. However, there are possible some simplifications obtained by reduction of the applications of $(3LL')$ if at least two of t, t', t'' are parameters. Consider the cases with at most one term t complex:

1. $a = b, a = c \Rightarrow b = c$
2. $t = b, t = c \Rightarrow b = c$
3. $a = t, a = b \Rightarrow t = b$
4. $a = b, a = t \Rightarrow b = t$

$(2LL')$ may be modified to cover identities from case 1 and 2:

$$(2LL'')\ \frac{\Gamma \Rightarrow \Delta, t = t' \qquad \Gamma \Rightarrow \Delta, \varphi(t)}{\Gamma \Rightarrow \Delta, \varphi(t')}$$

for $\varphi(t')$ being $\in$-atom or $=$-atom of the form $b = c$ (a third term in the premisses may be complex or a parameter). For cases 3 and 4 we may add the rules:

$$(Tr)\ \frac{\Gamma \Rightarrow \Delta, a = t \qquad t = b, \Gamma \Rightarrow \Delta}{a = b, \Gamma \Rightarrow \Delta}$$

or

$$(E)\ \frac{\Gamma \Rightarrow \Delta, a = t \qquad b = t, \Gamma \Rightarrow \Delta}{a = b, \Gamma \Rightarrow \Delta}$$

Any of them will do the task. For example, if we take (E) we have a direct proof of 4 and the following proof of 3:

$$\dfrac{a = t \Rightarrow a = t \qquad \dfrac{b = t \Rightarrow b = t \qquad \Rightarrow b = b \qquad t = b \Rightarrow t = b}{b = t \Rightarrow t = b}\ (3LL')}{a = t, a = b \Rightarrow t = b}\ (E)$$

As a result we have to keep $(3LL')$ only for all cases where at least two of t, t', t'' are complex terms at the price of adding (Tr) or (E). Let us call such a modified system GTNF'.

5 Conclusion

We have provided a proof theoretic treatment of the general theory of tfos introduced independently by several authors (S-theory), and proposed a modification of a different approach (T-theory) in a way which allows us to compare their relative strength. Moreover, we examined the ways in which both approaches may be extended to cover set theory NF of Quine. All obtained sequent systems satisfy the cut elimination theorem, but do not satisfy the subformula property. Hence, in the case of the systems for NF, we cannot obtain a syntactical consistency proof on the basis of the cut elimination theorem, because of the rules like $(3LL)$. Still these systems, in particular a system GTNF described in the last subsection, allow us to keep a stricter control over the construction of proof.

The natural next step of this research is connected with the application of, possibly modified, systems GS, GS', or (suitably restricted) rules of Tennant's approach, to other kinds of term-forming operators, and careful examination of their specific features.

Eventually it is also interesting to investigate if the obtained systems allow us to prove other desirable properties in constructive way. One of such important points is the interpolation theorem. Since it was proved semantically for the general S-theory in [4], it is an important task to find a constructive proof as well. However, the method of split-sequents due to Maehara, which is usually applied in the setting of sequent calculi, fails for the presented systems since it does not work with rules like $(a \Rightarrow)$. The problem is connected with the fact that the complex term occuring in the active formula in the premiss may contain some predicates which do not occur in the rest of the respective division of a split-sequent but occur in the interpolant (and of course in the other division of a split-sequent). In this case the interpolant of the premiss fails to be an interpolant of the conclusion, where the active formula is deleted. Only the weaker form of interpolation can be proved in which we require that interpolants have only parameters (but not predicates) common to both divisions of the split-sequent. It is an open problem if such difficulties can be overcome.

References

1. Corcoran, J., Herring, J.: Notes on a semantical analysis of variable-binding term operators. Logique et Anal. (N.S.) **55**, 644–657 (1971)
2. Corcoran, J., Hatcher, W.R., Herring, J.: Variable-binding term operators, Zeitschr. f. math. Logik u. Grund. d. Math. **18**, 177–182 (1972)
3. Da Costa, N.C.A.: Review of Corcoran, Hatcher and Herring 1972. Zentralblatt f. Math. **257**, 8–9 (1973)
4. Da Costa, N.C.A.: A model-theoretical approach to variable-binding term operators. In: Mathematical Logic in Latin America, pp. 133–162, North-Holland (1980)
5. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Synthese Library, vol. 277, Springer, Dordrecht (1998). https://doi.org/10.1007/978-94-011-5292-1
6. Hailperin, T.: Remarks on identity and description in firts-order axiom systems. J. Symb. Log. **19**(1), 14–20 (1954)
7. Hatcher, W., S.: Foundations of Mathematics. Saunders. Philadelphia (1968)
8. Hatcher, W.S.: The logical foundations of Mathematics. Pergamon Press (1982)
9. Holmes, M.R.: The set-theoretical program of Quine succeeded, but nobody noticed. Modern Logic. **4**(1), 1–47 (1994)
10. Indrzejczak, A.: Cut-free modal theory of definite descriptions. In: Bezhanishvili, N., D'Agostino, M., Studer, T. (eds.) Advances in Modal Logic 12, pp. 359–378. College Publications, Rickmansworth (2018)
11. Indrzejczak, A.: Fregean description theory in proof-theoretic setting. Logic Log. Philos. **28**(1), 137–155 (2019)
12. Indrzejczak, A.: Existence, definedness and definite descriptions in hybrid modal logic. In: Olivetti, N., Verbrugge, R., Negri, S., Sandu, G. (eds.) Advances in Modal Logic 13, pp. 349–368. College Publications, Rickmansworth (2020)
13. Indrzejczak, A.: Free definite description theory - sequent calculi and cut elimination. Logic Log. Philos. **29**(4), 505–539 (2020)
14. Indrzejczak, A.: Sequents and Trees. An Introduction to the Theory and Applications of Propositional Sequent Calculi. Birkhäuser (2021)
15. Indrzejczak, A.: Free Logics are cut-free. Stud. Logica. **109**, 859–886 (2021)
16. Indrzejczak, A.: A novel approach to equality. Synthese. **199**, 4749–4774 (2021)
17. Indrzejczak, A.: Russellian definite description theory–a proof-theoretic approach. Rev. Symbolic Logic. **16**(2), 624–649 (2023)
18. Indrzejczak, A., Zawidzki, M.: Tableaux for Free Logics with Descriptions. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 56–73. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_4
19. Indrzejczak, A., Zawidzki, M.: When Iota meets Lambda. Synthese **201**(2), 1–33 (2023)
20. Kalish, D., Montague, R., Mar, G.: Logic. Techniques of Formal Reasoning (2nd ed.). Oxford University Press, New York, Oxford (1980)
21. Kürbis, N.: A binary quantifier for definite descriptions in intuitionist negative free logic: natural deduction and normalization. Bull. Section Logic **48**(2), 81–97 (2019)
22. Kürbis, N.: Two treatments of definite descriptions in intuitionist negative free logic. Bull. Sect. Logic **48**(4), 299–317 (2019)
23. Kürbis, N.: Definite descriptions in intuitionist positive free logic. Logic Log. Philos. **30**(2), 327–358 (2021)
24. Kürbis, N.: Proof-theory and semantics for a theory of definite descriptions. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 95–111. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_6

25. Kürbis, N.: A binary quantifier for definite descriptions for cut free free logics. Stud. Logica. **110**(1), 219–239 (2022)
26. Lambert, K.: Free Logics. Character, and Some Applications Thereof. Academia Verlag. Sankt Augustin, Their Foundations (1997)
27. Negri, S., von Plato, J.: Structural Proof Theory. Cambridge University Press, Cambridge (2001)
28. Orlandelli, E.: Labelled calculi for quantified modal logics with definite descriptions. J. Log. Comput. **31**(3), 923–946 (2021)
29. Quine, W.V.: New foundations for mathematical logic. Amer. Math. Monthly **44**, 70–80 (1937)
30. Rosser, J.B.: Logic for Mathematicians. McGraw-Hill (1953)
31. Russell, B.: On Denoting. Mind XIV 479–494 (1905)
32. Scott, D.: Existence and description in formal logic. In: Shoenman, R. (ed.) Bertrand Russell, Philosopher of the Century, pp. 181–200. Georg Allen and Unwin Ltd., London (1967)
33. Tennant, N.: Natural Logic. Edinburgh (1978)
34. Tennant, N.: Anti-Realism and Logic. Oxford (1987)
35. Tennant, N.: A general theory of abstraction operators. Philosoph. Quat. **54**(214), 105–133 (2004)
36. Tennant, N.: The Logic of Number. Oxford (2022)
37. Whitehead, A.N., Russell, B.: Principia Mathematica, vol. I. Cambridge University Press, Cambridge (1910)

Theorem Proving

Lemmas: Generation, Selection, Application

Michael Rawson[1], Christoph Wernhard[2](✉), Zsolt Zombori[3,5], and Wolfgang Bibel[4]

[1] TU Wien, Vienna, Austria
michael@rawsons.uk
[2] University of Potsdam, Potsdam, Germany
info@christophwernhard.com
[3] Alfréd Rényi Institute of Mathematics, Budapest, Hungary
zombori@renyi.hu
[4] Technical University Darmstadt, Darmstadt, Germany
bibel@gmx.net
[5] Eötvös Loránd University, Budapest, Hungary

Abstract. Noting that lemmas are a key feature of mathematics, we engage in an investigation of the role of lemmas in automated theorem proving. The paper describes experiments with a combined system involving learning technology that generates useful lemmas for automated theorem provers, demonstrating improvement for several representative systems and solving a hard problem not solved by any system for twenty years. By focusing on condensed detachment problems we simplify the setting considerably, allowing us to get at the essence of lemmas and their role in proof search.

1 Introduction

Mathematics is built in a carefully structured way, with many disciplines and subdisciplines. These are characterized by concepts, definitions, axioms, theorems, lemmas, and so forth. There is no doubt that this inherent structure of mathematics is part of the discipline's long-lasting success.

Research into Automated Theorem Proving (ATP) to date has taken little notice of the information provided by this structure. Even state-of-the-art ATP systems ingest a conjecture together with pertinent definitions and axioms in a way completely agnostic to their place in the mathematical structure. A comparatively small but nevertheless important part of the structure of mathematics is

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495, by the North-German Supercomputing Alliance (HLRN), by the ERC grant CoG ARTIST 101002685, by the Hungarian National Excellence Grant 2018-1.2.1-NKP-00008 and the Hungarian Artificial Intelligence National Laboratory Program (RRF-2.3.1-21-2022-00004).

the identification and application of *lemmas*. It is this aspect which is the focus of the work presented here.

The purpose of lemmas in mathematics is at least threefold. First, and perhaps most importantly, lemmas support the search for proofs of assertions. If some lemma applies to a given problem, a proof may be found more easily. Second, it is often the case that a lemma may be applied more than once. If this happens, its use will shorten the length of the overall proof since the proof of the lemma need only be carried out once, not repeatedly for every application. Third, the structuring effect of proofs by the use of lemmas is an important feature for human comprehension of proofs. In our work we are motivated primarily by the first two of these three aspects.

These considerations give rise to the crucial question: how can we find useful lemmas for proving a given problem? Here we mean useful in the sense of the two aforementioned aspects: lemmas should be applicable to the problem at hand, preferably many times. In full generality this is a difficult question indeed, which will require much further research. In this first step we restrict the question to a narrow range of problems, known in literature as *condensed detachment* (CD) problems [41]. Proofs of CD problems can be represented in a simple and accessible form as *proof structure terms*, enabling structure enumeration to enhance proof search and lemma maintenance, as well as feature extraction for learning. Our investigation thus focuses on the question of how ATP performance may be improved for CD problems by the generation and selection of useful lemmas before search begins.

CD problems are of the form "axiom(s) and *Det* imply a goal" where *Det* represents the well-known modus ponens rule, or *condensed detachment*. They have a single unary predicate. A typical application is the investigation of an axiomatization of some propositional logic, whose connectives are then represented by function symbols. In order to support this study experimentally, we have built a combined system for dealing with these problems. It features SGCD [74] as prover and lemma generator along with a learning module based on either an easily-interpreted linear model over hand-engineered features, or a graph neural network supporting end-to-end learning directly from lemmas.

Our work results in a number of inter-related particular contributions:

1. Incorporation of proof structure terms into ATP with Machine Learning (ML). Consideration of features of the proof structure terms, explicitly in linear-model ML or implicitly in a neural ML model. A novel ATP/ML dataflow that is centered around proof structure terms.
2. Experimentally validated general insights into the use of learned lemmas for provers of different paradigms, with different ways to incorporate lemmas, and based on two alternate ML models. At the same time pushing forward the state of the art on proving CD problems. Insights include: SGCD is competitive with leading first-order provers; Learned lemmas significantly extend the set of problems provable by the leading first-order prover Vampire; Provers without internal lemma maintenance, such as Connection Method (CM) [6–8] systems, are drastically improved; Vampire and SGCD are able to handle a

few hundreds of supplied lemmas; Learning based on manual features and on automatic feature extraction perform similarly.
3. An automatic proof of the Meredith single axiom theorem LCL073-1, which has persisted in the TPTP rated 1.00 since 1997. The first and only system to succeed was OTTER [39], after intensive massaging by Wos [84]. It was proven by SGCD in a novel systematic way.
4. An implemented framework with the new techniques for generation, selection and application of lemmas.

Structure of the Paper. Section 2 presents condensed detachment and its embedding into the CM by way of so-called *D-terms*, as well as background material on lemmas and machine learning in ATP. Section 3 introduces a method for generating and selecting useful lemmas and presents experimental results with it, leading up to the proof of LCL073-1 in Sect. 4. We conclude with a summary and outlook for further work in this area in Sect. 5.

Supplementary material is provided in the appendix of the preprint version [54]. All experiments are fully reproducible and the artifacts are available at https://github.com/zsoltzombori/lemma, commit df2faaa. We use CD Tools [74] and PIE [71,72], implemented in SWI-Prolog [77], for reasoning tasks and PyTorch [47] for learning.

2 Background and Related Work

In a very general sense, lemmas in ATP factorize duplication. This may be between different proofs that make use of the same lemma, or within a single proof where a lemma is used multiple times. It may not even be a particular formula that is shared, but a *pattern*, such as a *resonator* [81]. In the presence of machine learning, we may think of even more abstract entities that are factorized: the *principles* by which proofs are written, repeated in different proofs or contexts.

Depending on the proving method, lemmas in ATP play different roles. Provers based on *saturation*, typically resolution/superposition (RS) systems [3], inherently operate by generating lemmas: a resolvent is itself a lemma derived from its parents. Nevertheless, one may ask for more meaningful lemmas than the clauses of the proof. This is addressed with *cut introduction* [14,20,78], which studies methods to obtain complex lemmas from resolution proofs. Such lemmas provide insight about the high-level structure of proofs, extract interesting concepts and support research into the correspondence between natural mathematical notions and possible proof compressions. Other approaches to interesting theorems or lemmas are described for example in [52,65].

Another question concerning lemmas and ATP systems is whether performance can be improved by supplementing the input with lemmas. This is particularly applicable if lemmas are obtained with methods that are *different* from those of the prover. Otherwise, it may have obtained these by itself.[1] As we will see, leading ATP systems such as Vampire and E [59] can indeed be improved in this way. Different *methods* does not necessarily mean different *systems*: it is possible to use different configurations of the same system for lemma generation and proving, as well as for intermediate operations. This was the workflow used by Larry Wos to prove the challenge problem LCL073-1 with OTTER [84]. Our SGCD system also supports this, which played a major role in its ability to prove the aforementioned challenge problem.

Lemmas play a quite different role for a family of provers which we call *CM-CT* for *Connection Method/Clausal Tableaux*, exemplified by PTTP [61], SETHEO [33], and leanCoP [45,46]. Underlying conceptual models are model elimination [35], clausal tableaux [31] and the CM. They enumerate proof structures while propagating variable bindings initialized by the goal through unification, and hence proceed in an inherently goal-driven way. While they are good at problems that benefit from goal direction, in general they are much weaker than RS provers and have not been among the top provers at *CASC* for about two decades. This is attributed to the fact that they do not re-use the proof of one subgoal as the solution of another: they do not use lemmas *internally*.

The lack of lemmas was identified early as a weakness of CM-CT [15], so there have been various proposed remedies [2,15,17,19,32,45,60,62]. Despite some insight and success, this did not yet elevate CM-CT to the level of the best RS systems. Nevertheless, the expectation remains that CM-CT provers would benefit from supplying lemmas as additional input. Hence, we included two CM-CT systems in our experiments, leanCoP and CMProver [12,71,72] and show that the expectation is greatly confirmed. Two other systems considered here, SGCD and CCS [73], can be viewed as CM-CT systems extended to support specific forms of lemma generation and application.

Lemmas can be maintained within the prover as an inherent part of the method, as in saturation. They may also be created and applied by different systems, or different instances of the same system [13,55]. Larry Wos calls this *lemma adjunction* [83]. Lemmas created by one system are passed to a second system in two principal ways. First, they can be passed as *additional axioms*, in the hope that the second system finds a shorter proof in the wider but shallower search space. Second, external lemmas can be used to *replace search*. The second system then starts with the given lemmas as if they were the cached result of its previous computation. Moreover, the provided lemmas can be restricted in advance by heuristic methods, such as by a machine-learned model. SGCD supports this *replacing* lemma incorporation. The basic distinction between augmenting and replacing search with lemmas was already observed by Owen L. Astrachan and Mark E. Stickel [2] in the context of improving CM-CT provers.

[1] We note here that in some cases systems *cannot* generate certain lemmas because of e.g. ordering restrictions.

2.1 Machine Learning for ATP

The past decade has seen numerous attempts to leverage machine learning in the automated theorem proving effort. Early systems mostly focused on premise selection, e.g. [1,68,70], aiming to reduce the number of axioms supplied as input to the prover, or on selection of heuristics, e.g. [11]. Other works provide internal guidance directly at the level of inferences during search, e.g. [18,24,25,27,34,53,85]. The emergence of generative language models has also led to some initial attempts at directly generating next proof steps, e.g. [48,49,67], moving the emphasis away from search.

In contrast to these lines of work, our focus is on learning the utility of lemmas. Close to our aims is [26,28], trying to identify globally useful lemmas in a collection of millions of proofs in HOL Light. Besides differences in the formal system, what distinguishes our work is that we learn a much more focused model: we put great emphasis on evaluating lemmas in the context of a particular goal and axiom set; in fact, our entire system was designed around the question whether a given lemma is moving the goal closer to the axioms. We argue that the D-term representation of all involved components (goal, lemma, axioms, proof) makes our framework particularly suitable for the lemma selection task.

We employ an iterative improvement approach first used in MaLARea [68]: in each iteration, we run proof search guided by a learned model, extract training data from proving attempts, and fit a new model to the new data. These steps can be repeated profitably until performance saturates.

2.2 Condensed Detachment: Proofs as Terms

Condensed detachment (CD) was developed in the mid-1950s by Carew A. Meredith as an evolution of *substitution and detachment* [30,43,50,51]. Reasoning steps are by *detachment*, or modus ponens, under implicit substitution by most general unifiers. Its primary application is the investigation of axiomatizations of propositional logics at a first-order meta-level. CD also provides a technical approach to the Curry-Howard correspondence, "formulas as types" [22,23] and is considered in witness theory [57]. Many early successes in ATP were on CD problems [40,66], but success was also found in the reverse direction. Refinements of the OTTER prover in the 1990s, some of which have found their ways into modern RS provers, were originally conceived and explored in the setting of CD [16,40,69,79–82,84].

From a first-order ATP perspective, a CD problem consists of *axioms*, i.e. positive unit clauses; a *goal theorem*, i.e. a single negative ground unit clause representing a universally-quantified atomic goal theorem after Skolemization; and the following ternary Horn clause that models detachment.

$$\mathit{Det} \overset{\text{def}}{=} \mathsf{P}(\mathsf{i}(x, y)) \land \mathsf{P}(x) \rightarrow \mathsf{P}(y).$$

The premises of *Det* are called the *major* and *minor* premise, respectively. All atoms in the problem have the same predicate P, which is unary and stands for something like *provable*. The formulas of the investigated propositional logic are expressed as terms, where the binary function symbol i stands for *implies*.

CD may be seen as an *inference rule*. From an ATP perspective, a *CD inference step* can be described as a hyperresolution from *Det* and two positive unit clauses to a third positive unit clause. A *CD proof* is a proof of a CD problem constructed with the CD inference rule. CD proofs can be contrasted with other types of proof, such as a proof with binary resolution steps yielding non-unit clauses. Prover9 [38] chooses positive hyperresolution by default as its only inference rule for CD problems and thus produces CD proofs for these.

It is, however, another aspect of CD that makes it of particular interest for developing new ATP methods, which only recently came to our attention in the ATP context [75]: the structure of CD proofs can be represented in a very simple and convenient way as full binary trees, or as terms. In ATP we find this aspect in the CM, where the proof structure as a whole is in focus, in contrast to extending a set of formulas by deduction [9]. This view of CD is made precise and elaborated upon in [76], on which the subsequent informal presentation is based. We call the structure representations of CD proofs *D-terms*. A D-term is a term recursively built from numeral constants and the binary function symbol D whose arguments are D-terms. In other words, it is a full binary tree where the leaf nodes are labeled with constants. Four examples of D-terms are

$$1, \quad 2, \quad \mathsf{D}(1,1), \quad \mathsf{D}(\mathsf{D}(2,1),\mathsf{D}(1,\mathsf{D}(2,1))).$$

A D-term represents the structure of a proof. A proof in full is represented by a D-term together with a mapping of constant D-terms to axioms. Conversion between CD proofs and D-terms is straightforward: the use of an axiom corresponds to a constant D-term, while an inference step corresponds to a D-term $\mathsf{D}(d_1, d_2)$ where d_1 is the D-term that proves the major premise and d_2 the minor.

Through first-order unification, constrained by axioms for the leaf nodes and the requirements of *Det* for inner nodes, it is possible to obtain a most general formula proven by a D-term [76]. We call it the *most general theorem* (MGT) of the D-term with respect to the axioms, unique up to renaming of variables. For a given axiom map, not all D-terms necessarily have an MGT: if unification fails, we say the D-term has no MGT. It is also possible that different D-terms have the same MGT, or that the MGT of one is subsumed by the MGT of another. A D-term is a proof of the problem if its MGT subsumes the goal theorem.

As an example, let the constant D-term 1 be mapped to $\mathsf{P}(\mathsf{i}(x,\mathsf{i}(x,x)))$, known as *Mingle* [66]. Then, the MGT of the D-term 1 is just this axiom. The MGT of the D-term $\mathsf{D}(1,1)$ is $\mathsf{P}(\mathsf{i}(\mathsf{i}(x,\mathsf{i}(x,x)),\mathsf{i}(x,\mathsf{i}(x,x))))$, that is, after renaming of variables, $\mathsf{P}(y)\sigma$ where σ is the most general unifier of the set of pairs $\{\{\mathsf{P}(\mathsf{i}(x,y)),\ \mathsf{P}(\mathsf{i}(x',\mathsf{i}(x',x')))\},\ \{\mathsf{P}(x),\ \mathsf{P}(\mathsf{i}(x'',\mathsf{i}(x'',x'')))\}\}$.

D-terms, as full binary trees, facilitate characterizing and investigating structural properties of proofs. While, for a variety of reasons, it is far from obvious how to measure the size of proofs obtained from ATP systems in general, for D-terms there are at least three straightforward size measures:

– The *tree size* of a D-term is the number of its inner nodes.

– The *height* of a D-term is the length of the longest root-leaf path.
– The *compacted size* of a D-term is the number of distinct compound subterms, or, in other words, the number of inner nodes of its minimal DAG.

Alternative names in the literature are *length* for compacted size, *level* for height and *CDcount* [69] for tree size. The D-term $\mathsf{D}(\mathsf{D}(1, \mathsf{D}(1, 1)), \mathsf{D}(\mathsf{D}(1, 1), 1))$, for example, has tree size 5, compacted size 4 and height 3. *Factor equations* provide a compact way of writing D-terms: distinct subproofs with multiple incoming edges in the DAG receive numeric labels, by which they are referenced. The D-term $\mathsf{D}(\mathsf{D}(1, 1), \mathsf{D}(\mathsf{D}(1, \mathsf{D}(1, 1)), \mathsf{D}(1, \mathsf{D}(1, 1))))$, for example, can be written as $2 = \mathsf{D}(1, 1),\ 3 = \mathsf{D}(1, 2),\ 4 = \mathsf{D}(2, \mathsf{D}(3, 3))$.

CD problems have core characteristics of first-order ATP problems: first-order variables, at least one binary function symbol and cyclic predicate dependency. But they are restricted: positive unit clauses, one negative ground clause, and one ternary Horn clause. Equality is not explicitly considered. The generalization of CD to arbitrary Horn problems is, however, not difficult [73].

2.3 Condensed Detachment for ATP and Lemmas

From an ATP point of view, D-terms provide access to proofs as a whole. This exposes properties of proofs that are not merely local to an inference step, but spread across the whole proof. It suggests a shift in the role of the calculus from providing a recipe for building the structure towards an inductive structure *specification*. Moreover, D-terms as objects provide insight into *all* proofs: for example, growth rates of the number of binary trees for tree size, height and compacted size are well-known with entries in *The On-Line Encyclopedia of Integer Sequences* [44] and provide upper bounds for the number of proofs [76]. A practical consequence for ATP is the justification of proof structure enumeration techniques where each structure appears at most once.

CD proofs suggest and allow for a specific form of lemmas, which we call *unit* or *subtree* lemmas, reflecting two views on them. As formulas, they are positive unit clauses, which can be re-used in different CD inference steps. In the structural view, they are subterms, or subtrees, of the overall D-term. If they occur multiply there, they are factored in the minimal DAG of the overall D-term. The views are linked in that the formula of a lemma is the MGT of its D-term. The *compacted size* measure specified above takes into account the compression achievable by unit/subtree lemmas. From the perspective of proof structure compression methods, unit/subtree lemmas have the property that the compression target is unique, because each tree is represented by a unique minimal DAG. CM-CT provers do not support such lemmas, which is the main reason for their notorious weakness on CD problems.

2.4 SGCD—Structure Generating Theorem Proving

SGCD *(Structure Generating Theorem Proving for Condensed Detachment)* [74] is the central system used in our experiments as prover as well as lemma generator. It realizes an approach to first-order theorem proving combining techniques known from the CM and RS that was not fully recognized before. It generalizes (for CD problems) bottom-up preprocessing for and with CM-CT provers [60] and hypertableaux [4]. SGCD works by enumeration of proof structures together with unification of associated formulas, which is also the core method of the CM-CT provers. Structures for which unification fails are excluded. Each structure appears at most once in the enumeration.

Let the proof structures be D-terms. Partition the set of all D-terms according to some *level* such that those in a lower level are strict subterms of those in a higher level. Tree size or height are examples of such a level. Let

`enum_dterm_mgt_pairs(`*+Level*`, `*?DTerm*`, `*?Formula*`)`

be a Prolog[2] predicate enumerating D-terms and corresponding MGTs at a certain level, with respect to given axioms that do not explicitly appear as parameter. We say that the predicate generates these pairs in an *axiom-driven* way. If the predicate is invoked with the formula argument instantiated by a ground formula, it enumerates D-terms that prove the formula at the specified level. The predicate is then used *goal-driven*, like a CM-CT prover. Invoking it for increasing level values realizes iterative deepening. There are further instantiation possibilities: if only the D-term is instantiated and the level is that of the D-term, its MGT is computed. If both D-term and formula are instantiated, the predicate acts as verifier.

The implementation includes several *generators*, concrete variants of the `enum_dterm_mgt_pairs` predicate for specific level characterizations. SGCD maintains a cache of $\langle \mathit{level}, \textit{D-term}, \mathit{formula} \rangle$ triples used to obtain solutions for subproblems in levels below the calling level. This cache is highly configurable. In particular, the number of entries can be limited, where only the best triples according to specified criteria are kept. Typical criteria are height or size of the formula, a heuristic shared with RS provers. Subsumed entries can be deleted, another feature in common with RS. Novel criteria are also supported, some of which relate the formula to the goal. Most criteria are based on the formula component of the triples, the MGT. Due to rigid variables [21], MGTs are not usually available in CM-CT provers [76] and cannot be used as a basis for heuristics.

When lemmas are provided to SGCD, they are used to initialize the cache, replacing search at levels lower than the calling level.[3] SGCD further maintains a set of *abandoned* $\langle \mathit{level}, \textit{D-term}, \mathit{formula} \rangle$ triples, those that are generated but do not qualify for entering the cache or were removed from the cache. These are kept as a source for heuristic evaluation of other triples and for lemma generation.

For theorem proving, SGCD proceeds as shown in Fig. 1. Input parameter g is the goal formula, while parameters *maxLevel* and *preAddMaxLevel* are configurable. `enum_dterm_mgt_pairs` represents a particular generator that is also configurable. It enumerates argument bindings nondeterministically: if it succeeds in the inner loop, an exception returns the D-term d. C is the

[2] Prolog serves here as a suitable specification language.

[3] Replacement can be subject to heuristic restrictions.

cache. The procedure merge_news_into_cache(N, C) merges newly generated $\langle \mathit{level}, \mathit{D\text{-}term}, \mathit{formula} \rangle$ triples N into the cache C. If *maxLevel* is configured as 0, the method proceeds in purely goal-driven mode with the inner loop performing iterative deepening on the level m. Similarity to CM-CT provers can be shown empirically by comparing the sets of solved TPTP problems [74]. Generally successful configurations of *preAddMaxLevel* typically have values 0–3.

```
$C := \emptyset$;
for $l := 0$ to $maxLevel$ do
    for $m := l$ to $l + preAddMaxLevel$ do
        enum_dterm_mgt_pairs($m, d, g$);
        throw proof_found($d$)
    $N := \{\langle l, d, f \rangle \mid$ enum_dterm_mgt_pairs($l, d, f$)$\}$;
    if $N = \emptyset$ then throw exhausted;
    $C :=$ merge_news_into_cache($N, C$)
```

Fig. 1. The nested loops of the SGCD theorem proving method.

3 Improving a Prover via Learned Lemma Selection

We employ machine learning to identify lemmas that can enhance proof search. Unlike the standard supervised scenario in which we learn from some training problems and evaluate performance on separate test problems, we take a reinforcement learning approach of self-improvement that has already been successfully applied in several theorem proving projects since [68]. In this approach, we perform proof search with a *base prover* on our entire problem set and learn from the proof attempts.[4] The learning-assisted prover is evaluated again in the problem set to see if it can find more or different problems. If there is improvement, the process can be repeated until performance saturates. In a bit more detail, our system has the following components.

1. **Base Prover**: Performs proof search and its main role is to provide training data to the utility model.
2. **Utility Model**: The model takes ⟨*conjecture*, *lemma*, *axioms*⟩ triples and outputs a utility score, i.e., some measure of how useful the lemma is for proving the conjecture from the axioms. The utility model is trained from the D-terms emitted by the base prover.
3. **Lemma Generator**: Produces a large set of candidate lemmas for each problem separately. All candidates are derivable from the axioms.
4. **Evaluated Prover**: For each problem, we evaluate the candidate sets with the utility model and select the best ones. These lemmas are provided to the evaluated prover which performs proof search on the problem set. The evaluated prover can be identical to or different from the base prover.

[4] We currently only learn from successful proof attempts and sketch an extension to learning from failure.

Base Prover. Any prover that emits proofs as D-terms is suitable as a base prover. Given a D-term proof tree P of some formula C from axiom set As, any connected subgraph S of P can be considered as the proof of a lemma L. If S is a full tree, it proves a unit lemma, which is the formula associated with its root. Otherwise, it proves a Horn clause, whose head is the root formula of S and whose body corresponds to the open leaves of S. We currently focus on unit lemmas and leave more general subgraphs for future work. To approximate the utility of lemma L for proving C from As, there are several easy-to-compute logical candidates, such as the reduction in tree size, tree height or compressed size. A more refined measure is obtained if we reprove C with the lemma L added to the axioms As and observe how the number of inference steps changes.[5] This is slower to compute, but takes into account the particularities of the base prover, hence provides more focused guidance. In our experiments, we find that the best performance is obtained by reproving and then computing utility U as the inference step reduction normalized into $[-1, 1]$, where -1 means that the problem could not be solved within the original inference limit and 1 is assigned to the lemma that yields the greatest speedup. We end up with tuples $\langle C, As, L, U \rangle$ to learn from.

Utility Model Training. We experiment with gradient-descent optimization for two classes of functions: linear models and graph neural networks (GNNs). Our linear model is based on 51 manually-identified features, some of them novel, described in [54, App. A]. For each feature f_i there is an associated weight parameter w_i to produce the final predicted utility

$$U(\boldsymbol{f}; \boldsymbol{w}) = \sum_i f_i w_i$$

The second, more involved model is a GNN. Describing this model is beyond the scope of this paper: see e.g. [58] for a gentle introduction. What is crucial for our purposes is that no manual feature extraction is involved: a specialized neural network processes the D-terms of involved formulas directly and learns to extract useful features during optimization. As input, the model is given a graph, losslessly encoding D-terms of the lemma to be evaluated, the conjecture and the axioms. The precise network architecture is provided in [54, App. B].

Candidate Lemma Generation. Candidate lemmas are generated separately for each problem via the structure enumeration mechanism of SGCD, as explained in Fig. 1. The goal g is provided and *preAddMaxLevel* is set to 0, making SGCD proceed axiom-driven, generating lemmas level by level. However, it does intersperse the goal-driven inner loop, which is only trying to prove the goal on the level directly above the last cached level. SGCD may terminate with

[5] The number of inferences is a measure provided by the Prolog engine and is not identical to the number of steps in the FOL calculus.

a proof, in which case further lemma generation is pointless. Otherwise it terminates after *maxLevel* is reached, generation of new levels is exhausted, or a time limit is reached. We then use the cache C and the abandoned triples as the generated output lemmas. Furthermore, there are many ways to configure SGCD. We obtained the best results generating by tree size and by PSP-level (explained below), combined with known good heuristic restrictions. In particular we restrict the size of the lemma formulas to the maximum of the size of the axioms and the goal, multiplied by some factor (usually 2–5). We also restrict the number of elements in the cache, typically to 1,000. The lemmas are sorted by formula size measures, smaller preferred, to determine which are retained in the cache.

Proof structure generation by PSP-level is a novel technique introduced in [74,76], based on an observation by Łukasiewicz and Meredith. In a detachment step, often the D-term that proves one premise is a subterm of the D-term that proves the other. We turn this relationship into a proof structure enumeration method: structures in level $n + 1$ are D-terms where one argument D-term is at level n and the other argument is a subterm of that D-term. The method is incomplete, but combines features of DAG enumeration while being compatible with a simple global lemma maintenance as realized with SGCD's cache [76].

Table 1. Features of the considered provers: whether their proofs are available as D-terms (possibly after some conversion), whether they were used with *replacing* lemma incorporation (Sect. 2), whether they operate goal-driven, and the underlying method.

	SGCD	Prover9	CMProver	leanCoP	CCS-Vanilla	Vampire	E
D-terms	●	●	●	–	●	–	–
Replacing lemmas	●	–	–	–	●	–	–
Goal-driven	●/–	–	●	●	●	–	–
CM-CT	–	–	●	●	–	–	–
RS	–	●	–	–	–	●	●

Evaluated Prover. For each problem, we evaluate the candidate set with the utility model and select k lemmas with the highest predicted utility, where k is a hyperparameter. The evaluated prover then tries to solve the problems with the help of the selected lemmas. The lemmas can either be treated as additional axioms—applicable to any prover—or have a specialized treatment if the prover provides for it: in particular, SGCD and CCS-Vanilla use the lemmas to replace inner lemma enumeration.[6] The evaluated prover can be any prover, since there is no specialized requirement to handle lemmas as new axioms. If, however, it

[6] Before the obtained input lemmas are passed to a prover we supplement them with the lemmas for all their subproofs, i.e. we close the set of D-terms under the subterm relationship. This proved beneficial in experiments (see, e.g., [54, App. D]). An alternative would be to perform this closure on all generated lemmas before selection.

is the base prover—or any other system that emits proofs as D-terms, then the learning procedure can be iterated as long as there are new problems solved.

3.1 Learning-Based Experiments

We experiment with a total of 312 CD problems, including all 196 pure CD problems from TPTP 8.1.2 [64], enriched with single-axiom versions of all the problems to which a technique by Tarski [37], as specified by Rezuş [56], was applicable. We test several representative ATP systems, including state-of-the-art systems for both general first-order reasoning and for CD problems.

Table 1 gives an overview of the considered provers. CCS-Vanilla is CCS [73] in a restricted configuration to find only those CD proofs with minimal compacted size, identifying problems that can clearly be solved with exhaustive search. It operates goal-driven, like the CM-CT provers, but by enumerating DAGs instead of trees through a local lemma maintenance mechanism. Vampire and E represent the state of the art of first-order ATP. Provers that produce D-terms as proofs (SGCD, Prover9, CMProver, CCS) can serve as base provers. We always rely on SGCD for lemma candidate generation. All provers are recent public versions: Vampire 4.5.1, E 2.6, leanCoP 2.1. We provide results in terms of *time* limits, although for the Prolog provers SGCD, CMProver and CCS-Vanilla we used a roughly-equivalent inference limit to avoid fluctuations due to server workload.

Improving the Base Prover. In our first experiment, we evaluate base provers after learning from their own proof attempts. The provers are given $k = 200$ best lemmas according to the linear utility model. Table 2[7] shows problems solved by four base provers without lemmas (*Base* case) and with two iterations of learning. The *Total* row gives the number of theorems proved by any of the three iterations shown. The stronger the base model, the harder it is to improve. CMProver and CCS-Vanilla are purely goal-driven and benefit greatly, reaching over 37% improvement for larger time limits. SGCD and Prover9 improve over 5% for shorter time limits, but this effect gradually vanishes as the time limit is increased.

Table 2. Number of problems solved over 2 iterations of training a linear model.

	SGCD				Prover9				CMProver				CCS-Vanilla			
Time	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m
Base	266	275	285	285	240	252	259	262	82	85	94	103	81	88	99	105
Iter 1	280	282	284	281	250	254	262	257	83	93	105	121	96	101	117	130
Iter 2	281	283	281	283	247	247	267	265	79	98	95	126	96	97	120	128
Total	282	284	286	286	253	258	269	267	91	105	112	141	106	105	133	145

An analysis, provided in [54, App. D], reveals that in the proofs not found during lemma generation and found by SGCD after the provision of lemmas,

[7] Further visualizations of our experiments are provided in [54, App. C].

63–96% of the distinct subterms originate from the lemmas, i.e., a substantial portion of the proofs are built up from the provided lemmas.

Learned Lemmas to Enhance Other Provers. Next, we fix SGCD as base prover and evaluate other provers, namely Vampire, E, Prover9 and leanCoP. Again, the provers are given $k = 200$ best lemmas according to the linear utility model. Table 3 shows the greatest boost is for the purely goal-driven leanCoP, where there is over 40% improvement for all time limits. Second is Vampire with 8–15% improvement, followed by Prover9 and E with around 3% improvement. Interestingly, E does not solve more problems with the lemmas, but it solves different ones, hence the improvement. These results suggest a great deal of transferability of the benefits of the lemma selector.

Table 3. Number of problems solved by Vampire (casc), E (autoschedule), Prover9 and leanCoP without and with additional lemmas using various time limits.

	Vampire				E				Prover9				leanCoP			
Time	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m	50 s	100 s	500 s	30 m
Base	221	224	252	263	253	264	275	281	236	244	257	260	70	71	77	77
Lemmas	249	257	274	283	256	266	275	275	246	250	261	269	100	103	111	113
Total	249	257	276	284	269	276	287	286	248	252	264	269	100	103	111	113

Changing the Number of Lemmas Added. Adding lemmas has potential to shorten proofs, but it also widens the search space, so it is not obvious how many lemmas are beneficial. In the next experiment, we again fix SGCD as base prover and evaluate SGCD and Vampire with different number of lemmas selected. Table 4 shows that as little as 25 added lemmas yield substantial improvement, 7% for Vampire and 4% for SGCD, and performance does not drop as we add more lemmas: even at 500 we see no negative effect of the expanded search space.

Table 4. Number of problems solved by Vampire (casc) and SGCD as we alter the number k of supplemented lemmas. We use a time limit of 100 s.

	Vampire						SGCD					
Lemma count	10	25	50	100	200	500	10	25	50	100	200	500
Base	227	227	227	227	227	227	275	275	275	275	275	275
Lemmas	226	242	246	258	257	258	278	285	284	281	283	284
Total	231	243	247	258	257	258	282	285	284	283	284	285

Linear vs GNN Model. The preceding experiments suggest that even a simple linear model can provide useful guidance when features are carefully selected. Table 5 shows that the GNN—which processes the formulas directly and has no access to expert designed features—also successfully learns to identify useful lemmas for SGCD and even slightly surpasses the linear model. `LCL125-1` can only be solved by the GNN-assisted prover, even at extremely large time limits.

Table 5. Number of problems solved by SGCD over 2 iterations of training both a linear and a graph neural network model, for time limits 50 s, 100 s, 500 s and 30 min.

	Linear				GNN			
Time	50 s	100 s	500 s	30m	50 s	100 s	500 s	30m
Base	266	275	285	285	266	275	285	285
Iter 1	280	282	284	281	272	282	283	284
Iter 2	281	283	281	283	279	282	282	284
Total	282	284	286	286	279	285	287	287

3.2 Discussion of Learning-Based Experiments

When enhanced by learning-based lemma selection, SGCD solves 287 of the 312 problems. These include 28 problems not solved by the leading first-order prover Vampire [29], which solves 263 problems in its *CASC* [63] portfolio mode. Supplemented with our lemmas, Vampire is boosted to 284 solved problems. In combination, boosted SGCD and Vampire give 293 solved problems. Taking into account the solutions obtained by further provers with our lemmas, we obtain a total of 297. For detailed results see [54, App. E] and http://cs.christophwernhard.com/cdtools/exp-lemmas/lemmas.html.

A notable observation is that all systems—with the exception of E—improve when provided with selected lemmas. We argue that our framework addresses fundamental weaknesses of both purely goal-driven systems such as CMProver, leanCoP and CCS-Vanilla, as well as those of saturation style systems such as Vampire and E. For the former, it is their inability to generate lemmas, which results in unduly long proofs. For the latter, it is their unrestricted expansion of the branching of the search space. We find that goal-driven systems demonstrate huge improvement when lemmas are added: usually 20–40% depending on the configuration. The improvement is much more modest for saturation style systems, partly because their baselines are already stronger and partly because learned lemma selection still has a large room for improvement. This is the focus of our immediate future work. SGCD already provides a balance between goal-driven search and axiom-driven lemma generation and we only see significant improvement from lemmas when the time limit on proof search is smaller. Our manual feature-based linear model allows for exploiting expert knowledge. However, we see more potential in automated feature extraction via GNNs. The fact that the two models perform similarly suggests that we are not providing enough training data for the GNN to manifest its full capabilities.

4 Proving `LCL073-1`

`LCL073-1` was proven by Meredith in the early 1950s with substitution and detachment [42] but it remains outstandingly hard for ATP, where it came to attention in 1992 [40]; TPTP reports rating 1.0 and status *Unknown* since 1997. Only Wos proved it in the year 2000 with several invocations of OTTER [84], transferring output and insight between runs. The problem has a single axiom,

$$\mathsf{P}(\mathsf{i}(\mathsf{i}(\mathsf{i}(\mathsf{i}(\mathsf{i}(x, y), \mathsf{i}(\mathsf{n}(z), \mathsf{n}(u))), z), v), \mathsf{i}(\mathsf{i}(v, x), \mathsf{i}(u, x)))),$$

and the goal $\mathsf{P}(\mathsf{i}(\mathsf{i}(\mathsf{a}, \mathsf{b}), \mathsf{i}(\mathsf{i}(\mathsf{b}, \mathsf{c}), \mathsf{i}(\mathsf{a}, \mathsf{c}))))$, known as *Syll* [66]. The wider context is showing that a single axiom entails the elements of a known axiomatization of a propositional logic. Experiments with SGCD in our workflow led to a proof of `LCL073-1` (Fig. 2, also [54, App. F]) surprisingly quickly. Its compacted size is 46, between that of Meredith (40, reconstructed with CD in [84]) and that of Wos (74). Our workflow is much simpler than Wos', basically the same as our other experiments but restricted to one phase of lemma generation and incorporation, with only heuristic lemma selection, no learning. Nevertheless, success is fragile with respect to configuration, where reasons for failure or success are not obvious.

$$\begin{gathered}2 = \mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(1, 1))),\ 3 = \mathsf{D}(2, 2),\ 4 = \mathsf{D}(1, 3),\ 5 = \mathsf{D}(1, 4),\ 6 = \mathsf{D}(5, 1),\ 7 = \mathsf{D}(5, 6),\\ 8 = \mathsf{D}(\mathsf{D}(\mathsf{D}(1, \mathsf{D}(1, 7)), 6), 1),\ 9 = \mathsf{D}(8, 6),\ 10 = \mathsf{D}(8, \mathsf{D}(1, 9)),\ 11 = \mathsf{D}(\mathsf{D}(1, \mathsf{D}(1, \mathsf{D}(4, 10))), 1),\\ 12 = \mathsf{D}(1, \mathsf{D}(6, \mathsf{D}(1, \mathsf{D}(\mathsf{D}(1, \mathsf{D}(9, \mathsf{D}(9, \mathsf{D}(\mathsf{D}(11, 3), 4)))), 1)))),\ 13 = \mathsf{D}(\mathsf{D}(\mathsf{D}(12, \mathsf{D}(5, \mathsf{D}(8, 12))), 1), 7),\\ 14 = \mathsf{D}(1, \mathsf{D}(13, \mathsf{D}(1, \mathsf{D}(13, 5)))),\ 15 = \mathsf{D}(\mathsf{D}(1, \mathsf{D}(13, \mathsf{D}(\mathsf{D}(\mathsf{D}(\mathsf{D}(13, 6), 9), 11), 10))), \mathsf{D}(14, \mathsf{D}(14, 1)))\end{gathered}$$

Fig. 2. The D-term of our proof of `LCL073-1` represented by factor equations.

Our configuration parameters are not problem specific, although we started out with lemma generation by PSP-level because it led earlier to a short proof of `LCL038-1` [74,76]. We first call SGCD to generate lemmas by PSP-level enumeration, configured with a cache size of 5,000, terminating after 60 s with exhaustion of the search space.[8] Lemma features are computed for the 98,198 generated lemmas and written to disk, taking another 120 s. Lemmas are then ordered lexicographically according to five features relating to sharing of symbols and subterms with the goal, and to formula dimensions, taking a further 70 s. These five features are **`lf_h_height`**, **`lf_h_excluded_goal_subterms`**, **`lf_h_tsize`**, **`lf_h_distinct_vars`**, **`dcterm_hash`**, see [54, App. A] for their specification. We now call SGCD again, configured such that it performs PSP-level enumeration for axiom-driven phases, interleaved with level enumeration by height for goal-driven phases with 0 as *preAddMaxLevel*. It incorporates the first 2,900 ordered

[8] Notebook hardware, Intel® Core™ i7-1260P processor, 32 GB RAM.

lemmas[9] as input by *replacement* (Sect. 2). The cache size limit is set to 1,500, a value used in other generally successful configurations. Formulas occurring as subformulas of an earlier-proven formula are excluded, a variation of the *organic* property [37, 76]. The proof is then found in 20 s, total time elapsed about 270 s.

The D-term dimensions $\langle$*compacted size*, *tree size*, *height*$\rangle$ are $\langle 46, 3276, 40\rangle$, compared to Meredith's $\langle 40, 6172, 30\rangle$[10] and Wos' $\langle 74, 9207, 48\rangle$. The maximal size (occurrences of non-constant function symbols) of a lemma formula (MGT of a subproof) in the proof is 19, the maximal height (tree height, disregarding the predicate symbol) 9, and the maximal number of variables 7. Of the 46 lemmas in the proof 12 are present in the 2,900 input lemmas. Among the 46 lemma formulas 35 are weakly organic [76] and 4 involve double negation. N-simplification [76] applies to 65 occurrences but does not effect a size reduction. The proof is S- and C-regular [76]. Certain configurations of SGCD for the proving phase also yield further proofs. In experiments so far, these are enumerated after the presented proof and have larger compacted size.

Proof structure enumeration by PSP-level [76] is the main key to finding our proof of `LCL073-1`. It is used for lemma generation and for axiom-driven proof search, whereas goal-driven phases use height instead. The structure of the proof reflects this: all steps with the exception of the root can be considered PSP steps, i.e. one premise is a subproof of the other. The particular challenge of the problem lies in the fact that it was solved by a human (Meredith). Unlike in recent ATP successes for Boolos' curious inference [5, 10], where the key is two particular second-order lemmas, the key here is a proof-structural *principle* for building-up proofs by lemmas. Intuitively it might express a form of economy, building proofs from proofs at hand, that belonged to Meredith's repertoire.

5 Conclusion

We presented encouraging results about the use of lemmas in proof search. Provers are provided with lemmas generated via structure enumeration, a feature of the CM, and filtered with either learned guidance or manual heuristics. As a first step with this new methodology, we focus on the class of CD problems where we obtained strong results with our own system and substantial improvement of general first-order provers based on different paradigms, including the long-time competition leader Vampire. Moreover, our approach has led to the—in a sense first—automatic proof for the well-known Meredith single axiom problem with TPTP difficulty rating 1.0.

An important and novel aspect in our work was the explicit consideration of proof structures, which for CD have a particularly simple form in D-terms. Proof structures of the CM have a direct correspondence to these [76], such that the

[9] 2,900 is one of the fragile parameters. Depending on features chosen for ordering lemmas, there are ranges around 3,000 where the problem is solved.

[10] The *length* reported in [84] is the compacted size if also the proofs of the two other goals required to prove completeness of the single axiom are considered. The notion of compacted size straightforwardly generalizes from trees to *sets* of trees [76].

CM may guide the way to generalizations for more expressive logics. Another course of generalization is to move from unit lemmas, i.e. sharing of *subtrees* of D-terms, to more powerful lemmas. Preliminary work shows a correspondence between Horn clause lemmas, D-terms with variables, proofs in the connection structure calculus [15], and combinatory compression [73].

The learning-based experiments show little difference in performance between using a simple linear model and a more sophisticated graph neural network. We believe this is due to the small problem corpus, which yields a limited training signal. Hence, we plan to scale the system up to larger problem sets.

Our work also sheds new light on perspectives for the CM. It is well-known that the lack of inherent lemma maintenance is a disadvantage of the CM compared to resolution, which can be overcome with the connection structure calculus [15], a generalization of the CM. Here we see in experiments a drastic improvement of the CM-CT provers by supplementing their input with externally generated lemmas. SGCD, which grew out of the CM-CT approach and integrates repeated lemma generation into the proving process, keeps up with RS provers on CD problems, and can even be applied to improve these by supplying its lemmas as additional input.

Acknowledgments. We thank Jens Otten for inspiring discussions at the outset of the current project and anonymous reviewers for helpful suggestions to improve the presentation. The Hungarian Artificial Intelligence National Laboratory (RRF-2.3.1-21-2022-00004) and the ELTE TKP 2021-NKTA-62 funding scheme.

References

1. Alemi, A.A., Chollet, F., Een, N., Irving, G., Szegedy, C., Urban, J.: DeepMath – deep sequence models for premise selection. In: Lee, D., et al. (eds.) NIPS 2016, pp. 2243–2251. Curran Associates Inc., USA (2016). http://dl.acm.org/citation.cfm?id=3157096.3157347
2. Astrachan, O.L., Stickel, M.E.: Caching and lemmaizing in model elimination theorem provers. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607, pp. 224–238. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55602-8_168
3. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chap. 2, pp. 19–99. Elsevier (2001). https://doi.org/10.1016/B978-044450813-3/50004-7
4. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. In: Alferes, J.J., Pereira, L.M., Orlowska, E. (eds.) JELIA 1996. LNCS, vol. 1126, pp. 1–17. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61630-6_1
5. Benzmüller, C., Fuenmayor, D., Steen, A., Sutcliffe, G.: Who finds the short proof? Logic J. IGPL (2023). https://doi.org/10.1093/jigpal/jzac082
6. Bibel, W.: Automated Theorem Proving, 2nd edn. Vieweg, Braunschweig (1987). First edition 1982. https://doi.org/10.1007/978-3-322-90102-6
7. Bibel, W.: Deduction: Automated Logic. Academic Press, Cambridge (1993)
8. Bibel, W., Otten, J.: From Schütte's formal systems to modern automated deduction. In: The Legacy of Kurt Schütte, pp. 217–251. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49424-7_13

9. Bonacina, M.P.: A taxonomy of theorem-proving strategies. In: Wooldridge, M.J., Veloso, M. (eds.) Artificial Intelligence Today. LNCS (LNAI), vol. 1600, pp. 43–84. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48317-9_3
10. Boolos, G.: A curious inference. J. Philos. Logic **16**, 1–12 (1987). https://doi.org/10.1007/BF00250612
11. Bridge, J.P., Holden, S.B., Paulson, L.C.: Machine learning for first-order theorem proving. J. Autom. Reason. **53**(2), 141–172 (2014). https://doi.org/10.1007/s10817-014-9301-5
12. Dahn, I., Wernhard, C.: First order proof problems extracted from an article in the Mizar mathematical library. In: Bonacina, M.P., Furbach, U. (eds.) FTP 1997, pp. 58–62. RISC-Linz Report Series No. 97-50, Joh. Kepler Univ. Linz (1997). https://www.logic.at/ftp97/papers/dahn.pdf
13. Denzinger, J., Kronenburg, M., Schulz, S.: DISCOUNT – a distributed and learning equational prover. J. Autom. Reason. **18**(2), 189–198 (1997). https://doi.org/10.1023/A:1005879229581
14. Ebner, G., Hetzl, S., Leitsch, A., Reis, G., Weller, D.: On the generation of quantified lemmas. J. Autom. Reason. **63**(1), 95–126 (2018). https://doi.org/10.1007/s10817-018-9462-8
15. Eder, E.: A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In: Börger, E., Büning, H.K., Richter, M.M. (eds.) CSL 1988. LNCS, vol. 385, pp. 80–98. Springer, Heidelberg (1989). https://doi.org/10.1007/BFb0026296
16. Fitelson, B., Wos, L.: Missing proofs found. J. Autom. Reason. **27**(2), 201–225 (2001). https://doi.org/10.1023/A:1010695827789
17. Fuchs, M.: Lemma generation for model elimination by combining top-down and bottom-up inference. In: Dean, T. (ed.) IJCAI 1999, pp. 4–9. Morgan Kaufmann (1999). http://ijcai.org/Proceedings/99-1/Papers/001.pdf
18. Gauthier, T., Kaliszyk, C., Urban, J., Kumar, R., Norrish, M.: Learning to prove with tactics. CoRR abs/1804.00596 (2018). https://doi.org/10.48550/arXiv.1804.00596
19. Hester, J.: Novel methods for first order automated theorem proving. Ph.D. thesis, University of Florida (2021)
20. Hetzl, S., Leitsch, A., Weller, D.: Towards algorithmic cut-introduction. In: Bjørner, N., Voronkov, A. (eds.) LPAR 2012. LNCS, vol. 7180, pp. 228–242. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28717-6_19
21. Hähnle, R.: Tableaux and related methods. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chap. 3, pp. 101–178. Elsevier (2001). https://doi.org/10.1016/b978-044450813-3/50005-9
22. Hindley, J.R.: Basic Simple Type Theory. Cambridge University Press, Cambridge (1997). https://doi.org/10.1017/CBO9780511608865
23. Hindley, J.R., Meredith, D.: Principal type-schemes and condensed detachment. J. Symbolic Logic **55**(1), 90–105 (1990). https://doi.org/10.2307/2274956
24. Holden, S.B.: Machine learning for automated theorem proving: learning to solve SAT and QSAT. Found. Trends® Mach. Learn. **14**(6), 807–989 (2021). https://doi.org/10.1561/2200000081
25. Jakubův, J., Urban, J.: ENIGMA: efficient learning-based inference guiding machine. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 292–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_20
26. Kaliszyk, C., Urban, J.: Learning-assisted theorem proving with millions of lemmas. J. Symb. Comput. **69**, 109–128 (2015). https://doi.org/10.1016/j.jsc.2014.09.032

27. Kaliszyk, C., Urban, J., Michalewski, H., Olšák, M.: Reinforcement learning of theorem proving. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) NeurIPS 2018, pp. 8836–8847 (2018). https://papers.nips.cc/paper/2018/file/55acf8539596d25624059980986aaa78-Paper.pdf
28. Kaliszyk, C., Urban, J., Vyskočil, J.: Lemmatization for stronger reasoning in large theories. In: Lutz, C., Ranise, S. (eds.) FroCoS 2015. LNCS (LNAI), vol. 9322, pp. 341–356. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24246-0_21
29. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
30. Lemmon, E.J., Meredith, C.A., Meredith, D., Prior, A.N., Thomas, I.: Calculi of pure strict implication. In: Davis, J.W., Hockney, D.J., Wilson, W.K. (eds.) Philosophical Logic, pp. 215–250. Springer, Dordrecht (1969). https://doi.org/10.1007/978-94-010-9614-0_17. Reprint of a technical report, Canterbury University College, Christchurch (1957)
31. Letz, R.: Tableau and Connection Calculi. Structure, Complexity, Implementation. Habilitationsschrift, TU München (1999). http://www2.tcs.ifi.lmu.de/~letz/habil.ps. Accessed 19 July 2023
32. Letz, R., Mayr, K., Goller, C.: Controlled integration of the cut rule into connection tableaux calculi. J. Autom. Reason. **13**(3), 297–337 (1994). https://doi.org/10.1007/BF00881947
33. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: a high-performance theorem prover. J. Autom. Reason. **8**(2), 183–212 (1992). https://doi.org/10.1007/BF00244282
34. Loos, S.M., Irving, G., Szegedy, C., Kaliszyk, C.: Deep network guided proof search. In: Eiter, T., Sands, D. (eds.) LPAR-21. EPiC, vol. 56, pp. 85–105 (2017). https://doi.org/10.29007/8mwc
35. Loveland, D.W.: Automated Theorem Proving: A Logical Basis. North-Holland, Amsterdam (1978)
36. Łukasiewicz, J.: Selected Works. North Holland (1970). Edited by L. Borkowski
37. Łukasiewicz, J., Tarski, A.: Untersuchungen über den Aussagenkalkül. Comptes rendus des séances de la Soc. d. Sciences et d. Lettres de Varsovie 23 (1930). English translation in [36], pp. 131–152
38. McCune, W.: Prover9 and Mace4 (2005–2010). http://www.cs.unm.edu/~mccune/prover9
39. McCune, W.: OTTER 3.3 reference manual. Technical report, ANL/MCS-TM-263, Argonne National Laboratory (2003). https://www.cs.unm.edu/~mccune/otter/Otter33.pdf. Accessed 19 July 2023
40. McCune, W., Wos, L.: Experiments in automated deduction with condensed detachment. In: Kapur, D. (ed.) CADE 1992. LNCS, vol. 607, pp. 209–223. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55602-8_167
41. Meredith, C.A., Prior, A.N.: Notes on the axiomatics of the propositional calculus. Notre Dame J. Formal Logic **4**(3), 171–187 (1963). https://doi.org/10.1305/ndjfl/1093957574
42. Meredith, C.A.: Single axioms for the systems (C, N), (C, O) and (A, N) of the two-valued propositional calculus. J. Comput. Syst. **1**, 155–164 (1953)
43. Meredith, D.: In memoriam: Carew Arthur Meredith (1904–1976). Notre Dame J. Formal Logic **18**(4), 513–516 (1977). https://doi.org/10.1305/ndjfl/1093888116
44. OEIS Foundation Inc.: The On-Line Encyclopedia of Integer Sequences (2021). http://oeis.org

45. Otten, J.: Restricting backtracking in connection calculi. AI Commun. **23**(2–3), 159–182 (2010). https://doi.org/10.3233/AIC-2010-0464
46. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. J. Symb. Comput. **36**(1–2), 139–161 (2003). https://doi.org/10.1016/S0747-7171(03)00037-3
47. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
48. Piotrowski, B., Urban, J.: Guiding inferences in connection tableau by recurrent neural networks. In: Benzmüller, C., Miller, B. (eds.) CICM 2020. LNCS (LNAI), vol. 12236, pp. 309–314. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53518-6_23
49. Polu, S., Sutskever, I.: Generative language modeling for automated theorem proving. CoRR abs/2009.03393 (2020). https://doi.org/10.48550/arXiv.2009.03393
50. Prior, A.N.: Logicians at play; or Syll, Simp and Hilbert. Australas. J. Philos. **34**(3), 182–192 (1956). https://doi.org/10.1080/00048405685200181
51. Prior, A.N.: Formal Logic, 2nd edn. Clarendon Press, Oxford (1962). https://doi.org/10.1093/acprof:oso/9780198241560.001.0001
52. Pudlák, P.: Search for faster and shorter proofs using machine generated lemmas. In: Sutcliffe, G., Schmidt, R., Schulz, S. (eds.) ESCoR 2006. CEUR Workshop Proceeding, vol. 192, pp. 34–53. CEUR-WS.org (2006). http://ceur-ws.org/Vol-192/paper03.pdf
53. Rawson, M., Reger, G.: lazyCoP: lazy paramodulation meets neurally guided search. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 187–199. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_11
54. Rawson, M., Wernhard, C., Zombori, Z., Bibel, W.: Lemmas: generation, selection, application. CoRR abs/2303.05854 (2023). https://doi.org/10.48550/arXiv.2303.05854
55. Reger, G., Tishkovsky, D., Voronkov, A.: Cooperating proof attempts. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 339–355. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_23
56. Rezuş, A.: Tarski's Claim thirty years later. In: Witness Theory - Notes on λ-calculus and Logic, Studies in Logic, vol. 84, pp. 217–225. College Publications (2020). Preprint (2016). http://www.equivalences.org/editions/proof-theory/ar-tc-20160512.pdf
57. Rezuş, A.: Witness Theory - Notes on λ-calculus and Logic. Studies in Logic, vol. 84. College Publications (2020)
58. Sanchez-Lengeling, B., Reif, E., Pearce, A., Wiltschko, A.B.: A gentle introduction to graph neural networks. Distill (2021). https://doi.org/10.23915/distill.00033
59. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
60. Schumann, J.M.P.: DELTA — a bottom-up preprocessor for top-down theorem provers. In: Bundy, A. (ed.) CADE 1994. LNCS, vol. 814, pp. 774–777. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58156-1_58
61. Stickel, M.E.: A Prolog technology theorem prover: implementation by an extended Prolog compiler. J. Autom. Reason. **4**(4), 353–380 (1988). https://doi.org/10.1007/BF00297245

62. Stickel, M.E.: Upside-down meta-interpretation of the model elimination theorem-proving procedure for deduction and abduction. J. Autom. Reason. **13**(2), 189–210 (1994). https://doi.org/10.1007/BF00881955
63. Sutcliffe, G.: The CADE ATP system competition – CASC. AI Mag. **37**(2), 99–101 (2016)
64. Sutcliffe, G.: The TPTP problem library and associated infrastructure. J. Autom. Reason. **59**(4), 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
65. Sutcliffe, G., Gao, Y., Colton, S.: A grand challenge of theorem discovery. In: Worksh. Challenges and Novel Applications for Automated Reasoning, 19th IJCAR, pp. 1–11 (2003). https://www.cs.miami.edu/home/geoff/Papers/Conference/2003_SGC03_CNAAR-1-11.pdf
66. Ulrich, D.: A legacy recalled and a tradition continued. J. Autom. Reason. **27**(2), 97–122 (2001). https://doi.org/10.1023/A:1010683508225
67. Urban, J., Jakubův, J.: First neural conjecturing datasets and experiments. In: Benzmüller, C., Miller, B. (eds.) CICM 2020. LNCS (LNAI), vol. 12236, pp. 315–323. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53518-6_24
68. Urban, J., Sutcliffe, G., Pudlák, P., Vyskočil, J.: MaLARea SG1 - machine learner for automated reasoning with semantic guidance. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 441–456. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_37
69. Veroff, R.: Finding shortest proofs: an application of linked inference rules. J. Autom. Reason. **27**(2), 123–139 (2001). https://doi.org/10.1023/A:1010635625063
70. Wang, M., Tang, Y., Wang, J., Deng, J.: Premise selection for theorem proving by deep graph embedding. In: Guyon, I., et al. (eds.) NIPS 2017, pp. 2783–2793 (2017). http://papers.nips.cc/paper/6871-premise-selection-for-theorem-proving-by-deep-graph-embedding
71. Wernhard, C.: The PIE system for proving, interpolating and eliminating. In: Fontaine, P., Schulz, S., Urban, J. (eds.) PAAR 2016. CEUR Workshop Proceedings, vol. 1635, pp. 125–138. CEUR-WS.org (2016). http://ceur-ws.org/Vol-1635/paper-11.pdf
72. Wernhard, C.: Facets of the *PIE* environment for proving, interpolating and eliminating on the basis of first-order logic. In: Hofstedt, P., Abreu, S., John, U., Kuchen, H., Seipel, D. (eds.) INAP/WLP/WFLP -2019. LNCS (LNAI), vol. 12057, pp. 160–177. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46714-2_11
73. Wernhard, C.: Generating compressed combinatory proof structures – an approach to automated first-order theorem proving. In: Konev, B., Schon, C., Steen, A. (eds.) PAAR 2022. CEUR Workshop Proceedings, vol. 3201. CEUR-WS.org (2022). https://arxiv.org/abs/2209.12592
74. Wernhard, C.: CD Tools – Condensed detachment and structure generating theorem proving (system description). CoRR abs/2207.08453 (2023). https://doi.org/10.48550/arXiv.2207.08453
75. Wernhard, C., Bibel, W.: Learning from Łukasiewicz and Meredith: investigations into proof structures. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 58–75. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_4
76. Wernhard, C., Bibel, W.: Investigations into proof structures. CoRR abs/2304.12827 (2023, submitted). https://doi.org/10.48550/arXiv.2304.12827
77. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-prolog. Theory Pract. Logic Program. **12**(1–2), 67–96 (2012). https://doi.org/10.1017/S1471068411000494

78. Woltzenlogel Paleo, B.: Atomic cut introduction by resolution: proof structuring and compression. In: Clarke, E.M., Voronkov, A. (eds.) LPAR 2010. LNCS (LNAI), vol. 6355, pp. 463–480. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17511-4_26
79. Wos, L., et al.: Automated reasoning contributes to mathematics and logic. In: Stickel, M.E. (ed.) CADE 1990. LNCS, vol. 449, pp. 485–499. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52885-7_109
80. Wos, L.: Automated reasoning and Bledsoe's dream for the field. In: Boyer, R.S. (ed.) Automated Reasoning: Essays in Honor of Woody Bledsoe, pp. 297–345. Automated Reasoning Series, Kluwer Academic Publishers (1991). https://doi.org/10.1007/978-94-011-3488-0_15
81. Wos, L.: The resonance strategy. Comput. Math. Appl. **29**(2), 133–178 (1995). https://doi.org/10.1016/0898-1221(94)00220-F
82. Wos, L.: The power of combining resonance with heat. J. Autom. Reason. **17**(1), 23–81 (1996). https://doi.org/10.1007/BF00247668
83. Wos, L.: Lemma inclusion versus lemma adjunction. Assoc. Autom. Reason. Newsl. **44** (1999). https://aarinc.org/Newsletters/044-1999-09.html. Accessed 19 July 2023
84. Wos, L.: Conquering the Meredith single axiom. J. Autom. Reason. **27**(2), 175–199 (2001). https://doi.org/10.1023/A:1010691726881
85. Zombori, Z., Urban, J., Brown, C.E.: Prolog technology reinforcement learning prover. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 489–507. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_33

Machine-Learned Premise Selection for Lean

Bartosz Piotrowski[1](✉), Ramon Fernández Mir[2], and Edward Ayers[3]

[1] University of Warsaw and Czech Technical University, Warsaw, Poland
bartoszpiotrowski@post.pl
[2] University of Edinburgh, Edinburgh, Scotland
[3] Carnegie Mellon University, Pittsburgh, USA

Abstract. We introduce a machine-learning-based tool for the Lean proof assistant that suggests relevant premises for theorems being proved by a user. The design principles for the tool are (1) tight integration with the proof assistant, (2) ease of use and installation, (3) a lightweight and fast approach. For this purpose, we designed a custom version of the random forest model, trained in an online fashion. It is implemented directly in Lean, which was possible thanks to the rich and efficient metaprogramming features of Lean 4. The random forest is trained on data extracted from mathlib – Lean's mathematics library. We experiment with various options for producing training features and labels. The advice from a trained model is accessible to the user via the `suggest_premises` tactic which can be called in an editor while constructing a proof interactively.

Keywords: premise selection · machine learning · Lean proof assistant

1 Introduction

Formalizing mathematics in proof assistants is an ambitious and hard undertaking. One of the major challenges in constructing formal proofs of theorems depending on multiple other results is the prerequisite of having a good familiarity with the structure and contents of the library. Tools for helping users search through formal libraries are currently limited.

In the case of the Lean proof assistant [13], users may look for relevant lemmas in its formal library, mathlib [5], either by (1) using general textual search tools and keywords, (2) browsing the related source files manually, (3) using mathlib's `suggest` or `library_search` tactics.

Approaches (1) and (2) are often slow and tedious. The limitation of approach (3) is the fact that `suggest` or `library_search` propose lemmas that strictly match the goal at the current proof state. This is often very useful, but it also means that these tactics often fail to direct the user to relevant lemmas that do not

The results were supported by the Hoskinson Center for Formal Mathematics (BP, RFM, EA), the Kościuszko Foundation (BP), and the Principal's Career Development Scholarship of the University of Edinburgh (RFM).

match the current goal exactly. They may also suggest too many trivial lemmas if the goal is simple.

The aim of this project is to make progress towards improving the situation of a Lean user looking for relevant lemmas while building proofs. We develop a new tool that efficiently computes a ranking of potentially useful lemmas selected by a machine learning (ML) model trained on data extracted from mathlib. This ranking can be accessed and used interactively via the `suggest_premises` tactic.

The project described here belongs to the already quite broad body of work dealing with the problem of fact selection for theorem proving [1,7,9,11,12,15, 16]. This problem, commonly referred to as the *premise selection* problem, is crucial when performing automated reasoning in large formal libraries – both in the context of *automated* (ATP) and *interactive* (ITP) theorem proving, and regardless of the underlying logical calculus. Most of the existing work on premise selection focuses on the ATP context. Our main contribution is the development of a premise selection tool that is practically usable in a proof assistant (Lean in that case), tightly integrated with it, lightweight, extendable, and equipped with a convenient interface. The tool is available in a public GitHub repository: https://github.com/BartoszPiotrowski/lean-premise-selection.

2 Dataset Collection

A crucial requirement of a useful ML model is a high-quality dataset of training examples. It should represent the learning task well and be suitable for the ML architecture being applied.

In this work, we use simple ML architectures that cannot process raw theorem statements and require *featurization* as a preprocessing step. The features need to be meaningful yet simple so that the model can use them appropriately. Our approach is described in Sect. 2.1. The notion of *relevant premise* may be understood differently depending on the context. In Sect. 2.2, we describe the different specifications of this notion that we used in our experiments.

The tool developed in this work is implemented and meant to be used in Lean 4 together with mathlib 4. However, since, at the time of writing, Lean 4's version of the library is still being ported from Lean 3, we use mathlib3port[1] as our main data source.

2.1 Features

The features, similar to those used in [8,15], consist of the symbols used in the theorem statement with different degrees of structure. In particular, three types of features are used: `names`, `bigrams` and `trigrams`.

As an illustration, take this theorem about groups with zero:

```
theorem div_ne_zero (ha : a ≠ 0) (hb : b ≠ 0) : a / b ≠ 0 := ...
```

This statement comes from one of the source files of mathlib. When producing the features for it, we do not use it directly as printed above but rather we take

[1] https://github.com/leanprover-community/mathlib3port (commit `f4e5dfe`).

its *elaborated* counterpart – a much more detailed version where all the hidden assumptions are made explicit by the Lean's elaborator so that the expression precisely conforms to Lean's dependent type theory.

The most basic form of featurization is the *bag-of-words* model, where we simply collect all the `names` (and numerical constants) involved in the theorem.

Following this definition, we obtain names $\neq$, `0`, and `/`, which are visible in the source version of the statement,[2] plus many more hidden names only appearing in the elaborated expression, e.g., `OfNat.ofNat` that is related to interpreting numerical literals as natural numbers.

During the featurization we distinguish features coming from the *conclusion* and the *hypotheses* (assumptions) of the theorem, and we mark them by prepending either `T` or `H`, respectively.

For our running example of theorem `div_ne_zero`, all this results in the list of `names` that looks as follows:

```
H:OfNat.ofNat H:MonoidWithZero.toZero H:0 H:Ne T:HDiv.hDiv T:0 T:Ne ...
```

It would be desirable, however, to keep track of which symbols appear next to each other in the syntactic trees of the theorem hypotheses and its statement. Thus, we extract `bigrams` that are formed by the head symbol and each of its arguments (separated by `/` below).

```
H:Ne/OfNat.ofNat H:OfNat.ofNat/0 T:OfNat.ofNat/0 T:Ne/OfNat.ofNat ...
```

Similarly, we also consider `trigrams`, taking all paths of length 3 from the syntactic tree of the expression.

```
H:Ne/OfNat.ofNat/0 H:Ne/OfNat.ofNat/Zero.toOfNat0 ...
```

2.2 Relevant Premises

To obtain the list of all the premises used in a proof of a given theorem it suffices to traverse the theorem's proof term[3] and keep track of all the constants whose type is a proposition. For instance, the raw list of premises that appear in the proof of `div_ne_zero` is:

```
GroupWithZero.noZeroDivisors
div_eq_mul_inv
mul_ne_zero
inv_ne_zero
Eq.refl
```

For more complicated examples, this approach results in a large number of premises including lemmas used *implicitly* by tactics (for instance, those picked by the 'simplify' tactic `simp`), or simple facts that a user would rarely write

[2] In fact, we use translations of these symbols from the elaborated counterpart of the theorem; so, for instance, we use `Ne` instead of the notation $\neq$, etc.

[3] A proof term is an internal Lean expression whose type is the theorem, constructed based on the proof written by a user, possibly using tactics.

Table 1. Filters' statistics. An example is a theorem with a non-empty list of premises. Because applying the `source` or `math` filter may result in an empty set of premises, the numbers of obtained training examples differ across the filters.

	`all`	`source`	`math`
Total premises	96 915	28 784	67 462
Total examples	41 755	20 571	40 187
Premises per example	3.12	2.35	2.09

explicitly. Three different filters are applied to mitigate this issue: `all`, `source`, and `math`. They are described below and their overall effect is shown in Table 1.

1. The `all` filter preserves almost all premises from the original, raw list, removing those that were generated automatically by Lean. They contain a leading underscore in their names, e.g., `RingTheory.MatrixAlgebra._auxLemma.1`. In our example, there are no such premises. Examples from this filter are not appropriate for training an ML advisor for interactive use as the suggestions would contain many lemmas used implicitly by tactics. Yet, such an advisor could be used for automated ITP approaches such as *hammers* [3].
2. The `source` filter leaves only those premises that appear in the proof's source code. The idea is to model the explicit usage of premises by a user. Following our example, we would take the following proof as a string and list only the three premises appearing there:

```
by rw [div_eq_mul_inv]; exact mul_ne_zero ha (inv_ne_zero hb)
```

3. The `math` filter preserves only lemmas that are clearly of mathematical nature, discarding basic, technical ones. The names of all theorems and definitions from mathlib are extracted and used as a *white list*. In particular, this means that many basic lemmas from Lean's core library (e.g. `Eq.refl` from our example) are filtered out.

In addition to our base datasets containing *one data point per theorem*, we also created a dataset (labeled as `intermediate`) representing *intermediate proof states*. In the standard data sets we recorded features of an initial proof state (the hypotheses and the conclusion of the theorem to be proved) and the premises used in a full proof. In the `intermediate` data set we instead record features of a proof state encountered *during* constructing a proof, and premises used in the next proof step only.

To this end, we used LeanInk,[4] a helper tool for Alectryon [17] – a toolkit that aids exploration of tactical proof scripts without running the proof assistant. Given a Lean file, LeanInk generates all the states that a user might be able to see in the *infoview* (a panel in Lean that displays goal states and other information about the prover's state) by clicking on the file. The file is split

[4] https://github.com/leanprover/LeanInk.

into fragments, each containing a string of Lean code, represented by a list of tokens, together with the proof states before and after. In this way, the file can be loaded statically simulating the effect of running Lean. Furthermore, it can be configured to keep track of typing information, which is key to detecting which tokens are premises. We modified LeanInk so that every fragment that appears inside a proof is treated as its own theorem by our extractor. We gather all the premises found in the list of tokens and featurize the hypotheses and goals in the "before" proof state.

This dataset consists of 91 292 examples and 143 165 premises, which gives an average of around 1.57 premises per example. It represents a more fine-grained use of the premises, which does not exactly correspond to our main objective of providing rankings of premises on the level of theorem statements. We treat it as an auxiliary dataset potentially useful for augmenting our base data sets.

3 Machine Learning Models

The task modelled here with ML is predicting a ranking of likely useful premises (lemmas and theorems) conditioned by the features of the statement of a theorem being proved by a user. The nature of this problem is different than common applications of classical ML: both the number of features and labels (premises) to predict is large, and the training examples are sparse in the feature space. Thus, we could not directly rely on traditional implementations of ML algorithms, and using custom-built versions was necessary. As one of our design requirements was tight integration with the proof assistant, we implemented the ML algorithms directly in Lean 4, without needing to call external tools. This also served as a test for the maturity and efficiency of Lean 4 as a programming language.

In Sects. 3.1 and 3.2 we describe two machine learning algorithms implemented in this work: k-nearest neighbours (k-NN) and random forest.

3.1 *k*-Nearest Neighbours

This is a classical and conceptually simple ML algorithm [6], which has already been used multiple times for premise selection [2,9,10]. It belongs to the *lazy learning* category, meaning that it does not result in a prediction model trained beforehand on the dataset, but rather the dataset is an input to the algorithm while producing the predictions.

Given an unlabeled example, k-NN produces a prediction by extracting the labels of the k most similar examples in the dataset and returning an averaged (or most frequent) label. In our case, the labels are lists of premises. We compose multiple labels into a ranking of premises according to the frequency of appearance in the concatenated labels.

The similarity measure in the feature space calculates how many features are shared between the two data points, but additionally puts more weight on those features that are rarer in the whole training dataset $\mathcal{D}$. The formula for the similarity of the two examples x_1 and x_2 associated with sets of features f_1 and f_2, respectively, is given below.

$$M(x_1, x_2) = \frac{\sum_{f \in f_1 \cap f_2} t(f)}{\sum_{f \in f_1} t(f) + \sum_{f \in f_2} t(f) - \sum_{f \in f_1 \cap f_2} t(f)}, \quad t(f) = \log \left(\frac{|\mathcal{D}|}{|\mathcal{D}_f|} \right)^2,$$

where $\mathcal{D}_f$ are those training examples that contain the feature f.

The advantages of k-NN are its simplicity and the lack of training. A disadvantage, however, is the need to traverse the whole training dataset in order to produce a single prediction (a ranking). This may be slow, and thus not optimal for interactive usage in proof assistants.

3.2 Random Forest

As an alternative to k-NN, we use *random forest* [4] – an ML algorithm from the *eager learning* category, with a separate training phase resulting in a prediction model consisting of a collection of decision trees. The leaves of the trees contain labels, and their nodes contain decision rules based on the features. In our case, the labels are sets of premises, and the rules are simple tests that check if a given feature appears in an example.

When predicting, unlabeled examples are passed down the trees to the leaves, the reached labels are recorded, and the final prediction is averaged across the trees via voting. The trees are trained in such a way as to avoid correlations between them, and the averaged prediction from them is of better quality than the prediction from a single tree.

Our version of random forest, adapted to deal with sparse binary features and a large number of labels, is similar to the one used in [19], where the task was to predict the next tactic progressing a proof in Coq proof assistant. There, the features were also sparse, however, the difference is that here we need to predict *sets* of labels (premises), not just one label (the next tactic).

Our random forest is trained in an *online* manner, i.e., it is updated sequentially with single training examples – not with the entire training dataset at once, as is typically done. The rationale for this is to make it easy to update the model with data coming from new theorems proved by a user. This allows the model to immediately provide suggestions taking into account these recently added theorems.[5]

Algorithm 1 provides a sketch of how a training example updates a tree – for all the details see the actual implementation in our public GitHub repository.[6] A crucial part of the algorithm is the MakeSplitRule function creating node splitting rules. Searching for the rules resulting in optimal splits would be costly, thus this function relies on heuristics.

Figure 1 schematically depicts how a simple decision tree from a trained random forest predicts a set of premises for an input example.

[5] This mode, however, has not yet been tested in the current stage of this work.

[6] The decision tree implementation is in a file PremiseSelection/Tree.lean.

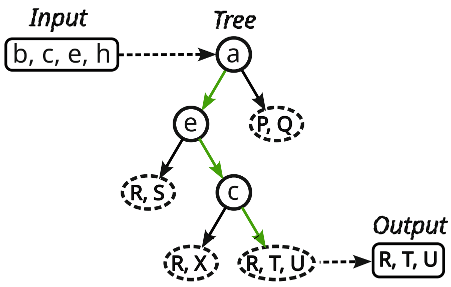


Fig. 1. A schematic example of a decision tree from a trained random forest. Lowercase letters (`a`, `b`, `c`, ...) designate features of theorem statements, whereas uppercase letters (`P`, `Q`, `R`, ...) designate names of premises. The input (a featurized theorem statement) is being passed down the tree (along the green arrows) so that each node tests for a presence of a single feature, and passes the input example to the left (or right) sub-tree in the negative (or positive) case. The output is a set of premises in the reached leaf. (Color figure online)

Algorithm 1. Updating a tree with a training example in a random forest.

```
1: function AddExampleToTree($T$, $e$)    ▷ $T$ – tree to update, $e$ – training example
2:     match $T$ with
3:         Node($R$, $T_l$, $T_r$):    ▷ $R$ – binary rule, $T_l$, $T_r$ – left and right subtrees
4:             match $R(e)$ with    ▷ passing example $e$ down the tree to a leaf
5:                 Left: return Node($R$, AddExampleToTree($T_l$, $e$), $T_r$)
6:                 Right: return Node($R$, $T_l$, AddExampleToTree($T_r$, $e$))
7:         Leaf($E$):    ▷ $E$ – examples stored in the leaf
8:             $E \leftarrow$ Append($E$, $e$)
9:             if SplitCondition($E$) then    ▷ testing if the leaf should be split
10:                $R \leftarrow$ MakeSplitRule($E$)    ▷ making semi-optimized new split rule
11:                $E_l$, $E_r \leftarrow$ Split($R$, $E$)    ▷ splitting examples into two parts
12:                return Node($R$, Leaf($E_l$), Leaf($E_r$))    ▷ new subtree growing the tree
13:            else
14:                return Leaf($E$)    ▷ the original leaf augmented with example $e$
```

4 Evaluation Setup and Results

To assess the performance of the ML algorithms, the data points extracted from mathlib were split into *training* and *testing* sets. The testing examples come from the modules that are *not* dependencies of any other modules (there are 592 of them). This simulates a realistic scenario in which a user utilizing the suggestion tool develops a new mathlib module. The rest of the modules (2436) served as the source of training examples.

Two measures of the quality of the rankings produced by ML are defined: Cover and Cover$_+$. Assuming a theorem T depends on the set of premises P of size n, and R is the ranking of premises predicted by the ML advisor for T, these measures are defined as follows:

$$\mathrm{Cover}(T) = \frac{\bigl|P \cap R[:n]\bigr|}{n}, \qquad \mathrm{Cover}_+(T) = \frac{\bigl|P \cap R[:n+10]\bigr|}{n},$$

where $R[:k]$ is a set of k initial premises from ranking R. Both Cover and Cover_+ return values in $[0, 1]$. Cover gives the score of 1 only for a "perfect" prediction where the premises actually used in the proof form an initial segment of the ranking. Cover_+ may also give a perfect score to less precise predictions. The rationale for Cover_+ is that the user in practice may look through 10 or more suggested premises. This is often more than the n premises actually used in the proof, so we consider initial segments of length $n + 10$ in Cover_+.

Both k-NN and random forest are evaluated on data subject to all three premise filters described in Sect. 2.2. For each of these variants of data, three combinations of features are tested: (1) `names` only, (2) `names` and `bigrams`, (3) `names`, `bigrams`, and `trigrams`. The hyper-parameters for the ML algorithms were selected by an experiment on a smaller dataset. For k-NN, the number of neighbours was fixed to 100. For random forest, the number of trees was set to 300, each example was used for training a particular decision tree with probability equal to 0.3, and the training algorithm passed through the whole training data 3 times.

Table 2 shows the results of the experiment. In terms of the Cover metric, random forest performed better than k-NN for all data configurations. However, for Cover_+ metric, k-NN surpassed random forest for the `math` filter.

It turned out that the union of `names` and `bigrams` constitutes the best features for all the filters and both ML algorithms. It likely means that the more complex `trigrams` did not help the algorithms to generalize well but rather caused *over-fitting* on the training set.

The results for the `all` filter appear to be much higher than for the other two filters. However, this is because applying `all` results in many simple examples containing just a few common, basic premises (e.g., just a single `rfl` lemma). They increase the average score.

Overall, random forest with `names` + `bigrams` (`n+b`) features gives the best results. An additional practical advantage of this model over k-NN is the speed of outputting predictions. For instance, for the `source` filter and `n+b` features, the average times of predicting a ranking of premises per theorem were 0.28 s and 5.65 s for random forest and k-NN, respectively.

Additionally, we evaluated the ML models on the `intermediate` dataset, using `n+b` features. The random forest achieved Cover $= 0.09$ and $\mathrm{Cover}_+ = 0.24$, whereas k-NN resulted in Cover $= 0.08$ and $\mathrm{Cover}_+ = 0.21$ on the testing part of the data. Then, we used the `intermediate` dataset in an attempt to improve the testing results on the base dataset with the `source` filter (as `intermediate` only contains premises exposed in the source files). We used the `intermediate` data as a *pre-training* dataset, first training a random forest on it, and later on the base data. We also used `intermediate` to *augment* the base data, mixing the two together. However, neither in the pre-training, nor in the augmentation mode statistically significant improvements in the testing performance were achieved. It is possible that the prediction quality from the practical perspective actually

Table 2. Average performance of random forest and k-NN on testing data, for three premises filters and three kinds of features. The type of features is indicated by a one-letter abbreviation: `n` = `names`, `b` = `bigrams`, `t` = `trigrams`. For each configuration, Cover and Cover+ measures are reported (the latter in brackets). In each row, the best Cover result is bolded.

premises	machine learning model					
	random forest			k-nearest neighbours		
	`n`	`n+b`	`n+b+t`	`n`	`n+b`	`n+b+t`
`all`	0.56 (0.67)	**0.57** (0.67)	0.47 (0.58)	0.51 (0.65)	0.52 (0.66)	0.51 (0.62)
`source`	0.28 (0.36)	**0.29** (0.36)	0.28 (0.36)	0.25 (0.35)	0.25 (0.36)	0.26 (0.35)
`math`	0.25 (0.32)	**0.26** (0.33)	0.16 (0.24)	0.22 (0.34)	0.23 (0.34)	0.16 (0.26)

improved, being more proof-state-dependent and not only theorem-dependent, but it did not manifest in our theorem-dependent evaluation.

The evaluation may be reproduced by following the instructions in the linked source code.[7]

5 Interactive Tool

The ML predictor is wrapped in an interactive tactic `suggest_premises` that users can type into their proof script. It will invoke the predictor and produce a list of suggestions. This list is displayed in the infoview. The display makes use of the new remote-procedure-call (RPC) feature in Lean 4 [14], to then asynchronously run various tactics for each suggestion. Given a suggested premise p, the system will attempt to run tactics `apply` p, `rw [` p`]` and `simp only [` p`]`, and return the first successful tactic application that advances the state. This will then be displayed to the user as shown in Fig. 2. She can select the resulting tactic to insert into the proof script. By using an asynchronous approach, we can display results rapidly without waiting for a slow tactic search to complete.

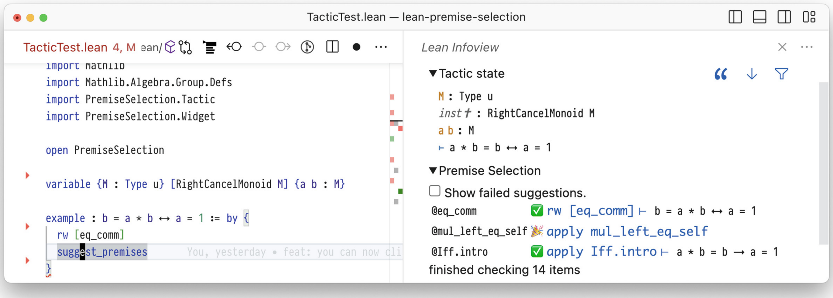
6 Future Work

There are several directions in which the current work may be developed.

The results may be improved by augmenting the dataset with, for instance, synthetic theorems, as well as developing better features, utilizing the well-defined structure of Lean expressions.

The evaluation may be extended to assess the proof-state level performance, and to compare with the standard Lean's suggestion tactics: `library_search`

[7] https://github.com/BartoszPiotrowski/lean-premise-selection#reproducing-evaluation.



```
import Mathlib
import Mathlib.Algebra.Group.Defs
import PremiseSelection.Tactic
import PremiseSelection.Widget

open PremiseSelection

variable {M : Type u} [RightCancelMonoid M] {a b : M}

example : b = a * b ↔ a = 1 := by {
  rw [eq_comm]
  suggest_premises      You, yesterday • feat: you can now cli
}
```

```
▼Tactic state
 M : Type u
 inst✝ : RightCancelMonoid M
 a b : M
 ⊢ a * b = b ↔ a = 1
▼Premise Selection
☐ Show failed suggestions.
@eq_comm          ✅ rw [eq_comm] ⊢ b = a * b ↔ a = 1
@mul_left_eq_self 🎉 apply mul_left_eq_self
@Iff.intro        ✅ apply Iff.intro ⊢ a * b = b → a = 1
finished checking 14 items
```

Fig. 2. The interactive tool in Visual Studio Code. The left pane shows the source file with the cursor over a `suggest_premises` tactic. The right pane shows the goal state at the cursor position and, below, the suggested lemmas to solve the goal. Suggestions annotated with a checkbox advance the goal state, suggestions annotated with confetti close the current goal. Clicking on a suggested tactic (e.g. `apply mul_left_eq_self`) automatically appends to the proof script on the left.

and `suggest`. It could be beneficial to combine these tactics – which use sctrict matching – with our tool based on statistical matching.

Applying modern neural architectures in place of the simpler ML algorithms used here is a promising path [7,12,16,18]. It would depart from our philosophy of a lightweight, self-contained approach as the suggestions would come from an external tool, possibly placed on a remote server. However, given the strength of the current neural networks, we could hope for higher-quality predictions. Moreover, neural models do not require hand-engineered features. The results presented here could serve as a baseline for comparison.

Finally, premise selection is an important component of ITP *hammer systems* [3]. The presented tool may be readily used for a hammer in Lean, which has not yet been developed.

References

1. Alama, J., Heskes, T., Kühlwein, D., Tsivtsivadze, E., Urban, J.: Premise selection for mathematics by corpus analysis and kernel methods. J. Autom. Reason. **52**(2), 191–213 (2014). https://doi.org/10.1007/s10817-013-9286-5
2. Blanchette, J.C., Greenaway, D., Kaliszyk, C., Kühlwein, D., Urban, J.: A learning-based fact selector for Isabelle/HOL. J. Autom. Reason. **57**(3), 219–244 (2016). https://doi.org/10.1007/s10817-016-9362-8
3. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formaliz. Reason. **9**(1), 101–148 (2016). https://doi.org/10.6092/issn.1972-5787/4593

4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. The mathlib Community. The lean mathematical library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, pp. 367–381. CPP 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372885.3373824
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. SSS, Springer, New York (2009). https://doi.org/10.1007/978-0-387-84858-7
7. Irving, G., Szegedy, C., Alemi, A.A., Eén, N., Chollet, F., Urban, J.: DeepMath - deep sequence models for premise selection. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016(December), pp. 5–10, 2016. Barcelona, Spain, pp. 2235–2243 (2016), https://proceedings.neurips.cc/paper/2016/hash/f197002b9a0853eca5e046d9ca4663d5-Abstract.html
8. Jakubův, J., Urban, J.: ENIGMA: efficient learning-based inference guiding machine. In: Geuvers, H., England, M., Hasan, O., Rabe, F., Teschke, O. (eds.) CICM 2017. LNCS (LNAI), vol. 10383, pp. 292–302. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62075-6_20
9. Kaliszyk, C., Urban, J.: Learning-assisted automated reasoning with Flyspeck. J. Autom. Reason. **53**(2), 173–213 (2014). https://doi.org/10.1007/s10817-014-9303-3
10. Kaliszyk, C., Urban, J.: MizAR 40 for Mizar 40. J. Autom. Reason. **55**(3), 245–256 (2015). https://doi.org/10.1007/s10817-015-9330-8
11. Kühlwein, D., van Laarhoven, T., Tsivtsivadze, E., Urban, J., Heskes, T.: Overview and evaluation of premise selection techniques for large theory mathematics. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 378–392. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_30
12. Mikula, M., et al.: Magnushammer: A transformer-based approach to premise selection. CoRR **abs/2303.04488** (2023).https://doi.org/10.48550/arXiv.2303.04488,https://doi.org/10.48550/arXiv.2303.04488
13. Moura, L., Ullrich, S.: The lean 4 theorem prover and programming language. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 625–635. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_37
14. Nawrocki, W., Ayers, E.W., Ebner, G.: An extensible user interface for Lean 4. In: 14th International Conference on Interactive Theorem Proving, ITP 2023, July 31-August 4, 2023, Białystok, Poland. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023)
15. Piotrowski, B., Urban, J.: ATPboost: Learning Premise Selection in Binary Setting with ATP Feedback. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 566–574. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_37
16. Piotrowski, B., Urban, J.: Stateful premise selection by recurrent neural networks. In: Albert, E., Kovács, L. (eds.) LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, 22–27 May 2020. EPiC Series in Computing, vol. 73, pp. 409–422. EasyChair (2020). https://doi.org/10.29007/j5hd

17. Pit-Claudel, C.: Untangling mechanized proofs. In: Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering, pp. 155–174. SLE 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3426425.3426940,https://pit-claudel.fr/clement/papers/alectryon-SLE20.pdf
18. Tworkowski, S., et al.: Formal premise selection with language models. In: The 7th Conference on Artificial Intelligence and Theorem Proving, AITP 2022(September), pp. 4–9, 2022. Aussois and online, France (2022). http://aitp-conference.org/2022/abstract/AITP_2022_paper_32.pdf
19. Zhang, L., Blaauwbroek, L., Piotrowski, B., Černỳ, P., Kaliszyk, C., Urban, J.: Online machine learning techniques for coq: a comparison. In: Kamareddine, F., Sacerdoti Coen, C. (eds.) CICM 2021. LNCS (LNAI), vol. 12833, pp. 67–83. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81097-9_5

gym-saturation: Gymnasium Environments for Saturation Provers (System description)

Boris Shminke(✉)

Université Côte d'Azur, CNRS, LJAD, Nice, France
boris.shminke@univ-cotedazur.fr

Abstract. This work describes a new version of a previously published Python package — gym-saturation: a collection of OpenAI Gym environments for guiding saturation-style provers based on the given clause algorithm with reinforcement learning. We contribute usage examples with two different provers: Vampire and iProver. We also have decoupled the proof state representation from reinforcement learning per se and provided examples of using a known ast2vec Python code embedding model as a first-order logic representation. In addition, we demonstrate how environment wrappers can transform a prover into a problem similar to a multi-armed bandit. We applied two reinforcement learning algorithms (Thompson sampling and Proximal policy optimisation) implemented in Ray RLlib to show the ease of experimentation with the new release of our package.

Keywords: Automated theorem proving · Reinforcement learning · Saturation-style proving · Machine learning

1 Introduction

This work describes a new version (0.10.0, released 2023.04.25) of a previously published [28] Python package — gym-saturation[1]: a collection of OpenAI Gym [6] environments for guiding saturation-style provers (using the given clause algorithm) with reinforcement learning (RL) algorithms. The new version partly implements the ideas of our project proposal [29]. The main changes from the previous release (0.2.9, on 2022.02.26) are:

- guiding two popular provers instead of a single experimental one (Sect. 3)
- pluggable first-order logic formulae embeddings support (Sect. 4)

[1] https://pypi.org/project/gym-saturation/.

This work has been supported by the French government, through the 3IA Côte d'Azur Investment in the Future project managed by the National Research Agency (ANR) with the reference numbers ANR-19-P3IA-0002.

– examples of experiments with different RL algorithms (Sect. 5)
– following the updated Gymnasium [35] API instead of the outdated OpenAI Gym

`gym-saturation` works with Python 3.8+. One can install it by `pip install gym-saturation` or `conda install -c conda-forge gym-saturation`. Then, provided Vampire and/or iProver binaries are on `PATH`, one can use it as any other Gymnasium environment:

```
import gymnasium

import gym_saturation

# v0 here is a version of the environment class, not the prover
env = gymnasium.make("Vampire-v0")  # or "iProver-v0"
# edit and uncomment the following line to set a non-default problem
# env.set_task("a-TPTP-problem-path")
observation, info = env.reset()
print("Starting proof state:")
env.render()
# truncation means finishing an episode in a non-terminal state
# e.g. because of the externally imposed time limit
terminated, truncated = False, False
while not (terminated or truncated):
    # apply policy (e.g. a random available action)
    action = env.action_space.sample(mask=observation["action_mask"])
    print("Given clause:", observation["real_obs"][action])
    observation, reward, terminated, truncated, info = env.step(action)
print("Final proof state:")
env.render()
env.close()
```

2 Related Work

Guiding provers with RL is a hot topic. Recent projects in this domain include TRAIL (Trial Reasoner for AI that Learns) [2], FLoP (Finding Longer Proofs) [37], and lazyCoP [26]. We will now compare the new `gym-saturation` features with these three projects.

Usually, one guides either a new prover created for that purpose (lazyCoP; FLoP builds on fCoP [14], an OCaml rewrite of older leanCoP [19]) or an experimental patched version of an existing one (TRAIL relies on a modified E [27]). Contrary to that, `gym-saturation` works with unmodified stable versions of Vampire [15] and iProver [10].

In addition, known RL-guiding projects are prover-dependent: FLoP could, in principle, work with both fCoP and leanCoP but reported only fCoP experiments. TRAIL claims to be reasoner-agnostic, but to our best knowledge, no one has tried it with anything but a patched E version it uses by default. [26] mentions an anonymous reviewer's suggestion to create a standalone tool for other existing systems, but we are not aware of further development in this direction. Quite the contrary, we have tested gym-saturation compatibility with two different provers (Vampire and iProver).

Deep learning models expect their input to be real-valued tensors and not, for example, character strings in the TPTP [32] language. Thus, one always uses a *representation* (or *embeddings*) — a function mapping a (parsed) logic formula to a real vector. In lazyCoP and FLoP parts of embedding functions belong to the underlying provers, making it harder to vary and experiment with (e.g., one needs Rust or OCaml programming skills to do it). gym-saturation leaves the choice of representation open and supports any mapping from TPTP-formatted string to real vectors. The version described in this work also provides a couple of default options.

3 Architecture and Implementation Details

3.1 Architecture

gym-saturation is compatible with Gymnasium [35], a maintained fork of now-outdated OpenAI Gym standard of RL-environments, and passes all required environment checks. As a result of our migration to Gymnasium, its maintainers featured gym-saturation in a curated list of third-party environments[2].

Previously, gym-saturation guided an experimental pure Python prover [28] which happened to be too slow and abandoned in favour of existing highly efficient provers: Vampire and iProver.

Although the gym-saturation user communicates with both iProver and Vampire in the same manner, under the hood, they use different protocols. For Vampire, we relied on the so-called manual (interactive) clause selection mode implemented several years ago for an unrelated task [11]. In this mode, Vampire interrupts the saturation loop and listens to standard input for a number of a given clause instead of applying heuristics. Independent of this mode, Vampire writes (or not, depending on the option show_all) newly inferred clauses to its standard output. Using Python package pexpect, we attach to Vampire's standard input and output, pass the action chosen by the agent to the former and read observations from the latter. In manual clause selection mode, Vampire works like a server awaiting a request with an action to which it replies (exactly what an environment typically does).

iProver recently added support of being guided by external agents. An agent has to be a TCP server satisfying a particular API specification. So, iProver behaves as a client which sends a request with observations to some server and awaits a reply containing an action. To make it work with gym-saturation, we implemented a *relay server*. It accepts a long-running TCP connection from a running iProver thread and stores its requests to a thread-safe queue, and pops

[2] https://gymnasium.farama.org/environments/third_party_environments/.

a response to it from another such queue filled by `gym-saturation` thread. See Fig. 1 for a communication scheme.

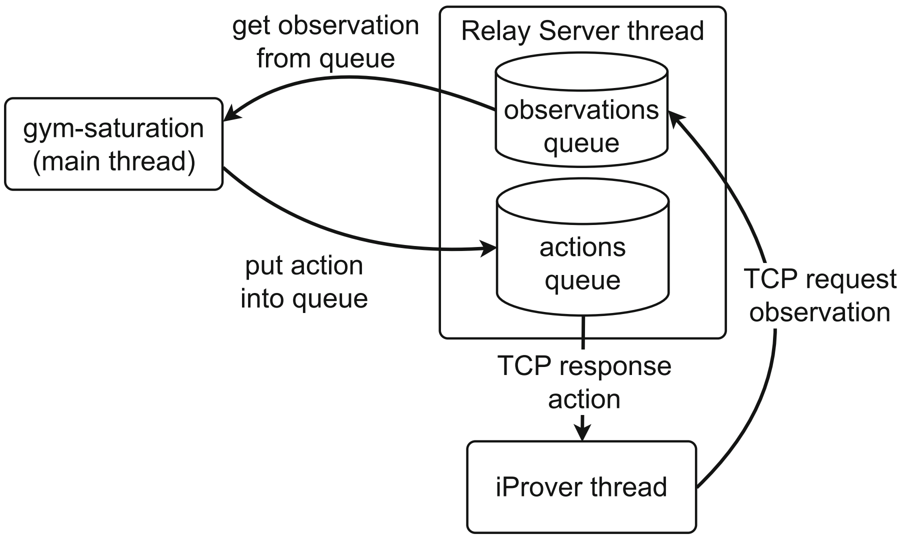


Fig. 1. `gym-saturation` interacting with iProver

3.2 Implementation Details

Clause Class. A clause is a Python data class having the following keys and respective values:

- `literals` — a string of clause literals in the TPTP format, e.g. `'member(X0,bb) | member(X0,b)'`
- `label` — a string label of a clause, e.g. '21'. Some provers (e.g. Vampire) use integer numbers for labelling clauses, but others (e.g. iProver) use an alphanumeric mixture (e.g. 'c_54')
- `role` — a string description of a clause role in a proof (hypothesis, negated conjecture, axiom, et cetera)
- `inference_rule` — a string name of an inference rule used to produce the clause. It includes not only resolution and superposition but also values like 'axiom' and 'input' (for theorem assumptions)
- `inference_parents` — a tuple of clause labels if needed by the inference rule ('axiom' doesn't need any, 'factoring' expects only one, 'resolution' — two, et cetera)
- `birth_step` — an integer step number when the clause appeared in the proof state. Axioms, assumptions, and the negated conjecture have birth step zero.

All these fields except the birth_step (computed by the environment itself) are already available as separate entities (and not parts of TPTP-formatted strings) in iProver and Vampire output.

Environment Class

Observation is a Python dictionary with several keys:

– real_obs is a tuple of all clauses (processed and unprocessed). It can be transformed to tensor representation by so-called observation wrappers[3]. The gym-saturation provides several such wrappers for cases of external embeddings service or hand-coded feature extraction function
– action_mask is a numpy [13] array of the size max_clauses (a parameter which one can set during the environment object instantiation) having a value 1.0 at index i if and only if a clause with a zero-based order number i currently exists and can be a given clause (e.g. not eliminated as redundant). All other values of action_mask are zeros. This array simplifies tensor operations on observation representations.

Limiting the total number of clauses in a proof state is a proxy of both random-access memory (each clause needs storage space) and time (a prover has to process each clause encountered) limits typical for the CASC [33] competition. One can add a standard Gymnasium time-limit wrapper to limit the number of steps in an episode. Setting wall-clock time and RAM limits is not typical for RL research.

Action is a zero-based order number of a clause from real_obs. If a respective action_mask is zero, an environment throws an exception during the execution of the step method.

Reward is 1.0 after a step if we found the refutation at this step and 0.0 otherwise. One can change this behaviour by either Gymnasium reward wrappers or by collecting trajectories in a local buffer and postprocessing them before feeding the trainer.

Episode is terminated when an empty clause $false appears in the proof state or if there are no more available actions.

Episode is truncated when there are more than max_clauses clauses in the proof state. Since the state is an (extendable) tuple, we don't raise an exception when a prover generates a few more clauses.

Info dictionary is always empty at every step by default.

[3] https://gymnasium.farama.org/api/wrappers/observation_wrappers/.

Render modes of the environment include two standard ones (`‘human’` and `‘ansi’`), the first one printing and the second one returning the same TPTP formatted string.

Multi-task Environment. The latest `gym-saturation` follows a Meta-World benchmark [36] style and defines `set_task` method with one argument — a TPTP problem full path. If one resets an environment without explicitly setting a task in advance, the environment defaults to a simple group theory problem (any idempotent element equals the identity). Having a default task helps us keep compatibility with algorithms not aware of multi-task RL. One can inherit from `gym-saturation` environment classes to set a random problem at every reset or implement any other desirable behaviour.

4 Representation Subsystem

4.1 Existing First-Order Formulae Representations and Related Projects

As mentioned in Sect. 2, to apply any deep reinforcement learning algorithm, one needs a representation of the environment state in a tensor form first. There are many known feature engineering procedures. It can be as simple as clause age and weight [25], or information extracted from a clause syntax tree [18] or an inference lineage of a clause [30]. Representing logic formulae as such is an active research domain: for example, in [23], the authors proposed more than a dozen different embedding techniques based on formulae syntax. In communities other than automated deduction, researchers also study first-order formulae representation: for example, in [5], the authors use semantics representation rather than syntax. One can also notice that first-order logic (FOL) is nothing more than a formal language, so abstract syntax trees of FOL are not, in principle, that different from those of programming language statements. And of course, encoding models for programming languages (like `code2vec` [4] for Java) exist, as well as commercially available solutions as GPT-3 [7] generic code embeddings and comparable free models like LLaMA [34].

To make the first step in this direction, we took advantage of existing pre-trained embedding models for programming languages and tried to apply them to a seemingly disconnected domain of automated provers.

4.2 `ast2vec` and Our Contributions to It

In [20], the authors proposed a particular neural network architecture they called *Recursive Tree Grammar Autoencoders (RTG-AE)*, which encodes abstract syntax trees produced by a programming language parser into real vectors. Being interested in education applications, they also published the pre-trained model for Python [21]. To make use of it for our purpose, we furnished several technical improvements to their code (our contribution is freely available[4]):

[4] https://gitlab.com/inpefess/ast2vec.

– a TorchServe [24] handler for HTTP POST requests for embeddings
– request caching with the Memcached server [9]
– Docker container to start the whole subsystem easily on any operating system

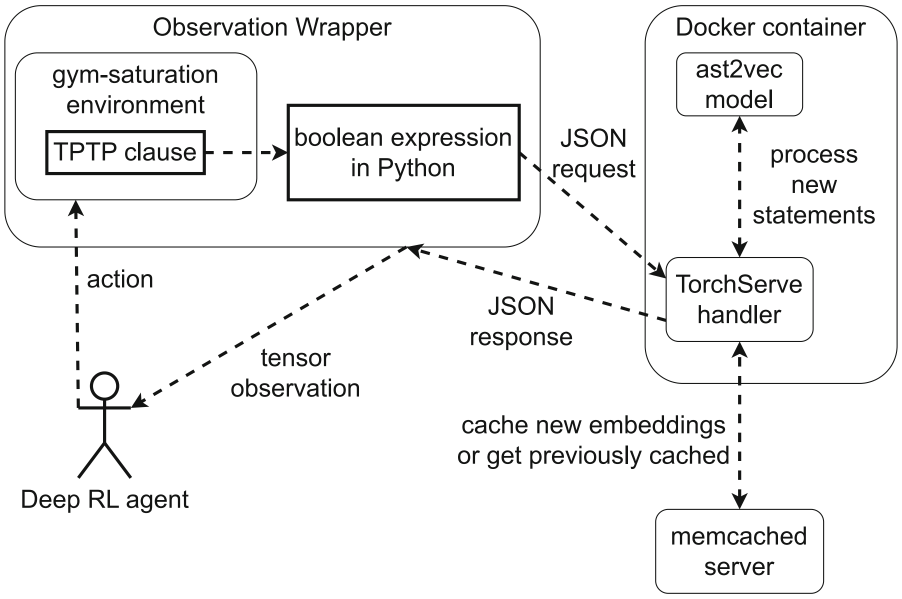


Fig. 2. gym-saturation communication with ast2vec

To integrate the `ast2vec` server with `gym-saturation` environments, we added Gymnasium observation wrappers, one of them mapping a clause in the TPTP language to a boolean-valued statement in Python (in particular, by replacing logic operation symbols, e.g. = in TPTP becomes == in Python). See Fig. 2 for a communication diagram. In principle, since a clause doesn't contain any quantifiers explicitly, one can rewrite it as a boolean-valued expression in many programming languages for which pre-trained embeddings might exist.

4.3 Latency Considerations

Looking at Fig. 2, one might wonder how efficient is such an architecture. The average response time observed in our experiments was 2 ms (with a 150 ms maximum). A typical natural language processing model which embeds whole texts has a latency from 40 ms to more than 600 ms [17] (depending on the model complexity and the length of a text to embed) when run on CPU, so there is no reason to believe that `ast2vec` is too slow. When evaluating a prover, one usually fixes the time limit: for example, 60 s is the default value for Vampire. Being written in C++ and with a cornucopia of optimisation tweaks, Vampire can generate around a million clauses during this relatively short timeframe. Thus, to be on par with Vampire, a representation service must have latency around 60 µs (orders of magnitude faster than we have). There can be several ways to lower the latency:

– inference in batches (one should train the embedding model to do it; `ast2vec` doesn't do it out of the box). The improvement may vary
– use GPU. NVIDIA reports around 20x improvement vs CPU [16]. However, throwing more GPUs won't be as efficient without batch inference from the previous point
– request an embedding for a binary object of an already parsed clause instead of a TPTP string. It means not repeating parsing already done by a prover, which might lower the latency substantially. To do this, one will have to patch an underlying prover to return binary objects instead of TPTP strings
– use RPC (remote procedure call) instead of REST protocol. TorchServe relies on REST and parcels in JSON format, and in gRPC [12], they prefer the binary `protobuf` format. One rarely expects sub-millisecond latency from REST, although for RPC, 150 µs is not unusual. This point doesn't make much sense without the previous one

5 Usage Examples

We provide examples of experiments easily possible with `gym-saturation` as a supplementary code to this paper[5]. We don't consider these experiments as being of any scientific significance per se, serving merely as illustrations and basic usage examples. Tweaking the RL algorithms' meta-parameters and deep neural network architectures is out of the scope of the present system description.

We coded these experiments in the Ray framework, which includes an RLlib — a library of popular RL algorithms. The Ray is compatible with Tensorflow [1] and PyTorch [22] deep learning frameworks, so it doesn't limit a potential `gym-saturation` user by one.

In the experiments, we try to solve `SET001-1` from the TPTP with `max_clauses=20` (having no more than twenty clauses in the proof state) for guiding Vampire and `max_clauses=15` for iProver. This difference is because even a random agent communicating to iProver manages to always solve `SET001-1` by generating no more than twenty clauses. We wanted training to start, but keep the examples as simple as possible, so we chose to harden the constraints instead of moving on to a more complicated problem.

In one experiment, we organise clauses in two priority queues (by age and weight) and use an action wrapper to map from a queue number (0 or 1) to the clause number. That means we don't implant these queues inside provers but follow a Gymnasium idiomatic way to extend environments. Of course, Vampire and iProver have these particular queues as part of their implementation, but our illustration shows one could use any other priorities instead. It transforms

[5] https://github.com/inpefess/ray-prover/releases/tag/v0.0.3.

our environment into a semblance of a 2-armed bandit, and we use Thompson sampling [3] to train. This experiment reflects ideas similar to those described in [31].

In another experiment, we use `ast2vec` server for getting clause embeddings and train a Proximal Policy Optimisation (PPO) algorithm as implemented in the Ray RLlib. The default policy network there is a fully connected one, and we used 256×20 tensors as its input (256 is an embedding size in `ast2vec`, and 20 is the maximal number of clauses we embed). So, the policy chooses a given clause given the embeddings of all clauses seen up to the current step (including those already chosen or judged to be redundant/subsumed). Such an approach is more similar to [37].

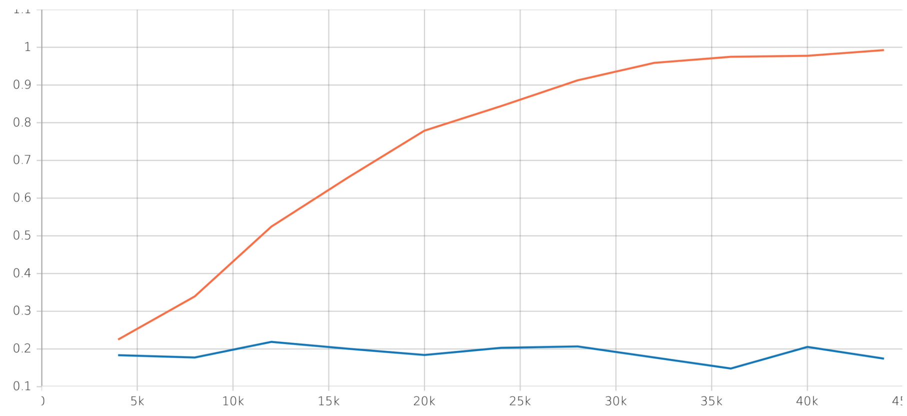


Fig. 3. Episode reward mean vs the total number of steps. The blue line is for a random agent and the orange one — for the PPO. Both agents guide Vampire (Color figure online)

We provide Fig. 3 as a typical training process chart.

6 Conclusion and Future Work

We contributed a new version of `gym-saturation`, which continued to be free and open-source software, easy to install and use while promising assistance in setting up experiments for RL research in the automated provers domain. In the new version, we enabled anyone interested to conduct experiments with RL algorithms independently of an underlying prover implementation. We also added the possibility of varying representations as external plug-ins for further experimentation. We hope that researchers having such an instrument can focus on more advanced questions, namely how to generate and prioritise training problems to better transfer search patterns learned on simpler theorems to harder ones.

Our experience with adding Vampire and iProver support to `gym-saturation` shows that working tightly with corresponding prover developers is not mandatory, although it might help immensely. Implementing the prover guidance through the standard I/O (as in Vampire) seems to be relatively easy, and we hope more provers will add similar functionality in future to be more ML-friendly. Such provers could then profit from using any other external guidance (see [8] for a different system using the same iProver technical features as we did).

We identify a discerning and computationally efficient representation service as a bottleneck for our approach and envision an upcoming project of creating a universal first-order logic embedding model usable not only by saturation-style provers but also tableaux-based ones, SMT-solvers, semantic reasoners, and beyond.

Acknowledgements. We would like to thank Konstantin Korovin for the productive discussion and for adding the external agents' communication feature to iProver, without which this work won't be possible. We also thank anonymous reviewers for their meticulous suggestions on improving the present paper.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). https://www.tensorflow.org/. Software available from tensorflow.org
2. Abdelaziz, I., et al.: Learning to guide a saturation-based theorem prover. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 738–751 (2023). https://doi.org/10.1109/TPAMI.2022.3140382
3. Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 127–135. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013). https://proceedings.mlr.press/v28/agrawal13.html
4. Alon, U., Zilberstein, M., Levy, O., Yahav, E.: Code2Vec: learning distributed representations of code. Proceed. ACM Programm. Lang. **3**(POPL), 1–29 (2019). https://doi.org/10.1145/3290353
5. Ballout, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Learning to classify logical formulas based on their semantic similarity. In: Aydoğan, R., Criado, N., Lang, J., Sanchez-Anguix, V., Serramia, M. (eds.) PRIMA 2022: Principles and Practice of Multi-Agent Systems, pp. 364–380. PRIMA 2022. LNCS, vol. 13753. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-21203-1_22
6. Brockman, G., et al.: OpenAI Gym. arXiv (2016).https://doi.org/10.48550/arXiv.1606.01540
7. Brown, T.B., et al.: Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS2020, Curran Associates Inc., Red Hook, NY, USA (2020). https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
8. Chvalovský, K., Korovin, K., Piepenbrock, J., Urban, J.: Guiding an instantiation prover with graph neural networks. In: Piskac, R., Voronkov, A. (eds.) Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence

and Reasoning. EPiC Series in Computing, vol. 94, pp. 112–123. EasyChair (2023). https://doi.org/10.29007/tp23. https://easychair.org/publications/paper/5z94
9. Danga Interactive Inc: Memcached (2023). https://github.com/memcached/memcached
10. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 388–397. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_24
11. Gleiss, B., Kovács, L., Schnedlitz, L.: Interactive visualization of saturation attempts in vampire. In: Ahrendt, W., Tapia Tarifa, S.L. (eds.) IFM 2019. LNCS, vol. 11918, pp. 504–513. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34968-4_28
12. gRPC authors: gRPC - An RPC library and framework (2023). https://github.com/grpc/grpc
13. Harris, C.R., et al.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2
14. Kaliszyk, C., Urban, J., Vyskočil, J.: Certified connection tableaux proofs for HOL light and TPTP. In: Proceedings of the 2015 Conference on Certified Programs and Proofs, pp. 59–66. CPP 2015, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2676724.2693176
15. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
16. Mukherjee, P., Weill, E., Taneja, R., Onofrio, D., Ko, Y.J., Sharma, S.: Real-time natural language understanding with BERT using TensorRT (2019). https://developer.nvidia.com/blog/nlu-with-tensorrt-bert/
17. Nguyen, V., Srihari, N., Chadha, P., Chen, C., Lee, J., Rodge, J.: Optimizing T5 and GPT-2 for real-time inference with NVIDIA TensorRT (2021). https://developer.nvidia.com/blog/optimizing-t5-and-gpt-2-for-real-time-inference-with-tensorrt/
18. Olsák, M., Kaliszyk, C., Urban, J.: Property invariant embedding for automated reasoning. In: Giacomo, G.D. et al. (eds.) ECAI 2020–24th European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 1395–1402. IOS Press (2020). https://doi.org/10.3233/FAIA200244
19. Otten, J., Bibel, W.: leanCoP: lean connection-based theorem proving. J. Symb. Comput. **36**(1), 139–161 (2003). https://doi.org/10.1016/S0747-7171(03)00037-3. First Order Theorem Proving
20. Paaßen, B., Koprinska, I., Yacef, K.: Recursive tree grammar autoencoders. Mach. Learn. **111**, 3393–3423 (2022). https://doi.org/10.1007/s10994-022-06223-7
21. Paassen, B., McBroom, J., Jeffries, B., Koprinska, I., Yacef, K.: Mapping python programs to vectors using recursive neural encodings. J. Educ. Data Min. **13**(3), 1–35 (2021). https://doi.org/10.5281/zenodo.5634224. https://jedm.educationaldatamining.org/index.php/JEDM/article/view/499
22. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019). https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
23. PurgaŁ, S., Parsert, J., Kaliszyk, C.: A study of continuous vector representations for theorem proving. J. Logic Comput. **31**(8), 2057–2083 (2021). https://doi.org/10.1093/logcom/exab006

24. PyTorch serve contributors: TorchServe (2023). https://github.com/pytorch/serve
25. Rawson, M., Reger, G.: Old Or heavy? Decaying gracefully with age/weight shapes. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 462–476. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_27
26. Rawson, M., Reger, G.: lazyCoP: lazy paramodulation meets neurally guided search. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 187–199. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_11
27. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
28. Shminke, B.: Gym-saturation: an OpenAI Gym environment for saturation provers. J. Open Source Softw. **7**(71), 3849 (2022). https://doi.org/10.21105/joss.03849
29. Shminke, B.: Project proposal: a modular reinforcement learning based automated theorem prover. arXiv (2022). https://doi.org/10.48550/ARXIV.2209.02562
30. Suda, M.: Improving ENIGMA-style clause selection while learning from history. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 543–561. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_31
31. Suda, M.: Vampire getting noisy: will random bits help conquer chaos? (System description). In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning. IJCAR 2022. LNCS, vol. 13385, pp. 659–667. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10769-6_38
32. Sutcliffe, G.: The TPTP problem library and associated infrastructure - from CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
33. Sutcliffe, G.: The 10th IJCAR automated theorem proving system competition - CASC-J10. AI Commun. **34**(2), 163–177 (2021). https://doi.org/10.3233/AIC-201566
34. Touvron, H., et al.: LLaMA: open and efficient foundation language models. arXiv (2023). https://doi.org/10.48550/arXiv.2302.13971
35. Towers, M., et al.: Gymnasium (2023). https://doi.org/10.5281/zenodo.8127026
36. Yu, T., et al.: Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. In: Kaelbling, L.P., Kragic, D., Sugiura, K. (eds.) Proceedings of the Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 100, pp. 1094–1100. PMLR (30 Oct-01 Nov 2020). https://proceedings.mlr.press/v100/yu20a.html
37. Zombori, Z., Csiszárik, A., Michalewski, H., Kaliszyk, C., Urban, J.: Towards finding longer proofs. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 167–186. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_10

Non-wellfounded Proofs

A Linear Perspective on Cut-Elimination for Non-wellfounded Sequent Calculi with Least and Greatest Fixed-Points

Alexis Saurin(✉)

IRIF, CNRS, Université de Paris Cité & INRIA, Paris, France
alexis.saurin@irif.fr

Abstract. This paper establishes cut-elimination for $\mu\mathsf{LL}^\infty$, $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$, that are non-wellfounded sequent calculi with least and greatest fixed-points, by expanding on prior works by Santocanale and Fortier [20] as well as Baelde et al. [3,4]. The paper studies a fixed-point encoding of LL exponentials in order to deduce those cut-elimination results from that of $\mu\mathsf{MALL}^\infty$. Cut-elimination for $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$ is obtained by developing appropriate linear decorations for those logics.

Keywords: LL · μ-calculus · Non-wellfounded proofs · cut elimination

1 Introduction

On the Non-Wellfounded Proof-Theory of Fixed-Point Logics. In the context of logics with induction and coinduction (such as logics with inductive definitions *à la* Martin Löf [6,9,10,25], or variants of the μ-calculus [11,22,23]), the need for a (co)inductive invariant (in the form of the Park's rule for induction) is replaced by the ability to pursue the proof infinitely, admitting non-wellfounded branches, when considering non-wellfounded and circular proofs (also called cyclic, or regular proofs, since the proof tree is a regular tree, with finitely many distinct subtrees). In such frameworks, sequent proofs may be finitely branching but non-wellfounded derivation trees and infinite branches shall satisfy some validity condition. (Otherwise one could derive any judgement, see Fig. 1(a).) Various validity conditions have been considered in the literature [3].

The non-wellfounded and circular proof-theory of fixed-points attracted a growing attention first motivated by proof-search [1,7,8,16–18,28] and more recently by a Curry-Howard perspective, studying the dynamics of the cut-elimination in those logics [4,20,29] where formulas correspond to (co)inductive types. Notice also that when interested in the computational content of proofs, we will not focus solely on the regular fragment as we expect, for instance, that we can write a regular program that computes a non-ultimately periodic stream.

This work was partially funded by the ANR project RECIPROG, project reference ANR-21-CE48-019-01.

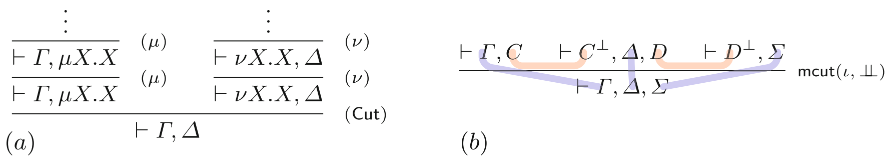


Fig. 1. (a) Example of an invalid circular pre-proof (b) Schema of the multicut rule

Cut-Elimination and LL. When studying the structure of proofs and their cut-elimination properties, LL, Girard's Linear Logic [21], is a logic of choice: the careful treatment of structural rules gives access to a lot of information and a fine-grained control over cut-reduction. The constrained use of structural rules indeed renders the cut-elimination theorem more informative than in LJ and of course LK. Interestingly it provided a positive feedback on the understanding of LJ and LK: by decorating intuitionistic and classical proofs with enough exponential modalities $(!, ?)$, they can become LL proofs and one can therefore refine the original cut-elimination relations [12,21]. This approach impacted the understanding of evaluation strategies of programming languages such as call-by-name and call-by-value notably. Another way to view this is by noting that, in LK, the additive and multiplicative presentations of conjunction (resp. disjunction) can be shown to be interderivable thanks to structural rules. This fails in LL and it is the reason why LL has well-established additive – $\oplus, \&, \top, 0$ – (resp. multiplicative $⅋, \otimes, \perp, 1$) *fragments*. It is the role of the exponential fragment to relate the additive and multiplicative worlds, by mean of the fundamental equivalence: $!A \otimes !B \dashv\vdash !(A \& B)$ (and its dual, $?A ⅋ ?B \dashv\vdash ?(A \oplus B)$). The exponential modalities are precisely introduced where structural rules are needed to restore the equivalence between the additive and multiplicative conjunctions; in categorical models of LL [26], this principle is referred to as Seely isomorphisms.

Cut-Elimination for Non-Wellfounded Proofs. Proving cut-elimination results for non-wellfounded proofs in the presence of least and greatest fixed-points requires to use reasoning techniques coping with the non-inductive structure of the considered formulas (fixed-points formulas regenerate) and proof objects (which are non-wellfounded). For instance, Santocanale and Fortier [20] proved cut-elimination for the regular fragment of non-wellfounded proofs of purely additive linear logic with fixed points, $\mu\mathsf{ALL}^\infty$, while Baelde *et al.* [4] proved cut-elimination for non-wellfounded proofs with additive and multiplicative connectives, $\mu\mathsf{MALL}^\infty$. In both cases, the proof relies on a generalization of the cut-rule, the *multicut* rule (which abstracts a portion of a proof tree constituted only of cut inferences see Fig. 1(b)) and on a reasoning by contradiction to prove that one can eliminate cuts at the limit of an infinite cut reduction sequence, while preserving the validity condition. Baelde *et al.* [3,4] use a so-called "locative" approach by modelling sequents as sets of formulas paired with addresses which determines uniquely the formula occurrence in a sequent and makes explicit the ancestor relation used to trace the progress along branches.

Moreover, the cut-elimination proof proceeds by a rather complex semantical, roundabout, argument relying on a soundness theorem.

In a slightly different direction, Das and Pous [15] proved a cut-elimination result for Kleene algebras and their variants. This can be viewed as a non-commutative version of intuitionistic MALL with a particular form of inductive construction, Kleene's star. Kuperberg et al [24] and more specifically Pinault's PhD thesis [27] as well as Das [13] examine non-wellfounded versions of System T based on [15], exploring the computational content of non-wellfounded proofs.

Neither Santocanale and Fortier's [20,29], nor Baelde et al. [3,4] works captured full linear logic: the exponentials are missing and the proofs cannot deal with them in a simple way. Indeed, the proof for $\mu\mathsf{ALL}$ strongly relies on the assumption the sequents are pairs of formulas $(A \vdash B)$ while in $\mu\mathsf{MALL}$, the locative approach taken by Baelde et al. is not well-suited to work with structural rules: the extension of the proof would be possible though highly technical. In contrast, our motto in the present work is to work with traditional sequents as lists of formulas and to exploit the (co)inductive nature of LL exponentials.

On the (Co)Inductive Nature of Exponential Modalities in Linear Logic. The original works by Baelde and Miller on fixed-points in linear logic [2,5] focus on $\mu\mathsf{MALL}$ only and present an encoding of the exponential modalities of LL using least and greatest fixed points. Indeed, the ? and ! modalities have an infinitary character which is well-known from the early days of linear logic (see Section V.5 of Girard's seminal paper [21]) and which is in fact respectively inductive for ? and coinductive for !; let us discuss it briefly here.

One can decide to contract a ?-statement any *finite* number of times before it is ultimately weakened or derelicted. It is therefore natural to represent $?A$ with formula $?^\bullet A = \mu X.A \oplus (\bot \oplus (X ⅋ X))$: A allows for dereliction, $\bot$ for weakening and $X ⅋ X$ will regenerate, by unfolding, two copies of $?^\bullet A$, making the contraction derivable. The $\oplus$ and μ connectives respectively provide the ability to choose either of those three inferences and to repeat finitely this process.

On the other hand, a !-formula is a formula which, during cut-elimination, shall maintain a proper interaction with *any number* of contractions, weakenings or derelictions: a proof concluded with a promotion shall be able to react to any number of duplications or erasure before the promotion actually interact with a dereliction to open the *exponential box*: from that follows the coinductive character of $!A$ modelled as $!^\bullet A = \nu X.A \& (1 \& (X \otimes X))$.

As discussed above and formally established by Baelde and Miller [5], the exponential rules can be derived in the finitary sequent calculus $\mu\mathsf{MALL}$: to any LL provable sequent can be associated a provable $\mu\mathsf{MALL}$ sequent via the above translations of the exponentials. However, until now one can hardly say more about this embedding for two deep reasons: (i) the fundamental Seely isomorphisms which relate the additive and multiplicative versions of conjunction (resp. disjunction) are still derivable through this encoding but they are no more isomorphisms and (ii) on the provability level as well, the encoding is not faithful: the $\mu\mathsf{MALL}$ provability of the translation of an LL sequent s does not entail the LL provability of s itself (counter-example due to Das [14]). A contribution of the present paper is to put to work Baelde and Miller's encoding, showing that, in the case of non-wellfounded proofs, its structure is faithful enough to extract information of the cut-reduction behaviour of the logic.

Contributions and Organization of the Paper. The main result of this paper is a cut-elimination theorem for $\mu\mathsf{LL}^\infty$, the non-wellfounded sequent calculus for linear logic extended with least and greatest fixed points. Our proof proceeds by encoding LL exponentials in $\mu\mathsf{MALL}^\infty$ and studying $\mu\mathsf{LL}^\infty$ cut-reduction sequences through their simulation in $\mu\mathsf{MALL}^\infty$ which may be a *transfinite* sequence. In Sect. 2, we introduce our logics, $\mu\mathsf{MALL}^\infty$, $\mu\mathsf{LL}^\infty$, $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$, altogether with their non-wellfounded proofs and validity conditions. We adapt $\mu\mathsf{MALL}^\infty$ cut-elimination theorem [4] to our setting where sequents are lists and prove a compression lemma for $\mu\mathsf{MALL}^\infty$ transfinite cut-reduction sequences. Section 3 constitutes the core of our paper: we define $\mu\mathsf{LL}^\infty$ cut-reduction rules, study the encoding of exponentials in $\mu\mathsf{MALL}^\infty$ and show that $\mu\mathsf{LL}^\infty$ cut-reduction steps can be simulated in $\mu\mathsf{MALL}^\infty$, before proving $\mu\mathsf{LL}^\infty$ cut-elimination theorem. We prove in Sect. 4, as corollaries, cut-elimination for $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$, the non-wellfounded sequent-calculi for classical and intuitionistic logic. While our result for $\mu\mathsf{LL}^\infty$ shows that any fair cut-reduction sequence produces a cut-free valid proof, our two other cut-elimination results are truly (infinitary) weak-normalization results. We finally conclude in Sect. 5 with perspectives. A major advantage of our approach is that $\mu\mathsf{MALL}^\infty$ cut-elimination proof and, to some extent, the validity conditions, are regarded as black boxes, simplifying the presentation of the proof and making it reusable wrt. other validity conditions or $\mu\mathsf{MALL}^\infty$ proof techniques. An additional by-product of our approach, to the theory of linear logic, is to illustrate the fact that Seely isomorphisms are not needed to reach a cut-free proof.

A companion technical report containing additional details on the definitions as well as full proofs is available online [30].

2 Non-Wellfounded Proofs: $\mu\mathsf{MALL}^\infty$, $\mu\mathsf{LL}^\infty$, $\mu\mathsf{LK}^\infty$, $\mu\mathsf{LJ}^\infty$

2.1 μ-Signatures and Formulas

Definition 1 (μ-signature). *A* **μ-signature** *is a set $\mathcal{C}$ of pairs (c, p) of a connective symbol c and a tuple p of elements of $\{+, -\}$. The arity of c, $\mathsf{ar}(c)$, is the length of p, while the elements of p indicate the mono/antitonicity of the connective in the given component. The empty tuple will be denoted as $()$[1].*

Example 2 (μ-signature associated with $\mu\mathsf{MALL}, \mu\mathsf{LL}, \mu\mathsf{LK}, \mu\mathsf{LJ}$). The μ-signatures associated with $\mu\mathsf{MALL}, \mu\mathsf{LL}, \mu\mathsf{LK}, \mu\mathsf{LJ}$ are:

[1] μ-signature can be enriched to consider quantifiers but we restrict to the propositional case here.

- μMALL signature: $\mathcal{C}_{\mu\mathsf{MALL}} = \{⅋, \otimes, \oplus, \&\} \times \{(+,+)\} \cup \{0, 1, \top, \bot\} \times \{()\}$;
- one-sided μLL signature: $\mathcal{C}_{\mu\mathsf{LL}_1} = \mathcal{C}_{\mu\mathsf{MALL}} \cup \{!, ?\} \times \{(+)\}$;
- two-sided μLL signature: $\mathcal{C}_{\mu\mathsf{LL}_2} = \mathcal{C}_{\mu\mathsf{LL}_1} \cup \{(\multimap, (-,+)), (\cdot^{\bot}, (-))\}$;
- μLK signature: $\mathcal{C}_{\mu\mathsf{LK}} = \{\wedge, \vee\} \times \{(+,+)\} \cup \{(\Rightarrow, (-,+))\} \cup \{\top, \mathsf{F}\} \times \{()\}$;
- μLJ signature: $\mathcal{C}_{\mu\mathsf{LJ}} = \mathcal{C}_{\mu\mathsf{LK}}$.

Definition 3 (Pre-formulas). *Given a μ-signature $\mathcal{C}$, a countable set $\mathcal{V}$ of fixed-point variables and a set of atomic formulas $\mathcal{A}$, the set of* **pre-formulas** *over $\mathcal{S}$ is defined as the least set $\mathcal{F}_{\mathcal{S}}$ such that: (α) $\mathcal{A} \cup \mathcal{V} \subseteq \mathcal{F}_{\mathcal{S}}$; (β) for every c of arity n in $\mathcal{C}$ and $F_1, \ldots, F_n \in \mathcal{F}_{\mathcal{S}}$, $c(F_1, \ldots, F_n) \in \mathcal{F}_{\mathcal{S}}$; (γ) for every $X \in \mathcal{V}$ and pre-formula $F \in \mathcal{F}_{\mathcal{S}}$, $\mu X.F \in \mathcal{F}_{\mathcal{S}}$ and $\nu X.F \in \mathcal{F}_{\mathcal{S}}$.*

Definition 4 (Positive and negative occurrences of a variable). *Given a μ-signature $\mathcal{C}$ and a fixed-point variable $X \in \mathcal{V}$, one defines by induction on pre-formulas the fact, for X, to occur positively (resp. negatively) in a pre-formula : (α) X occurs positively in X; (β) X occurs positively (resp. negatively) in $c(F_1, \ldots, F_n)$, for $(c, p) \in \mathcal{C}$, if there is some $1 \leq i \leq n$ such that X occurs positively (resp. negatively) in F_i and $p_i = +$ or there is some $1 \leq i \leq n$ such that X occurs negatively (resp. positively) in F_i and $p_i = -$; (γ) X occurs positively (resp. negatively) in $\sigma Y.F$, for $\sigma \in \{\mu, \nu\}$, if $Y \neq X$ and X occurs positively (resp. negatively) in F.*

Definition 5 (μ-formula). *A μ-formula F over a signature $\mathcal{S}$ is a pre-formula containing no free fixed-point variable and such that for any sub-pre-formula of F of the form $\sigma X.G$, all occurrences of X in G are positive.*

Definition 6. *One-sided μLL formulas are those formulas defined over the signature $\mathcal{C}_{\mu\mathsf{LL}_1}$ together with a set of atomic formulas $\{a, a^{\bot} \mid a \in \mathcal{A}\}$ for a countable set $\mathcal{A}$. Negation $(\_)^{\bot}$ is the involution on pre-formulas defined by:*
$(a^{\bot})^{\bot} = a$; $\bot^{\bot} = 1$; $\top^{\bot} = 0$; $(F ⅋ G)^{\bot} = F^{\bot} \otimes G^{\bot}$; $(F \oplus G)^{\bot} = F^{\bot} \& G^{\bot}$; $(?\,F)^{\bot} = !\,F^{\bot}$; $X^{\bot} = X$; $(\nu X.F)^{\bot} = \mu X.F^{\bot}$.

Definition 7 (μ-Fischer-Ladner subformulas). *Given a μ-signature $\mathcal{C}$ and a μ-formula F, $FL(F)$ is the least set of formulas such that:*

- *$F \in FL(F)$;*
- *$c(F_1, \ldots, F_n) \in FL(F) \Rightarrow F_1, \ldots, F_n \in FL(F)$ for $c \in \mathcal{C}$;*
- *$\sigma X.B \in FL(F) \Rightarrow B[\sigma X.B/X] \in FL(F)$ for $\sigma \in \{\mu, \nu\}$.*

Example 8. Let us consider $F = \nu X.((a ⅋ a^{\bot}) \otimes (!X \otimes \mu Y.X))$. $FL(F)$ is the set $\{F, (a ⅋ a^{\bot}) \otimes (!\,F \otimes \mu Y.F), a ⅋ a^{\bot}, a, a^{\bot}, !\,F \otimes \mu Y.F, !\,F, \mu Y.F\}$.

The finiteness of $FL(F)$ makes it an adequate notion of subformula:

Proposition 9. *For any μ-signature $\mathcal{S}$ and μ-formula F, $FL(F)$ is finite.*

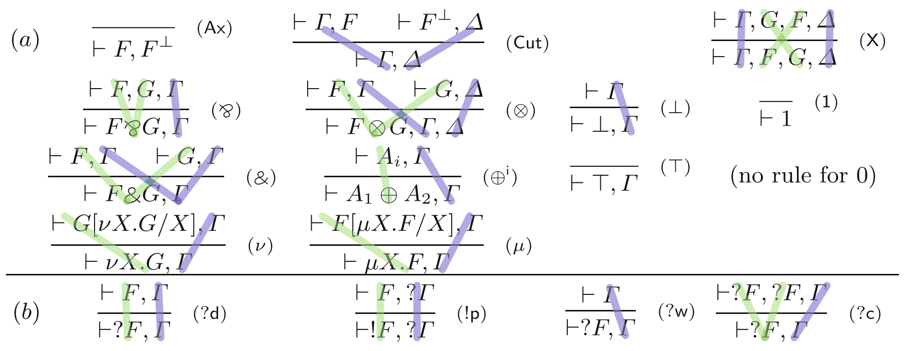

(a)
$\vdash F, F^\perp$ (Ax)
$\vdash \Gamma, F \quad \vdash F^\perp, \Delta$ / $\vdash \Gamma, \Delta$ (Cut)
$\vdash \Gamma, G, F, \Delta$ / $\vdash \Gamma, F, G, \Delta$ (X)
$\vdash F, G, \Gamma$ / $\vdash F ⅋ G, \Gamma$ (⅋)
$\vdash F, \Gamma \quad \vdash G, \Delta$ / $\vdash F \otimes G, \Gamma, \Delta$ ($\otimes$)
$\vdash \Gamma$ / $\vdash \bot, \Gamma$ ($\bot$)
$\vdash 1$ (1)
$\vdash F, \Gamma \quad \vdash G, \Gamma$ / $\vdash F \& G, \Gamma$ (&)
$\vdash A_i, \Gamma$ / $\vdash A_1 \oplus A_2, \Gamma$ ($\oplus^i$)
$\vdash \top, \Gamma$ ($\top$)
(no rule for 0)
$\vdash G[\nu X.G/X], \Gamma$ / $\vdash \nu X.G, \Gamma$ (ν)
$\vdash F[\mu X.F/X], \Gamma$ / $\vdash \mu X.F, \Gamma$ (μ)
(b)
$\vdash F, \Gamma$ / $\vdash ?F, \Gamma$ (?d)
$\vdash F, ?\Gamma$ / $\vdash !F, ?\Gamma$ (!p)
$\vdash \Gamma$ / $\vdash ?F, \Gamma$ (?w)
$\vdash ?F, ?F, \Gamma$ / $\vdash ?F, \Gamma$ (?c)

Fig. 2. (a) $\mu\mathsf{MALL}^\infty$ Inferences (b)$\mu\mathsf{LL}^\infty$ Exponential Inferences

2.2 μMALL$^\infty$, μLL$^\infty$, μLK$^\infty$ & μLJ$^\infty$ Inference Rules

Now, we define the inference rules associated with the above μ-signatures.

Definition 10 (Sequents and inferences). *A* **sequent** $s = \Gamma \vdash \Delta$ *over a μ-signature $\mathcal{S}$ is a pair of finite lists Γ, Δ of $\mathcal{S}$-formulas: Γ is the* **antecedent** *and Δ the* **succedent**. *An* **inference rule** *r, usually presented by a schema, is the data of a* **conclusion sequent**, **premise sequents**, *together with an* **ancestor relation** *relating formulas of the conclusion with formulas of the premises. A rule has a subset of distinguished* **principal formulas** *of the conclusion.*

Convention 1. *In the following, the ancestor relation will be depicted as colored lines joining related formulas. The* **principal** *formulas of an inference are the formulas which are explicitly spelled out in the conclusion sequent of an inference, not described via a context meta-variable. A formula occurrence of an inference is said to be* **active** *if it is principal or related to a principal formula by the ancestor relation. We will freely use the derived rules obtained by* **pre- and post-composition with the exchange rule**, *adapting the ancestry relation accordingly. Finally, for one-sided sequent calculi with an involutive negation $\cdot^\perp$, we may write $\Gamma \vdash \Delta$ for sequents $\vdash \Gamma^\perp, \Delta$ to clarify the computational behaviour of our examples (keeping the rule names unchanged).*

Definition 11 ($\mu\mathsf{MALL}^\infty, \mu\mathsf{LL}^\infty, \mu\mathsf{LK}^\infty, \mu\mathsf{LJ}^\infty$)**.** *$\mu\mathsf{MALL}^\infty$ inferences are given in Fig. 2. Those for one-sided $\mu\mathsf{LL}^\infty$ in Fig. 2(a) and 2(b). Those for $\mu\mathsf{LK}^\infty$ in Fig. 3. Those for $\mu\mathsf{LJ}^\infty$ by considering only inference from Fig. 3 where the succedent of both premises and conclusion sequents are singletons.*

In the above sequent calculi, every inference but the cut satisfies the subformula property *wrt.* FL-subformulae. The 2-sided $\mu\mathsf{LL}^\infty$ sequent calculus, over $\mathcal{C}_{\mu\mathsf{LL}_2}$, is defined as usual and not recalled here for space constraints.

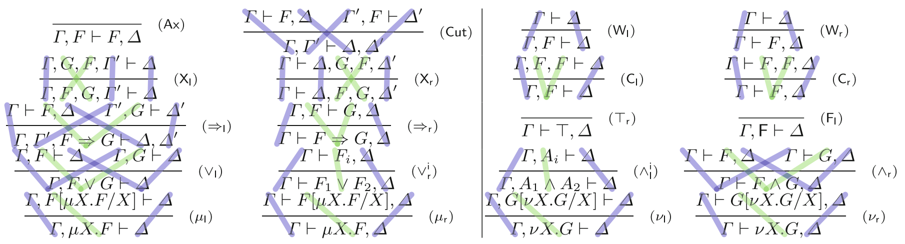


Fig. 3. $\mu\mathsf{LK}^\infty$ Two-sided Inferences

2.3 Pre-proofs and Validity Conditions

Definition 12 (Pre-proofs). *The set* $\mathsf{P}_{\mathcal{S},\mathcal{I}}$ *of* $\mathcal{I}$-**pre-proofs** *associated to some of the above* μ*-signatures* $\mathcal{S}$ *and sets of inferences* $\mathcal{I}$ *is the set of* **finite or infinite** *trees whose nodes are correctly labelled with inferences and sequents.*

Pre-proofs are equipped with a metric structure as follows: we define a ***distance*** $\mathsf{d} : \mathsf{P}_{\mathcal{S},\mathcal{I}} \times \mathsf{P}_{\mathcal{S},\mathcal{I}} \to \mathbb{R}$ as: $\mathsf{d}(\pi, \pi') = 0$ if $\pi = \pi'$ and $\mathsf{d}(\pi, \pi') = 2^{-k}$ where k is the length of the shortest position where π and π' differ otherwise.

Example 13. Consider $\mu\mathsf{LJ}$ formulas $N = \mu X.\top \vee X$ and $S = \nu X.N \wedge X$. They represent nats and streams of nats. The $\mu\mathsf{LJ}^\infty$ derivations of Fig. 4 respectively represent natural numbers, successor function, $n{::}n+1{::}n+2{::}\dots$, the double functions and the function that builds a stream enumerating the natural numbers from its input: the cut-elimination process considered below will ensure that cutting π_{k} with π_{enum} will infinitarily reduce to $\pi^{\mathsf{k}}_{\mathsf{from}}$. Figure 5 shows other examples of $\mu\mathsf{LL}^\infty$ pre-proofs, discussed with the validity condition.

The back-edge arrow to a lower sequent is notation to describe a fixed-point definition of the proof object: the subproof rooted in the source is equal to the proof rooted in the target. Trivially there is a unique solution.

In the following, we assume given a μ-signature $\mathcal{S}$ and a sequent calculus S for this signature and we shall define the valid S-proofs as a subset of S-pre-proofs, by introduction a ***thread-based validity condition***.

Definition 14 (Thread and validity). *Given a pre-proof* π *and an infinite branch* $\beta = (s_i)_{i\in\omega}$ *in* π*, a* **thread** *for* β *is an infinite sequence* θ *of formula occurrences such that* $\forall i \in \omega$*,* θ_i *is a formula occurrence of* s_i *and* θ_i *and* θ_{i+1} *are ancestor of each other.* θ *is said to* **support** β*.*

A formula F *is* **recurring** *in a thread* θ *of* β *if there are infinitely many* i *such that* θ_i *is an occurrence of* F*.*

A thread θ *is* **valid** *if it contains infinitely often the principal formula (occurrence) of a* ν *or* μ *rule and if the set of recurring formulas of* θ *has a least element (for the usual subformula ordering) which is (i) a* ν *formula when the least element occurs in the succedents or (ii) a* μ *formula if it occurs in the antecedents. A pre-proof is* **valid** *if all its infinite branches have a suffix supported by a valid thread.*

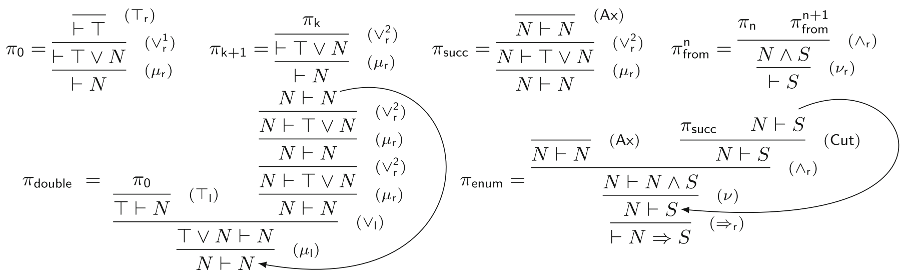


Fig. 4. Examples of $\mu\mathsf{LJ}^\infty$ pre-proofs

Example 15 ((Non-)valid pre-proofs). Consider the pre-proof in Fig. 5(a), with $F = \nu X.((a⅋a^\perp) \otimes (!X \otimes \mu Y.X))$ and $G = \mu Y.F$. The rightmost branch is supported by the green thread for which the least recurring formula is F, a ν-formula. All other branches are valid: this pre-proof is valid. Consider now the same pre-proof but with $F = \nu X.((a⅋a^\perp)\otimes(!X\otimes G))$ and $G = \mu Y.\nu X.((a⅋a^\perp)\otimes (!X \otimes Y))$. G is now a subformula of F and G, a μ-formula, and becomes the least recurring formula of all threads along the right-most infinite branch. This branch is invalid: the pre-proof is not a proof. Examples of $\mu\mathsf{LL}^\infty$ invalid pre-proofs are given in Fig. 1(a),5(b–c). In Fig. 4, π_{double} has a left thread on N while $\pi^{\mathsf{n}}_{\mathsf{from}}, \pi_{\mathsf{enum}}$ have right threads on S: they are valid.

2.4 Non-Locative $\mu\mathsf{MALL}^\infty$ Cut-Elimination Theorem

The validity condition defines a subset of pre-proofs, ensuring good properties for those non-wellfounded derivations that satisfy the validity condition. In this paper, we will mainly be interested in cut-elimination theorem, which was proved for $\mu\mathsf{MALL}^\infty$ [4] and that we review in this subsection. In [4], a somehow stronger result than cut-elimination is proved: infinitary strong normalization with respect to the class of *fair* reduction sequences.

The only new result developed in this subsection is the lifting of the occurrence-based cut-elimination result of [4] to our setting system, for which we first introduce the multicut inference and review the main multicut-reduction steps for $\mu\mathsf{MALL}^\infty$ before defining fair reductions. The cut-elimination results of [4,20] do not rewrite cuts, *per se*, but subtrees of cuts in the form of an abstraction called ***multicut*** which is a variable arity inference defined as follows:

Definition 16. *The* **multicut inference** *is given by the data of (i) a conclusion sequent s, (ii) a non-empty list of premises $(s_1, \dots, s_n), n \geq 1$, (iii) an* **ancestor relation** *ι which is an injective map from the conclusion formulas to the premise formulas and relates identical formulas and additionally (iv) a* **cut-connectedness relation** *$\perp\!\!\!\perp$ which is a total, symmetric, binary relation among the formula occurrences of the premises which are not ancestor of a conclusion*

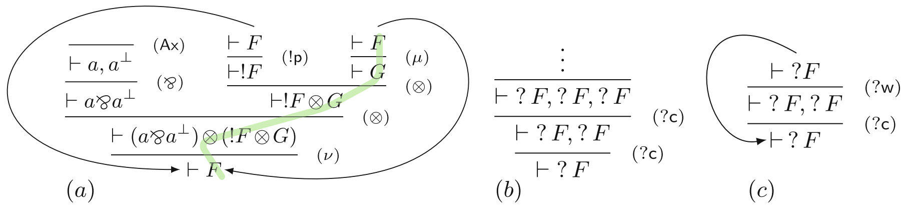


Fig. 5. Examples of valid and invalid pre-proofs

formula, which relates dual formulas[2] *and which satisfies a connectedness and acyclicity condition (see [3,4]). The multicut inference has no principal formula.*

We write this multicut rule as: $\dfrac{s_1 \quad \dots \quad s_n}{s}\ \mathsf{mcut}(\iota, \perp\!\!\!\perp)$.

In the following, we only consider $\mu\mathsf{MALL}^\infty$ pre-proofs with specific multicuts:

Definition 17 ($\mu\mathsf{MALL}^\infty_\mathsf{m}$). *$\mu\mathsf{MALL}^\infty_\mathsf{m}$ (pre)proofs are those (pre)proofs built from $\mu\mathsf{MALL}^\infty$ inferences and the multicut, such that (i) any branch contains at most one multicut and (ii) any occurrence of a cut is above a multicut inference.*

In the following, we shall always assume, even without mentioning it, that we consider proofs in $\mu\mathsf{MALL}^\infty_\mathsf{m}$ (as well as $\mu\mathsf{LL}^\infty_\mathsf{m}$, $\mu\mathsf{LJ}^\infty_\mathsf{m}$, $\mu\mathsf{LK}^\infty_\mathsf{m}$). We need the following definition (from [4]), identifying the premises of an mcut which are cut-connected to a given formula occurrence:

Definition 18 (Restriction of a mcut-context). *Consider an occurrence of a mcut $\dfrac{s_1 \quad \dots \quad s_n}{s}\ \mathsf{mcut}(\iota, \perp\!\!\!\perp)$ and assume s_i to be $\vdash F_1, \dots, F_k$. We define $\mathcal{C}_{F_j}, 1 \leq j \leq k$, to be the least set of sequent occurrences contained in $\{s_1, \dots, s_n\}$ such that:*
(i) If $\exists k, l$ such that $(k, l) \perp\!\!\!\perp (i, j)$, then $s_k \in \mathcal{C}_{F_j}$;
(ii) for any $k, k' \neq i$, if $s_k \in \mathcal{C}_{F_j}$ and $\exists l, l'$ such that $(k, l) \perp\!\!\!\perp (k', l')$, then $s_{k'} \in \mathcal{C}_{F_j}$.
We define $\mathcal{C}_\emptyset = \emptyset$ and $\mathcal{C}_{F,\Gamma} = \mathcal{C}_F \cup \mathcal{C}_\Gamma$.

When relating $\mu\mathsf{LL}^\infty$ and $\mu\mathsf{MALL}^\infty$ mcut-sequences below, we shall consider not only finite sequence nor ω-indexed sequences but also transfinite sequences. Those are sequences of triples of a proof, a redex and the position of the redex in the proof tree. A position p has a ***depth*** $\mathsf{dpth}(p)$ which is its length.

Definition 19 (mcut-reduction rules, transfinite sequences). *$\mu\mathsf{MALL}^\infty$ mcut-reduction sequences are directly adapted from [3,4]. Given an ordinal λ, a* **transfinite reduction sequence** *of length λ, or $\lambda\mathsf{TRS}$, is a λ-indexed sequence $(\pi_i, r_i, p_i)_{i\in\lambda}$ such that $\pi_i \longrightarrow^{p_i}_{r_i} \pi_{i+1}$, for any i such that $i + 1 \in \lambda$, where the reduction occurs at position p_i reducing mcut-redex r_i.*

[2] When working with two-sided sequents, $\perp\!\!\!\perp$ will relate identical formulas, one in a succedent, the other in an antecedent.

Definition 20 (Weak and strong convergence). *A (transfinite) mcut reduction sequence* $(\pi_i, r_i, p_i)_{i\in\alpha}$ *is* **weakly converging** *if for any limit ordinal* $\beta \in \alpha$, $lim(\pi_i)_{i\in\beta} = \pi_\beta$. $(\pi_i, r_i, p_i)_{i\in\alpha}$ *is* **strongly converging** *if it is weakly converging and moreover for any limit ordinal* $\beta \in \alpha$, $lim(\mathsf{dpth}(p_i))_{i\in\beta} = +\infty$.

Remark 21. The cut-reduction rules preserve the property that every branch of a proof has at most one multicut inference: $\mu\mathsf{MALL}^\infty_\mathsf{m}$ is closed by cut-reduction.

A $\mu\mathsf{MALL}^\infty_\mathsf{m}$ pre-proof π may contain multiple cut-redexes: $\pi \longrightarrow^{p_1}_{r_1} \pi_1$ and $\pi \longrightarrow^{p_2}_{r_2} \pi_2$. As usual, a notion of residual associates to (r_1, p_1), a set of redexes of π_2, $(r_1, p_1)/(r_2, p_2)$ which is generalized to reduction sequences: $(r_1, p_1)/\sigma$.

Definition 22 (Fair reduction sequences). *A reduction sequence* $(\pi_i, r_i, p_i)_{i\in\omega}$ *is* **fair** *if for all* $i \in \omega$ *and* r, p *such that* $\pi_i \longrightarrow^p_r \pi'$ *there is some* $j \geq i$ *such that* π_j *does not contain a residual of* (r, p) *anymore.*

Theorem 23. *Every fair mcut-reduction sequence of* $\mu\mathsf{MALL}^\infty$ *valid proofs of* $\vdash \Gamma$ *(strongly) converges to a cut-free valid proof of* $\vdash \Gamma$.

2.5 Compressing Transfinite $\mu\mathsf{MALL}^\infty$ Cut-Reduction Sequences

In the previous paragraph, we introduced not only ω-indexed sequences, but transfinite $\mu\mathsf{MALL}^\infty$ cut-reduction sequences as we shall need reduction beyond ω when simulating $\mu\mathsf{LL}^\infty$ cut-elimination in $\mu\mathsf{MALL}^\infty$. We shall now prove that a class of transfinite $\mu\mathsf{MALL}^\infty$ mcut-reduction sequences can be compressed to $\omega\mathsf{TRS}$. This result can be viewed as adapting to our setting the compression lemma from infinitary rewriting [31], even though we require more on the structure of the compressed sequences as it will be useful to establish $\mu\mathsf{LL}^\infty$ cut-elimination.

Definition 24 (Depth-increasing). *A* $\mu\mathsf{MALL}^\infty$ *cut reduction sequence* $\sigma = (\pi_i, r_i, p_i)_{i\in\omega}$ *is* **depth-increasing** *if* $(\mathsf{dpth}(p_i))_{i\in\omega}$ *is (weakly) increasing.*

Definition 25 (Reordering). *An mcut reduction sequence* $\sigma = (\pi_i, r_i, p_i)_{i\in\alpha}$ *is a reordering of* $\sigma' = (\pi'_i, r'_i, p'_i)_{i\in\beta}$ *if there is a bijection* o *between* α *and* β *such that for any* $i \in \alpha$, $(r'_{o(i)}, p'_{o(i)}) = (r_i, p_i)$.

Proposition 26 (Compression lemma). *Let* $\sigma = (\pi_i, r_i, p_i)_{i\in\alpha}$ *be a strongly converging* $\mu\mathsf{MALL}^\infty$ *transfinite cut-reduction sequence. There exists a* $\mu\mathsf{MALL}^\infty$ *cut-reduction sequence* $\mathsf{Comp}(\sigma) = (\pi'_i, r'_i, p'_i)_{i\in\beta}$ *which is a reordering of* σ, *depth-increasing, strongly converging with the same limit as* σ *and such that* $\beta = \alpha$ *if* α *is finite and* $\beta = \omega$ *otherwise.*

3 Cut-Elimination Theorem for $\mu\mathsf{LL}^\infty$

The aim of this section is to prove the following theorem:

Theorem 27. *For any valid $\mu\mathsf{LL}^\infty$ proof π, fair $\mu\mathsf{LL}^\infty$ mcut-sequences from π converge to cut-free $\mu\mathsf{LL}^\infty$ proofs.*

The idea of the proof and outline of the present section are as follows:

1. We shall first define the cut-reduction rules for $\mu\mathsf{LL}^\infty$ by extending $\mu\mathsf{MALL}^\infty$ *multicut*-reduction with rules for reducing exponential cuts.
2. We then encode exponentials with fixed-points and translate $\mu\mathsf{LL}^\infty$ sequents (resp. pre-proofs) into $\mu\mathsf{MALL}^\infty$, preserving validity both ways.
3. We will then simulate $\mu\mathsf{LL}^\infty$ reductions in $\mu\mathsf{MALL}^\infty$: a single $\mu\mathsf{LL}^\infty$ step may require an infinite, or even transfinite, $\mu\mathsf{MALL}^\infty$ mcut-reduction sequence.
4. Finally, we will study the simulation of fair $\mu\mathsf{LL}^\infty$ cut-reduction sequences. Even though the simulation of $\mu\mathsf{LL}^\infty$ sequences builds transfinite sequences, we shall see that one can associate a(n almost) fair $\mu\mathsf{MALL}^\infty$ mcut-reduction sequence to any fair $\mu\mathsf{LL}^\infty$ mcut-reduction sequence, and conclude.
   The next four subsections will closely follow the above pathway.

3.1 Cut-Elimination Rules for $\mu\mathsf{LL}^\infty$

$\mu\mathsf{LL}^\infty$ mcut-reduction is defined by extending $\mu\mathsf{MALL}^\infty$ multicut-reduction with the steps given in Fig. 6. The reduction rules for the exponentials assume a condition on the premisses of the multi-cut rule: all the proofs (hereditarily) cut-connected to some distinguished formula must have promotions as last inferences.

Definition 28 (($!\mathsf{p}$)-ready contexts). *A subset of the subproofs of a multicut is said to be* ($!\mathsf{p}$)*-ready if all its elements are concluded with an* ($!\mathsf{p}$) *rule.* $\mathcal{C}^!$ *will denote a* ($!\mathsf{p}$)*-ready context and* $\mathcal{C}^!_\Gamma$ *a context restriction which is* ($!\mathsf{p}$)*-ready.*

Remark 29. The condition for triggering the exponential key reductions $(?\mathsf{w})/(!\mathsf{p})$ and $(?\mathsf{c})/(!\mathsf{p})$ as well as the ($!\mathsf{p}$)-commutation rule is expressed in terms of ($!\mathsf{p}$)-readiness: for every ?-formula $?G$ in the context of a promotion which shall either commute or cut-reduce with a ?-rule, we require that $\mathcal{C}_{?G}$ is ($!\mathsf{p}$)-ready.

3.2 Embedding $\mu\mathsf{LL}^\infty$ in $\mu\mathsf{MALL}^\infty$

To extend the cut-elimination result from $\mu\mathsf{MALL}^\infty$ to $\mu\mathsf{LL}^\infty$, we encode the exponential connectives using fixed points as follows, following Baelde [2]:

Definition 30. $?^\bullet(F) = \mu X.F \oplus (\bot \oplus (X \parr X))$; $!^\bullet(F) = \nu X.F \& (1 \& (X \otimes X))$

This straightforwardly induces an embedding of $\mu\mathsf{LL}^\infty$ into $\mu\mathsf{MALL}^\infty$:

Definition 31 (Embedding of $\mu\mathsf{LL}^\infty$sequents into $\mu\mathsf{MALL}^\infty$).

$$\begin{array}{lll|lll}
(a)^\bullet & = a & \textit{if } a \textit{ is an atom} & (\sigma X.F)^\bullet & = \sigma X.(F)^\bullet & , \sigma \in \{\mu, \nu\} \\
(u)^\bullet & = u & \textit{if } u \in \{1, \bot, \top, 0\} & (?F)^\bullet & = ?^\bullet(F^\bullet) & \\
(A \star B)^\bullet & = (A)^\bullet \star (B)^\bullet & \textit{if } \star \in \{\&, \oplus, \parr, \otimes\} & (!F)^\bullet & = !^\bullet(F^\bullet) &
\end{array}$$

$$\dfrac{\mathcal{C} \quad \dfrac{\vdash \Delta, F}{\vdash \Delta, ?F}\,(?\mathsf{d})}{\vdash \Sigma, ?F}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\mathcal{C} \quad \vdash \Delta, F}{\vdash \Sigma, F}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp)}{\vdash \Sigma, ?F}\,(?\mathsf{d})$$

$$\dfrac{\mathcal{C} \quad \dfrac{\vdash \Delta, ?F, ?F}{\vdash \Delta, ?F}\,(?\mathsf{c})}{\vdash \Sigma, ?F}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\mathcal{C} \quad \vdash \Delta, ?F, ?F}{\vdash \Sigma, ?F, ?F}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp)}{\vdash \Sigma, ?F}\,(?\mathsf{c})$$

$$\dfrac{\mathcal{C} \quad \dfrac{\vdash \Delta}{\vdash \Delta, ?F}\,(?\mathsf{w})}{\vdash \Sigma, ?F}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\mathcal{C} \quad \vdash \Delta}{\vdash \Sigma}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp)}{\vdash \Sigma, ?F}\,(?\mathsf{w})$$

$$\dfrac{\dfrac{\vdash F, ?\Gamma}{\vdash !F, ?\Gamma}\,(!\mathsf{p}) \quad \mathcal{C}^{!}}{\vdash !F, ?\Sigma}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\vdash F, ?\Gamma \quad \mathcal{C}^{!}}{\vdash F, ?\Sigma}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp)}{\vdash !F, ?\Sigma}\,(!\mathsf{p})$$

$$\dfrac{\mathcal{C} \quad \dfrac{\vdash F, \Gamma}{\vdash ?F, \Gamma}\,(?\mathsf{d}) \quad \dfrac{\vdash F^{\perp}, ?\Delta}{\vdash !F^{\perp}, ?\Delta}\,(!\mathsf{p})}{\vdash \Sigma}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\mathcal{C} \quad \vdash F, \Gamma \quad \vdash F^{\perp}, ?\Delta}{\vdash \Sigma}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp')$$

where $?F \perp\!\!\!\perp !F^{\perp}$ and $\perp\!\!\!\perp'$ coincides with $\perp\!\!\!\perp$ except for $F \perp\!\!\!\perp' F^{\perp}$.

$$\dfrac{\mathcal{C}_{\Gamma} \quad \mathcal{C}^{!}_{?\,F} \quad \dfrac{\vdash ?F, ?F, \Gamma}{\vdash ?F, \Gamma}\,(?\mathsf{c})}{\vdash \Gamma', ?\,\Sigma}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\mathcal{C}_{\Gamma} \quad \mathcal{C}^{!}_{?\,F} \quad \mathcal{C}^{!}_{?\,F} \quad \vdash ?F, ?F, \Gamma}{\vdash \Gamma', ?\Sigma, ?\Sigma}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp')}{\vdash \Gamma', ?\Sigma}\,(?\mathsf{c})^{\star}$$

where $\mathcal{C}^{!}_{?\,F} \neq \emptyset$, $\perp\!\!\!\perp'$ corresponds to $\perp\!\!\!\perp$ on $\mathcal{C}_{\Gamma}$ and is a "duplication" of $\perp\!\!\!\perp$ on $\mathcal{C}^{!}_{?\,F}$ and each copy of $?\,F$ is in $\perp\!\!\!\perp'$-relation with the corresponding copy of $!\,F^{\perp}$ in $\mathcal{C}^{!}_{?\,F}$.

$$\dfrac{\mathcal{C}_{\Gamma} \quad \mathcal{C}^{!}_{?\,F} \quad \dfrac{\vdash \Gamma}{\vdash ?F, \Gamma}\,(?\mathsf{w})}{\vdash \Gamma', ?\Sigma}\,\mathsf{mcut}(\iota, \perp\!\!\!\perp) \quad \xrightarrow[r]{} \quad \dfrac{\dfrac{\mathcal{C}_{\Gamma} \quad \vdash \Gamma}{\vdash \Gamma'}\,\mathsf{mcut}(\iota', \perp\!\!\!\perp')}{\vdash \Gamma', ?\Sigma}\,(?\mathsf{w})^{\star}$$

where $\mathcal{C}^{!}_{?\,F} \neq \emptyset$ and $\perp\!\!\!\perp'$ corresponds to the restriction of $\perp\!\!\!\perp$ on $\mathcal{C}_{\Gamma}, \Gamma$.

Fig. 6. $\mu\mathsf{LL}^{\infty}$ mcut-reduction rules

Definition 32 ($\mu\mathsf{MALL}^{\infty}$derivability of the exponentials). *$\mu\mathsf{LL}^{\infty}$ exponential rules can be encoded in $\mu\mathsf{MALL}^{\infty}$ as shown in Fig. 7. We denote the derivable rules by $?\mathsf{d}^{\bullet}, ?\mathsf{c}^{\bullet}$, $?\mathsf{w}^{\bullet}$ and $!\mathsf{p}^{\bullet}$ respectively. ($!\mathsf{p}^{\bullet}$ uses a circular proof.)*

Proposition 33 (Preservation of validity). *π is a valid $\mu\mathsf{LL}^{\infty}$ proof of $\vdash \Gamma$ iff $\pi^{\bullet}$ is a valid $\mu\mathsf{MALL}^{\infty}$ proof of $\vdash \Gamma^{\bullet}$.*

Proof (Proof sketch). We simply relate the infinite branches in both pre-proofs. Assuming that π is valid, consider the special case of an infinite branch β of $\pi^{\bullet}$ that, when entering the encoding of a promotion, follows the left-most premise of the $(\&)$ rule. To such an infinite branch it is easy to associate an infinite branch b of π. b is valid and supported by a thread t with least formula $\nu X.F$. $(\nu X.F)^{\bullet}$ is the least recurring formula in the thread θ associated with t in β: β is valid.

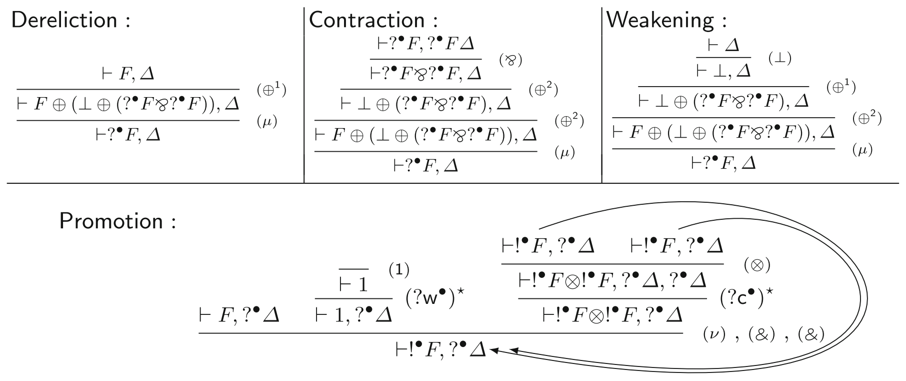


Fig. 7. $\mu\mathsf{MALL}^\infty$ encoding of the exponential inferences

3.3 Simulation of $\mu\mathsf{LL}^\infty$ Cut-Elimination Steps

Now we have to show that $\mu\mathsf{LL}^\infty$ cut-elimination steps can be simulated by the previous encoding. *E.g.*, the commutation rule for dereliction is simulated by a $(\mu)/(\mathsf{Cut})$ commutation followed by a $(\oplus)/(\mathsf{Cut})$ commutation as follows:

$$\cfrac{\cfrac{\vdash F, G, \Gamma}{\vdash ?^\bullet F, G, \Gamma}\,(?\mathsf{d}^\bullet) \qquad \vdash G^\perp, \Delta}{\vdash ?^\bullet F, \Gamma, \Delta}\,(\mathsf{Cut}) \quad \longrightarrow^2 \quad \cfrac{\cfrac{\vdash F, G, \Gamma \qquad \vdash G^\perp, \Delta}{\vdash F, \Gamma, \Delta}\,(\mathsf{Cut})}{\vdash ?^\bullet F, \Gamma, \Delta}\,(?\mathsf{d}^\bullet)$$

The challenge is to show that the simulation of reductions also holds (i) for the reductions involving $(!p)$ as well as (ii) for reductions occurring ***above*** a promotion rule (aka. in a box) since the encoding of $[!p]$ uses an infinite, circular derivation. In the promotion commutation case for instance, we have:

$$\cfrac{\cfrac{\vdash F, ?^\bullet G, ?^\bullet \Gamma}{\vdash !^\bullet F, ?^\bullet G, ?^\bullet \Gamma}\,(!\mathsf{p}^\bullet) \qquad \cfrac{\vdash G^\perp, ?^\bullet \Delta}{\vdash !^\bullet G^\perp, ?^\bullet \Delta}\,(!\mathsf{p}^\bullet)}{\vdash !^\bullet F, ?^\bullet \Gamma, ?^\bullet \Delta}\,(\mathsf{Cut}) \quad \longrightarrow_\omega \quad \cfrac{\cfrac{\vdash F, ?^\bullet G, ?^\bullet \Gamma \qquad \cfrac{\vdash G^\perp, ?^\bullet \Delta}{\vdash !^\bullet G^\perp, ?^\bullet \Delta}\,(!\mathsf{p}^\bullet)}{\vdash F, ?^\bullet \Gamma, ?^\bullet \Delta}\,(\mathsf{Cut})}{\vdash !^\bullet F, ?^\bullet \Gamma, ?^\bullet \Delta}\,(!\mathsf{p}^\bullet)$$

Proposition 34. *Each $\mu\mathsf{LL}^\infty$ mcut-reduction r can be simulated in $\mu\mathsf{MALL}^\infty$ by a (possibly infinite) sequence of mcut-reductions, denoted $r^\bullet$.*

Remark 35. Conversely, one can wonder whether a possible reduction in $\pi^\bullet$ necessarily comes from the simulation of a reduction step in π. It is *almost* the case except when the reduction in $\pi^\bullet$ comes from exponential cuts requiring a $(!\mathsf{p})$-ready context (*ie.* $(!\mathsf{p})$ commutation as well as $(?\mathsf{w})/(!\mathsf{p})$ and $(?\mathsf{c})/(!\mathsf{p})$ key cases, see above): in those cases indeed, if the context is "partially ready" – meaning that some, but not all, the required premises are promoted – a prefix of the sequence simulating the reduction step can indeed be performed, before being stuck. As consequence – and we shall exploit it in the next section when proving $\mu\mathsf{LL}^\infty$ cut-elimination – the simulation of a fair reduction sequence is not necessarily fair, *but only as long as the above cases are involved*:

Proposition 36. *There exists a fair reduction ρ from some $\mu\mathsf{LL}^\infty$ (pre-)proof π such that $\rho^\bullet$ is an ω-indexed unfair $\mu\mathsf{MALL}^\infty$ cut-reduction sequence.*

3.4 Proof of $\mu\mathsf{LL}^\infty$ Cut-Elimination Theorem

$\mu\mathsf{LL}^\infty$ cut-elimination theorem follows from the following two lemmas:

Lemma 37. *Let π be a $\mu\mathsf{LL}^\infty$-proof of $\vdash \Gamma$ and $\sigma = (\pi_i, r_i, p_i)_{i\in\omega}$ a fair $\mu\mathsf{LL}^\infty$ cut-reduction sequence from π. σ converges to a cut-free $\mu\mathsf{LL}^\infty$-pre-proof of $\vdash \Gamma$.*

Lemma 38. *Let π be a $\mu\mathsf{LL}^\infty$ pre-proof of $\vdash \Gamma$ and let us consider a cut-reduction sequence $\sigma = (\pi_i, r_i, p_i)_{i\in\omega}$ in $\mu\mathsf{LL}^\infty$ from π that converges to a cut-free $\mu\mathsf{LL}^\infty$ pre-proof π'. $\sigma^\bullet$ is a strongly converging (possibly transfinite) sequence.*

Proof (Sketch for Thm. 27). Let π be a $\mu\mathsf{LL}^\infty$-proof of $\vdash \Gamma$ and $\sigma = (\pi_i, r_i, p_i)_{i\in\omega}$ be a fair $\mu\mathsf{LL}^\infty$ mcut-reduction sequence from π. Consider the associated (transfinite) $\mu\mathsf{MALL}^\infty$ mcut-reduction sequence $\sigma^\bullet$ from $\pi^\bullet$ obtained by simulation. By Lemma 37, σ converges (*strongly*) to a cut-free $\mu\mathsf{LL}^\infty$ pre-proof π'.

Let us prove that π' is valid. By Lemma 38, $\sigma^\bullet$ is a *transfinite* mcut-reduction sequence from $\pi^\bullet$ *strongly converging* to $\pi'^\bullet$. By Prop. 26, $\sigma^\bullet$ can be compressed into $\rho = (\pi'_i, r'_i, p'_i)_{i\in\omega}$ an *ω-indexed* depth-increasing $\mu\mathsf{MALL}^\infty$ mcut-reduction sequence which converges to $\pi'^\bullet$ and contains the same reductions as $\sigma^\bullet$. By Proposition 36, ρ may not be fair: this prevents us from concluding directly by Proposition 33 but we can still conclude. Let us consider ρ_f a fair reduction sequence obtained from ρ by reducing those redexes which cause the lack of fairness of ρ and let us consider the limit of ρ_f, π_f. To any infinite branch β of $\pi'^\bullet$, one can associate a branch β_f of π_f: it coincides with β except when the next inference of β_f is on a $(!\,F)^\bullet$ (in a sequent, say, $\vdash (!\,F)^\bullet, ?^\bullet\Delta^\bullet$ which is not principal along β). In that case, we expand β_f by following the unique premise of the (ν) rule, the second premise of the first $(\&)$ rule and the first premise of the second $(\&)$ rule, reaching $\vdash 1, ?^\bullet\Delta^\bullet$, in which case we know that the 1 is not principal (and never will be) and we follow back β. β_f has exactly the same threads as β: finite threads may only be extended *finitely* on occurrences of $(!\,F)^\bullet$. Since ρ_f is fair, β_f is valid and so is β.

We can then conclude that $\pi'^\bullet$ is cut-free and valid and, using preservation of validity (Proposition 33), that π' is a valid cut-free $\mu\mathsf{LL}^\infty$-proof. □

Infinitary cut-elimination for $\mu\mathsf{LL}^\infty$ two-sided sequent calculus is an easy corollary of Theorem 27. Indeed, fair cut-reduction sequences in two-sided $\mu\mathsf{LL}^\infty$ are mapped to fair reduction sequences in one-sided $\mu\mathsf{LL}^\infty$ from which follows:

Corollary 39. *Fair 2-sided $\mu\mathsf{LL}^\infty$ valid mcut-reduction sequences eliminate cuts.*

4 Cut-Elimination Theorem for $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$

Cut-elimination theorems for both $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$ can be established as corollaries of Theorem 27. For lack of space, we directly go to our results and postpone to future work a detailed study of the generalizations to non-wellfounded sequent calculi of the linear embeddings of LK and LJ into LL developed since Girard seminal paper. We shall comment on those translations in the conclusion.

4.1 $\mu\mathsf{LK}^\infty$ Cut-Elimination: Skeletons and Decorations

To any $\mu\mathsf{LL}^\infty$ formulas and $\mu\mathsf{LL}^\infty$ proofs, one can associate their skeletons, that is corresponding $\mu\mathsf{LK}^\infty$ formulas and proofs, after erasing of the linear information:

Definition 40 (Skeleton). $\mathsf{Sk}(A)$ *is defined by induction on* $A \in \mu\mathsf{LL}^\infty$*:*

$$\begin{array}{r|c|c}
\mathsf{Sk}(A \otimes B) = \mathsf{Sk}(A) \wedge \mathsf{Sk}(B) & \mathsf{Sk}(A \parr B) = \mathsf{Sk}(A) \vee \mathsf{Sk}(B) & \mathsf{Sk}(!\,A) = \mathsf{Sk}(A) \\
\mathsf{Sk}(A \& B) = \mathsf{Sk}(A) \wedge \mathsf{Sk}(B) & \mathsf{Sk}(A \oplus B) = \mathsf{Sk}(A) \vee \mathsf{Sk}(B) & \mathsf{Sk}(?\,A) = \mathsf{Sk}(A) \\
\mathsf{Sk}(1) = \mathsf{Sk}(\top) = \top & \mathsf{Sk}(\bot) = \mathsf{Sk}(0) = \mathsf{F} & \mathsf{Sk}(a) = a \\
\mathsf{Sk}(A \multimap B) = \mathsf{Sk}(A) \Rightarrow \mathsf{Sk}(B) & \mathsf{Sk}(\sigma X.A) = \sigma X.\mathsf{Sk}(A) & \mathsf{Sk}(X) = X
\end{array}$$

with $\sigma \in \{\mu, \nu\}$*.*

Given a 2-sided $\mu\mathsf{LL}^\infty$ *pre-proof* π *of* $\Gamma \vdash \Delta$ *with last rule* r *and premises* $(\pi_i)_{1 \leq i \leq n}$*,* $\mathsf{Sk}(\pi)$ *is the* $\mu\mathsf{LK}^\infty$ *pre-proof of* $\mathsf{Sk}(\Gamma) \vdash \mathsf{Sk}(\Delta)$ *defined corecursively, by case on* r*: (i) if* $r \in \{(!\mathsf{p}), (?\mathsf{d})\}$*,* $\mathsf{Sk}(\pi) = \mathsf{Sk}(\pi_1)$*; (ii) otherwise, apply the* $\mu\mathsf{LK}^\infty$ *rule corresponding to* r *with premises* $(\mathsf{Sk}(\pi_i))_{1 \leq i \leq n}$*.*

Proposition 41. $\mathsf{Sk}(\cdot)$ *transports valid* $\mu\mathsf{LL}^\infty$*-proofs to valid* $\mu\mathsf{LK}^\infty$ *proofs.*

$\mu\mathsf{LK}^\infty$ cut-elimination follows from the existence of $\mu\mathsf{LK}^\infty$ linear decorations.

Proposition 42. *For any* $\mu\mathsf{LK}^\infty$ *sequent* s *and any* $\mu\mathsf{LK}^\infty$ *proof* π *of* s*, there is a linear decoration of* π*, that is a* $\mu\mathsf{LL}^\infty$ *proof* π^d *such that* $\mathsf{Sk}(\pi^d) = \pi$*.*

Definition 43 ($\mu\mathsf{LK}^\infty$cut-reduction). $\mu\mathsf{LK}^\infty$ *mcut-reduction relation is defined as follows:* $\longrightarrow_{\mu\mathsf{LK}^\infty} = \{(\mathsf{Sk}(\pi), \mathsf{Sk}(\pi')) \mid \pi \longrightarrow_{mcut} \pi' \;\&\; \pi \neq \pi'\}$*.*

Theorem 44. $\mu\mathsf{LK}^\infty$ *enjoys cut-elimination.*

4.2 $\mu\mathsf{LJ}^\infty$ Cut-Elimination

The linear decoration for $\mu\mathsf{LJ}^\infty$ is simply Girard's call-by-value translation [21] extended to fixed-points on formulas and proofs as follows:

$$[X]^j = !\,X; \qquad [\mu X.F]^j = !\,\mu X.[F]^j; \qquad [\nu X.F]^j = !\,\nu X.[F]^j.$$

$$\left[\dfrac{\dfrac{\pi}{\Gamma \vdash F[\sigma X.F/X]}}{\Gamma \vdash \sigma X.F}\,(\sigma_\mathsf{r})\right]^j = \dfrac{\dfrac{\dfrac{[\pi]^j}{[\Gamma]^j \vdash [F]^j[\sigma X.[F]^j/X]}}{[\Gamma]^j \vdash \sigma X.[F]^j}\,(\sigma_\mathsf{r})}{[\Gamma]^j \vdash [\sigma X.F]^j}\,(!\mathsf{p}_\mathsf{r}) \quad \text{and}$$

$$\left[\dfrac{\dfrac{\pi}{\Gamma, F[\sigma X.F/X] \vdash G}}{\Gamma, \sigma X.F \vdash G}\,(\sigma_\mathsf{l})\right]^j = \dfrac{\dfrac{\dfrac{[\pi]^j}{[\Gamma]^j, [F]^j[\sigma X.[F]^j/X], \vdash [G]^j}}{[\Gamma]^j, \sigma X.[F]^j \vdash [G]^j}\,(\sigma_\mathsf{l})}{[\Gamma]^j, [\sigma X.F]^j \vdash [G]^j}\,(!\mathsf{d}_\mathsf{l})\ .$$

The translation is consistent with $\mu\mathsf{LJ}^\infty$- and $\mu\mathsf{LL}^\infty$-positivity conditions.

Definition 45 ($\mu\mathsf{ILL}^\infty$)**.** *$\mu\mathsf{ILL}$ formulas are defined inductively as:*
$I, J ::= a \mid {!}X \mid I \multimap J \mid I \& J \mid I \oplus J \mid \top \mid 0 \mid \mu X.I \mid \nu X.I \mid {!}I.$

A $\mu\mathsf{ILL}$ sequent is a sequent of $\mu\mathsf{ILL}$ formulas with exactly one formula in the succedent. A $\mu\mathsf{ILL}^\infty$ proof is a $\mu\mathsf{LL}^\infty$ proof containing only $\mu\mathsf{ILL}$ sequents.

The translation preserves validity, following from $[X]^j = {!}X$, by induction.

Lemma 46. *The following hold:*

- *For any $\mu\mathsf{LJ}$ formulas A, B, $\sigma \in \{\mu, \nu\}$, $[A[\sigma X.B/X]]^j = [A]^j[\sigma X.[B]^j/X]$.*
- *For any $\mu\mathsf{LJ}$ formula A, $[A]^j$ is a $\mu\mathsf{ILL}$ formula.*
- *If π is a $\mu\mathsf{LJ}^\infty$ proof of $\Gamma \vdash F$, then $[\pi]^j$ is a $\mu\mathsf{ILL}^\infty$ proof of $[\Gamma]^j \vdash [F]^j$.*

On $\mu\mathsf{ILL}^\infty$ proofs, the skeletons of the previous section can be reused: $\mathsf{Sk}(\cdot)$ transports valid $\mu\mathsf{ILL}^\infty$ proof to valid $\mu\mathsf{LJ}^\infty$ proofs. Moreover $\mu\mathsf{ILL}^\infty$ proofs are closed by $\mu\mathsf{LL}^\infty$ cut-reductions from which we deduce, as for $\mu\mathsf{LK}^\infty$, that:

Theorem 47. *$\mu\mathsf{LJ}^\infty$ enjoys cut-elimination.*

5 Conclusion

In the present paper, we established several cut-elimination results for non-wellfounded proof systems for logics with least and greatest fixed-points expanding on previous works [4,20]: (i) for $\mu\mathsf{MALL}^\infty$ with sequents as lists in contrast sequents as sets of locative occurrences [4], (ii) for the 1-sided and 2-sided sequent calculi of $\mu\mathsf{LL}^\infty$, (iii) for $\mu\mathsf{LK}^\infty$ and (iv) for $\mu\mathsf{LJ}^\infty$. We also established additional results from a compression lemma for $\mu\mathsf{MALL}^\infty$ strongly converging cut-reduction sequences to linear embeddings of $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$ into $\mu\mathsf{LL}^\infty$.

On the Meaning and Expressiveness of Tree-Exponential Modalities. The proof of our main result proceeds by encoding LL exponentials in $\mu\mathsf{MALL}^\infty$ following an encoding first considered by Baelde and Miller, and studying $\mu\mathsf{LL}^\infty$ cut-reduction sequences through their simulation in $\mu\mathsf{MALL}^\infty$, which was first conjectured in Doumane's thesis [18]. We think that the present paper does not only demonstrate the usefulness of the encoding but that it also suggests new questions. Indeed, this encoding has interesting features:

- this "rigid" tree-like exponential does not exhibit the Seely isomorphism but, even though those isomorphisms are common in axiomatizations of categorical models of linear logic, it is not necessary to have them as isomorphisms to build a denotational model of linear logic (that is, which quotients proofs up to cut-equivalence): the present work is actually an example of this fact. (They are crucial, though, to encode the λ-calculus in linear logic, as additional equations are needed, which are realized by Seely isos.)

– These exponentials allow for a realization of a somehow non-uniform promotion: indeed, while a proof of $\vdash\ !^\bullet F, ?^\bullet \Gamma$ has to provide a proof of $\vdash F, ?^\bullet \Gamma$, the circular definition of the promotion is not the only possible definition: one can consider as well promotions that would provide a distinct value each time a box is opened (*e.g.* a proof of $\vdash\ !^\bullet \mu X.1 \oplus X$ may provide distinct integers depending on how structural rules managed the resource). See [30] for a detailed discussion.

This tree-like exponential is being investigated with Ehrhard and Jafarrahmani.

Benefiting from Advances in Infinitary Rewriting. Our cut-elimination proof by encoding $\mu\mathsf{LL}^\infty$ into $\mu\mathsf{MALL}^\infty$ relies on a simulation of reductions sequences which makes use of transfinite reductions sequences and compression results. Those techniques are inspired and adapted from the literature on infinitary rewriting. We plan to make clearer the connection between non-wellfounded proof theory and infinitary rewriting in the future, even though in the present state it was not possible to readily apply results from infinitary rewriting such as the compression lemma which we has to reprove in our setting [31]. Moreover, we did not make use of coinductive formulations of infinitary rewriting [19]. That is another direction for future work: currently, we do not know how to use those formulations of infinitary rewriting because the sequences we consider by simulation are not given as (strongly) converging sequences. We plan to reconsider this and benefit from the coinductive approach to infinite reduction sequences.

On Linear Translations for Fixed-Point Logics and Non-Wellfounded Proofs. We obtained a cut-elimination theorem for $\mu\mathsf{LK}^\infty$ and $\mu\mathsf{LJ}^\infty$ thanks to linear translations which deserve some comments. While the linear translation used for $\mu\mathsf{LJ}^\infty$ is standard (it is a call-by-value translation dating back to Girard's seminal paper), the treatment of classical logic was more complex. Indeed, usual linear translation for classical logic introduce, at places, cuts. Due to the sensitivity of the straight-thread validity condition with respect to the presence of cuts in cycles, we could not use those translations. However, we plan to investigate whether a more standard translation can be used in the specific case of bouncing validity [3].

A Treatment of Cut-Elimination Which Is Agnostic to Validity Conditions. Last but not least, a major advantage of our approach is that $\mu\mathsf{MALL}^\infty$ cut-elimination proof and, to some extent, the validity conditions, are regarded as black boxes, simplifying the presentation of the proof and making it reusable wrt. other validity condition or $\mu\mathsf{MALL}^\infty$ proof techniques. The proof seems to be reusable easily with bouncing validity for instance (even though setting up an adequate definition of bouncing validity for $\mu\mathsf{LL}^\infty$ is quite tricky). A fragment which seems promising and that we wish to investigate in the near future, is $\mu\mathsf{MELL}^\infty$ equipped with bouncing validity [3].

Acknowledgements. First, I would like to deeply thank David Baelde and Amina Doumane for their extensive collaboration and brilliant ideas on the topic of $\mu\mathsf{MALL}^\infty$;

the idea of a cut-elimination proof exploiting the fixed-point encoding of the exponentials emerged in joint discussions with them. Many thanks to Esaïe Bauer, Anupam Das, Abhishek De, Claudia Faggian, Guilhem Jaber, Farzad Jafarrahmani, Paul-André Melliès and Luc Pellissier for helpful discussions and constructive feedback on earlier versions of this draft. Last, the author would like to thank the anonymous reviewers for their work and for bringing very relevant suggestions for the present paper as well as for future works.

References

1. Afshari, B., Leigh, G.E.: On closure ordinals for the modal mu-calculus. In Rocca, S.R.D., (eds), Computer Science Logic 2013 (CSL 2013). vol. 23 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 30–44, Dagstuhl, Germany (2013). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. http://drops.dagstuhl.de/opus/volltexte/2013/4188, https://doi.org/10.4230/LIPIcs.CSL.2013.30
2. Baelde, D.: Least and greatest fixed points in linear logic. ACM Trans. Comput. Logic **13**(1) (2012). https://doi.org/10.1145/2071368.2071370
3. Baelde, D., Doumane, A., Kuperberg, D., Saurin, A.: Bouncing threads for circular and non-wellfounded proofs: towards compositionality with circular proofs. In: Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 63:1–63:13. ACM (2022)
4. Baelde, D., Doumane, A., Saurin, A.: Infinitary proof theory: the multiplicative additive case. In: 25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France, volume 62 of LIPIcs, pp. 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016). http://www.dagstuhl.de/dagpub/978-3-95977-022-4
5. Baelde, D., Miller, D.: Least and greatest fixed points in linear logic. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 92–106. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75560-9_9
6. Brotherston, J.: Sequent Calculus Proof Systems for Inductive Definitions. PhD thesis, University of Edinburgh, November 2006
7. Brotherston, J., Distefano, D., Petersen, R.L.: Automated cyclic entailment proofs in separation logic. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS (LNAI), vol. 6803, pp. 131–146. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22438-6_12
8. Brotherston, J., Gorogiannis, N., Petersen, R.L.: A generic cyclic theorem prover. In: Jhala, R., Igarashi, A. (eds.) APLAS 2012. LNCS, vol. 7705, pp. 350–367. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35182-2_25
9. Brotherston, J., Simpson, A.: Complete sequent calculi for induction and infinite descent. In: 22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007), pp. 51–62. IEEE (2007)
10. Brotherston, J., Simpson, A.: Sequent calculi for induction and infinite descent. J. Logic Comput. **21**(6), 1177–1216 (2011)
11. Brünnler, K., Studer, T.: Syntactic cut-elimination for a fragment of the modal mu-calculus. Ann. Pure Appl. Logic **163**(12), 1838–1853 (2012). https://www.sciencedirect.com/science/article/pii/S0168007212000760, https://doi.org/10.1016/j.apal.2012.04.006
12. Danos, V., Joinet, J.-B., Schellinx, H.: A new deconstructive logic: linear logic. J. Symb. Log. **62**(3), 755–807 (1997)

13. Das, A.: A circular version of Gödel's T and its abstraction complexity. CoRR, abs/2012.14421 (2020)
14. Das, A.: Personal communication, June 2023
15. Das, A., Pous, D.: Non-wellfounded proof theory for (Kleene+action)(algebras +lattices). In: Annual Conference for Computer Science Logic (CSL). vol. 119 of LIPIcs, pp. 19:1–19:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018)
16. Dax, C., Hofmann, M., Lange, M.: A proof system for the linear time μ-calculus. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 273–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11944836_26
17. Doumane, A.: Constructive completeness for the linear-time μ-calculus. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, 20–23 June 2017, pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005075
18. Doumane, A.: On the infinitary proof theory of logics with fixed points. PhD thesis, Paris Diderot University (2017)
19. Endrullis, J., Hansen, H.H., Hendriks, D., Polonsky, A., Silva, A.: Coinductive foundations of infinitary rewriting and infinitary equational logic. Log. Methods Comput. Sci. **14**(1) (2018). https://doi.org/10.23638/LMCS-14(1:3)2018
20. Fortier, J., Santocanale, L.: Cuts for circular proofs: semantics and cut-elimination. In: Rocca, S.R.D. (eds.) Computer Science Logic 2013 (CSL 2013), CSL 2013, 2–5 September 2013, Torino, Italy, volume 23 of LIPIcs, pp. 248–262. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2013)
21. Girard, J.-Y.: Linear logic. Theor. Comput. Sci. **50**(1), 1–101 (1987). https://doi.org/10.1016/0304-3975(87)90045-4
22. Kozen, D.: Results on the propositional μ-calculus. Theor. Comput. Sci. **27**(3), 333–354 (1983). Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer (1982). http://www.sciencedirect.com/science/article/pii/0304397582901256, https://doi.org/10.1016/0304-3975(82)90125-6
23. Kozen, D.: A finite model theorem for the propositional μ-calculus. Studia Logica **47**(3), 233–241 (1988). https://doi.org/10.1007/BF00370554
24. Kuperberg, D., Pinault, L., Pous, D.: Cyclic proofs, system T, and the power of contraction. Proc. ACM Program. Lang. **5**, 1–28 (2021)
25. Martin-Löf, P.: Hauptsatz for the intuitionistic theory of iterated inductive definitions. In: Fenstad, J.E. (edn.) Proceedings of the Second Scandinavian Logic Symposium. vol. 63 of Studies in Logic and the Foundations of Mathematics, pp. 179–216. Elsevier (1971). https://www.sciencedirect.com/science/article/pii/S0049237X08708474, https://doi.org/10.1016/S0049-237X(08)70847-4
26. Melliès, P.-A.: Categorical semantics of linear logic. In: Interactive models of computation and program behavior, Panoramas et Synthèses, pp. 213. Société Mathématique de France (2009)
27. Pinault, L.: From automata to cyclic proofs : equivalence algorithms and descriptive complexity. (Des automates aux preuves cycliques : algorithmes d'équivalence et complexité descriptive). PhD thesis, University of Lyon, France (2021)
28. Rowe, R.N.S., Brotherston, J.: Automatic cyclic termination proofs for recursive procedures in separation logic. In: Bertot, Y., Vafeiadis, V., (eds). Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, 16–17 January 2017, pp. 53–65. ACM (2017). https://doi.org/10.1145/3018610.3018623

29. Santocanale, L.: A calculus of circular proofs and its categorical semantics. In: Nielsen, M., Engberg, U. (eds.) FoSSaCS 2002. LNCS, vol. 2303, pp. 357–371. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45931-6_25
30. Saurin, A.: A linear perspective on cut-elimination for non-wellfounded sequent calculi with least and greatest fixed points (extended version). long version of the present paper (2023). https://hal.science/hal-04169137
31. Terese. Term Rewriting Systems. vol. 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2003)

Ill-Founded Proof Systems for Intuitionistic Linear-Time Temporal Logic

Bahareh Afshari[1], Lide Grotenhuis[2], Graham E. Leigh[1], and Lukas Zenger[3](✉)

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Gothenburg, Sweden
{bahareh.afshari,graham.leigh}@gu.se

[2] Institute of Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands
l.m.grotenhuis@uva.nl

[3] Institute of Computer Science, University of Bern, Bern, Switzerland
lukas.zenger@unibe.ch

Abstract. We introduce ill-founded sequent calculi for two intuitionistic linear-time temporal logics. Both logics are based on the language of intuitionistic propositional logic with 'next' and 'until' operators and are evaluated on dynamic Kripke models wherein the intuitionistic and temporal accessibility relations are assumed to satisfy one of two natural confluence properties: forward confluence in one case, and both forward and backward confluence in the other. The presented sequent calculi are cut-free and incorporate a simple form of formula nesting. Soundness of the calculi is shown by a standard argument and completeness via proof search.

Keywords: Sequent calculus · Intuitionistic logic · Temporal logic · Ill-founded proofs

1 Introduction

Intuitionistic modal and temporal logics have found tangible applications in computer science [7,9,12,13,16,22] and with that comes the motivation for developing succinct proof systems that facilitate establishing fundamental properties such as decidability and algorithmic proof search. Temporal logic describes a range of modal logics in which modal and 'fixed point' operators are interpreted as temporal relations. An important example is linear-time temporal logic LTL, whose temporal operators include a 'next' operator X and an 'until' operator U. The formula $\mathsf{X}A$ is interpreted as 'A is true in the next time-step', and $A \mathsf{U} B$ as 'A is true until B is true'. The until operator satisfies the equivalence

$$A \mathsf{U} B \qquad \text{iff} \qquad B \vee (A \wedge \mathsf{X}(A \mathsf{U} B)),$$

Supported by the Swiss National Science Foundation [200021L_196176], Dutch Research Council [OCENW.M20.048], the Knut and Alice Wallenberg Foundation [2020.0199], and the Swedish Research Council [2017-05111].

showing that $A \mathsf{U} B$ is a *fixed point* of the propositional function $p \mapsto B \vee (A \wedge \mathsf{X} p)$.

Advances in the proof theory of temporal logics evidence that ill-founded calculi are particularly suitable for capturing the behaviour of fixed point operators in a syntactic way [1,8,10,15]. So far, the study of such proof systems has focused on classical logic and their applicability to intuitionistic temporal logics remains largely unexplored. One of the obstacles in directly applying the techniques from the classical setting is the interaction of the temporal and intuitionistic relation in the intuitionistic Kripke semantics.

A standard way to present the semantics of intuitionistic propositional logic is in terms of Kripke models $(W, \leq, V)$, where $\leq$ is a partial order on the set of worlds W and V a valuation that is monotone in $\leq$. A key property of this semantics is the monotonicity lemma: for all $v, v' \in W$, if $v \leq v'$ and $v \models A$ then $v' \models A$. The semantics of intuitionistic modal/temporal logics can be given in terms of intuitionistic Kripke models $(W, \leq, V)$ equipped with an additional relation R on W used to interpret the modal operators. In order to keep the monotonicity property, modalities are interpreted as follows.

$$\begin{array}{lll} w \models \Box A & \text{iff} & \forall w' \geq w \, \forall v (w' R v \text{ implies } v \models A) \\ w \models \Diamond A & \text{iff} & \forall w' \geq w \, \exists v (w' R v \text{ and } v \models A) \end{array}$$

One can also use the classical truth conditions for modalities and instead impose confluence properties on R and $\leq$ to ensure monotonicity. Two confluence properties considered in the literature are:

Forward confluence if $v \geq w$ and wRw' then there exists $v' \geq w'$ with vRv'.
Backward confluence if wRw' and $w' \leq v'$ then there exists $v \geq w$ with vRv'.

In the setting of intuitionistic LTL, forward confluence alone suffices to obtain the monotonicity lemma [3]. Since Simpson [20] argues that an intuitionistic reading of possible world semantics results in models that also satisfy backward confluence, *intuitionistic modal logic* is generally used to refer to the logic obtained when adopting both conditions. Nevertheless, logics corresponding to weaker frame conditions, often called *constructive modal logics*, have also received considerable interest (see e.g. [2,23]).

In this work, the language of linear temporal logic is interpreted over models satisfying forward confluence and models satisfying both forward and backward confluence; following the terminology in [3], they are referred to as *expanding* and *persistent* models, respectively. To date, neither logic has been given a sound and complete axiomatisation.[1] For each of the resulting logics, we present a cut-free, ill-founded sequent calculus. Both calculi employ a simple form of nested sequents so that formulas can be operated on at different temporal steps. This form of nesting has been used by Kojima and Igarashi [14] to obtain a finitary calculus for a constructive interpretation of LTL without the until operator.

A standard technique for showing completeness of an ill-founded calculus is to set up a proof search game between two players, Prover and Refuter, such

[1] A Hilbert-style axiomatisation exists for the ‘eventually’ only fragment over expanding models [5] but the case of persistent models is unknown.

that a winning strategy for Prover corresponds to a proof and a winning strategy for Refuter to a countermodel (see e.g. [1,19]). When applying this technique to the intuitionistic case, one needs to ensure that the constructed 'countermodel' satisfies the required frame conditions. We present such a proof search game for both logics. The use of nested sequents is crucial for the game as it enables postponing the application of non-invertible rules until all relevant information about future time steps is determined.

Intuitionistic temporal logics have been studied in different contexts, the most notable of which are *metaprogramming* and *topological dynamics*. The former involves the addition of temporal operators to λ-calculi with the aim of modelling aspects of metaprogramming such as staged computation (see e.g. [7,21,24]). The latter concerns the use of intuitionistic temporal logics to reason about dynamical topological systems. Fernández-Duque [11] introduced the logic $\mathsf{ITL}^{\mathsf{c}}$, in which formulas of LTL are interpreted in general topological models, and showed that its restriction to the 'eventually' operator $\Diamond$ is decidable.[2] Boudou et al. [4] show the decidability of the same fragment interpreted in expanding models, denoted by $\mathsf{ITL}^{\mathsf{e}}$, and provide a Hilbert-style axiomatisation for both logics in [5]. A calculus with ω-branching inference rules is given in [6] for $\mathsf{ITL}^{\mathsf{e}}$ extended with the 'henceforth' operator. To date, no recursive axiomatisation of the validities in persistent models is known.

Outline. Section 2 introduces the syntax and semantics of intuitionistic linear temporal logic iLTL. Section 3 presents the proof system $\mathsf{iLTL}^{\mathsf{nest}}_{\mathsf{e}}$, which is proven sound and complete with respect to expanding models in Sects. 4 and 5. In Sect. 6, we outline how $\mathsf{iLTL}^{\mathsf{nest}}_{\mathsf{e}}$ can be adapted to obtain a system $\mathsf{iLTL}^{\mathsf{nest}}_{\mathsf{p}}$ that is sound and complete with respect to persistent models.

2 Syntax and Semantics

Fix a countable set Prop of atomic propositions. *Formulas* of iLTL are defined inductively as follows:

$$A, B ::= \bot \mid p \mid A \wedge B \mid A \vee B \mid A \rightarrow B \mid \mathsf{X}A \mid A \,\mathsf{U}\, B$$

where $p \in \mathsf{Prop}$. We denote formulas by A, B, etc., and atomic propositions by p, q, etc. We define the formula $X^n A$ inductively by $X^0 A := A$ and $X^{n+1}A := XX^n A$.

Formulas of iLTL are evaluated on *dynamic models*, which are intuitionistic Kripke models equipped with a time function that maps each world to its temporal successor.

Definition 1. *A* dynamic model *is a tuple $M = (W, \leq, f, V)$ where*

1. *W is a non-empty set,*
2. *$\leq$ is a partial order on W,*
3. *$f : W \longrightarrow W$ is a function and*

[2] In our notation, the eventually operator $\Diamond$ can be defined as $\Diamond A := \top \,\mathsf{U}\, A$.

$$\begin{tikzcd}
v \arrow[r, "f"] & f(v) & & v \arrow[r, dashed, "f"] & v' \\
w \arrow[u, "\leq"] \arrow[r, "f"'] & f(w) \arrow[u, dashed, "\leq"'] & & w \arrow[u, dashed, "\leq"] \arrow[r, "f"'] & f(w) \arrow[u, "\leq"']
\end{tikzcd}$$

Fig. 1. Forward and backward confluence.

4. $V : W \longrightarrow \mathcal{P}(Prop)$ is a valuation function that is monotone in $\leq$, i.e., for all $w, v \in W$, if $w \leq v$, then $V(w) \subseteq V(v)$.

Elements of W are called *worlds*. If $w, v \in W$ such that $f(w) = v$, then v is called the *temporal successor* of w. If $w \leq v$, then v is called an *intuitionistic successor* of w. We inductively define $f^0(w) := w$ and $f^{n+1}(w) := f(f^n(w))$.

Given a dynamic model $M = (W, \leq, V, f)$, the *truth relation* $M, w \models A$ where $w \in W$ is defined inductively on A as follows.

$M, w \not\models \bot$
$M, w \models p$ iff $p \in V(w)$,
$M, w \models A \wedge B$ iff $M, w \models A$ and $M, w \models B$,
$M, w \models A \vee B$ iff $M, w \models A$ or $M, w \models B$,
$M, w \models A \to B$ iff for all $v \geq w$ if $M, v \models A$, then $M, v \models B$,
$M, w \models \mathsf{X}A$ iff $M, f(w) \models A$,
$M, w \models A \,\mathsf{U}\, B$ iff there exists an $n < \omega$ such that $M, f^n(w) \models B$ and for all $m < n$ we have $M, f^m(w) \models A$.

Validity and satisfiability over a class of dynamic models are defined in the standard way.

We consider dynamic models that satisfy certain confluence properties, namely forward and backward confluence, which are illustrated in Fig. 1.

Definition 2. *A dynamic model $M = (W, \leq, f, V)$ is*

- expanding *if M is forward confluent: for all $w, v \in W$,*

$$\textit{if } w \leq v, \textit{ then } f(w) \leq f(v),$$

- persistent *if M is expanding and backward confluent: for all $w, v' \in W$,*

$$\textit{if } v' \geq f(w), \textit{ then there exists } v \geq w \textit{ with } f(v) = v'.$$

We denote by $\mathsf{iLTL_e}$ and $\mathsf{iLTL_p}$ the set of iLTL-validities over expanding and persistent models, respectively. It is easy to check that the temporal version of the **K**-axiom, namely $\mathsf{X}(A \to B) \to (\mathsf{X}A \to \mathsf{X}B)$, is valid over expanding models. The converse $(\mathsf{X}A \to \mathsf{X}B) \to \mathsf{X}(A \to B)$ is only valid over persistent models, and so we have $\mathsf{iLTL_e} \subsetneq \mathsf{iLTL_p}$.

With a straightforward induction, one can prove the monotonicity lemma for expanding models. Note that the lemma thus also holds for persistent models.

Lemma 1. *Let $M = (W, \leq, V, f)$ be an expanding model, $w, v \in W$ and A a formula. If $M, w \models A$ and $w \leq v$, then $M, v \models A$.*

3 Nested Ill-Founded Proofs

In this section we present an ill-founded sequent calculus that is sound and complete with respect to the class of expanding models. Proofs in this calculus are finitely-branching trees that admit infinitely long branches. Importantly, the calculus has no explicit induction rule and does not make use of the cut-rule.

To ensure soundness, infinite branches are required to satisfy a global soundness condition, which is presented in a standard way using formula traces. To ensure completeness, the calculus incorporates a simple form of *nesting.*

Definition 3. *A* nested iLTL-formula *is a tuple* (A, n)*, denoted by* A^n*, with* A *an* iLTL*-formula and* $n < \omega$*. A* sequent *is an ordered pair* $\langle \Gamma, \Delta \rangle$*, written as* $\Gamma \Rightarrow \Delta$*, where* Γ *and* Δ *are finite sets of nested formulas.*

For the remainder of this paper we call nested formulas simply formulas. Formulas that are not nested are called *plain.* Observe that sequents $\Gamma \Rightarrow \Delta$ may contain multiple formulas in Δ, i.e. we consider a multi-succedent calculus. The intended interpretation of a nested formula A^n is the plain formula $\mathsf{X}^n A$. The interpretation of a sequent $A_1^{m_1}, \ldots, A_k^{m_k} \Rightarrow B_1^{n_1}, \ldots, B_l^{n_l}$ is the plain formula

$$\bigwedge_{1 \leq i \leq k} \mathsf{X}^{m_i} A_i \to \bigvee_{1 \leq j \leq l} \mathsf{X}^{n_j} B_j$$

We write $M, w \models A^n$ if $M, w \models \mathsf{X}^n A$ and $M, w \models \Gamma \Rightarrow \Delta$ if M, w satisfies the interpretation of $\Gamma \Rightarrow \Delta$. For any set Γ of nested formulas, we define $\Gamma^{+1} = \{A^{n+1} : A^n \in \Gamma\}$.

Definition 4. *The sequent calculus* $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ *consists of the rules in Fig. 2. Rules without premises are called* axioms.

The propositional rules of $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ are based on the multi-succedent calculus **G3im** from Negri and von Plato [18] and the nesting is inspired by the work of Kojima and Igarashi [14]. The rule $\to$L differs from the presentation in Negri and von Plato insofar that there is no weakening in the left premise, resulting in invertibility of $\to$L. The choice to use a multi-succedent instead of a single-succedent calculus is motivated by the former's better compatibility with proof search. Observe that the rule $\to$R has only a single formula in the succedent of the premise, ensuring that the law of excluded middle is not derivable. Moreover, $\to$R can only be applied to implications with nesting level 0. Relaxing this restriction by allowing implications of arbitrary nesting depth is unsound for expanding models but sound and complete for persistent models (see Sect. 6).

The U-rules capture the equivalence $A \mathbin{\mathsf{U}} B \equiv B \vee (A \wedge \mathsf{X}(A \mathbin{\mathsf{U}} B))$ and the 'shift' rule S captures modal necessitation. The rules XL and XR are purely structural as $\mathsf{X}A^n$ has the same interpretation as A^{n+1}. Moreover, note that all rules except S and $\to$R are invertible in the sense that the conclusion is valid

$$\frac{}{\Gamma, A^n \Rightarrow A^n, \Delta}\ \text{id} \qquad \frac{}{\Gamma, \bot^n \Rightarrow \Delta}\ \bot$$

$$\frac{\Gamma, A^n, B^n \Rightarrow \Delta}{\Gamma, A \wedge B^n \Rightarrow \Delta}\ \wedge\text{L} \qquad \frac{\Gamma \Rightarrow A^n, \Delta \quad \Gamma \Rightarrow B^n, \Delta}{\Gamma \Rightarrow A \wedge B^n, \Delta}\ \wedge\text{R}$$

$$\frac{\Gamma, A^n \Rightarrow \Delta \quad \Gamma, B^n \Rightarrow \Delta}{\Gamma, A \vee B^n \Rightarrow \Delta}\ \vee\text{L} \qquad \frac{\Gamma \Rightarrow A^n, B^n, \Delta}{\Gamma \Rightarrow A \vee B^n, \Delta}\ \vee\text{R}$$

$$\frac{\Gamma, A \to B^n \Rightarrow A^n, \Delta \quad \Gamma, B^n \Rightarrow \Delta}{\Gamma, A \to B^n \Rightarrow \Delta}\ \to\text{L} \qquad \frac{\Gamma, A^0 \Rightarrow B^0}{\Gamma \Rightarrow A \to B^0, \Delta}\ \to\text{R}$$

$$\frac{\Gamma, A^{n+1} \Rightarrow \Delta}{\Gamma, \mathsf{X}A^n \Rightarrow \Delta}\ \mathsf{X}\text{L} \qquad \frac{\Gamma \Rightarrow A^{n+1}, \Delta}{\Gamma \Rightarrow \mathsf{X}A^n, \Delta}\ \mathsf{X}\text{R}$$

$$\frac{\Gamma, B^n \Rightarrow \Delta \quad \Gamma, A^n, \mathsf{X}(A \,\mathsf{U}\, B)^n \Rightarrow \Delta}{\Gamma, A \,\mathsf{U}\, B^n \Rightarrow \Delta}\ \mathsf{UL} \qquad \frac{\Gamma \Rightarrow B^n, A^n, \Delta \quad \Gamma \Rightarrow B^n, \mathsf{X}(A \,\mathsf{U}\, B)^n, \Delta}{\Gamma \Rightarrow A \,\mathsf{U}\, B^n, \Delta}\ \mathsf{UR}$$

$$\frac{\Gamma \Rightarrow \Delta}{\Sigma, \Gamma^{+1} \Rightarrow \Delta^{+1}, \Pi}\ \text{S}$$

Fig. 2. Rules of the system $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$. The symbols Γ, Δ, Σ and Π range over arbitrary finite sets of nested formulas which may be empty.

if and only if the premises are.[3] We will therefore refer to S and →R as the *non-invertible rules* and to all other rules as the *invertible rules*.

It will be helpful to refer to formulas according to their role in a particular rule application. For each rule, the distinguished formula in the conclusion is called *principal* and the distinguished formulas in the premises are called its *residuals*; for example, in →L the principal formula is $A \to B^n$ and its residuals are $A \to B^n$, A^n and B^n. In S, all formulas in the conclusion are principal and each formula in the premise is the residual of its corresponding principal formula; in particular, formulas in Σ and Π have no residual. In every rule application, any formula that is neither principal nor residual is called a *side formula*.

A *derivation* in $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ of a sequent σ is a finite or infinite tree whose nodes are labelled according to the rules of $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ and whose root is labelled by σ. We will read trees 'upwards', so the nodes labelled by premises are viewed as successors of the node labelled by the conclusion. A *path* through such a derivation T is a finite or infinite sequence $\rho_0, \rho_1, \ldots$ of nodes of T such that for each index i, ρ_{i+1} is a direct successor of ρ_i in T.

Definition 5. *Let ρ be a path through a derivation T. A* (formula) trace *on ρ is a finite or infinite sequence of nested formulas $A_0, A_1, \ldots$ such that for each index i the following hold.*

1. *A_i occurs on the left-hand side of the sequent labelling ρ_i;*

[3] This is a semantic notion of invertibility. The syntactic invertibility of these rules, meaning that the conclusion is provable if and only if the premises are, will follow from soundness and completeness.

2. *if A_i is a principal formula in the rule applied at ρ_i, then A_{i+1} is a residual formula of A_i in ρ_{i+1};*
3. *if A_i is a side formula in the rule applied at ρ_i, then $A_{i+1} = A_i$.*

For any rule R, we say that a trace $(A_i)_i$ *actively passes through* R if there is an index j such that A_j is a principal formula in an application of R.

Definition 6. *A formula trace is* good *if it actively passes through infinitely many applications of the rule* UL.

The following lemma describes a straightforward yet key property of good formula traces.

Lemma 2. *If $(A_i)_i$ is a good formula trace, then there is a plain formula of the form $A \mathsf{U} B$ and some $j < \omega$ such that for all $k \geq j$, A_k is of the form $A \mathsf{U} B^{m_k}$ or $\mathsf{X}(A \mathsf{U} B)^{m_k}$ for $m_k < \omega$.*

A proof in $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ is defined as follows.

Definition 7. *An $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$-derivation T of a sequent σ is a* proof of σ *if*

1. *every leaf in T is labelled by an axiom;*
2. *every infinite branch of T contains an infinite path that has a good formula trace.*

4 Soundness

This section establishes soundness of $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ with respect to the class of expanding models. The proof proceeds via a standard argument using *signatures*: maps that associate a natural number to each 'relevant' formula in a sequent σ. We assume towards a contradiction that there is a proof π of an invalid sequent σ. Then, using a countermodel of σ, we find an infinite path ρ of invalid sequents in π and assign a signature to each of them. By ensuring that these signatures never increase and decrease when passing through the UL-rule, it then follows that a good formula trace on ρ corresponds to an infinite descent of natural numbers. The aforementioned 'relevant' formulas are called *eventualities*.

For a sequent σ, let Γ_σ and Δ_σ denote, respectively, the left-hand and right-hand side of σ.

Definition 8. *An* eventuality *is a formula of the form $\mathsf{X}^j(A \mathsf{U} B)^n$ with $n, j < \omega$. Given a sequent σ, a formula E is an* eventuality of σ *if E is an eventuality occurring in Γ_σ.*

Let U^k be the operator defined inductively by $A \,\mathsf{U}^0\, B = B$ and $A \,\mathsf{U}^{k+1} B = A \wedge \mathsf{X}(A \,\mathsf{U}^k B)$. For an eventuality $E = \mathsf{X}^j(A \mathsf{U} B)^n$ and $k < \omega$ define

$$E[k] := \mathsf{X}^j(A \,\mathsf{U}^k B)^n.$$

Given a sequent σ, a *signature for* σ is a map τ which assigns a natural number to each eventuality of σ. By $\Gamma_\sigma[\tau]$ we denote the set obtained from Γ_σ by replacing each eventuality E with $E[\tau(E)]$. Furthermore, we let $\sigma[\tau]$ denote the sequent $\Gamma_\sigma[\tau] \Rightarrow \Delta_\sigma$.

Theorem 1. *Every sequent provable in* $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ *is valid over the class of expanding models.*

Proof. Let π be a $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$-proof of σ and suppose, for contradiction, that σ is not valid. Let $M = (W, \leq, f, V)$ be an expanding model and $w \in W$ such that $M, w \not\models \sigma$. For brevity, we will identify each node in π with the sequent that labels it.

We will inductively define a path $(\sigma_i)_i$ of sequents through π, a sequence of worlds $(w_i)_i$ in M and a sequence of signatures $(\tau_i)_i$ such that the following hold for every $i < \omega$:

1. τ_i is a signature for σ_i;
2. $w_i \models \bigwedge \Gamma_{\sigma_i}[\tau_i]$ and $w_i \not\models \bigvee \Delta_{\sigma_i}$ (and thus $w_i \not\models \sigma_i[\tau_i]$ and $w_i \not\models \sigma_i$);
3. for every eventuality E of σ_i, the following hold:
   (a) $\tau_i(E)$ is the least natural number k such that $w_i \models E[k]$;
   (b) if E is a side formula in the rule application with conclusion σ_i, then E is an eventuality of σ_{i+1} and $\tau_{i+1}(E) \leq \tau_i(E)$.

We define $(\sigma_i)_i$, $(w_i)_i$ and $(\tau_i)_i$ as follows.

Set $\sigma_0 = \sigma$. Since $w \not\models \sigma$, there exists a $v \geq w$ such that $v \models \bigwedge \Gamma_\sigma$ and $v \not\models \bigvee \Delta_\sigma$. Set $w_0 = v$ and for every eventuality E in Γ_σ, define $\tau_0(E)$ to be the least k such that $w_0 \models E[k]$.

Suppose σ_i, w_i and τ_i are given. We use a case distinction based on the rule applied at σ_i in π (i.e. the rule that has σ_i as conclusion). Note that this rule cannot be an axiom, since $w_i \not\models \sigma_i$. We only show the cases $\to$R, XL, S, and UL; the other cases are treated in a straightforward way.

$\to$R Suppose $\sigma_i = (\Gamma \Rightarrow A \to B^0, \Delta)$ with $A \to B^0$ principal in the rule application. Let $\sigma_{i+1} = (\Gamma, A^0 \Rightarrow B^0)$. Since $w_i \not\models A \to B^0$, there exists a $w_{i+1} \geq w_i$ such that $w_{i+1} \models A^0$ and $w_{i+1} \not\models B^0$. For any eventuality E in $\Gamma \cup \{A^0\}$, let τ_{i+1} map E to the least k such that $w_{i+1} \models E[k]$. Since $w_{i+1} \geq w_i$, by monotonicity (Lemma 1) we have $\tau_{i+1}(E) \leq \tau_i(E)$ for each eventuality E in Γ.

XL Suppose $\sigma_i = (\Gamma, \mathsf{X}A^n \Rightarrow \Delta)$ with $\mathsf{X}A^n$ principal in the rule application. Let $\sigma_{i+1} = (\Gamma, A^{n+1} \Rightarrow \Delta)$ and $w_{i+1} = w_i$. If A^{n+1} is an eventuality, define $\tau_{i+1}(A^{n+1}) := \tau_i(\mathsf{X}A^n)$. On other eventualities, τ_{i+1} acts as τ_i.

S Suppose $\sigma_i = (\Sigma, \Gamma^{+1} \Rightarrow \Delta^{+1}, \Pi)$ such that $\Gamma \Rightarrow \Delta$ is the premise of the rule application. Let $\sigma_{i+1} = (\Gamma \Rightarrow \Delta)$, $w_{i+1} = f(w_i)$ and $\tau_{i+1}(A^n) = \tau_i(A^{n+1})$ for every eventuality A^n in Γ.

UL Suppose $\sigma_i = (\Gamma, A \mathsf{U} B^n \Rightarrow \Delta)$ with $A \mathsf{U} B^n$ principal in the rule application. If $\tau_i(A \mathsf{U} B^n) = 0$, let $\sigma_{i+1} = (\Gamma, B^n \Rightarrow \Delta)$ and $w_{i+1} = w_i$. If B^n is an eventuality and not in Γ, let τ_{i+1} map B^n to the least k such that $w_{i+1} \models B^n[k]$. On other eventualities, τ_{i+1} acts as τ_i.
Alternatively, if $\tau_i(A \mathsf{U} B^n) > 0$, let $\sigma_{i+1} = (\Gamma, A^n, \mathsf{X}(A \mathsf{U} B)^n \Rightarrow \Delta)$ and $w_{i+1} = w_i$. If A^n is an eventuality and not in Γ, let τ_{i+1} map A^n to the least k such that $w_{i+1} \models A^n[k]$. Define $\tau_{i+1}(\mathsf{X}(A \mathsf{U} B)^n) = \tau_i(A \mathsf{U} B^n) - 1$. On other eventualities, τ_{i+1} acts as τ_i.

It is easy to verify that $(\sigma_i)_i$, $(w_i)_i$ and $(\tau_i)_i$ satisfy properties 1–3.

Since π is a proof, the infinite branch $(\sigma_i)_i$ must contain a good trace $(A_i)_{i\geq j}$ starting in some sequent σ_j. By Lemma 2, we may assume that this trace only passes actively through the rules XL, S and UL, and it cannot pass through the latter in a degenerative way.[4] Now consider the infinite sequence $(\tau_i(A_i))_{i\geq j}$ of natural numbers. Note that, by property 3(b), if A_i is a side formula then $\tau_{i+1}(A_{i+1}) \leq \tau_i(A_i)$. Moreover, if A_i is principal in an application of XL or S then $\tau_{i+1}(A_{i+1}) = \tau_i(A_i)$, and if A_i is principal in a (non-degenerative) application of UL then $\tau_{i+1}(A_{i+1}) < \tau_i(A_i)$. As the trace is good, the latter case occurs infinitely often, and so we obtain an infinite, strictly decreasing sequence of natural numbers and thereby a contradiction.

5 Completeness

This section establishes completeness of $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ with respect to the class of expanding models. For each sequent σ we construct an infinite two-player game between *Prover* (Prov) and *Refuter* (Ref) such that a winning strategy for Prov corresponds to a proof of σ and a winning strategy for Ref to the existence of a countermodel for σ. The game will be played on a *proof search tree*, which is a finitely branching, ill-founded tree that presents a systematic search for a proof of σ. In this tree, non-invertible rules will only be applied to *saturated* sequents.

Definition 9. *A sequent $\Gamma \Rightarrow \Delta$ is* left-saturated *if the following hold.*

1. *if $A \wedge B^n \in \Gamma$, then $A^n, B^n \in \Gamma$;*
2. *if $A \vee B^n \in \Gamma$, then $A^n \in \Gamma$ or $B^n \in \Gamma$;*
3. *if $A \rightarrow B^n \in \Gamma$, then $A^n \in \Delta$ or $B^n \in \Gamma$;*
4. *if $\mathsf{X}A^n \in \Gamma$, then $A^{n+1} \in \Gamma$;*
5. *if $A \,\mathsf{U}\, B^n \in \Gamma$, then there exists an $m \geq n$ such that $B^m \in \Gamma$ and $A^k \in \Gamma$ for all $n \leq k < m$.*

The sequent is saturated *if, in addition,*

6. *if $A \wedge B^n \in \Delta$, then $A^n \in \Delta$ or $B^n \in \Delta$;*
7. *if $A \vee B^n \in \Delta$, then $A^n, B^n \in \Delta$;*
8. *if $\mathsf{X}A^n \in \Delta$, then $A^{n+1} \in \Delta$;*
9. *if $A \,\mathsf{U}\, B^0 \in \Delta$, then $B^0, A^0 \in \Delta$ or $B^0, \mathsf{X}(A \,\mathsf{U}\, B)^0 \in \Delta$.*

Given a sequent σ, we say that a formula ϕ is saturated in σ *if σ satisfies the relevant saturation clause for ϕ.*

Note that the saturation clause for right U-formulas is restricted to the zeroth nesting level. The saturation clause for left U-formulas is needed to ensure that the valuation of the countermodel constructed from a failed proof search is monotone. This will become evident once we define such countermodels later in this section.

[4] Formally, a trace $(A_i)_i$ *passes degeneratively through* UL if there is an A_j of the form $A \,\mathsf{U}\, B^n$ such that $A_{j+1} \in \{A^n, B^n\}$.

As we are working with set sequents, formulas can simultaneously function as principal and side formulas. To avoid creating infinite branches with no good trace, one needs to be explicit about how rules may be applied in the proof search tree. We call an application of a rule *succinct* if the principal formula(s) is not also a side formula, and *preserving* if the principal formula(s) is also a side formula. For example, an application of XL of the form given in Fig. 2 is succinct if $\mathsf{X}A^n \notin \Gamma$ and preserving if $\mathsf{X}A^n \in \Gamma$. Rule applications of $\rightarrow$R and S are always succinct. Note that succinct and preserving are dual notions; we find it useful to refer to them as separate concepts as they each highlight a key property of the proof search tree.

Definition 10. *A* proof search tree T *for a sequent* σ *is a finite or infinite tree whose nodes are labelled according to the rules of* $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ *and in which the following holds.*

1. *The root of* T *is labelled by* σ*.*
2. *A node of* T *is a leaf if and only if it is labelled by an axiom.*
3. *Every left rule application is succinct.*
4. *Every right rule application except* $\rightarrow$R *is preserving.*
5. *No invertible rule is applied to a sequent in which the principal formula is already saturated.*
6. *Instead of the rules* $\rightarrow$R *and* S*, we have the rule*

$$\frac{\Sigma, \Gamma^{+1}, A_0^0 \Rightarrow B_0^0 \quad \dots \quad \Sigma, \Gamma^{+1}, A_k^0 \Rightarrow B_k^0 \quad \Gamma \Rightarrow \Delta}{\Sigma, \Gamma^{+1} \Rightarrow (A_0 \rightarrow B_0)^0, \dots, (A_k \rightarrow B_k)^0, \Delta^{+1}, \Pi}\ \mathrm{C},$$

where it is required that every formula in $\Sigma \cup \Pi$ *is of nesting level* 0*,* Π *does not contain a formula of the form* $A \rightarrow B^0$ *and the conclusion is a saturated sequent. We call the premises of the form* $\Sigma, \Gamma^{+1}, A_i^0 \Rightarrow B_i^0$ *the* left premises*, and* $\Gamma \Rightarrow \Delta$ *the* right premise *of* C*.*

The ‘choice’ rule C represents a choice between non-invertible rules that Prov has to make once the sequent is saturated. Note that the empty sequent is saturated; an empty sequent in a proof search tree can only be the conclusion of a C-rule and has another empty sequent as its only direct successor.

Given a sequent σ, one can build a proof search tree as follows. First try to saturate all left formulas by succinctly applying invertible left rules. If a left-saturated sequent is obtained, saturate all right formulas by preservingly applying invertible right rules, then apply C and start over. Observe that it is possible that some branches in a proof search tree do not contain a saturated sequent due the fifth saturation clause.

The following lemmas describe some key properties of proof search trees. For a node s in a proof search tree, we write $\Gamma_s \Rightarrow \Delta_s$ to denote the sequent labelling the node s.

Lemma 3. *If* T *is a proof search tree wherein* $s \in T$ *is the conclusion of a* C*-application with right premise* $t \in T$*, then the following hold.*

1. *t is labelled by a left-saturated sequent;*
2. *if $r \geq t$ and no* C*-application occurs between t and r, then $\Gamma_r = \Gamma_t$.*

Lemma 4. *Every infinite branch of a proof search tree T contains infinitely many applications of* UL *or* C*.*

Proof. Let $(\rho_i)_{i<\lambda}$ be a branch of T (where $\lambda \leq \omega$). Suppose there exists a suffix $(\rho_i)_{j \leq i<\lambda}$ that contains no applications of UL or C. Due to property 3 to 5 of the proof search tree, there exists a $k \geq j$ such that all formulas except for left U-formulas will be saturated in ρ_k. The only rules which may be applied at that point are UL or C, showing that $(\rho_i)_{i<\lambda}$ must be finite.

Lemma 5. *Every infinite branch of a proof search tree T that contains only finitely many* C*-applications has a suffix with a good formula trace.*

Proof. Let β be an infinite branch of T with finitely many C-applications. Let ρ be a suffix of β that starts after the last C-application. By the previous lemma, ρ must contain infinitely many applications of UL. We show that ρ contains a good trace.

Consider the tree T_ρ of formula traces on ρ (add a fresh node as the root). Now let T'_ρ be the tree obtained from T_ρ by identifying consecutive nodes that are labelled by the same formula. Note that T'_ρ cannot be finite, since ρ must contain infinitely many applications of UL and this rule may not be applied to formulas that also function as a side formula. By König's lemma, T'_ρ contains an infinite branch. Note that this branch corresponds to an infinite formula trace $(A_i)_i$ on ρ that does not stagnate on a side formula, that is, $(A_i)_i$ actively passes through a left rule infinitely often. Property 3 to 5 of the proof search tree and absence of C-applications then imply that $(A_i)_i$ actively passes through UL infinitely often.

We are now ready to define the notion of a refutation which corresponds to a winning strategy for Ref.

Definition 11. *A* refutation *of a sequent σ is a subtree R of a proof search tree T for σ such that the following hold.*

1. *R contains the root of T.*
2. *Every branch of R is infinite.*
3. *If a node s in R is (labelled by) the conclusion of an application of* C *in T, then R contains all direct successors of s in T.*
4. *If a node s in R is (labelled by) the conclusion of an application of any rule other than* C *in T, then R contains exactly one direct successor of s in T.*
5. *No infinite branch of R contains a path with a good formula trace.*

Note that the final condition above together with Lemma 4 implies that every branch in a refutation must contain infinitely many applications of the C-rule.

Proposition 1. *Every sequent with a refutation has an expanding countermodel.*

The proof of the above proposition is provided in the following section. For now, we turn to defining the proof search game which is instrumental in the completeness proof.

Given a sequent σ and a proof search tree T for σ, the *proof search game* $\mathcal{G}(T, \sigma)$ is defined as follows. The game is played by two *players* Prov and Ref. The *arena* of the game is the proof search tree T. Each *play* starts in the root of T, which is labelled by σ. If the current play is in position t, where t is a node of T, and t is owned by player $\mathsf{P} \in \{\mathsf{Prov}, \mathsf{Ref}\}$, then P *plays* by choosing a direct successor of t in T. Prov owns all positions that are conclusions of applications of the C-rule while every other position is owned by Ref. If a play reaches a node that has no successors (i.e. an axiom), then the play ends and is called *finite*; otherwise the play is called *infinite*. Observe that every play directly corresponds to a branch of T. The *winning conditions* are as follows: finite plays are won by Prov and infinite plays are won by Prov if the infinite branch of T to which the play corresponds contains a good trace, and won by Ref otherwise. We make use of the standard notion of a *(winning) strategy* for players. The following lemma is then a straightforward consequence of the winning conditions of the game $\mathcal{G}(T, \sigma)$.

Lemma 6. *If there is a winning strategy for* Prov *in* $\mathcal{G}(T, \sigma)$*, then* σ *has a* $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$*-proof, and if there is a winning strategy for* Ref*, then* σ *has a refutation.*

As the set of winning plays (for each player) is Borel, it follows from Martin's determinacy theorem [17] that the game $\mathcal{G}(T, \sigma)$ is determined for any sequent σ and proof search tree T. That is, exactly one player has a winning strategy in $\mathcal{G}(T, \sigma)$. As every sequent has a proof search tree, completeness of $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ is then obtained as a direct consequence of Proposition 1 and Lemma 6.

Theorem 2. *Every sequent valid over the class of expanding models is provable in* $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$.

5.1 Proof of Proposition 1

Let R be a refutation of σ. Recall that, for any node $s \in R$, $\Gamma_s \Rightarrow \Delta_s$ denotes the sequent labelling s. We define a dynamic model $M = (W, \leq, f, V)$ as follows.

1. $W = R/{\sim}$, where $s \sim t$ iff there exists a path between s and t in which no C-application occurs.
2. Define the function f by
$$f(w) = v \text{ iff there exist s} \in \text{w and t} \in \text{v such that s is the conclusion}$$
$$\text{and t is the } \textit{right} \text{ premise of the same C}-\text{application.}$$
Note that f is a total function, since every branch of R contains infinitely many C-applications and every C-application has a right premise.
3. First define the relation $\leq_0$ on W by
$$w \leq_0 v \text{ iff there exist s } \in \text{w and t } \in \text{v such that s is the conclusion}$$
$$\text{and t a } \textit{left} \text{ premise of the same C}-\text{application.}$$

Then let $\leq$ be the transitive reflexive closure of the relation

$$\leq_1 := \{(f^n(w), f^n(v)) : w \leq_0 v \text{ and } n < \omega\}.$$

4. Define the valuation by $V(w) = \{p \in \mathsf{Prop} : p^0 \in \Gamma_w\}$ where $\Gamma_w = \bigcup_{s \in w} \Gamma_s$.

Similar to Γ_w we write Δ_w for $\bigcup_{s \in w} \Delta_s$.

Lemma 7. *M is an expanding model.*

Proof. Forward confluence follows directly from the definition of $\leq_1$. For monotonicity of the valuation, note that it suffices to show that the relation $\leq_1$ is monotone in V. In the following, we write $[t]$ for the equivalence class of t with respect to $\sim$.

Let $w, v \in W$ with $w \leq_1 v$. Then there exist $n < \omega$ and $s, t \in R$ such that $w = f^n([s])$, $v = f^n([t])$ and t is a left premise of a C-application on s. Note that this means that w is reached from $[s]$ by applying the C-rule n times while always taking the right premise, and similarly for v and $[t]$. From Lemma 3 and definition of C, it follows that for any atomic proposition p,

$$p^0 \in \Gamma_{f^n([s])} \text{ implies } p^n \in \Gamma_{[s]}, \tag{1}$$

$$p^n \in \Gamma_{[t]} \quad \text{ implies } p^0 \in \Gamma_{f^n([t])}. \tag{2}$$

So we have the following chain of implications

$$p^0 \in \Gamma_{f^n([s])} \overset{(1)}{\Longrightarrow} p^n \in \Gamma_{[s]} \Longrightarrow p^n \in \Gamma_{[t]} \overset{(2)}{\Longrightarrow} p^0 \in \Gamma_{f^n([t])},$$

where the middle implication follows from the definition of C. This shows $V(w) \subseteq V(v)$ as required.

Lemma 8. *For any $w \in W$, we have $M, w \models \bigwedge \Gamma_w$ and $M, w \not\models \bigvee \Delta_w$.*

Proof. Let A be a formula. By induction on the logical complexity of A, we simultaneously prove that for any $w \in W$ and $n < \omega$ we have (a) $w \models A^n$ if $A^n \in \Gamma_w$ and (b) $w \not\models A^n$ if $A^n \in \Delta_w$.

We only treat the propositional case and the connectives $\rightarrow$ and U. The proof relies on the C-rule being applied only on a saturated conclusion. Thus the sequent $\Gamma_w \Rightarrow \Delta_w$ is saturated for every $w \in W$.

We begin with (a). Suppose $A^n \in \Gamma_w$. If $A \in \mathsf{Prop}$, then $A^0 \in \Gamma_{f^n(w)}$ and thus $w \models \mathsf{X}^n A$. If $A = B \,\mathsf{U}\, C$, by saturation there exists an $m \geq n$ such that $C^m \in \Gamma_w$ and $B^k \in \Gamma_w$ for all $n \leq k < m$. Thus $w \models A^n$ by the IH. This leaves the case $A = B \rightarrow C$. Let $s \in w$ be the (unique) conclusion of a C-application. By definition of $\rightarrow$L, we have $C^n \in \Gamma_w$ or $B \rightarrow C^n \in \Gamma_s$. In the first case, the IH implies $w \models C^n$ hence $w \models A^n$. The second case is more involved. We have $A^0 \in \Gamma_{f^n(w)}$ by Lemma 3. Define $u := f^n(w)$ and let $v \geq u$. We will restrict ourselves to the case that $v \geq_1 u$; the argument can be extended to the general case using the monotonicity lemma. Let $r, t \in R$ and $m < \omega$ be such that $u = f^m([r])$, $v = f^m([t])$ and t is a left premise of a C-application with conclusion r. Since $A^0 \in \Gamma_u$, we have $A^m \in \Gamma_r$ (by Lemma 3), which implies that $A^m \in \Gamma_t$. As before, we then have $C^m \in \Gamma_{[t]}$ or $A^m \in \Gamma_{t'}$, where $t' \in [t]$ is the conclusion of a C-application. This implies $v \models C^0$ (by the IH) or $A^0 \in \Gamma_v$ (by Lemma 3). In the second case, saturation implies that $C^0 \in \Gamma_v$ or $B^0 \in \Delta_v$. Applying the IH, either $v \models C^0$ or $v \not\models B^0$. Thus $u \models A^0$ and thereby $w \models A^n$.

We now consider (b). Suppose $A^n \in \Delta_w$. If $A \in \mathsf{Prop}$, then $A^n \notin \Gamma_w$ since no sequent in R can be an axiom. By the same argument used to obtain (1), we have $A^0 \notin \Gamma_{f^n(w)}$ and thus $w \not\models \mathsf{X}^n A$. If $A = B \to C$, then $A^0 \in \Delta_{f^n(w)}$. As the C-rule must be applied to some (namely the highest) sequent in the equivalence class $f^n(w)$, it must be the case that $f^n(w)$ has an intuitionistic successor v such that $B^0 \in \Gamma_v$ and $C^0 \in \Delta_v$. The IH then implies $f^n(w) \not\models A^0$ and thus $w \not\models A^n$.

Finally, if $A = B \mathsf{U} C$, then $A^0 \in \Delta_{f^n(w)}$ because UR-applications are preserving. Saturation and the IH implies $f^n(w) \not\models C^0$ and either $f^n(w) \not\models B^0$ or $A^1 \in \Delta_{f^n(w)}$. Similarly, for every $m \geq n$, if $A^1 \in \Delta_{f^m(w)}$ then $f^{m+1}(w) \not\models C^0$ and either $f^{m+1}(w) \not\models B^0$ or $A^1 \in \Delta_{f^{m+1}(w)}$. So either there exists an $m \geq n$ such that $f^m(w) \not\models B^0$ and $f^k(w) \not\models C^0$ for all $n \leq k \leq m$, or $f^m(w) \not\models C^0$ for all $m \geq n$. Either way, $w \not\models A^n$.

We conclude that the expanding model M falsifies σ.

5.2 A Sequent Unprovable with Bounded Nesting

We have shown that the calculus $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ is complete with respect to the class of expanding models via a proof search argument. However, our argument does not yield regular completeness. Observe that in the construction of the proof search tree, there is no bound given on the nesting depth occurring in sequents. Indeed, in order to saturate U-formulas on the left, one has to keep unfolding them until the left premise is chosen, which, in case of a successful branch, might never happen. Hence, proofs might have arbitrary large nesting depth and there is thus no guarantee that infinite branches will contain repetitions. This observation raises the question of whether the completeness proof can be adapted to obtain a bound on the nesting depth occurring in $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$-proofs. Unfortunately, this is not possible, as there are sequents that are not provable in $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ with bounded nesting depth. An example for this is the sequent

$$\Diamond(A \vee B)^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0,$$

where $\Diamond A := \top \mathsf{U} A$ and $\top := \bot \to \bot$. For brevity, instead of the U-rules we will use the following rules for $\Diamond$.

$$\frac{\Gamma, A^n \Rightarrow \Delta \quad \Gamma, \mathsf{X}\Diamond A^n \Rightarrow \Delta}{\Gamma, \Diamond A^n \Rightarrow \Delta}\ \Diamond\mathrm{L} \qquad \frac{\Gamma \Rightarrow A^n, \mathsf{X}\Diamond A^n, \Delta}{\Gamma \Rightarrow \Diamond A^n, \Delta}\ \Diamond\mathrm{R}$$

It is easy to see that any formula in the $\Diamond$-fragment of iLTL is provable in $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ if and only if it is provable in $\mathsf{iLTL}_{\mathsf{e}}^{\mathsf{nest}}$ with the $\Diamond$-rules instead of the U-rules.

Let us now consider the following proof π of the sequent $\Diamond(A \vee B)^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$.

$$
\begin{prooftree}
\AxiomC{π_0}
\noLine
\UnaryInfC{$A \vee B^0 \Rightarrow \Delta$}
\AxiomC{π_1}
\noLine
\UnaryInfC{$A \vee B^1 \Rightarrow \Delta$}
\AxiomC{$\vdots$}
\noLine
\UnaryInfC{$\Diamond(A \vee B)^2 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\RightLabel{XL}
\UnaryInfC{$\mathsf{X}\Diamond(A \vee B)^1 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\RightLabel{$\Diamond$L}
\BinaryInfC{$\Diamond(A \vee B)^1 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\RightLabel{XL}
\UnaryInfC{$\mathsf{X}\Diamond(A \vee B)^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\RightLabel{$\Diamond$L}
\BinaryInfC{$\Diamond(A \vee B)^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\end{prooftree}
$$

The subproof π_0 is given as follows.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{id}
\UnaryInfC{$A^0, C^0 \Rightarrow A^0, \mathsf{X}\Diamond A^0$}
\RightLabel{$\Diamond$R}
\UnaryInfC{$A^0, C^0 \Rightarrow \Diamond A^0$}
\RightLabel{$\to$R}
\UnaryInfC{$A^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\AxiomC{}
\RightLabel{id}
\UnaryInfC{$B^0, C^0 \Rightarrow B^0, \mathsf{X}\Diamond B^0$}
\RightLabel{$\Diamond$R}
\UnaryInfC{$B^0, C^0 \Rightarrow \Diamond B^0$}
\RightLabel{$\to$R}
\UnaryInfC{$B^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\RightLabel{$\vee$L}
\BinaryInfC{$A \vee B^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$}
\end{prooftree}
$$

The subproof π_1 is similar, the only difference being that the formulas $\Diamond A^0$ and $\Diamond B^0$ have to be unfolded twice to reach an axiom instead of just once. In the same way, we obtain the subproofs π_i for each $i < \omega$.

Note that π is indeed a proof, as it contains only one infinite branch and this branch contains a good trace, and that the nesting depth in π is unbounded. Furthermore, note that *any* proof of this sequent will have an infinite branch on the right with unbounded nesting levels. Working bottom-up, applying any other rule than $\Diamond$L to the root sequent results in an unprovable sequent, and applying any rule other than XL to its right premise results in an unprovable sequent as well. The same argument applies to each sequent in the right-most branch of π.

Interestingly, allowing analytic cuts there is a proof of this sequent with nesting depth bounded by 1, the cut formula being $\Diamond A \vee \Diamond B^0$.

6 Persistency

The system $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ can be adapted to a sound and complete proof system for the logic iLTL_p of validities over persistent models.

Definition 12. *The sequent calculus* $\mathsf{iLTL}_\mathsf{p}^\mathsf{nest}$ *consists the rules of* $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ *except* S *and* $\to$R *which are replaced by*

$$
\frac{\Gamma, A^n \Rightarrow B^n}{\Gamma \Rightarrow A \to B^n, \Delta}\ \to\mathrm{R}_\mathsf{p}
$$

Derivations, paths, (good) formula traces and proofs are defined for $\mathsf{iLTL}_\mathsf{p}^\mathsf{nest}$ just as for $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$, and it is easy to see that Lemma 2 still holds. To prove soundness, one can simply follow the proof of Theorem 1 and in the case for $\to\mathrm{R}_\mathsf{p}$ invoke the validity of $(\mathsf{X}A \to \mathsf{X}B) \to \mathsf{X}(A \to B)$ over the class of persistent models.

To show completeness, we will adapt the proof search for $\mathsf{iLTL}_\mathsf{e}^\mathsf{nest}$ by introducing different levels of saturation.

Definition 13. *Let $k < \omega$. A sequent $\Gamma \Rightarrow \Delta$ is* k-saturated *if it satisfies clauses 1-8 of Definition 9 and the additional clause*

9. *for all $n \leq k$, if $A \mathsf{U} B^n \in \Delta$, then $B^n, A^n \in \Delta$ or $B^n, \mathsf{X}(A \mathsf{U} B)^n \in \Delta$.*

Given a sequent σ, we say that a formula A is k-saturated in σ *if σ satisfies the relevant k-saturation clause for A.*

Note that 0-saturation is equivalent to our earlier notion of saturation.

To keep track of the level of saturation in sequents, the proof search tree will be labelled by *indexed sequents* $\Gamma \Rightarrow_k \Delta$, that is, sequents equipped with a natural number $k < \omega$.

Definition 14. *A* persistent proof search tree T *for a sequent $\Gamma \Rightarrow \Delta$ is a finite or infinite tree whose nodes are labelled with indexed sequents following the rules of $\mathsf{iLTL}_{\mathsf{p}}^{\mathsf{nest}}$ such that:*

1. *The root of T is labelled by $\Gamma \Rightarrow_0 \Delta$.*
2. *A node of T is a leaf if and only if it is labelled by an axiom.*
3. *Invertible rule applications leave the index of a sequent unchanged.*
4. *Every left rule application is succinct.*
5. *Every right rule application apart from $\rightarrow\mathrm{R}_{\mathsf{p}}$ is preserving.*
6. *No invertible rule is applied to a sequent of index k in which the principal formula is already k-saturated.*
7. *In place of the rule $\rightarrow\mathrm{R}_{\mathsf{p}}$, the rule*

$$\frac{\Gamma, A_0^k \Rightarrow_0 B_0^k \quad \cdots \quad \Gamma, A_j^k \Rightarrow_0 B_j^k \quad \Gamma \Rightarrow_{k+1} (A_0 \rightarrow B_0)^k, \ldots, (A_j \rightarrow B_j)^k, \Delta}{\Gamma \Rightarrow_k (A_0 \rightarrow B_0)^k, \ldots, (A_j \rightarrow B_j)^k, \Delta}\ \mathrm{C}_{\mathsf{p}}$$

is utilised, where Δ may not contain a formula of the form $A \rightarrow B^k$ and the conclusion of the rule is a k-saturated sequent.

It is easy to see that every sequent has a persistent proof search tree and that Lemma 3, 4 and 5 also hold for persistent proof search trees. Following Definition 11, we define a *persistent refutation* as a subtree of a persistent proof search tree satisfying properties 1-5 of Definition 11, with C replaced by C_{p}. As before, the fifth property ensures that every branch in a persistent refutation passes through the C_{p}-rule infinitely often.

Via a game-theoretic argument, we obtain completeness of $\mathsf{iLTL}_{\mathsf{p}}^{\mathsf{nest}}$ as a corollary of the following proposition.

Proposition 2. *If a sequent σ has a persistent refutation, then it has a persistent countermodel.*

Due to space limit the proof is omitted. The main difference to the proof of Proposition 1 is that, when constructing a persistent countermodel from a persistent refutation, right premises of the C_{p}-rule are not viewed as temporal successors but as a further description of the current world w. In the limit, this description fully determines the temporal 'successors' $f^n(w)$ for every n, whereby these successors can be added accordingly. Due to this limit construction, worlds in the obtained countermodel may have infinitely many intuitionistic successors, which is not the case for the countermodel obtained in Proposition 1.

7 Conclusion

This investigation is part of a larger programme of devising sequent calculi for intuitionistic modal logic with fixed points to establish fundamental properties such as decidability and algorithmic proof search. To this aim, we introduce ill-founded cut-free sequent calculi for intuitionistic linear-time temporal logic over expanding and persistent models, denoted $\mathsf{iLTL_e}$ and $\mathsf{iLTL_p}$ respectively. The presented systems and the techniques devised to establish soundness and completeness are inspired by the study of ill-founded proof systems for classical temporal logics. In particular, we have illustrated how the method of proof search can be adapted to the intuitionistic realm.

A natural direction for future research is to extend $\mathsf{iLTL_e}$ and $\mathsf{iLTL_p}$ to logics containing greatest fixed point operators such as 'henceforth' and, more generally, 'release'. The latter is the classical dual of U which is not definable from U in the intuitionistic setting [3]. Although we believe that our approach can be extended to handle more expressive temporal logics, an adaptation of the proof search strategy is by no means trivial. The presence of greatest fixed point formulas on the left-hand side of a sequent presents a challenge in ensuring that the model constructed from a refutation satisfies monotonicity.

Another possible direction is to devise complete cyclic calculi for iLTL-based logics. The main difficulty in turning an ill-founded proof into a cyclic one lies in our reliance on nested sequents. In the completeness proof, there is no guarantee that every infinite branch in a proof contains a repeated sequent. Indeed, as shown in Sect. 5.2, the sequent $\Diamond(A \vee B)^0 \Rightarrow C \to \Diamond A^0, C \to \Diamond B^0$ admits a proof in $\mathsf{iLTL_e^{nest}}$ only with an unbounded nesting depth. This implies that a simple definition of repetition in an infinite branch will not result in a complete cyclic system. Incorporating the cut-rule into the systems, one can obtain a proof of the sequent wherein the nesting depth is at most 1. Since the required application of cut in this example requires only analytic formulas, it is worthwhile investigating whether the presented systems can be turned into cyclic systems with analytic cuts.

References

1. Afshari, B., Leigh, G.E., Menéndez Turata, G.: A cyclic proof system for full computation tree logic. In: Klin, B., Pimentel, E. (eds.) 31st EACSL Annual Conference on Computer Science Logic (CSL 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 252, pp. 1–19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.CSL.2023.5
2. Alechina, N., Mendler, M., de Paiva, V., Ritter, E.: Categorical and Kripke semantics for constructive S4 modal logic. In: Fribourg, L. (ed.) CSL 2001. LNCS, vol. 2142, pp. 292–307. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44802-0_21
3. Balbiani, P., Boudou, J., Diéguez, M., Fernández-Duque, D.: Intuitionistic linear temporal logics. ACM Trans. Comput. Logic **21**(2), 3365833 (2019). https://doi.org/10.1145/3365833

4. Boudou, J., Diéguez, M., Fernández-Duque, D.: A decidable intuitionistic temporal logic. In: Goranko, V., Dam, M. (eds.) 26th EACSL Annual Conference on Computer Science Logic (CSL 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 82, pp. 1–17. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2017). https://doi.org/10.4230/LIPIcs.CSL.2017.14. http://drops.dagstuhl.de/opus/volltexte/2017/7701
5. Boudou, J., Diéguez, M., Fernández-Duque, D.: Complete intuitionistic temporal logics for topological dynamics. J. Symb. Log. **87**(3), 995–1022 (2022)
6. Chopoghloo, S., Moniri, M.: A strongly complete axiomatization of intuitionistic temporal logic. J. Log. Comput. **31**(7), 1640–1659 (2021). https://doi.org/10.1093/logcom/exab041
7. Davies, R., Pfenning, F.: A modal analysis of staged computation. J. ACM **48**(3), 555–604 (2001). https://doi.org/10.1145/382780.382785
8. Dax, C., Hofmann, M., Lange, M.: A proof system for the linear time μ-calculus. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 273–284. Springer, Heidelberg (2006). https://doi.org/10.1007/11944836_26
9. De Paiva, V., Artemov, S.: Journal of Applied Logics, Volume 8, Number 8, September 2021. Special Issue: Intuitionistic Modal Logic and Applications. College Publications (2021). https://books.google.se/books?id=45ipzgEACAAJ
10. Doumane, A.: Constructive completeness for the linear-time μ-calculus. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, 20–23 June 2017, pp. 1–12. IEEE Computer Society (2017)
11. Fernández-Duque, D.: The intuitionistic temporal logic of dynamical systems. Logic. Methods Comput. Sci. **14**, 35 (2018)
12. Jeltsch, W.: Temporal logic with "until", functional reactive programming with processes, and concrete process categories. In: Proceedings of the 7th workshop on Programming languages meets program verification, pp. 69–78 (2013)
13. Kamide, N., Wansing, H.: Combining linear-time temporal logic with constructiveness and paraconsistency. J. Appl. Log. **8**(1), 33–61 (2010)
14. Kojima, K., Igarashi, A.: Constructive linear-time temporal logic: Proof systems and Kripke semantics. Inf. Comput. **209**, 1491–1503 (2011). https://doi.org/10.1016/j.ic.2010.09.008
15. Kokkinis, I., Studer, T.: Cyclic proofs for linear temporal logic. Ontos Math. Logic **6**, 171–192 (2016)
16. Maier, P.: Intuitionistic LTL and a new characterization of safety and liveness. In: Marcinkowski, J., Tarlecki, A. (eds.) Computer Science Logic, pp. 295–309. Springer, Berlin Heidelberg, Berlin, Heidelberg (2004)
17. Martin, D.A.: Borel determinacy. Annal. Math. **102**(2), 363–371 (1975). http://www.jstor.org/stable/1971035
18. Negri, S., von Plato, J., Ranta, A.: Structural Proof Theory. Cambridge University Press, New York (2001)
19. Niwinski, D., Walukiewicz, I.: Games for the mu-calculus. Theoret. Comput. Sci. **163**(1&2), 99–116 (1996)
20. Simpson, A.: The proof theory and semantics of intuitionistic modal logic, Ph. D. thesis, University of Edinburgh (1994)
21. Taha, W., Nielsen, M.F.: Environment classifiers. SIGPLAN Not. **38**(1), 26–37 (2003)
22. Tsukada, T., Igarashi, A.: A logical foundation for environment classifiers. In: Curien, P.-L. (ed.) TLCA 2009. LNCS, vol. 5608, pp. 341–355. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02273-9_25

23. Wijesekera, D.: Constructive modal logics I. Ann. Pure Appl. Logic **50**(3), 271–301 (1990)
24. Yuse, Y., Igarashi, A.: A modal type system for multi-level generating extensions with persistent code. In: International Conference on Principles and Practice of Declarative Programming: Proceedings of the 8th ACM SIGPLAN symposium on Principles and practice of declarative programming; 10–12 July 2006, pp. 201–212. PPDP 2006, ACM (2006)

Proof Systems for the Modal μ-Calculus Obtained by Determinizing Automata

Maurice Dekker, Johannes Kloibhofer, Johannes Marti(✉), and Yde Venema

ILLC, University of Amsterdam, Amsterdam, Netherlands
pmauricedekker@gmail.com, {j.kloibhofer,y.venema}@uva.nl,
johannes.marti@gmail.com

Abstract. Automata operating on infinite objects feature prominently in the theory of the modal μ-calculus. One such application concerns the tableau games introduced by Niwiński & Walukiewicz, of which the winning condition for infinite plays can be naturally checked by a nondeterministic parity stream automaton. Inspired by work of Jungteerapanich and Stirling we show how determinization constructions of this automaton may be used to directly obtain proof systems for the μ-calculus. More concretely, we introduce a binary tree construction for determinizing nondeterministic parity stream automata. Using this construction we define the annotated cyclic proof system BT, where formulas are annotated by tuples of binary strings. Soundness and Completeness of this system follow almost immediately from the correctness of the determinization method.

Keywords: modal mu-calculus · derivation system · determinisation of Büchi and parity automata · non-wellfounded and cyclic proofs

1 Introduction

The Modal μ-calculus. The modal μ-calculus is a natural extension of basic modal logic with explicit least and greatest fixpoint operators. Allowing the formulation of various recursive phenomena, this extension raises the expressive power of the language (at least when it comes to bisimulation-invariant properties of transition systems) to that of monadic second-order logic [12]. The μ-calculus is generally regarded as a universal specification language, since it embeds most other logics that are used for this purpose, such as LTL, CTL, CTL* and PDL. Despite its expressive power the μ-calculus has still reasonable computational properties; its model checking problem is in quasi-polynomial time [4] and its satisfiability problem is EXPTIME-complete [7]. Another interesting feature of the theory of the modal μ-calculus lies in its connections with other fields, in particular the theory of finite automata operating on infinite objects, and that of infinite games.

Derivation Systems. Given the importance of the modal μ-calculus, there is a natural interest in the development and study of derivation systems for its validities. And indeed, already in [15] Kozen proposed an axiomatization. Despite the naturality of this axiom system, he only established a partial completeness result, and it took a substantial amount of time before Walukiewicz [25] managed to prove soundness and completeness for the full language.

Kozen's axiomatization amounts to a Hilbert-style derivation system, making it less attractive for proof search. If one is interested in a cut-free system, a good starting point is the two-player tableau-style game introduced by Niwiński & Walukiewicz [19]. Here we will present their system in the shape of a derivation system NW (this change of perspective can be justified by identifying winning strategies for one of the players in the game with NW-proofs). NW is a one-sided sequent system which allows for infinite proofs: although its proof rules are completely standard (and finitary), due to the unfolding rules for the fixpoint operators, derivations may have infinite branches. A crucial aspect of the NW-system is that one has to keep track of the *traces* of individual formulas along the infinite branches. A derivation will only count as a proper proof if each of its infinite branches is *successful*, in the sense that it carries a so-called ν-trace: a trace which is dominated by a *greatest* fixpoint operator.

This condition is easy to formulate but not so nice to work with. One could describe the subsequent developments in the proof theory for the modal μ-calculus as a series of modifications of the system NW which aim to get a grip on the complexities and intricacies of the above-mentioned traces, and in particular, to use the resulting "trace management" for the introduction of finitary, cyclic proof systems. Landmark results were obtained by Jungteerapanich [13] and Stirling [23], who introduced cyclic proof systems for the μ-calculus, two calculi that we will identify here under the name JS.

Automata and Derivation Systems. Applications of automata theory are ubiquitous in the theory of the modal μ-calculus, and the area of proof theory is no exception. In particular, Niwiński & Walukiewicz [19] observed that infinite matches of their game, corresponding to infinite branches in an NW-derivation, can be seen as infinite words or *streams* over some finite alphabet. It follows that *stream automata* (automata operating on infinite words) can be used to determine whether such a match/branch carries a ν-trace. Niwiński & Walukiewicz used this perspective to link their results to the exponential-time complexity of the satisfiability problem for the μ-calculus.

A key contribution of Jungteerapanich and Stirling [13,23] was to bring automata *inside* the proof system. The basic idea would be to decorate each sequent in a derivation with a state of the stream automaton which recognizes whether an infinite branch is successful or not; starting from the root, the successive states decorating the sequents on a given branch simply correspond to a run of the automaton on this branch. For this idea to work one needs the stream automaton to be *deterministic.* To see this, observe that two successful but distinct branches in a derivation would generally require two distinct runs, and in the case of a nondeterministic automaton, these two runs might already diverge before the two branches split.

Interestingly, there is a natural stream automaton recognizing the successful branches of an NW-derivation: One may simply take the states of such an automaton to be the formulas in the (Fischer-Ladner) *closure* of the root sequent. But given the *nondeterministic* format of this automaton, before it can be used in a proof system, we need to transform it into an equivalent deterministic one. This explains the relevance of constructions for determinizing stream automata to the proof theory of the modal μ-calculus.

Determinization of Stream Automata. Using the ideas we just sketched, one may obtain sound and complete derivation systems for the modal μ-calculus in an easy way. For any deterministic automaton $\mathbb{A}$ that recognizes the successful branches in NW-derivations, one could simply introduce new-style sequents consisting of an NW-sequent decorated with a state of $\mathbb{A}$, and adapt the proof rules of NW incorporating the transition map of $\mathbb{A}$. This could be done in such a way that the stream of decorations of an infinite branch corresponds to the run of $\mathbb{A}$ on the stream of sequents of the same branch. The trace condition of NW-derivations could then be replaced by the acceptance condition of $\mathbb{A}$ (which is generally much simpler, since it does not refer to traces).

More interesting is to use specific determinization constructions, in order to design more attractive proof systems or to prove results *about* the derivation system (and thus, potentially, about the μ-calculus). In particular, some determinization constructions are based on a *power construction*, meaning that the states of the deterministic automaton consist of *macrostates* (*subsets* of the nondeterministic original) with some additional structure. Such constructions allow for proof calculi where this additional structure is incorporated into the sequents. For instance, the derivation system JS is based on the well-known Safra construction [20], in which the states of the deterministic automaton consist of macrostates of the original automaton that are organised by means of so-called *Safra trees*. Concretely, the (augmented) sequents in JS consist of a set of *annotated formulas*, with the annotations indicating the position of the formula in the Safra tree and a so-called *control* which provides additional information on the Safra tree.

Our Contribution. Our overall goal is to explicitize the role of automata theory in the design of derivation systems for the modal μ-calculus (and other fixpoint logics). Our point is that distinct determinization constructions lead to distinct sequent system, and that we may look for alternatives to the Safra construction. Concretely the contribution of this paper is threefold:

1. We provide a new determinization construction for both Büchi and parity stream automata which is based on binary trees. Our construction is similar to constructions related to so-called profile trees [8,16].
2. We apply our construction to obtain a new derivation system BT for the modal μ-calculus. While our system is similar in spirit to the system JS, a

key difference is that our sequents consist of annotated formulas, and nothing else.

3. We establish the soundness and completeness of BT. A distinguishing feature of our approach is that (up to some optimizations) this result is a *direct* consequence of the soundness and completeness of NW and the adequacy of our determinization construction.

Related Work. There is an extensive literature on applications of automata theory in the theory of the modal μ-calculus, among others [6,11,12,26]. Jungteerapanich and Stirling [13,23] were the first to obtain an annotated proof system inspired by the determinization of automata. The proof system Focus for the alternation-free μ-calculus designed by Marti & Venema [18] originates with a rather simple determinization construction for so-called weak automata. In [17], Leigh & Wehr also take a rather general approach towards the use of determinization constructions in the design of derivation systems, but they confine attention to the Safra construction.

Overview of Paper. In the next section we provide the necessary background material on binary trees, on ω-automata, on the modal μ-calculus and the proof system NW; doing so we fix our notation. In Sect. 3 we introduce a new determinization method for nondeterministic Büchi and parity automata. We will use this construction to prove the soundness and completeness of the proof system BT, which we introduce in Sect. 4. All missing proofs can be found in the extended version of this paper [5].

2 Preliminaries

Binary Trees. We let 2^* denote the set of *binary strings*; we write $<$ for the lexicographical order of 2^*, and $\sqsubseteq$ for the (initial) substring relation given by $s \sqsubseteq t$ if $sr = t$ for some r. *Substitution* for binary strings is defined in the following way: Let $s, t, \tilde{s}, r \in 2^*$ be such that $s = t\tilde{s}$, then $s[t \backslash r]$ denotes the binary string $r\tilde{s}$. A *binary tree* is a finite set of binary strings $T \subseteq 2^*$ such that $s0 \in T \Rightarrow s \in T$ and $s0 \in T \Leftrightarrow s1 \in T$. Here we let $\text{leaves}(T) = \{s \in T \mid s0 \notin T\}$ denote its set of *leaves*, and $\text{minL}(T)$ its *minimal* leaf of T, i.e. the unique leaf of the form $0 \cdots 0$. A set of binary strings L is a *set of leaves of a binary trees* if for all $s \neq t \in L$ we have $s \not\sqsubseteq t$ and $\text{tree}(L) = \{s \in 2^* \mid \exists t \in L : s \sqsubseteq t\}$ is a binary tree.

Stream Automata. A *non-deterministic automaton* over a finite alphabet Σ is a quadruple $\mathbb{A} = \langle A, \Delta, a_I, \text{Acc} \rangle$, where A is a finite set, $\Delta : A \times \Sigma \to \mathcal{P}(A)$ is the transition function of $\mathbb{A}$, $a_I \in A$ its initial state and $\text{Acc} \subseteq A^\omega$ its acceptance condition. An automaton is called *deterministic* if $|\Delta(a, y)| = 1$ for all pairs $(a, y) \in A \times \Sigma$. A *run* of an automaton $\mathbb{A}$ on a stream $w = y_0 y_1 y_2 ... \in \Sigma^\omega$ is a stream $a_0 a_1 a_2 ... \in A^\omega$ such that $a_0 = a_I$ and $a_{i+1} \in \Delta(a_i, y_i)$ for all $i \in \omega$. A stream w is *accepted* by $\mathbb{A}$ if there is a run of $\mathbb{A}$ on w, which is in Acc; we define $\mathcal{L}(\mathbb{A})$ to be the set of all accepting streams of $\mathbb{A}$.

The acceptance condition can be given in different ways: A *Büchi* condition is given as a subset $F \subseteq A$. The corresponding acceptance condition is the set of runs, which contain infinitely many states in F. A *parity* condition is given as a map $\Omega : A \to \omega$. The corresponding acceptance condition is the set of runs α such that $\min\{\Omega(a) \mid a \text{ occurs infinitely often in } \alpha\}$ is even. A *Rabin* condition is given as a set $R = ((G_i, B_i))_{i \in I}$ of pairs of subsets of A. The corresponding acceptance condition is the set of runs α for which there exists $i \in I$ such that α contains infinitely many states in G_i and finitely many in B_i. Automata with these acceptance conditions are called *Büchi*, *parity* and *Rabin automata*, respectively.

Modal μ-calculus: Syntax. The set $\mathcal{L}_\mu$ of *formulas* of the modal μ-calculus is generated by the grammar

$$\varphi ::= p \mid \overline{p} \mid \bot \mid \top \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \Diamond\varphi \mid \Box\varphi \mid \mu x.\varphi \mid \nu x.\varphi,$$

where p and x are taken from a fixed set Prop of propositional variables and in formulas of the form $\mu x.\varphi$ and $\nu x.\varphi$ there are no occurrences of $\overline{x}$ in φ.

Formulas of the form $\mu x.\varphi$ ($\nu x.\varphi$) are called *μ-formulas* (*ν-formulas*, respectively); formulas of either kind are called *fixpoint formulas*. We write $\eta, \lambda \in \{\mu, \nu\}$ to denote an arbitrary fixpoint operator. We use standard terminology and notation for the binding of variables by the fixpoint operators and for substitutions, and make sure only to apply substitution in situations where no variable capture will occur. An important use of the substitution operation concerns the *unfolding* $\chi[\xi/x]$ of a fixpoint formula $\xi = \eta x.\chi$.

Given two formulas $\varphi, \psi \in \mathcal{L}_\mu$ we write $\varphi \to_C \psi$ if ψ is either a direct boolean or modal subformula of φ, or else φ is a fixpoint formula and ψ is its unfolding. The *closure* $\mathsf{Clos}(\Phi) \subseteq \mathcal{L}_\mu$ of $\Phi \subseteq \mathcal{L}_\mu$ is the least superset of Φ that is closed under this relation. It is well known that $\mathsf{Clos}(\Phi)$ is finite iff Φ is finite. A *trace* is a sequence $(\varphi_n)_{n<\kappa}$, with $\kappa \leq \omega$, such that $\varphi_n \to_C \varphi_{n+1}$, for all $n + 1 < \kappa$.

We define a *dependence order* on the fixpoint formulas occurring in Φ, written $\mathsf{Fix}(\Phi)$, by setting $\eta x.\varphi <_\Phi \lambda y.\psi$ (where smaller in $<_\Phi$ means being of higher priority) if $\mathsf{Clos}(\eta x.\varphi) = \mathsf{Clos}(\lambda y.\psi)$ and $\eta x.\varphi$ is a subformula of $\lambda y.\psi$. One may define a *parity function* $\Omega : \mathsf{Fix}(\Phi) \to \omega$, which respects this order (i.e., $\Omega(\eta x.\varphi) < \Omega(\lambda y.\psi)$ if $\eta x.\varphi <_\Phi \lambda y \psi$) and satisfies $\Omega(\eta x.\varphi)$ is even iff $\eta = \nu$. Let $\max_\Omega(\Phi) = \max\{\Omega(\nu x.\varphi) \mid \nu x.\varphi \in \mathsf{Fix}(\Phi)\}$.

It is well known that any infinite trace $\tau = (\varphi_n)_{n<\kappa}$ features a unique formula φ that occurs infinitely often on τ and is a subformula of φ_n for cofinitely many n. This formula is always a fixpoint formula, and where it is of the form $\eta x.\psi$ we call τ an *η-trace*.

Since the semantics of the modal μ-calculus only plays an indirect role in our paper, we refrain from giving the definition.

Non-wellfounded Proofs. A sequent Γ is a finite set of μ-calculus formulas, possibly with additional structure such as annotations. Rules have the following form, possibly with additional side conditions:

$$R\colon \frac{\Gamma_1 \quad \cdots \quad \Gamma_n}{\Gamma} \qquad\qquad \begin{array}{c} [\Gamma]^{\mathsf{x}} \\ \vdots \\ \mathsf{D}^{\mathsf{x}}\colon \ \frac{\Gamma}{\Gamma} \end{array}$$

A rule R, where $n = 0$, is called an axiom. The rules D^{x} are called *discharge* rules. Each discharge rule is marked by a unique *discharge token* taken from a fixed infinite set $\mathcal{D} = \{\mathsf{x}, \mathsf{y}, ...\}$.

Definition 1. *A* derivation system $\mathcal{P}$ *is a set of rules. A* $\mathcal{P}$ derivation $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ *is a quintuple such that* (T, P) *is a, possibly infinite, tree with nodes* T *and parent relation* P*;* S *is a function that maps every node* $u \in T$ *to a non-empty sequent* Σ_u*;* R *is a function that maps every node* $u \in T$ *to its* label $\mathsf{R}(u)$*, which is either (i) the name of a rule in* $\mathcal{P}$ *or (ii) a discharge token; and* f *is a partial function that maps some nodes* $u \in T$ *to its* principal formula $\mathsf{f}(u) \in \mathsf{S}(u)$*. If a specific formula* φ *in the conclusion of a rule is designated, then* $\mathsf{f}(u) = \varphi$ *and otherwise* $\mathsf{f}(u)$ *is undefined. To qualify as a derivation, such a quintuple is required to satisfy the following conditions:*

1. *If a node is labeled with the name of a rule then it has as many children as the rule has premises, and the annotated sequents at the node and its children match the specification of the rules.*
2. *If a node is labeled with a discharge token then it is a leaf. For every leaf* l *that is labeled with a discharge token* $\mathsf{x} \in \mathcal{D}$ *there is exactly one node* $u \in T$ *that is labeled with* D^{x}*. This node* u *and its child are proper ancestors of* l*. In this situation we call* l *a* discharged leaf*, and* u *its* companion*; we write* c *for the function that maps a discharged leaf* l *to its companion* $c(l)$ *and write* $p(l)$ *for the path in* T *from* $c(l)$ *to* l*.*

A derivation $\pi' = (T', P', \mathsf{S}', \mathsf{R}', \mathsf{f}')$ is a *subderivation* of $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ if (T', P') is a subtree of (T, P) and $\mathsf{S}', \mathsf{R}', \mathsf{f}'$ and $\mathsf{S}, \mathsf{R}, \mathsf{f}$ are equal on (T', P'). A derivation π is called *regular* if it has finitely many distinct subderivations.

Definition 2. *Let* $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ *be a derivation. We define two graphs we are interested in: (i) The usual* proof tree $\mathcal{T}_\pi = (T, P)$ *and (ii) the* proof tree with back edges $\mathcal{T}_\pi^C = (T, P^C)$*, where* $P^C = P \cup \{(l, c(l)) \mid l \text{ is a discharged leaf}\}$ *is the parent relation plus back-edges for every discharged leaf.*

A strongly connected subgraph *in* $\mathcal{T}_\pi^C$ *is a set* S *of nodes, such that for every* $u, v \in S$ *there is a* P^C*-path from* u *to* v*.*

The NW *Proof System.* The rules of the derivation system NW, which is directly based on the tableau games introduced by Niwiński & Walukiewicz [19], are given in Fig. 1.

In order to decide whether an NW derivation qualifies as a proper *proof*, one has to keep track of the development of individual formulas along infinite branches of the proofs.

$$\mathsf{Ax1}\ \frac{}{p, \bar{p}, \Gamma} \qquad \mathsf{Ax2}\ \frac{}{\top, \Gamma} \qquad \mathsf{R}_\vee\ \frac{\varphi, \psi, \Gamma}{\varphi \vee \psi, \Gamma} \qquad \mathsf{R}_\wedge\ \frac{\varphi, \Gamma \qquad \psi, \Gamma}{\varphi \wedge \psi, \Gamma}$$

$$\mathsf{R}_\Box\ \frac{\varphi, \Gamma}{\Box\varphi, \Diamond\Gamma, \Delta} \qquad \mathsf{R}_\mu\ \frac{\varphi[\mu x.\varphi/x], \Gamma}{\mu x.\varphi, \Gamma} \qquad \mathsf{R}_\nu\ \frac{\varphi[\nu x.\varphi/x], \Gamma}{\nu x.\varphi, \Gamma}$$

Fig. 1. Rules of NW

Definition 3. *Let Γ, Γ' be sequents, ξ be a formula such that Γ is the conclusion and Γ' is a premise of a rule in Fig. 1 with principal formula ξ. We define the* active *and* passive trail relation $\mathsf{A}_{\Gamma,\xi,\Gamma'}, \mathsf{P}_{\Gamma,\xi,\Gamma'} \subseteq \Gamma \times \Gamma'$. *Both relations are defined via a case distinction on ξ:*

Case $\xi = \Box\varphi$: *Then $\Gamma = \Box\varphi, \Diamond\Lambda, \Delta$ and $\Gamma' = \varphi, \Lambda$. We define $\mathsf{A}_{\Gamma,\xi,\Gamma'}, = \{(\Box\varphi, \varphi)\} \cup \{(\Diamond\chi, \chi) \mid \chi \in \Lambda\}$ and $\mathsf{P}_{\Gamma,\xi,\Gamma'} = \varnothing$.*

Case $\xi = \varphi \vee \psi$: *Then $\Gamma = \varphi \vee \psi, \Lambda$ and $\Gamma' = \varphi, \psi, \Lambda$. We define $\mathsf{A}_{\Gamma,\xi,\Gamma'} = \{(\varphi \vee \psi, \varphi), (\varphi \vee \psi, \psi)\}$ and $\mathsf{P}_{\Gamma,\xi,\Gamma'} = \{(\chi, \chi) \mid \chi \in \Lambda\}$.*

The relations for the remaining rules are defined analogously.

The trail relation $\mathsf{T}_{\Gamma,\xi,\Gamma'} \subseteq \Gamma \times \Gamma'$ *is defined as $\mathsf{A}_{\Gamma,\xi,\Gamma'} \cup \mathsf{P}_{\Gamma,\xi,\Gamma'}$. Finally, for nodes u, v in an* NW *proof π, such that $P(u, v)$, we define $\mathsf{T}_{u,v} = \mathsf{T}_{\mathsf{S}(u),\mathsf{f}(u),\mathsf{S}(v)}$*

Note that for any two nodes u, v with $P(u, v)$ and $(\varphi, \psi) \in \mathsf{T}_{u,v}$, we have either $(\varphi, \psi) \in \mathsf{A}_{u,v}$ and $\varphi \to_C \psi$, or else $(\varphi, \psi) \in \mathsf{P}_{u,v}$ and $\varphi = \psi$. The idea is that A connects the active formulas in the premise and conclusion, whereas P connects the side formulas.

Definition 4. *Let $\pi = (T, P, \mathsf{S}, \mathsf{R}, \mathsf{f})$ be an* NW *derivation. A* branch *of π is simply a (finite or infinite) branch of the underlying tree (T, P) of π. A* trail *on a branch $\alpha = (v_n)_{n<\kappa}$ is a sequence $\tau = (\varphi_n)_{n<\kappa}$ of formulas such that $(\varphi_i, \varphi_{i+1}) \in \mathsf{T}_{v_i,v_{i+1}}$, whenever $i + 1 < \kappa$. We obtain the* tightening $\widehat{\tau}$ *of such a τ by erasing all φ_{i+1} from τ for which $(\varphi_i, \varphi_{i+1})$ belongs to the passive trail relation $\mathsf{P}_{v_i,v_{i+1}}$. We call τ a* ν-trail *if its tightening $\widehat{\tau}$ is a ν-trace (and so, in particular, it is infinite).*

Definition 5. *An* NW proof *π is an* NW *derivation such that on every infinite branch of π there is a ν-trail. We write* $\mathsf{NW} \vdash \Gamma$ *if there is an* NW *proof of Γ, i.e., an* NW *proof, where Γ is the sequent at the root of the proof.*

Soundness and Completeness of NW for guarded formulas, (i.e., where in every subformula $\eta x.\psi$ all free occurrences of x in ψ are in the scope of a modality) follows from the results by Niwiński & Walukiewicz [19]. As pointed out in [2], it follows from [24] and [10] that the result in fact holds for arbitrary formulas. By Theorem 6.3 in [19], NW-proofs can be assumed to be regular, and this observation applies to unguarded formulas as well.

Theorem 1 (Soundness & Completeness). *Let Γ be a sequent, then $\bigvee \Gamma$ is valid iff* $\mathsf{NW} \vdash \Gamma$ *iff Γ has a regular* NW*-proof.*

3 Determinization of Automata with Binary Trees

3.1 Büchi automata

Let Σ be an alphabet and $\mathbb{B} = \langle B, \Delta, b_I, F\rangle$ a nondeterministic Büchi automaton over Σ. We want to present an equivalent deterministic Rabin automaton.

The *run tree* of $\mathbb{B}$ on a word $w = (w_i)_{i\in\omega}$ is a pair $\mathsf{R} = (R, l)$, where R is the full infinite binary tree and l labels every node s with $B_s \subseteq B$, such that $l(\epsilon) = \{b_I\}$ and for $|s| = i$: $l(s1) = \Delta(B_s, w_i) \cap F$ and $l(s0) = \Delta(B_s, w_i) \cap \overline{F}$, where we define $\Delta(B_s, y) = \bigcup_{b\in B_s} \Delta(b, y)$. It describes all possible runs of $\mathbb{B}$ on w, using the 1 s to keep track of visited states in F.

The *profile tree*, introduced in [9], is a pruned version of the run tree, where 1) at each level all but the (lexicographically) greatest occurrence of a state b are removed and 2) nodes labelled by the empty set are deleted. This results in a tree of bounded width, where every node has 0,1 or 2 children. It can be proved that $\mathbb{B}$ accepts a stream w iff the corresponding profile tree has a branch with infinitely many 1 s.

In [8] a determinization method was defined, where macrostates encode levels of the profile tree. In our approach macrostates encode a compressed version of the whole profile tree up to some level: Nodes u, v are identified (iteratively), if v is the unique child of u. This results in finite binary trees, where leaves are labelled by subsets of B. In every step of the transition function we add one level of the run tree and then prune and compress the tree to obtain a binary tree again. Whenever a 1 is compressed (in the sense of a node being identified with its right child) we know that a state in F has been visited and mark the node green. A run of the deterministic automaton is accepted if there is a node, which never gets removed and is marked green infinitely often. Figure 2 contains an example of this determinization construction.

Formally we define the deterministic Rabin automaton $\mathbb{B}^D = \langle B^D, \delta, b'_I, R\rangle$ as follows: An element S in the carrier B^D of $\mathbb{B}^D$ is called a *macrostate* and consists of

- a subset B_S of B,
- a map $f : B_S \to 2^*$, such that[1] $\mathrm{ran}(f)$ is a set of leaves of a binary tree and
- a colouring map $c : \mathrm{tree}(\mathrm{ran}(f)) \to \{\mathrm{green}, \mathrm{red}, \mathrm{white}\}$.

We define T^S to be the binary tree $\mathrm{tree}(\mathrm{ran}(f))$, that has $\mathrm{ran}(f)$ as its leaves and say that a binary string s is *in play* if $s \in T^S$. If it is clear from the context we occasionally abbreviate T^S by T. We will sometimes denote a macrostate by a set of pairs (b, s), usually written as b^s, where $b \in B_S$ and $s = f(b)$ and deal with the colouring c implicitly.

The initial macrostate b'_I consists of the singleton $\{b_I^\epsilon\}$, where $c(\epsilon) = \mathrm{white}$. To define the transition function δ let S be in B^D and $y \in \Sigma$. We define $\delta(S, y) = S'$, where starting from the empty set we build up S' in the following steps:

1. <u>Move:</u> For every $a^s \in S$ and $b \in \Delta(a, y)$, add b^s to S'.

[1] Here $\mathrm{ran}(f)$ denotes the co-domain of f.

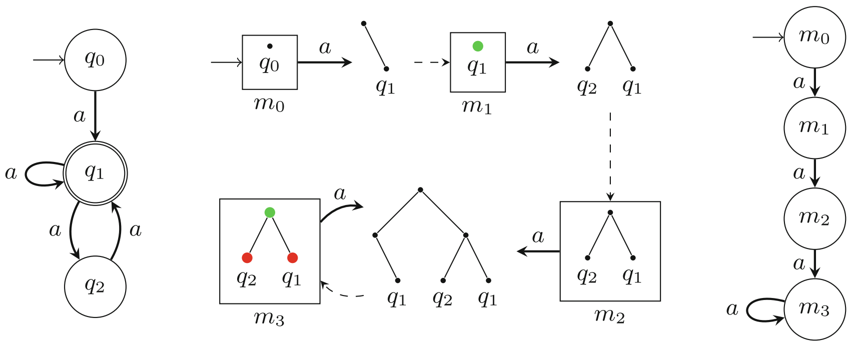


Fig. 2. A nondeterministic Büchi automaton $\mathbb{B}$ on the left and its determinization $\mathbb{B}^D$ on the right. The diagram in the middle shows the internal structure of the macrostates m_0, m_1, m_2 and m_3 of $\mathbb{B}^D$. Binary trees are represented in the obvious way (i.e., the root is the string ϵ, and for every node the left child appends 0 and the right child appends 1). The transitions of $\mathbb{B}^D$ are split in two parts: In the first part one level of the run tree is added, corresponding to the steps 1 and 2 in the definition of the transition function. In the second part (the dashed arrows) the tree is pruned and compressed, corresponding to the steps 3 and 4. The acceptance condition of $\mathbb{B}^D$ is such that the word a^ω is accepted by $\mathbb{B}^D$ because the string ϵ is always in play, marked green infinitely often and never red.

2. Append: For every $a^s \in S'$, where $a \notin F$, change a^s to a^{s0}. For every $a^s \in S'$, where $a \in F$, change a^s to a^{s1}.
3. Resolve: If a^s and a^t are in S', where $s < t$, delete a^s.
4. Compress/Colour: Let $c(t) =$ white for every $t \in T^{S'}$. Now we compress and colour T in the following way, until there exists no *witness* $t \in T$, such that (a) or (b) is applicable:[2]
   (a) For any $t \in T$, such that $t0 \in T$ and $t1 \notin T$, change every $a^s \in S'$, where $t0 \sqsubseteq s$, to $a^{s[t0\backslash t]}$. For any $s \in T$, where $t \sqsubset s$, let $c(s) =$ red.
   (b) For any $t \in T$, such that $t0 \notin T$ and $t1 \in T$, change every $a^s \in S'$, where $t1 \sqsubseteq s$, to $a^{s[t1\backslash t]}$. For any $s \in T$ such that $t = s0 \cdots 0$, let $c(s) =$ green, if $c(s) \neq$ red. In particular let $c(t) =$ green if $c(t) \neq$ red. For any $s \in T$, where $t \sqsubset s$, let $c(s) =$ red.

We define B^D as the set of macrostates that can be reached from b'_I in this way.

A run of $\mathbb{B}^D$ is accepting if there is a binary string s, which is in play cofinitely often such that $c(s)$ is green infinitely often and red only finitely often.

Theorem 2. *$\mathbb{B}$ accepts a word w iff $\mathbb{B}^D$ accepts w.*

Remark 1. For a Büchi automaton of n states, our construction yields a deterministic automaton $\mathbb{B}^D$ with $n^{\mathcal{O}(n)}$ states and a Rabin condition of $\mathcal{O}(2^n)$ pairs,

[2] As shown in Proposition 1 of [5] this procedure does not depend on the order in which witnesses are chosen, and thus produces a unique binary tree.

see Lemma 5 of [5]. With some adaptations we could also match the optimal Rabin condition, which is known to be linear-size [20].

This can be achieved by adding an labelling function as follows: Let $L = \{1, \ldots, 2n-1\}$ be the set of potential labels. Macrostates are defined as before, where an additional injective function $l : T^S \to L$ is added. For the initial state we let $l(\epsilon) = 1$. The steps 1–4 in the transition function remain the same, where we add a final step 5 in which we define the new labeling function l': We let $l'(s) = l(s)$ for all s that already occurred in T^S and for all $s \in T^{S'} \setminus T^S$ we let $c(s) = \text{red}$ and choose new, distinct labels in L, i.e. ones which do not occur in $\text{ran}(l)$. The binary tree $T^{S'}$ has at most n leaves, hence it has at most $2n-1$ many nodes and this is always possible.

The new acceptance condition has the following form: A run of the automaton is accepting if there is a label $k \in L$, such that $c(l^{-1}(k))$ is green infinitely often and red only finitely often. Here $c(l^{-1}(k))$ is defined to be red if $k \notin \text{ran}(l)$. This is a Rabin condition with $\mathcal{O}(n)$ pairs. Notably we still have $n^{\mathcal{O}(n)}$ macrostates, thus the determination method is optimal.

3.2 Parity Automata

We now extend the approach to parity automata. Let Σ be an alphabet and $\mathbb{A} = \langle A, \Delta_A, a_I, \Omega\rangle$ be a nondeterministic parity automaton.

In order to present the intuitive idea behind the construction we first transform $\mathbb{A}$ into an equivalent nondeterministic Büchi automaton $\mathbb{B}$. Let m be the maximal even priority of Ω. For even $k = 0, 2, ...m$ we define $\mathbb{A}_0, \mathbb{A}_2, ..., \mathbb{A}_n$ as copies of $\mathbb{A}$ without the states of priority smaller than k, i.e. $\mathbb{A}_k = \langle A_k, \Delta_k, F_k\rangle$ with $A_k = \{a_k \mid a \in A \wedge \Omega(a) \geq k\}$, $\Delta_k = \Delta_A|_{A_k}$ and $F_k = \{a_k \in A_k \mid \Omega(a) = k\}$. Now we define the nondeterministic Büchi automaton $\mathbb{B} = \langle B, \Delta_B, b_I, F\rangle$:[3]

$$B = A \cup \bigcup_{\substack{k=0\\k \text{ even}}}^{m} A_k, \qquad b_I = a_I, \qquad F = \bigcup_{\substack{k=0\\k \text{ even}}}^{m} F_k,$$

$$\Delta_B = \Delta_A \cup \bigcup_{\substack{k=0\\k \text{ even}}}^{m} \Delta_k \cup \{(a, y, b_k) \in A \times \Sigma \times A_k \mid b \in \Delta_A(a, y), k = 0, 2, ..., m\}.$$

Although $\mathbb{A}_k$ is not an automaton, as it does not have an initial state, we can define the Büchi automaton $\mathbb{A} \cup \mathbb{A}_k = \langle A \cup A_k, \Delta_B|_{A \cup A_k}, a_I, F_k\rangle$ for $k = 0, ..., m$.

The intuition behind the determinization of the parity automaton $\mathbb{A}$ is the following: We apply the binary tree construction to every automaton $\mathbb{A} \cup \mathbb{A}_k$ for $k = 0, 2, ..., m$, which is possible as there are no paths from A_k to A_j if $k \neq j$ and none of the accepting states of $\mathbb{B}$ are in the set A. The annotation of a state $a \in \mathbb{A}$ will then be the tuple $(s_0, s_2, ..., s_m)$, where s_k is the annotation at the state $a_k \in \mathbb{A} \cup \mathbb{A}_k$. Note that the automaton $\mathbb{A}^D$ will be different from the automaton obtained from the binary tree construction on the whole $\mathbb{B}$.

[3] For easier notation we represent the transition function $B \times \Sigma \to \mathcal{P}(B)$ by its corresponding relation (i.e., subset of $B \times \Sigma \times B$).

To make that formal we need some definitions. A *treetop* L is a set of leaves of a binary tree, where potentially the minimal leaf is missing, i.e. L is a finite set of binary strings such that for all $s \neq t \in L$ it holds $s \not\sqsubseteq t$ and $\text{tree}(L) = \{s \in 2^* \mid \exists t \in L : s \sqsubseteq t\} \cup \{s0 \mid s = 0 \cdots 0 \text{ and } s1 \in L\}$ is a binary tree.

For even m let $\text{TSeq}(m) = \{(s_0, s_2, ..., s_m) \mid s_0, s_2, ..., s_m \in 2^*\}$ be the set of sequences of length $\frac{m}{2} + 1$, where $s_0, ..., s_m$ are binary strings. Let π_k be the projection function, which maps $\sigma = (s_0, ..., s_m)$ to s_k for $k = 0, 2, ..., m$. We define a partial order $<$ on $\text{TSeq}(m)$: Let $(s_0, ..., s_m) < (t_0, ..., t_m)$ if there exists $l \in \{0, ..., m\}$ such that $s_l < t_l$ and $s_j = t_j$ for $j = 0, ..., l-2$.

We now define the deterministic Rabin automaton $\mathbb{A}^D = \langle A^D, \delta_A, a'_I, R_A \rangle$. Let m be the maximal even priority of Ω in $\mathbb{A}$. An element S in the carrier A^D of $\mathbb{A}^D$ consists of a tuple $(A_S, f, c_0, ..., c_m)$, where

- A_S is a subset of A,
- $f : A_S \to \text{TSeq}(m)$, such that $\text{ran}(\pi_k \circ f)$ is a treetop for $k = 0, ..., m$ and
- c_k is a colouring map from $\text{tree}(\text{ran}(\pi_k \circ f)) \to \{\text{green}, \text{red}, \text{white}\}$ for $k = 0, 2, ..., m$.

We define T_k^S to be the binary tree $\text{tree}(\text{ran}(\pi_k \circ f))$ for $k = 0, 2, ..., m$ and say a binary string s is *in play at position* k if $s \in T_k^S$. If the context is clear we will abbreviate T_k^S with T_k. Again we sometimes denote a macrostate by a set of pairs (a, σ), usually written as a^σ, where $a \in A_S$ and $\sigma = f(a)$ and deal with the colourings c_k implicitly.

The initial macrostate a'_I consists of the singleton $\{a_I^{(\epsilon, ..., \epsilon)}\}$. To define the transition function δ_A let S be in A^D and $y \in \Sigma$. We define $\delta_A(S, y) = S'$, where S' is constructed in the following steps:

1. (a) <u>Move:</u> For every $a^\sigma \in S$ and $b \in \Delta_A(a, y)$, add b^σ to S'.
   (b) <u>Reduce:</u> For every $a^\sigma \in S'$, change a^σ to $a^{\sigma'}$, where σ' is obtained from $\sigma = (s_0, ..., s_m)$ by replacing every s_j with $j > \Omega(a)$ by $\text{minL}(T_j)$.
2. <u>Append:</u> For every $a^\sigma \in S'$ and $\sigma = (s_0, ..., s_m)$, change a^σ to $a^{\sigma'}$, where $\sigma' = (s_0 0, ..., s_{k-2} 0, s_k 1, s_{k+2} 0, ..., s_m 0)$ if $\Omega(a) = k$ is even, and $\sigma' = (s_0 0, ..., s_m 0)$ if $\Omega(a) = k$ is odd.
3. <u>Resolve:</u> If a^σ and a^τ are in S' and $\sigma < \tau$, delete a^σ.
4. <u>Compress/Colour:</u> Do for every $k = 0, 2, ..., m$: Let $c_k(t) = \text{white}$ for any $t \in T_k$. Now we compress and colour T_k inductively in the following way, until there exists no *witness* $t \in T_k$, such that (a) or (b) is applicable:
   (a) For any $t \in T_k$, such that $t0 \in T_k$ and $t1 \notin T_k$, change every $a^\sigma \in S'$, where $\sigma = (s_0, ..., s_m)$, and $t0 \sqsubseteq s_k$, to $a^{\sigma'}$, where $\sigma' = (s_0, ..., s_k[t0 \backslash t], ..., s_m)$. For any $s \in T_k$, where $t \sqsubset s$, let $c_k(s) = \text{red}$.
   (b) For any $t \in T_k$, such that $t0 \notin T_k$, $t1 \in T_k$ and $t \neq 0 \cdots 0$, change every $a^\sigma \in S'$, where $\sigma = (s_0, ... s_m)$, and $t1 \sqsubseteq s_k$, to $a^{\sigma'}$, where $\sigma' = (s_0, ..., s_k[t1 \backslash t], ..., s_m)$. For any $s \in T_k$ such that $t = s0 \cdots 0$, let $c_k(s) = \text{green}$, if $c_k(s) \neq \text{red}$. In particular let $c_k(t) = \text{green}$ if $c_k(t) \neq \text{red}$. For any $s \in T_k$, where $t \sqsubset s$, let $c_k(s) = \text{red}$.

A run of $\mathbb{A}^D$ is accepting if there is $k \in \{0, 2, ..., m\}$ and a binary string s, which is in play at position k cofinitely often such that $c_k(s)$ is green infinitely often and red only finitely often.

Theorem 3. *Let $\mathbb{A}$ be a parity automaton and $\mathbb{A}^D$ the deterministic Rabin automaton defined from $\mathbb{A}$. Then $L(\mathbb{A}) = L(\mathbb{A}^D)$.*

Remark 2. For a parity automaton $\mathbb{A}$ of size n with highest even priority m, our construction produces a deterministic Rabin automaton with $n^{\mathcal{O}(m \cdot n)}$ macrostates and $\mathcal{O}(m \cdot 2^n)$ Rabin pairs, see Lemma 6 of [5].

4 BT Proofs

4.1 Proof Systems

We present two non-wellfounded proof systems for the modal μ-calculus, namely BT and BT^∞. The idea is that annotated sequents in the BT system correspond to macrostates of $\mathbb{A}^D$, where $\mathbb{A}$ is a nondeterministic parity automaton checking the trace condition in an NW proof. The rules of BT resemble the transition function of $\mathbb{A}^D$.

Let Φ be a set of formulas, the sequent we want to prove, and let $m = \max_\Omega(\Phi)$ be the maximal even priority of Ω. *Annotated sequents* are sets of pairs (φ, σ), usually written as φ^σ, where $\varphi \in \mathsf{Clos}(\Phi)$ and $\sigma \in \mathrm{TSeq}(m)$. For an annotated sequent Γ we let Γ^N be the set of annotations occurring in Γ, i.e. $\Gamma^N = \{\sigma \in \mathrm{TSeq}(m) \mid \exists \varphi \text{ s.t. } \varphi^\sigma \in \Gamma\}$. We let Γ_k^N be the set of binary strings occurring at the k-th position of the annotations in Γ, i.e., $\Gamma_k^N = \pi_k[\Gamma^N]$. We say that a string s *occurs in* Γ_k^N if there exists $t \in \Gamma_k^N$ such that $s \sqsubseteq t$.

For $\sigma = (s_0, ..., s_m) \in \mathrm{TSeq}(m)$ we define $\sigma \cdot 1_k = (s_0, ..., s_k 1, ..., s_m)$ and $\sigma \cdot 0_k = (s_0, ..., s_k 0, ..., s_m)$. For an annotated sequent Γ we let $\Gamma^{\cdot 0_k}$ denote the annotated sequent $\{\varphi^{\sigma \cdot 0_k} \mid \varphi^\sigma \in \Gamma\}$.

Let Γ be an annotated sequent and $\varphi^\sigma \in \Gamma$. We define $\sigma \restriction k^\Gamma$ to be the tuple of binary strings obtained from $\sigma = (s_0, ..., s_m)$ by replacing every s_j with $j > k$ by $\mathrm{minL}(\mathrm{tree}(\Gamma_j^N))$. If the context Γ is clear we write $\sigma \restriction k$ instead of $\sigma \restriction k^\Gamma$.

The rules $\mathsf{Compress}_k^{s0}$ and $\mathsf{Compress}_k^{s1}$ are schemata for $k = 0, 2, ..., m$ and $s \in 2^*$. In these rules the notation $\varphi_i^{(..., st_i, ...)}$ is to be read such that st_i is the binary string in the k-th position of the annotation. We will write $\mathsf{Compress}$ for any of those rules and write $\mathsf{Compress}_k^s$ for either $\mathsf{Compress}_k^{s0}$ or $\mathsf{Compress}_k^{s1}$.

Note that, if one ignores the annotations, the rules $\mathsf{Ax1}$, $\mathsf{Ax2}$, $\mathsf{R}_\vee$, $\mathsf{R}_\wedge$, R_μ, R_ν and $\mathsf{R}_\Box$ in Fig. 3 are the same as the rules of NW. As mentioned above annotated sequents in the BT system correspond to macrostates of $\mathbb{A}^D$, where $\mathbb{A}$ is a nondeterministic parity automaton checking the trace condition in an NW proof. The rules of BT correspond to the transition function δ_A of $\mathbb{A}^D$, where the transformations of δ_A are distributed over multiple rules: Step 1(a) of δ_A is carried out in every rule and step 1(b) and step 2 correspond to the modification of the annotations in the rules R_μ and R_ν. Notably, we do not add zeros to the annotations if the zeros would get deleted anyway in step 4 of the transition function. The rules $\mathsf{Resolve}$ and $\mathsf{Compress}$ are additional and correspond to steps 3 and 4 of δ_A.

$$\mathsf{Ax1}\colon \frac{}{p^\sigma, \bar{p}^\tau, \Gamma} \qquad \mathsf{Ax2}\colon \frac{}{\top^\sigma, \Gamma} \qquad \mathsf{R}_\vee\colon \frac{\varphi^\sigma, \psi^\sigma, \Gamma}{(\varphi \vee \psi)^\sigma, \Gamma} \qquad \mathsf{R}_\wedge\colon \frac{\varphi^\sigma, \Gamma \quad \psi^\sigma, \Gamma}{(\varphi \wedge \psi)^\sigma, \Gamma}$$

$$\mathsf{R}_\mu\colon \frac{\varphi[x \backslash \mu x.\varphi]^{\sigma \restriction \Omega(\mu x.\varphi)}, \Gamma}{\mu x.\varphi^\sigma, \Gamma} \qquad \mathsf{R}_\nu\colon \frac{\varphi[x \backslash \nu x.\varphi]^{\sigma \restriction k \cdot 1_k}, \Gamma^{\cdot 0_k}}{\nu x.\varphi^\sigma, \Gamma} \quad \text{where } k = \Omega(\nu x.\varphi)$$

$$\mathsf{R}_\Box\colon \frac{\varphi^\sigma, \Gamma}{\Box\varphi^\sigma, \Diamond\Gamma, \Delta} \qquad \mathsf{Resolve}\colon \frac{\varphi^\sigma, \Gamma}{\varphi^\sigma, \varphi^\tau, \Gamma} \quad \text{where } \sigma > \tau \qquad \mathsf{D}^{\mathsf{x}}\colon \begin{array}{c} [\Gamma]^{\mathsf{x}} \\ \vdots \\ \dfrac{\Gamma}{\Gamma} \end{array}$$

$$\mathsf{Compress}_k^{s0}\colon \frac{\varphi_1^{(\dots,st_1,\dots)}, \dots, \varphi_n^{(\dots,st_n,\dots)}, \Gamma}{\varphi_1^{(\dots,s0t_1,\dots)}, \dots, \varphi_n^{(\dots,s0t_n,\dots)}, \Gamma} \quad \text{where } s \text{ does not occur in } \Gamma_k^N$$

$$\mathsf{Compress}_k^{s1}\colon \frac{\varphi_1^{(\dots,st_1,\dots)}, \dots, \varphi_n^{(\dots,st_n,\dots)}, \Gamma}{\varphi_1^{(\dots,s1t_1,\dots)}, \dots, \varphi_n^{(\dots,s1t_n,\dots)}, \Gamma} \quad \text{where } s \text{ does not occur in } \Gamma_k^N \text{ and } s \neq 0 \cdots 0$$

Fig. 3. Rules of BT

Definition 6. *A* BT derivation π *is a derivation defined from the rules in Fig. 3, such that the rules are applied with the following priority: first* Resolve*, then* Compress*, and then all other rules.*

Just as annotated sequents correspond to macrostates of the deterministic automaton $\mathbb{A}^D$, the soundness condition of BT^∞ and BT correspond to the acceptance condition of $\mathbb{A}^D$: We say that a pair (k, s) is preserved at a node, if s is in play at position k at the corresponding macrostate and not marked red; and progresses if it is marked green.

Definition 7. *Let π be a* BT *derivation of Φ, $m = \max_\Omega(\Phi)$ and S be a set of nodes in π. Let $k \in \{0, 2, \dots, m\}$ and $s \in 2^*$. We say that the pair (k, s)*

- *is* preserved *on S if*
  - *s occurs in $\mathsf{S}(v)_k^N$ for every v in S and*
  - *if $\mathsf{R}(v) = \mathsf{Compress}_k^t$ for a node v in S, then $t \not\sqsubseteq s$,*
- progresses *(infinitely often) on S if there is $s' = s0 \cdots 0$ such that $\mathsf{R}(v) = \mathsf{Compress}_k^{s'1}$ for some v in S (for infinitely many $v \in S$).*

Definition 8. *Let π be a* BT *derivation. An infinite branch $\alpha = (u_i)_{i \in \omega}$ in π is* successful *if there are N and (k, s) such that (k, s) is preserved and progresses infinitely often on $\{u_i \mid i \geq N\}$. A* BT^∞ proof *is a* BT *derivation without occurrences of* D^{x} *and such that all infinite branches are successful. A* BT proof *is a finite* BT *derivation such that for each* strongly connected subgraph *S in $\mathcal{T}_\pi^C$ there exists (k, s) that is preserved and progresses on S.*

We write $\mathsf{BT} \vdash \Gamma$ *(*$\mathsf{BT}^\infty \vdash \Gamma$*) if there is a* BT *(*$\mathsf{BT}^\infty$*) proof of Γ, i.e., a proof, where Γ is the sequent at the root of the proof.*

Remark 3. In the proof system JS introduced by Jungteerapanich and Stirling [13,23] annotated sequents are of the form $\theta \vdash \varphi_1^{a_1}, ..., \varphi_n^{a_n}$, where $a_1, ..., a_n$ are sequences of names and the so-called *control* θ is a linear order on all names occurring in the sequent. In contrast to JS our sequents consist of formulas with annotations and nothing else, that is, no control. On the other hand the soundness condition of BT is less local: It speaks about strongly connected subgraphs, whereas in JS only paths between leafs and its companions have to be checked. We see that the control in JS gives information on the structure of the cyclic proof tree. Interestingly, we could also add a control to our sequents and obtain a soundness condition that talks about paths, if desired. Similarly, in [1] a control was added to a cyclic system for the first-order μ-calculus introduced by [22] to obtain a path-based system.

4.2 Soundness and Completeness

The intuitive idea behind the BT^∞ proof system is the following: Starting with an NW proof, we can define a nondeterministic parity automaton $\mathbb{A}$, that checks if an infinite branch carries a ν-trail. Using the determinization method from Sect. 3 we simulate macrostates of $\mathbb{A}^D$ by annotated formulas in the proof system. Thus an infinite branch in BT^∞ resembles an infinite run of $\mathbb{A}^D$. This will be formalised in the Soundness and Completeness proofs.

Tracking Automaton. Let Φ be a sequent of formulas, $\eta x_1.\psi_1, ..., \eta x_n.\psi_n$ the fixpoint formulas in $\mathsf{Fix}(\Phi)$ and Ω the parity function on $\mathsf{Fix}(\Phi)$.

We define a nondeterministic parity automaton that checks if there is a ν-trail on an infinite branch of some NW proof of Φ. The alphabet Σ consists of all triples (Γ, ξ, Γ'), where $\Gamma \subseteq \mathsf{Clos}(\Phi)$ is the conclusion and $\Gamma' \subseteq \mathsf{Clos}(\Phi)$ is the premise of a rule in Fig. 1 with principal formula ξ. We define the following nondeterministic parity automaton $\mathbb{A} = (A, \Delta, a_I, \Omega_A)$:

- $A = a_I \cup \mathsf{Clos}(\Phi) \cup \{\eta x.\psi^* \mid \eta x.\psi \in \mathsf{Clos}(\Phi)\}$,
- For each $\gamma \in A$ and $(\Gamma, \xi, \Gamma') \in \Sigma$:
  1. if $\gamma = a_I$, then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \Phi$,
  2. if $\gamma = \xi = \eta x.\psi$ then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\eta x.\psi^*\}$,
  3. if $\gamma = \eta x.\psi^*$, then $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\gamma' \mid (\psi[x \backslash \eta x.\psi], \gamma') \in \mathsf{T}_{\Gamma,\xi,\Gamma'}\}$ and
  4. else $\Delta(\gamma, (\Gamma, \xi, \Gamma')) = \{\gamma' \mid (\gamma, \gamma') \in \mathsf{T}_{\Gamma,\xi,\Gamma'}\}$.
- For all states $\eta x.\psi^*$ let $\Omega_A(\eta x.\psi^*) = \Omega(\eta x.\psi)$. For all other states a let $\Omega_A(a) = \max_\Omega(\Phi)$ if $\max_\Omega(\Phi)$ is odd and $\Omega_A(a) = \max_\Omega(\Phi) + 1$ else.

Let $\alpha = (v_n)_{n\in\omega}$ be an infinite branch in an NW-proof π. We define $w(\alpha) \in \Sigma^\omega$ to be the infinite word $(\mathsf{S}(v_0), \mathsf{f}(v_0), \mathsf{S}(v_0))(\mathsf{S}(v_0), \mathsf{f}(v_0), \mathsf{S}(v_1))(\mathsf{S}(v_1), \mathsf{f}(v_1), \mathsf{S}(v_2))....$

Lemma 1. *Let α be an infinite branch in an* NW *proof. Then α carries a ν-trail iff $w(\alpha) \in \mathcal{L}(\mathbb{A})$.*

Combining Lemma 1 and Theorem 3 from Sect. 3 we get

Lemma 2. *Let π be an* NW *derivation. Then π is an* NW *proof iff for every infinite branch α in π it holds $w(\alpha) \in \mathcal{L}(\mathbb{A}^D)$.*

Lemma 3. *Let Γ be a sequent. Then* $\mathsf{NW} \vdash \Gamma$ *iff* $\mathsf{BT} \vdash \Gamma^\epsilon$.

Proof (Sketch). Let π be an NW proof of a sequent Γ. Inductively we translate every node v in π to a node v' plus some additional nodes, such that v' is labeled by the same sequent as v plus annotations. This can be achieved by replacing every rule in NW by its corresponding rule in BT and adding the rules $\mathsf{Resolve}$ and $\mathsf{Compress}$ whenever applicable. This yields a BT derivation ρ. It remains to show that every infinite branch $\alpha = (v_i)_{i\in\omega}$ in ρ is successful. Let $\hat{\alpha}$ be the corresponding infinite branch in π. Due to Lemma 2 it holds that $\hat{\alpha} \in \mathcal{L}(\mathbb{A}^D)$. Thus there is (k, s) such that s is in play at position k cofinitely often and $c_k(s)$ is green infinitely often and red only finitely often. As the annotations in α resemble the annotations in the run of $\mathbb{A}^D$ on $\hat{\alpha}$ it follows that there is some $N \in \omega$ such that (k, s) is preserved and progresses infinitely often on $\{v_i \mid i \geq N\}$.

Conversely let ρ be a BT proof of Γ^ϵ. We let π be the NW derivation defined from ρ by omitting the rules $\mathsf{Resolve}$ and $\mathsf{Compress}$ and reducing the other rules to the corresponding NW rules. We have to show that every infinite branch α in π is successful. Let $\alpha' = (v_i)_{i\in\omega}$ be the corresponding infinite branch in ρ. Because ρ is a BT proof there is N,(k, s) such that (k, s) is preserved and progresses infinitely often on $\{v_i \mid i \geq N\}$. Again the annotations in α' resemble the annotations in the run of $\mathbb{A}^D$ on α, thus (k, s) witnesses the acceptance of the run of $\mathcal{L}(\mathbb{A}^D)$ on α and Lemma 2 concludes the proof.

Theorem 4 (Soundness and Completeness). *Let Γ be a sequent. Then there is a* BT^∞*-proof of Γ^ϵ iff $\bigvee \Gamma$ is valid.*

Proof. This follows from Lemma 3 and Theorem 1.

4.3 Cyclic BT Proofs

As NW proofs can be assumed to be regular and annotations are added deterministically we can also assume BT^∞ proofs to be regular. A standard argument then transforms regular BT^∞ proofs into BT proofs and vice versa.

Lemma 4. *An annotated sequent is provable in* BT *iff it is provable in* BT^∞.

Theorem 5 (Soundness and Completeness). *Let Γ be a sequent. Then there is a* BT*-proof of Γ^ϵ iff $\bigvee \Gamma$ is valid..*

Remark 4. The number of distinct subtrees in a regular BT^∞ proof can be bounded by the number of distinct annotated sequents. This follows because the same statement holds for NW proofs [19] and because in the proof of Lemma 3 annotations and extra rules are added deterministically to sequents in NW proofs.

Let Φ be a sequent, $n = |\mathsf{Clos}(\Phi)|$ and $m = \max_\Omega(\Phi)$. There are at most $n^{\mathcal{O}(m\cdot n)}$ many distinct annotated sequents occurring in a proof of Φ, because annotated sequents resemble macrostates in $\mathbb{A}^D$ and as seen in Remark 2 there are at most $n^{\mathcal{O}(m \cdot n)}$ distinct macrostates in $\mathbb{A}^D$.

Combining these two observations with the proof of Lemma 4 yields that the height of a BT proof of a sequent Φ can be bound by $n^{\mathcal{O}(m \cdot n)}$. This is the same complexity as in JS [13].

Remark 5. Given a BT derivation π, we can check if π is a BT proof in coNP. We can give the following algorithm in NP, that checks if π is not a BT proof: Choose non-deterministically a strongly connected subgraph S and check if there exists (k, s) that is preserved and progresses on S, the latter can be done in polynomial time. The complexity of proof checking can be compared to linear time in JS and PSPACE in NW. Note that, if we add a control to the BT proof system, the soundness condition boils down to checking paths between leafs and its companions. In that case proof checking could also be done in linear time.

5 Conclusions and Future Work

We hope that this paper contributes to the theory of non-wellfounded and cyclic proof systems by discussing applications of automata theory in the field. We have argued for the relevance of the notion of determinizing stream automata in the design of proof systems for the modal μ-calculus. More concretely, we have introduced a determinization construction based on binary trees and used this to obtain a new derivation system BT which is cyclic, cutfree, and sound and complete for the collection of valid $\mathcal{L}_\mu$-formulas. In the remainder of this concluding section we point out some directions for future research.

First of all, our approach is not restricted to the modal μ-calculus, but will apply to non-wellfounded and cyclic derivation systems for many other logics as well. For instance, in the proof systems LKID$^\omega$ [3] for first-order logic with inductive definitions, cyclic arithmetic CA [21] and similar systems the trace condition is of the form that on every infinite branch there is a term/variable which progresses infinitely often. This condition can be checked by a nondeterministic Büchi automaton and thus our method would yield an annotated proof system, where the annotations are binary strings, which label the terms/variables.

Second, in Remark 3 we discussed some relative advantages and disadvantages of the systems JS and BT. It would be interesting to either design a system that combines the advantages of both systems (i.e. sequents consisting of annotated formulas only as in BT, and a local condition for proof checking as in JS), or prove that such a system cannot exist.

Finally, it would be interesting (and in fact, it was one of the original aims of our work), to connect annotation-based sequent calculi such as JS and BT to Kozen's Hilbert-style proof system and to see whether a more structured automata-theoretic approach would yield an alternative proof of Walukiewicz' completeness result. Note that this was also the goal of Afshari & Leigh [2]; unfortunately, it was recently shown by the second author [14] that the system Clo, a key system in Afshari & Leigh's approach linking JS to Kozen's axiomatization, is in fact incomplete.

References

1. Afshari, B., Enqvist, S., Leigh, G.E.: Cyclic proofs for the first-order μ-calculus. Logic J. IGPL (2022). https://doi.org/10.1093/jigpal/jzac053
2. Afshari, B., Leigh, G.E.: Cut-free completeness for modal μ-calculus. In: Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavík, Iceland. IEEE Press (2017)
3. Brotherston, J.: Sequent calculus proof systems for inductive definitions. Ph.D. thesis (2006). https://era.ed.ac.uk/handle/1842/1458
4. Calude, C., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, (STOC 2017), pp. 252–263 (2017)
5. Dekker, M., Kloibhofer, J., Marti, J., Venema, Y.: Proof systems for the modal μ-calculus obtained by determinizing automata (2023). https://doi.org/10.48550/arXiv.2307.06897
6. Doumane, A.: Constructive completeness for the linear-time μ-calculus. In: 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pp. 1–12 (2017). https://doi.org/10.1109/LICS.2017.8005075
7. Emerson, E., Jutla, C.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (1999)
8. Fogarty, S., Kupferman, O., Vardi, M.Y., Wilke, T.: Profile trees for Büchi word automata, with application to determinization. Inf. Comput. **245**, 136–151 (2015)
9. Fogarty, S., Kupferman, O., Wilke, T., Vardi, M.: Unifying Büchi complementation constructions. Log. Methods Comput. Sci. **9**(1) (2013). https://doi.org/10.2168%2Flmcs-9%281%3A13%292013
10. Friedmann, O., Lange, M.: Deciding the unguarded modal μ-calculus. J. Appl. Non-Class. Logics **23**(4), 353–371 (2013). https://doi.org/10.1080/11663081.2013.861181
11. Janin, D., Walukiewicz, I.: Automata for the modal μ-calculus and related results. In: Wiedermann, J., Hájek, P. (eds.) MFCS 1995. LNCS, vol. 969, pp. 552–562. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60246-1_160
12. Janin, D., Walukiewicz, I.: On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In: Montanari, U., Sassone, V. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 263–277. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61604-7_60
13. Jungteerapanich, N.: Tableau systems for the modal μ-calculus. Ph.D. thesis, School of Informatics; The University of Edinburgh (2010). http://hdl.handle.net/1842/4208
14. Kloibhofer, J.: A note on the incompleteness of Afshari & Leigh's system Clo (2023). https://doi.org/10.48550/arXiv.2307.06846
15. Kozen, D.: Results on the propositional μ-calculus. Theoret. Comput. Sci. **27**, 333–354 (1983)
16. Löding, C., Pirogov, A.: Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp. Schloss Dagstuhl - Leibniz-Zentrum für Informatik GmbH, Wadern/Saarbruecken, Germany (2019). http://drops.dagstuhl.de/opus/volltexte/2019/10696/
17. Leigh, G.E., Wehr, D.: From GTC to reset: generating reset proof systems from cyclic proof systems. Technical report (2023). https://doi.org/10.48550/arXiv.2301.07544. http://arxiv.org/abs/2301.07544

18. Marti, J., Venema, Y.: A focus system for the alternation-free μ-calculus. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 371–388. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_22
19. Niwinski, D., Walukiewicz, I.: Games for the μ-Calculus. Theor. Comput. Sci. **163**(1&2), 99–116 (1996). https://doi.org/10.1016/0304-3975(95)00136-0
20. Safra, S.: On the complexity of ω-automata. In: Proceedings of the 29th Symposium on the Foundations of Computer Science, pp. 319–327. IEEE Computer Society Press (1988)
21. Simpson, A.: Cyclic arithmetic is equivalent to peano arithmetic. In: Esparza, J., Murawski, A.S. (eds.) FoSSaCS 2017. LNCS, vol. 10203, pp. 283–300. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54458-7_17
22. Sprenger, C., Dam, M.: On the structure of inductive reasoning: circular and tree-shaped proofs in the μcalculus. In: Gordon, A.D. (ed.) FoSSaCS 2003. LNCS, vol. 2620, pp. 425–440. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36576-1_27
23. Stirling, C.: A tableau proof system with names for modal μ-calculus. In: Voronkov, A., Korovina, M. (eds.) HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday. EPiC Series in Computing, vol. 42, pp. 306–318. EasyChair (2014). https://doi.org/10.29007/lwqm
24. Studer, T.: On the proof theory of the modal μ-calculus. Studia Logica **89**(3), 343–363 (2008). https://doi.org/10.1007/s11225-008-9133-6
25. Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus. Inf. Comput. **157**, 142–182 (2000)
26. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Belgian Math. Soc. **8**, 359–391 (2001)

Modal Logics

Extensions of K5: Proof Theory and Uniform Lyndon Interpolation

Iris van der Giessen[1](✉), Raheleh Jalali[2,3], and Roman Kuznets[4]

[1] University of Birmingham, Birmingham, UK
i.vandergiessen@bham.ac.uk
[2] Utrecht University, Utrecht, Netherlands
[3] Czech Academy of Sciences, Prague, Czechia
[4] TU Wien, Vienna, Austria
roman@logic.at

Abstract. We introduce a Gentzen-style framework, called *layered sequent calculi*, for modal logic K5 and its extensions KD5, K45, KD45, KB5, and S5 with the goal to investigate the uniform Lyndon interpolation property (ULIP), which implies both the uniform interpolation property and the Lyndon interpolation property. We obtain complexity-optimal decision procedures for all logics and present a constructive proof of the ULIP for K5, which to the best of our knowledge, is the first such syntactic proof. To prove that the interpolant is correct, we use model-theoretic methods, especially bisimulation modulo literals.

1 Introduction

The uniform interpolation property (UIP) is an important property of a logic. It strengthens the Craig interpolation property (CIP) by making interpolants depend on only one formula of an implication, either the premise or conclusion. A lot of work has gone into proving the UIP, and it is shown to be useful in various areas of computer science, including knowledge representation [17] and description logics [25]. Early results on the UIP in modal logic include positive results proved semantically for logics GL and K (independently in [9,32,35]) and negative results for logics S4 [10] and K4 [5]. A proof-theoretic method to prove the UIP was first proposed in [30] for intuitionistic propositional logic and later adapted to modal logics, such as K and T in [5]. A general proof-theoretic method of proving the UIP for many classical and intuitionistic (non-)normal modal logics and substructural (modal) logics based on the form of their sequent-calculi rules was developed in the series of papers [2,3,16].

I. van der Giessen—Supported by a UKRI Future Leaders Fellowship, 'Structure vs Invariants in Proofs', project reference MR/S035540/1.
R. Jalali—Acknowledges the support of the Netherlands Organization for Scientific Research under grant 639.073.807 and the Czech Science Foundation Grant No. 22-06414L.
R. Kuznets—Supported by the Austrian Science Fund (FWF) ByzDEL project (P33600).

Apart from the UIP, we are also interested in the uniform Lyndon interpolation property (ULIP) that is a strengthening of the UIP in the sense that interpolants must respect the polarities of the propositional variables involved. Kurahashi [18] first introduced this property and proved it for several normal modal logics, by employing a semantic method using layered bisimulations. A sequent-based proof-theoretic method was used in [1] to show the ULIP for several non-normal modal logics and conditional logics.

Our long-term goal is to provide a general proof-theoretic method to (re)prove the UIP for modal logics via multisequent calculi (i.e., nested sequents, hypersequents, labelled hypersequents, etc.). Unlike many other ways of proving interpolation, the proof-theoretic treatment is constructive in that it additionally yields an algorithm for constructing uniform interpolants. Towards this goal, we build on the modular treatment of multicomponent calculi to prove the CIP for modal and intermediate logics in [8,19,21,23,24]. First steps have been made by reproving the UIP for modal logics K, D, and T via nested sequents [12] and for $\mathsf{S5}$ via hypersequents [11,13], the first time this is proved proof-theoretically for $\mathsf{S5}$.

In this paper, we focus on logics $\mathsf{K5}$, $\mathsf{KD5}$, $\mathsf{K45}$, $\mathsf{KD45}$, $\mathsf{KB5}$, and $\mathsf{S5}$. The ULIP for these logics was derived in [18, Prop. 3] from the logics' local tabularity [28] and Lyndon interpolation property (LIP) [20].

Towards a modular proof-theoretic treatment, we introduce a new form of multisequent calculi for these logics that we call *layered sequent calculi*, the structure of which is inspired by the structure of the Kripke frames for the concerned logics from [27]. For $\mathsf{S5}$, this results in standard hypersequents [4,26,31]. For $\mathsf{K5}$ and $\mathsf{KD5}$, the presented calculi are similar to grafted hypersequent calculi in [22] but without explicit weakening. Other, less related, proof systems include analytic cut-free sequent systems for $\mathsf{K5}$ and $\mathsf{KD5}$ [34], cut-free sequent calculi for $\mathsf{K45}$ and $\mathsf{KD45}$ [33], and nested sequent calculi for modal logics [7].

The layered sequent calculi introduced in this paper adopt a strong version of termination that only relies on a local loop-check based on saturation. For all concerned logics, this yields a decision procedure that runs in co-NP time, which is, therefore, optimal [15]. We provide a semantic completeness proof via a countermodel construction from failed proof search.

Finally, layered sequents are used to provide the first proof-theoretic proof of the ULIP for $\mathsf{K5}$. The method is adapted from [11,13] in which the UIP is proved for $\mathsf{S5}$ based on hypersequents. We provide an algorithm to construct uniform Lyndon interpolants purely by syntactic means using the termination strategy of the proof search. To show the correctness of the constructed interpolants, we use model-theoretic techniques inspired by bisimulation quantification in the setting of uniform Lyndon interpolation [18].

An extended version of the paper with more detailed proofs is found in [14].

2 Preliminaries

The language of modal logics consists of a set Pr of countably many (*propositional*) *atoms* $p, q, \ldots$, their *negations* $\overline{p}, \overline{q}, \ldots$, *propositional connectives* $\wedge$ and $\vee$,

Table 1. Modal axioms and their corresponding frame conditions.

Axiom	Formula	Frame condition
k	$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$	none
5	$\Diamond\varphi \to \Box\Diamond\varphi$	Euclidean: $wRv \wedge wRu \Rightarrow vRu$
4	$\Box\varphi \to \Box\Box\varphi$	transitive: $wRv \wedge vRu \Rightarrow wRu$
d	$\Box\varphi \to \Diamond\varphi$	serial: $\forall w \exists v(wRv)$
b	$\varphi \to \Box\Diamond\varphi$	symmetric: $wRv \Rightarrow vRw$
t	$\Box\varphi \to \varphi$	reflexive: $\forall w(wRw)$

boolean constants $\top$ and $\bot$, and *modal operators* $\Box$ and $\Diamond$. A *literal* ℓ is either an atom or its negation, and the set of all literals is denoted by Lit. We define *modal formulas* in the usual way and denote them by lowercase Greek letters $\varphi, \psi, \ldots$. We define $\overline{\varphi}$ using the usual De Morgan laws to push the negation inwards (in particular, $\overline{\overline{p}} := p$) and $\varphi \to \psi := \overline{\varphi} \vee \psi$. We use uppercase Greek letters $\Gamma, \Delta, \ldots$ to refer to finite *multisets* of formulas. We write Γ, Δ to mean $\Gamma \cup \Delta$ and Γ, φ to mean $\Gamma \cup \{\varphi\}$. The set of literals of a formula φ, denoted $\mathsf{Lit}(\varphi)$, is defined recursively: $\mathsf{Lit}(\top) = \mathsf{Lit}(\bot) = \varnothing$, $\mathsf{Lit}(\ell) = \ell$ for $\ell \in \mathsf{Lit}$, $\mathsf{Lit}(\varphi \wedge \psi) = \mathsf{Lit}(\varphi \vee \psi) = \mathsf{Lit}(\varphi) \cup \mathsf{Lit}(\psi)$, and $\mathsf{Lit}(\Box\varphi) = \mathsf{Lit}(\Diamond\varphi) = \mathsf{Lit}(\varphi)$.

We consider extensions of K5 with any combination of axioms 4, d, b, and t (Table 1). Several of the 16 combinations coincide, resulting in 6 logics: K5, KD5, K45, KD45, KB5, and S5 (Table 2). Throughout the paper, we assume $\mathsf{L} \in \{\mathsf{K5}, \mathsf{KD5}, \mathsf{K45}, \mathsf{KD45}, \mathsf{KB5}, \mathsf{S5}\}$ and write $\vdash_{\mathsf{L}} \varphi$ iff $\varphi \in \mathsf{L}$.

Definition 1 (Logic K5). Modal logic K5 *is axiomatized by the classical tautologies, axioms* k *and* 5*, and rules modus ponens (from φ and $\varphi \to \psi$ infer ψ) and necessitation (from φ infer $\Box\varphi$).*

Throughout the paper we employ the semantics of Kripke frames and models.

Definition 2 (Kripke semantics). *A* Kripke frame *is a pair (W, R) where W is a nonempty set of* worlds *and $R \subseteq W \times W$ a binary relation. A* Kripke model *is a triple (W, R, V) where (W, R) is a Kripke frame and $V\colon \mathsf{Pr} \to \mathcal{P}(W)$ is a* valuation function. *A formula φ is defined to be* true *at a world w in a model $\mathcal{M} = (W, R, V)$, denoted $\mathcal{M}, w \vDash \varphi$, as follows: $\mathcal{M}, w \vDash \top$, $\mathcal{M}, w \nvDash \bot$ and*

$$\begin{array}{lll}
\mathcal{M}, w \vDash p & \textit{iff} & w \in V(p) \\
\mathcal{M}, w \vDash \overline{p} & \textit{iff} & w \notin V(p) \\
\mathcal{M}, w \vDash \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \vDash \varphi \textit{ and } \mathcal{M}, w \vDash \psi \\
\mathcal{M}, w \vDash \varphi \vee \psi & \textit{iff} & \mathcal{M}, w \vDash \varphi \textit{ or } \mathcal{M}, w \vDash \psi \\
\mathcal{M}, w \vDash \Box\varphi & \textit{iff} & \textit{for all } v \in W \textit{ such that } wRv,\ \mathcal{M}, v \vDash \varphi \\
\mathcal{M}, w \vDash \Diamond\varphi & \textit{iff} & \textit{there exists } v \in W \textit{ such that } wRv \textit{ and } \mathcal{M}, v \vDash \varphi.
\end{array}$$

Formula φ is valid *in $\mathcal{M} = (W, R, V)$, denoted $\mathcal{M} \vDash \varphi$, iff for all $w \in W$, $\mathcal{M}, w \vDash \varphi$. We call $\varnothing \neq C \subseteq W$ a* cluster *(in $\mathcal{M}$) iff $C \times C \subseteq R$, i.e., the relation R is* total *on C. We write wRC iff wRv for all $v \in C$.*

Table 2. Semantics for extensions of K5 (see [27,29]). Everywhere not $\rho R\rho$ for the root ρ, set C is a finite cluster, and $\sqcup$ denotes disjoint union.

Logic L	Axiomatization	Class of L-frames (W, R)
K5	Definition 1	$W = \{\rho\}$ or $W = \{\rho\} \sqcup C$
KD5	K5 + d	$W = \{\rho\} \sqcup C$
K45	K5 + 4	$W = \{\rho\}$ or ($W = \{\rho\} \sqcup C$ and ρRC)
KD45	K5 + d + 4	$W = \{\rho\} \sqcup C$ and ρRC
KB5	K5 + b	$W = \{\rho\}$ or $W = C$
S5	K5 + t	$W = C$

We work with specific classes of Kripke models sound and complete w.r.t. the logics. The respective frame conditions for the logic L, called L-*frames*, are defined in Table 2. A model (W, R, V) is an L-*model* iff (W, R) is an L-frame. Table 2 is a refinement of Theorem 3, particularly shown for K45, KD45, and KB5 in [29]. More precisely, we consider rooted frames and completeness w.r.t. the root, i.e., $\vdash_{\mathsf{L}} \varphi$ iff for all L-models $\mathcal{M}$ with root ρ, $\mathcal{M}, \rho \vDash \varphi$ (we often denote the if-condition as $\vDash_{\mathsf{L}} \varphi$). For each logic, this follows from easy bisimulation arguments.

Theorem 3 ([27]). *Any normal modal logic containing* K5 *is sound and complete w.r.t. a class of finite Euclidean Kripke frames* (W, R) *of one of the following forms:* (a) $W = \{\rho\}$ *consists of a singleton root and* $R = \varnothing$, (b) *the whole* W *is a cluster (any world can be considered its root), or* (c) $W \backslash \{\rho\}$ *is a cluster for a (unique) root* $\rho \in W$ *such that* $\rho R w$ *for some* $w \in W \backslash \{\rho\}$ *while not* $\rho R \rho$.

Definition 4 (UIP and ULIP). *A logic* L *has the* uniform interpolation property (UIP) *iff for any formula* φ *and* $p \in \mathsf{Pr}$ *there is a formula* $\forall p\varphi$ *such that*

(1) $\mathsf{Lit}(\forall p\varphi) \subseteq \mathsf{Lit}(\varphi) \setminus \{p, \overline{p}\}$,
(2) $\vdash_{\mathsf{L}} \forall p\varphi \to \varphi$, *and*
(3) $\vdash_{\mathsf{L}} \psi \to \varphi$ *implies* $\vdash_{\mathsf{L}} \psi \to \forall p\varphi$ *for any formula* ψ *with* $p, \overline{p} \notin \mathsf{Lit}(\psi)$.

A logic L *has the* uniform Lyndon interpolation property (ULIP) *[1,18] iff for any formula* φ *and* $\ell \in \mathsf{Lit}$, *there is a formula* $\forall \ell\varphi$ *such that*

(i) $\mathsf{Lit}(\forall \ell\varphi) \subseteq \mathsf{Lit}(\varphi) \setminus \{\ell\}$,
(ii) $\vdash_{\mathsf{L}} \forall \ell\varphi \to \varphi$, *and*
(iii) $\vdash_{\mathsf{L}} \psi \to \varphi$ *implies* $\vdash_{\mathsf{L}} \psi \to \forall \ell\varphi$ *for any formula* ψ *with* $\ell \notin \mathsf{Lit}(\psi)$.

We call $\forall p\varphi$ ($\forall \ell\varphi$) *the* uniform (Lyndon) interpolant of φ w.r.t. atom p (literal ℓ).

These are often called *pre-interpolants* as opposed to their dual *post-interpolants* that, in classical logic, can be defined as $\exists p\varphi = \overline{\forall p\overline{\varphi}}$ and $\exists \ell\varphi = \overline{\forall \overline{\ell}\overline{\varphi}}$ (see, e.g., [1,5,11,18] for more explanations).

Theorem 5. *If a logic* L *has the ULIP, then it also has the UIP.*

Proof. We define a uniform interpolant of φ w.r.t. atom p as a uniform Lyndon interpolant $\forall p \forall \overline{p} \varphi$ of $\forall \overline{p} \varphi$ w.r.t. literal p. We need to demonstrate conditions LIP(1)–(3) from Definition 4. First, it follows from ULIP(i) that $\mathsf{Lit}(\forall p \forall \overline{p} \varphi) \subseteq \mathsf{Lit}(\forall \overline{p} \varphi) \setminus \{p\} \subseteq \mathsf{Lit}(\varphi) \setminus \{p, \overline{p}\}$. Second, $\vdash_{\mathsf{L}} \forall p \forall \overline{p} \varphi \to \forall \overline{p} \varphi$ and $\vdash_{\mathsf{L}} \forall \overline{p} \varphi \to \varphi$ by ULIP(ii), hence, $\vdash_{\mathsf{L}} \forall p \forall \overline{p} \varphi \to \varphi$. Finally, if $\vdash_{\mathsf{L}} \psi \to \varphi$ where $p, \overline{p} \notin \mathsf{Lit}(\psi)$, then by ULIP(iii), $\vdash_{\mathsf{L}} \psi \to \forall \overline{p} \varphi$ as $\overline{p} \notin \mathsf{Lit}(\psi)$ and $\vdash_{\mathsf{L}} \psi \to \forall p \forall \overline{p} \varphi$ as $p \notin \mathsf{Lit}(\psi)$. $\square$

3 Layered Sequents

Definition 6 (Layered sequents). *A* layered sequent *is a generalized one-sided sequent of the form*

$$\mathcal{G} = \Gamma_1, \ldots, \Gamma_n, [\Sigma_1], \ldots, [\Sigma_m], [[\Pi_1]], \ldots, [[\Pi_k]] \tag{1}$$

where $\Gamma_i, \Sigma_i, \Pi_i$ *are finite multisets of formulas,* $n, m, k \geq 0$*, and if* $k \geq 1$*, then* $m \geq 1$*. A layered sequent is an* L*-sequent iff it satisfies the conditions in the rightmost column of Table 3. Each* Σ_i*, each* Π_i*, and* $\bigcup_i \Gamma_i$ *is called a* sequent component *of* $\mathcal{G}$*. The* formula interpretation *of a layered sequent* $\mathcal{G}$ *above is:*

$$\iota(\mathcal{G}) = \bigvee\nolimits_{i=1}^{n} \left(\bigvee \Gamma_i\right) \vee \bigvee\nolimits_{i=1}^{m} \Box\left(\bigvee \Sigma_i\right) \vee \bigvee\nolimits_{i=1}^{k} \Box\Box\left(\bigvee \Pi_i\right).$$

Layered sequents are denoted by $\mathcal{G}$ and $\mathcal{H}$. The structure of a layered sequent can be viewed as at most two layers of hypersequents ([ ]-*components* Σ_i and [[ ]]-*components* Π_i forming the first and second layer respectively) possibly nested on top of the sequent component $\bigcup_i \Gamma_i$ as the root. Following the arboreal terminology from [22], the root is called the *trunk* while [ ]- and [[ ]]-components form the *crown*. Analogously to nested sequents representing tree-like Kripke models, the structure of L-sequents is in line with the structure of L-models introduced in Sect. 2. We view sequents components as freely permutable, e.g., $[[\Pi_1]], \Gamma_1, [\Sigma_1], \Gamma_2$ and $\Gamma_1, \Gamma_2, [\Sigma_1], [[\Pi_1]]$ represent the same layered sequent.

Table 3. Layered sequent calculi $\mathsf{L.L}$: in addition to explicitly stated rules, all $\mathsf{L.L}$ have axioms $\mathsf{id_P}$ and $\mathsf{id_\top}$ and rules $\vee$, $\wedge$, $\Diamond_c$, and t (see Fig. 1). Note that the rules of system $\mathsf{L.L}$ may only be applied to L-sequents.

Calculus	Sequent rules								Conditions on layered sequents
L.K5	$\Box_t$		$\Diamond_t$		$\Box_{c'}$				$n \geq 1,\ m, k \geq 0$
L.KD5	$\Box_t$		$\Diamond_t$		$\Box_{c'}$	d_t		$\mathsf{d}_{c'}$	$n \geq 1,\ m, k \geq 0$
L.K45	$\Box_t$		$\Diamond_t$	$\Box_c$					$n \geq 1,\ m \geq 0,\ k = 0$
L.KD45	$\Box_t$		$\Diamond_t$	$\Box_c$		d_t	d_c		$n \geq 1,\ m \geq 0,\ k = 0$
L.KB5		$\Box_{t'}$		$\Box_c$					$n = 0, m \geq 2, k = 0$ or $n = 1, m = 0, k = 0$
L.S5				$\Box_c$					$n = 0,\ m \geq 1,\ k = 0$

Remark 7. The layered calculi presented here generalize grafted hypersequents of [22] and, hence, similarly combine features of hypersequents and nested sequents. In particular, layered sequents are generally neither pure hypersequents (except for the case of S5) nor bounded-depth nested sequents. The latter is due to the fact that the defining property of nested sequents is the tree structure of the sequent components, whereas the crown components of a layered sequent form a cluster. Although formally grafted hypersequents are defined with one layer only, this syntactic choice is more of a syntactic sugar than a real distinction. Indeed, the close relationship of one-layer grafted hypersequents for K5 and KD5 in [22] to the two-layer layered sequents presented here clearly manifests itself when translating grafted hypersequents into the prefixed-tableau format (see grafted tableau system for K5 [22, Sect. 6]). There prefixes for the crown are separated into two types, limbs and twigs, which match the separation into [ ]- and [[ ]]-components.

For a layered sequent (1), we assign labels to the components as follows: the trunk is labeled $\bullet$, [ ]-components get distinct labels $\bullet 1, \bullet 2, \ldots$, and [[ ]]-components get distinct labels $1, 2, \ldots$. We let $\sigma, \tau, \ldots$ range over these labels. The set of labels is denoted $\mathit{Lab}(\mathcal{G})$ and $\sigma \in \mathcal{G}$ means $\sigma \in \mathit{Lab}(\mathcal{G})$. We write $\sigma : \varphi \in \mathcal{G}$ (or $\sigma : \varphi$ if no confusion occurs) when a formula φ occurs in a sequent component of $\mathcal{G}$ labeled by σ.

Example 8. $\mathcal{G} = \varphi, \psi, [\chi], [\xi], [[\theta]]$ is a layered sequent with the trunk and three crown components: two [ ]-components and one [[ ]]-component. Since it has both the trunk and a [[ ]]-component, it can only be a K5- or KD5-sequent. A corresponding labeled sequent is $\mathcal{G} = \varphi_\bullet, \psi_\bullet, [\chi]_{\bullet 1}, [\xi]_{\bullet 2}, [[\theta]]_1$, with the set $\mathit{Lab}(\mathcal{G}) = \{\bullet, \bullet 1, \bullet 2, 1\}$ of four labels. Similarly, for the KB5/S5-sequent $\mathcal{H} = [\sigma], [\delta]$, a corresponding labeled sequent is $\mathcal{H} = [\sigma]_{\bullet 1}, [\delta]_{\bullet 2}$ with $\mathit{Lab}(\mathcal{H}) = \{\bullet 1, \bullet 2\}$.

$$\mathsf{id_P}\,\frac{}{\mathcal{G}\{p, \overline{p}\}} \qquad \mathsf{id}_\top\,\frac{}{\mathcal{G}\{\top\}} \qquad \wedge\,\frac{\mathcal{G}\{\varphi \wedge \psi, \varphi\} \quad \mathcal{G}\{\varphi \wedge \psi, \psi\}}{\mathcal{G}\{\varphi \wedge \psi\}}$$

$$\vee\,\frac{\mathcal{G}\{\varphi \vee \psi, \varphi, \psi\}}{\mathcal{G}\{\varphi \vee \psi\}} \qquad \Box_t\,\frac{\mathcal{G}, \Box\varphi, [\varphi]}{\mathcal{G}, \Box\varphi} \qquad \Box_{t'}\,\frac{[\varSigma, \Box\varphi], [\varphi]}{\varSigma, \Box\varphi} \qquad \Diamond_t\,\frac{\mathcal{G}, \Diamond\varphi, [\varSigma, \varphi]}{\mathcal{G}, \Diamond\varphi, [\varSigma]}$$

$$\Box_c\,\frac{\mathcal{G}, \llbracket \varSigma, \Box\varphi \rrbracket, [\varphi]}{\mathcal{G}, \llbracket \varSigma, \Box\varphi \rrbracket} \qquad \Box_{c'}\,\frac{\mathcal{G}, \llbracket \varSigma, \Box\varphi \rrbracket, [[\varphi]]}{\mathcal{G}, \llbracket \varSigma, \Box\varphi \rrbracket} \qquad \Diamond_c\,\frac{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket, \llparenthesis \varPi, \varphi \rrparenthesis}{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket, \llparenthesis \varPi \rrparenthesis}$$

$$\mathsf{d}_t\,\frac{\mathcal{G}, \Diamond\varphi, [\varphi]}{\mathcal{G}, \Diamond\varphi} \qquad \mathsf{d}_c\,\frac{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket, [\varphi]}{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket} \qquad \mathsf{d}_{c'}\,\frac{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket, [[\varphi]]}{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket} \qquad \mathsf{t}\,\frac{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi, \varphi \rrbracket}{\mathcal{G}, \llbracket \varSigma, \Diamond\varphi \rrbracket}$$

Fig. 1. Layered sequent rules: brackets $\llbracket\ \rrbracket$ and $\llparenthesis\ \rrparenthesis$ range over both [ ] and [[ ]].

We sometimes use *unary contexts*, i.e., layered sequents with exactly one *hole*, denoted $\{\ \}$. Such contexts are denoted by $\mathcal{G}\{\ \}$. The insertion $\mathcal{G}\{\Gamma\}$ of a finite multiset Γ into $\mathcal{G}\{\ \}$ is obtained by replacing $\{\ \}$ with Γ. The hole $\{\ \}$ in a component σ can also be labeled $\mathcal{G}\{\ \}_\sigma$. We use the notations $[\![\ ]\!]$ and $(\!|\ |\!)$ to refer to either of $[\ ]$ or $[[\ ]]$.

Using Fig. 1 and the middle column of Table 3, we define layered sequent calculi L.K5, L.KD5, L.K45, L.KD45, L.KB5, and L.S5, where L.L is the calculus for the logic L. Following the terminology from [22], we split all modal rules into *trunk rules* (subscript t) and *crown rules* (subscript c) depending on the position of the *principal* formula. We write $\vdash_{\mathsf{L.L}} \mathcal{G}$ iff $\mathcal{G}$ is derivable in L.L.

Definition 9 (Saturation). *Labeled formula $\sigma : \varphi \in \mathcal{G}$ is* saturated for L.L *iff*

- *φ equals p or $\overline{p}$ for an atom p, or equals $\bot$, or equals $\top$;*
- *$\varphi = \varphi_1 \wedge \varphi_2$ and $\sigma : \varphi_i \in \mathcal{G}$ for some i;*
- *$\varphi = \varphi_1 \vee \varphi_2$ and both $\sigma : \varphi_1 \in \mathcal{G}$ and $\sigma : \varphi_2 \in \mathcal{G}$;*
- *$\varphi = \Box\varphi'$, the unique rule applicable to $\sigma : \Box\varphi'$ in L.L is either $\Box_t$ or $\Box_c$ (i.e., a rule creating a $[\ ]$-component), and $\bullet i : \varphi' \in \mathcal{G}$ for some i;*
- *$\varphi = \Box\varphi'$, the unique rule applicable to $\sigma : \Box\varphi'$ in L.L is $\Box_{c'}$ (i.e., a rule creating a $[[\ ]]$-component), and $i : \varphi' \in \mathcal{G}$ for some i.*

In addition, we define for any label σ and formula φ:

- *$\sigma : \Diamond\varphi$ is* saturated w.r.t. *$\bullet \in Lab(\mathcal{G})$;*
- *$\sigma : \Diamond\varphi$ is* saturated w.r.t. a label *$\bullet i \in Lab(\mathcal{G})$ iff $\bullet i : \varphi \in \mathcal{G}$;*
- *$\sigma : \Diamond\varphi$ is* saturated w.r.t. a label *$i \in Lab(\mathcal{G})$ iff $\sigma = \bullet$ or $i : \varphi \in \mathcal{G}$;*
- *$\sigma : \Diamond\varphi$ is* d_t-saturated *iff $\sigma \neq \bullet$ or $\bullet i : \varphi \in \mathcal{G}$ for some i;*
- *$\sigma : \Diamond\varphi$ is* d_c-saturated *iff $\sigma = \bullet$ or $\bullet i : \varphi \in \mathcal{G}$ for some i;*
- *$\sigma : \Diamond\varphi$ is* d'_c-saturated *iff $\sigma = \bullet$ or $i : \varphi \in \mathcal{G}$ for some i.*

$\mathcal{G}$ is propositionally saturated *iff all $\vee$- and $\wedge$-formulas are saturated in $\mathcal{G}$. L-sequent $\mathcal{G}$ is* L-saturated *iff a) each non-$\Diamond$ formula is saturated, b) each $\sigma : \Diamond\varphi$ is saturated w.r.t. every label in $Lab(\mathcal{G})$, c) each $\sigma : \Diamond\varphi$ is* d*-saturated whenever* $\mathsf{d} \in \mathsf{L.L} \cap \{\mathsf{d}_t, \mathsf{d}_c, \mathsf{d}_{c'}\}$*, and d) $\mathcal{G}$ is not of the from $\mathcal{H}\{\top\}$ or $\mathcal{H}\{q, \overline{q}\}$ for some* $q \in \mathsf{Pr}$.

Theorem 10. *Proof search in* L.L *modulo saturation terminates and provides an optimal-complexity decision algorithm, i.e., runs in co-NP time.*

Proof. Given a proof search of layered sequent $\mathcal{G}$, for each layered sequent $\mathcal{H}$ in this proof search, consider its labeled formulas as a set $F_{\mathcal{H}} = \{\sigma : \varphi \mid \sigma : \varphi \in \mathcal{H}\}$. Let s be the number of subformulas occurring in $\mathcal{G}$ and N be the number of sequent components in $\mathcal{G}$. Since we only apply rules (that do not equal $\mathsf{id_P}$ or $\mathsf{id}_\top$) to non-saturated sequents, sets $F_{\mathcal{H}}$ will grow for each premise. Going bottom-up in the proof search, at most s labels of the form $\bullet i$ and at most s labels of the form i can be created, and each label can have at most s formulas. Therefore, the cardinality of sets $F_{\mathcal{H}}$ are bounded by $s(N+s+s)$, which is polynomial in the size of $F_{\mathcal{G}}$. Hence, the proof search terminates modulo saturation. Moreover, since each added labeled formula is linear in the size $F_{\mathcal{G}}$ and the non-deterministic branching in the proof search is bounded by $(N+s+s)s(N+s+s)$, again a polynomial in the size of $F_{\mathcal{G}}$, this algorithm is co-NP, i.e., provides an optimal decision procedure for the logic. $\square$

Definition 11 (Interpretations). *An* interpretation of an L-sequent $\mathcal{G}$ into an L-model $\mathcal{M} = (W, R, V)$ *is a function* $\mathcal{I} : Lab(\mathcal{G}) \to W$ *such that the following conditions apply whenever the respective type of labels exists in* $\mathcal{G}$*:*

1. $\mathcal{I}(\bullet) = \rho$*, where* ρ *is the root of* $\mathcal{M}$*;*
2. $\mathcal{I}(\bullet) R\, \mathcal{I}(\bullet i)$ *for each label of the form* $\bullet i \in Lab(\mathcal{G})$*;*
3. $\mathcal{I}(\bullet i) R\, \mathcal{I}(j)$ *and* $\mathcal{I}(j) R\, \mathcal{I}(\bullet i)$ *for all labels of the form* $\bullet i$ *and* j *in* $Lab(\mathcal{G})$*;*
4. *Not* $\mathcal{I}(\bullet) R\, \mathcal{I}(j)$ *for any label of the form* $j \in Lab(\mathcal{G})$*.*

Note that none of the conditions (1)–(4) apply to layered $\mathsf{S5}$-sequents.

Definition 12 (Sequent semantics). *For any given interpretation* $\mathcal{I}$ *of an* L*-sequent* $\mathcal{G}$ *into an* L*-model* $\mathcal{M}$*,*

$$\mathcal{M}, \mathcal{I} \vDash \mathcal{G} \qquad \textit{iff} \qquad \mathcal{M}, \mathcal{I}(\sigma) \vDash \varphi \textit{ for some } \sigma : \varphi \in \mathcal{G}.$$

$\mathcal{G}$ *is* valid *in* L*, denoted* $\vDash_{\mathsf{L}} \mathcal{G}$*, iff* $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$ *for all* L*-models* $\mathcal{M}$ *and interpretations* $\mathcal{I}$ *of* $\mathcal{G}$ *into* $\mathcal{M}$*. We omit* L *and* $\mathcal{M}$ *when clear from the context.*

The proof of the following theorem is based on a countermodel construction (for more standard parts of the proof we refer to the Appendix of [14]):

Theorem 13 (Soundness and completeness). *For any* L*-sequent* $\mathcal{G}$*,*

$$\vdash_{\mathsf{L.L}} \mathcal{G} \qquad \Longleftrightarrow \qquad \vDash_{\mathsf{L}} \iota(\mathcal{G}) \qquad \Longleftrightarrow \qquad \vDash_{\mathsf{L}} \mathcal{G}.$$

Proof. We show a cycle of implications. The left-to-middle implication, i.e., that $\vdash_{\mathsf{L.L}} \mathcal{G} \Longrightarrow \vDash_{\mathsf{L}} \iota(\mathcal{G})$, can be proved by induction on the $\mathsf{L.L}$-derivation of $\mathcal{G}$.

For the middle-to-right implication, i.e., $\vDash_{\mathsf{L}} \iota(\mathcal{G}) \Longrightarrow \vDash_{\mathsf{L}} \mathcal{G}$, let $\mathcal{G}$ be a sequent of form (1). We prove that $\mathcal{M}, \mathcal{I} \nvDash \mathcal{G}$ implies $\mathcal{M}, \mathcal{I}(\bullet) \nvDash \iota(\mathcal{G})$ (if $n = 0$, use 1 in place of $\bullet$). By definition, $\mathcal{I}(\bullet)$ is the root of $\mathcal{M}$. If $\mathcal{M}, \mathcal{I} \nvDash \mathcal{G}$, then $\mathcal{I}(\bullet) \nvDash \varphi$ for all $\varphi \in \bigcup_{i=1}^{n} \Gamma_i$, for each $1 \leq i \leq m$ we have $\mathcal{I}(\bullet i) \nvDash \psi$ for all $\psi \in \Sigma_i$, and for each $1 \leq i \leq k$ we have $\mathcal{I}(i) \nvDash \chi$ for all $\chi \in \Pi_i$. By Definition 11, in case $k \geq 1$ label $\bullet 1$ is in $\mathcal{G}$ and $\mathcal{I}(\bullet) R \mathcal{I}(\bullet 1) R \mathcal{I}(i)$ for each $1 \leq i \leq k$. Therefore $\mathcal{M}, \mathcal{I}(\bullet) \nvDash \iota(\mathcal{G})$.

Finally, we prove the right-to-left implication by contraposition using a countermodel construction: from a failed proof search of $\mathcal{G}$, construct an L-model refuting $\mathcal{G}$ from (1). In a failed proof-search tree (Theorem 10), since $\nvdash_{\mathsf{L.L}} \mathcal{G}$, at least one saturated leaf

$$\mathcal{G}' = \Gamma', [\Sigma_1'], \ldots, [\Sigma_m'], [\Sigma_1''], \ldots, [\Sigma_{m'}''], [[\Pi_1']], \ldots, [[\Pi_k']], [[\Pi_1'']], \ldots, [[\Pi_{k'}'']],$$

is such that $\bigcup_i \Gamma_i \subseteq \Gamma'$, $\Sigma_i \subseteq \Sigma_i'$, and $\Pi_i \subseteq \Pi_i'$ (or for $\mathsf{KB5}$, if $\mathcal{G} = \Gamma$, then $\mathcal{G}' = \Gamma'$ for $\Gamma \subseteq \Gamma'$ or $[\Sigma], [\Sigma_1], \ldots, [\Sigma_m]$ with $\Gamma \subseteq \Sigma$). Define $\mathcal{M} = (W, R, V)$:

$$W = Lab(\mathcal{G}'), \qquad V(p) = \{\sigma \mid \sigma : \overline{p} \in \mathcal{G}'\},$$
$$R = \{(\bullet, \bullet i) \mid \bullet i \in Lab(\mathcal{G}')\} \cup \{(\sigma, \tau) \mid \sigma, \tau \in Lab(\mathcal{G}'), \sigma, \tau \neq \bullet\}.$$

Since $\mathcal{G}'$ is saturated, $\mathcal{M}$ is an L-model. Taking $\mathcal{I}$ of $\mathcal{G}$ into $\mathcal{M}$ as the identity function (or $\mathcal{I}(\bullet) = 1$ in case of $\mathsf{KB5}$), we have $\mathcal{M}, \mathcal{I} \nvDash \mathcal{G}$ as desired. $\square$

4 Uniform Lyndon Interpolation

Definition 14 (Multiformulas). *The grammar*

$$\mho ::= \sigma : \varphi \mid (\mho \owedge \mho) \mid (\mho \ovee \mho)$$

defines multiformulas, *where $\sigma : \varphi$ is a labeled formula. $Lab(\mho)$ denotes the set of labels of $\mho$. An* interpretation *$\mathcal{I}$ of a layered sequent $\mathcal{G}$ into a model $\mathcal{M}$ is called an* interpretation of a multiformula $\mho$ into $\mathcal{M}$ *iff $Lab(\mho) \subseteq Lab(\mathcal{G})$. If $\mathcal{I}$ is an interpretation of $\mho$ into $\mathcal{M}$, we define $\mathcal{M}, \mathcal{I} \vDash \mho$ as follows:*

$\mathcal{M}, \mathcal{I} \vDash \sigma : \varphi$	*iff*	$\mathcal{M}, \mathcal{I}(\sigma) \vDash \varphi$,
$\mathcal{M}, \mathcal{I} \vDash \mho_1 \owedge \mho_2$	*iff*	$\mathcal{M}, \mathcal{I} \vDash \mho_1$ *and* $\mathcal{M}, \mathcal{I} \vDash \mho_2$,
$\mathcal{M}, \mathcal{I} \vDash \mho_1 \ovee \mho_2$	*iff*	$\mathcal{M}, \mathcal{I} \vDash \mho_i$ *for at least one* $i = 1, 2$.

Multiformulas $\mho_1$ and $\mho_2$ are said to be equivalent, *denoted $\mho_1 \equiv_{\mathsf{L}} \mho_2$, or simply $\mho_1 \equiv \mho_2$, iff $\mathcal{M}, \mathcal{I} \vDash \mho_1 \Leftrightarrow \mathcal{M}, \mathcal{I} \vDash \mho_2$ for any interpretation $\mathcal{I}$ of both $\mho_1$ and $\mho_2$ into an L-model $\mathcal{M}$.*

Lemma 15 ([21]). *Any multiformula $\mho$ can be transformed into an equivalent one in SDNF (SCNF) as a $\ovee$-disjunction ($\owedge$-conjunction) of $\owedge$-conjunctions ($\ovee$-disjunctions) of labeled formulas $\sigma : \varphi$ such that each label of $\mho$ occurs exactly once per conjunct (disjunct).*

Definition 16 (Bisimilarity) . *Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$ be models and $\ell \in \mathsf{Lit}$. We say $\mathcal{M}'$ is ℓ-*bisimilar *to $\mathcal{M}$, denoted $\mathcal{M}' \leq_\ell \mathcal{M}$ iff there is a nonempty binary relation $Z \subseteq W \times W'$, called an ℓ-*bisimulation *between $\mathcal{M}$ and $\mathcal{M}'$, such that the following hold for every $w \in W$ and $w' \in W'$:*

literals$_\ell$. *if wZw', then a) $\mathcal{M}, w \vDash q$ iff $\mathcal{M}', w' \vDash q$ for all atoms $q \notin \{\ell, \overline{\ell}\}$ and b) if $\mathcal{M}', w' \vDash \ell$, then $\mathcal{M}, w \vDash \ell$;*
forth. *if wZw' and wRv, then there exists $v' \in W'$ such that vZv' and $w'R'v'$;*
back. *if wZw' and $w'R'v'$, then there exists $v \in W$ such that vZv' and wRv.*

$\mathcal{M}$ and $\mathcal{M}'$ are bisimilar, *denoted $\mathcal{M} \sim \mathcal{M}'$, iff there is a relation $Z \neq \varnothing$ satisfying **forth** and **back**, as well as part a) of **literals$_\ell$** for any $p \in \mathsf{Pr}$, in which case Z is called a* bisimulation. *We write (similarly for $\sim$ instead of $\leq_\ell$):*

- *$(\mathcal{M}', w') \leq_\ell (\mathcal{M}, w)$ iff there is an ℓ-bisimulation Z, such that wZw';*
- *$(\mathcal{M}', \mathcal{I}') \leq_\ell (\mathcal{M}, \mathcal{I})$ for functions $\mathcal{I} : X \to W$ and $\mathcal{I}' : X \to W'$ iff there is an ℓ-bisimulation Z such that $\mathcal{I}(\sigma)\, Z\, \mathcal{I}'(\sigma)$ for each $\sigma \in X$.*

Note that $\leq_\ell$ is a preorder and we have $\mathcal{M}' \leq_\ell \mathcal{M}$ iff $\mathcal{M} \leq_{\overline{\ell}} \mathcal{M}'$. By analogy with [6, Theorem 2.20], we have the following immediate observation, which additionally holds for multiformulas $\mho$ (we provide a proof in [14]):

Lemma 17. *Let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations of a layered sequent $\mathcal{G}$ into models $\mathcal{M}$ and $\mathcal{M}'$ respectively.*

1. *Let $\ell \notin \mathsf{Lit}(\mathcal{G})$. If $(\mathcal{M}', \mathcal{I}') \leq_\ell (\mathcal{M}, \mathcal{I})$, then $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$ implies $\mathcal{M}', \mathcal{I}' \vDash \mathcal{G}$.*
2. *If $(\mathcal{M}, \mathcal{I}) \sim (\mathcal{M}', \mathcal{I}')$, then $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$ iff $\mathcal{M}', \mathcal{I}' \vDash \mathcal{G}$.*

Definition 18 (BLUIP). *Logic* L *is said to have the* bisimulation layered-sequent uniform interpolation property (BLUIP) *iff for every literal* ℓ *and every* L*-sequent* $\mathcal{G}$*, there is a multiformula* $A_\ell(\mathcal{G})$*, called* BLU interpolant*, such that:*

(i) $\mathsf{Lit}\big(A_\ell(\mathcal{G})\big) \subseteq \mathsf{Lit}(\mathcal{G})\backslash\{\ell\}$ *and* $Lab\big(A_\ell(\mathcal{G})\big) \subseteq Lab(\mathcal{G})$*;*
(ii) for each interpretation $\mathcal{I}$ *of* $\mathcal{G}$ *into an* L*-model* $\mathcal{M}$*,*

$$\mathcal{M},\mathcal{I} \vDash A_\ell(\mathcal{G}) \quad \textit{implies} \quad \mathcal{M},\mathcal{I} \vDash \mathcal{G};$$

(iii) for each L*-model* $\mathcal{M}$ *and interpretation* $\mathcal{I}$ *of* $\mathcal{G}$ *into* $\mathcal{M}$*, if* $\mathcal{M},\mathcal{I} \nvDash A_\ell(\mathcal{G})$*, then there is an* L*-model* $\mathcal{M}'$ *and interpretation* $\mathcal{I}'$ *of* $\mathcal{G}$ *into* $\mathcal{M}'$ *such that*

$$(\mathcal{M}',\mathcal{I}') \leq_\ell (\mathcal{M},\mathcal{I}) \textit{ and } \mathcal{M}',\mathcal{I}' \nvDash \mathcal{G}.$$

Lemma 19. *The BLUIP for* L *implies the ULIP for* L*.*

Proof. Let $\forall\ell\varphi = A_\ell(\varphi)$. We prove the properties of Definition 4. Variable property is immediate. For Property (ii), assume $\nvdash_{\mathsf{L}} A_\ell(\varphi) \to \varphi$. By completeness, we have $\mathcal{M},\rho \vDash A_\ell(\varphi)$ and $\mathcal{M},\rho \nvDash \varphi$ for some L-model $\mathcal{M}$ with root ρ. As ρ is the root, it can be considered as an interpretation by Definition 11. By condition (ii) from Definition 18 we get a contradiction. For (iii), let ψ be a formula such that $\ell \notin \mathsf{Lit}(\psi)$ and suppose $\nvdash_{\mathsf{L}} \psi \to A_\ell(\varphi)$. So there is an L-model $\mathcal{M}$ with root ρ such that $\mathcal{M},\rho \vDash \psi$ and $\mathcal{M},\rho \nvDash A_\ell(\varphi)$. Again, ρ is treated as an interpretation, and by (iii) from Definition 18, there is an L-model $\mathcal{M}'$ with root ρ' such that $(\mathcal{M}',\rho') \leq_\ell (\mathcal{M},\rho)$ and $\mathcal{M}',\rho' \nvDash \varphi$. By Lemma 17, $\mathcal{M}',\rho' \vDash \psi$, hence $\nvdash_{\mathsf{L}} \psi \to \varphi$ as desired. □

To show that calculus $\mathsf{L.K5}$ enjoys the BLUIP for $\mathsf{K5}$, we need two important ingredients: some model modifications that are closed under bisimulation and an algorithm to compute uniform Lyndon interpolants.

Definition 20 (Copying). *Let* $\mathcal{M} = (W, R, V)$ *be a* $\mathsf{K5}$*-model with root* ρ *and cluster* C*. Model* $\mathcal{N}' = (W \sqcup \{w_c\}, R', V')$ *is obtained by* copying $w \in C$ *iff* $R' = R \sqcup (\{w_c\} \times C) \sqcup (C \times \{w_c\}) \sqcup \{(\rho, w_c) \mid (\rho, w) \in R\} \sqcup \{(w_c, w_c)\}$*, and* $V'(p) = V(p) \sqcup \{w_c \mid w \in V(p)\}$ *for any* $p \in \mathsf{Pr}$*. Model* $\mathcal{N}'' = (W \sqcup \{w_c\}, R'', V')$ *is obtained by* copying w away from the root *iff* $R'' = R' \setminus \{(\rho, w_c)\}$*.*

Lemma 21. *Let model* $\mathcal{N}$ *be obtained by copying a world* w *from a* $\mathsf{K5}$*-model* $\mathcal{M}$ *(away from the root). Let* $\mathcal{I}\colon X \to \mathcal{M}$ *and* $\mathcal{I}'\colon X \to \mathcal{N}$ *be interpretations such that for each* $x \in X$*, either* $\mathcal{I}(x) = \mathcal{I}'(x)$ *or* $\mathcal{I}(x) = w$ *while* $\mathcal{I}'(x) = w_c$*. Then,* $\mathcal{N}$ *is a* $\mathsf{K5}$*-model and* $(\mathcal{M},\mathcal{I}) \sim (\mathcal{N},\mathcal{I}')$*.*

In the construction of interpolants, we use the following rules d'_t and dd and sets $\mathcal{G}_c$ and $\Box\Diamond\mathcal{G}_c$ of formulas from the crown of $\mathcal{G}$:

$$\mathcal{G}_c = \{\varphi \mid \sigma : \varphi \in \mathcal{G}, \sigma \neq \bullet\} \qquad \Box\Diamond\mathcal{G}_c = \{\Box\varphi \mid \Box\varphi \in \mathcal{G}_c\} \sqcup \{\Diamond\varphi \mid \Diamond\varphi \in \mathcal{G}_c\}$$

$$\mathsf{d}'_t\,\frac{\Gamma, \big[\{\psi \mid \Diamond\psi \in \Gamma\}\big] \quad \Gamma, \Diamond\top}{\Gamma} \quad \text{and} \quad \mathsf{dd}\,\frac{\mathcal{G}, \big[\{\psi \mid \Diamond\psi \in \mathcal{G}\}\big], \big[\big[\{\chi \mid \Diamond\chi \in \mathcal{G}_c\}\big]\big]}{\mathcal{G}}$$

Rule d'_t shows similarities with rule d_t from logics $\mathsf{KD5}$ and $\mathsf{KD45}$, but is only applied in the absence of the crown. Rule d'_t is sound for $\mathsf{K5}$ because it can be viewed as a composition of an (admissible) cut on $\Box\bot$ and $\Diamond\top$ in the trunk, followed by $\Box_t$ in the left premise on $\Box\bot$ that creates the first crown component (though $\bot$ is dropped from it), which is populated using several $\Diamond_t$-rules for $\Diamond\psi \in \varGamma$. The label of this crown component is always $\bullet 1$. Rule dd provides extra information in the calculation of the uniform interpolant and is needed primarily for technical reasons. We highlight the two new sequent components created by the last instance of dd using special placeholder labels $\bullet\mathsf{d}$ and d for the respective brackets. These labels are purely for readability purposes and revert to the standard $\bullet j$ and k labels after the next instance of dd.

Table 4. Recursive construction of $A_\ell(t, \varSigma_c; \mathcal{G})$ for $\mathcal{G}$ that are not $\mathsf{K5}$-saturated.

	$\mathcal{G}$ matches	$A_\ell(t, \varSigma_c; \mathcal{G})$ equals
1.	$\mathcal{G}'\{\top\}_\sigma$	$\sigma : \top$
2.	$\mathcal{G}'\{q, \overline{q}\}_\sigma$	$\sigma : \top$
3.	$\mathcal{G}'\{\varphi \vee \psi\}$	$A_\ell\big(t, \varSigma_c; \mathcal{G}'\{\varphi \vee \psi, \varphi, \psi\}\big)$
4.	$\mathcal{G}'\{\varphi \wedge \psi\}$	$A_\ell\big(t, \varSigma_c; \mathcal{G}'\{\varphi \wedge \psi, \varphi\}\big) \owedge A_\ell\big(t, \varSigma_c; \mathcal{G}'\{\varphi \wedge \psi, \psi\}\big)$
5.	$\mathcal{G}', \Box\varphi$	$\bigowedge_{i=1}^{h} \left(\bullet : \Box\delta_i \ovee \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\right)$ where j is the smallest integer such that $\bullet j \notin \mathcal{G}$ and the SCNF of $A_\ell\big(t, \varSigma_c; \mathcal{G}', \Box\varphi, [\varphi]_{\bullet j}\big)$ is $\bigowedge_{i=1}^{h} \left(\bullet j : \delta_i \ovee \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\right)$,
6.	$\mathcal{G}', [\![\varSigma, \Box\varphi]\!]_\sigma$	$\bigowedge_{i=1}^{h} \left(\sigma : \Box\delta_i \ovee \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\right)$ where j is the smallest integer such that $j \notin \mathcal{G}$ and the SCNF of $A_\ell\big(t, \varSigma_c; \mathcal{G}', [\![\varSigma, \Box\varphi]\!]_\sigma, [[\varphi]]_j\big)$ is $\bigowedge_{i=1}^{h} \left(j : \delta_i \ovee \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\right)$,
7.	$\mathcal{G}', \Diamond\varphi, [\varSigma]$	$A_\ell\big(t, \varSigma_c; \mathcal{G}', \Diamond\varphi, [\varSigma, \varphi]\big)$
8.	$\mathcal{G}', [\![\varSigma, \Diamond\varphi]\!]$	$A_\ell\big(t, \varSigma_c; \mathcal{G}', [\![\varSigma, \Diamond\varphi, \varphi]\!]\big)$
9.	$\mathcal{G}', [\![\varSigma, \Diamond\varphi]\!], (\!|\Pi|\!)$	$A_\ell\big(t, \varSigma_c; \mathcal{G}', [\![\varSigma, \Diamond\varphi]\!], (\!|\Pi, \varphi|\!)\big)$

To compute a uniform Lyndon interpolant $\forall\ell\xi$ for a formula ξ, we first compute a BLU interpolant $A_\ell(0, \varnothing; \xi_\bullet)$ by using the recursive function $A_\ell(t, \varSigma_c; \mathcal{G})$ with three parameters we present below. The main parameter is a $\mathsf{K5}$-sequent $\mathcal{G}$, while the other two parameters are auxiliary: $t \in \{0, 1\}$ is a boolean variable such that $t = 1$ guarantees that rule dd has been applied at least once for the case when $\mathcal{G}$ contains diamond formulas; $\varSigma_c \subseteq \Box\Diamond\mathcal{G}_c$ is a set of modal formulas that provides a bookkeeping strategy to prevent redundant applications of rule dd.

To calculate $A_\ell(t, \varSigma_c; \mathcal{G})$ our algorithm makes a choice of which row from Table 4 to apply by trying each of the following steps in the specified order:

1. If possible, apply rows 1–2, i.e., stop and return $A_\ell(t, \varSigma_c; \mathcal{G}) = \sigma : \top$.

2. If some formula $\varphi \vee \psi$ (resp. $\varphi \wedge \psi$) from $\mathcal{G}$ is not saturated, compute $A_\ell(t, \Sigma_c; \mathcal{G})$ according to row 3 (resp. 4) applied to this formula.
3. If some formula $\Box\varphi \in \mathcal{G}$ is not saturated (resp. $\Diamond\varphi \in \mathcal{G}$ is not saturated w.r.t. $\sigma \in \mathcal{G}$), compute $A_\ell(t, \Sigma_c; \mathcal{G})$ according to the unique respective row among 5–9 applicable to this formula (w.r.t. σ).
4. If Steps 1–3 do not apply, i.e., $\mathcal{G}$ is saturated, proceed as follows:
   (a) if $\mathcal{G}$ has no $\Diamond$-formulas, stop and return $A_\ell(t, \Sigma_c; \mathcal{G}) = \mathsf{LitDis}_\ell(\mathcal{G})$ where

$$\mathsf{LitDis}_\ell(\mathcal{G}) = \bigovee_{\sigma:\ell' \in \mathcal{G}, \ell' \in \mathsf{Lit}\setminus\{\ell\}} \sigma : \ell' \tag{2}$$

   (b) else, if $\mathcal{G} = \Gamma$ consists of the trunk only, apply rule d'_t as follows:

$$A_\ell(t, \Sigma_c; \Gamma) =$$
$$\Big(\bullet : \Box\bot \ovee \bigovee_{i=1}^{h} \big(\bullet : \Diamond\delta_i \owedge \bullet : \gamma_i\big)\Big) \owedge \big(\bullet : \Diamond\top \ovee \mathsf{LitDis}_\ell(\Gamma)\big) \tag{3}$$

   where the SDNF of $A_\ell\Big(0, \Sigma_c; \quad \Gamma, \big[\{\psi \mid \Diamond\psi \in \Gamma\}\big]_{\bullet 1}\Big)$ is

$$\bigovee_{i=1}^{h} \Big(\bullet 1 : \delta_i \owedge \bullet : \gamma_i\Big) \tag{4}$$

   (c) else, if $t = 1$ and $\Box\Diamond\mathcal{G}_c \subseteq \Sigma_c$, stop and return $A_\ell(t, \Sigma_c; \mathcal{G}) = \mathsf{LitDis}_\ell(\mathcal{G})$.
   (d) else, apply the rule dd as follows (where w.l.o.g. $\bullet 1 \in \mathcal{G}$):

$$A_\ell(t, \Sigma_c; \mathcal{G}) = \bigovee_{i=1}^{h} \Big(\bullet : \Diamond\delta_i \owedge \bullet 1 : \Diamond\delta'_i \owedge \bigowedge_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\Big) \tag{5}$$

   where SDNF of $A_\ell\Big(1, \Box\Diamond\mathcal{G}_c;\ \mathcal{G}, \big[\{\psi \mid \Diamond\psi \in \mathcal{G}\}\big]_{\bullet\mathsf{d}}, \big[[\{\chi \mid \Diamond\chi \in \mathcal{G}_c\}]\big]_{\mathsf{d}}\Big)$ is

$$\bigovee_{i=1}^{h} \Big(\bullet\mathsf{d} : \delta_i \owedge \mathsf{d} : \delta'_i \owedge \bigowedge_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau}\Big) \tag{6}$$

The computation of the algorithm can be seen as a proof search tree (extended with rules d'_t and dd). In this proof search, call $A_\ell(t, \Sigma_c; \mathcal{G})$ is *sufficient* (to be a BLU interpolant for $\mathcal{G}$) if each branch going up from it either stops in Steps 1 or 4a or continues via Steps 4b or 4d. Otherwise, it is *insufficient*, if one of the branches stops in Step 4c, say, calculating $A_\ell(1, \Sigma_c; \mathcal{H})$. In this case, $A_\ell(1, \Sigma_c; \mathcal{H})$ is not generally a BLU interpolant for $\mathcal{H}$, but these leaves provide enough information to find a BLU interpolant from some sequent down the proof search tree.

Example 22. Consider the layered sequent $\mathcal{G} = \varphi$ for $\varphi = \overline{p} \vee \Diamond\Diamond(p \vee q)$. We show how to construct $A_\ell(0, \varnothing; \varphi)$ for $\ell = p$. First, we compute the proof search tree decorated with (t, Σ_c) to the left of each line, according to the algorithm, using the following abbreviations $\Gamma = \varphi, \overline{p}, \Diamond\Diamond(p \vee q)$ and $\Sigma_1 = \Diamond(p \vee q), p \vee q, p, q$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(1,\{\Diamond(p\vee q)\})\quad \Gamma,[\Sigma_1]_{\bullet 1},[\Diamond(p\vee q),p\vee q,p,q]_{\bullet \mathsf{d}},[[p\vee q,p,q]]_{\mathsf{d}}}{(1,\{\Diamond(p\vee q)\})\quad \Gamma,[\Sigma_1]_{\bullet 1},[\Diamond(p\vee q),p\vee q]_{\bullet \mathsf{d}},[[p\vee q]]_{\mathsf{d}}}\,\vee}{(0,\varnothing)\quad \Gamma,[\Diamond(p\vee q),p\vee q,p,q]_{\bullet 1}}\,\mathsf{dd}}{(0,\varnothing)\quad \Gamma,[\Diamond(p\vee q),p\vee q]_{\bullet 1}}\,\vee}{(0,\varnothing)\quad \Gamma,[\Diamond(p\vee q)]_{\bullet 1}}\,\mathsf{t}\qquad\qquad \Gamma,\Diamond\top}{(0,\varnothing)\quad \varphi,\overline{p},\Diamond\Diamond(p\vee q)}\,\mathsf{d}'_t}{(0,\varnothing)\quad \overline{p}\vee\Diamond\Diamond(p\vee q)}\,\vee$$

$\mathcal{H}=\varphi,\overline{p},\Diamond\Diamond(p\vee q),[\Diamond(p\vee q),p\vee q,p,q]_{\bullet 1},[\Diamond(p\vee q),p\vee q,p,q]_{\bullet \mathsf{d}},[[p\vee q,p,q]]_{\mathsf{d}}$ in the left leaf is a saturated sequent with $\Diamond$-formulas, crown components, $t=1$, and $\Box\Diamond\mathcal{H}_c=\{\Diamond(p\vee q)\}\subseteq\{\Diamond(p\vee q)\}=\Sigma_c$. Hence, by Step 4c,

$$A_p(1,\{\Diamond(p\vee q)\};\mathcal{H}) \quad=\quad \bullet:\overline{p}\ \ovee\ \bullet 1:q\ \ovee\ \bullet\mathsf{d}:q\ \ovee\ \mathsf{d}:q. \tag{7}$$

Applications of rule $\vee$ do not change the interpolant (Step 2, row 3). To compute $A_p(0,\varnothing;\Gamma,[\Sigma_1]_{\bullet 1})$ for the conclusion of dd, we convert (7) into an SDNF

$$\Bigl(\bullet:\overline{p}\ \owedge \bigowedge_{\sigma\in\{\bullet 1,\bullet\mathsf{d},\mathsf{d}\}}\sigma:\top\Bigr)\ \ovee \bigovee_{\tau\in\{\bullet 1,\bullet\mathsf{d},\mathsf{d}\}}\Bigl(\tau:q\ \owedge \bigowedge_{\sigma\in\{\bullet,\bullet 1,\bullet\mathsf{d},\mathsf{d}\}\setminus\{\tau\}}\sigma:\top\Bigr).$$

Now, by Step (d), and converting into a new SDNF, we get $A_p(0,\varnothing;\Gamma,[\Sigma_1]_{\bullet 1})\equiv$

$$\bigl(\bullet:(\overline{p}\wedge\Diamond\top)\ \owedge\ \bullet 1:(\top\wedge\Diamond\top)\bigr)\ \ovee\ \bigl(\bullet:(\top\wedge\Diamond\top)\ \owedge\ \bullet 1:(q\wedge\Diamond\top)\bigr)\ovee$$
$$\bigl(\bullet:(\top\wedge\Diamond q)\ \owedge\ \bullet 1:(\top\wedge\Diamond\top)\bigr)\ \ovee\ \bigl(\bullet:(\top\wedge\Diamond\top)\ \owedge\ \bullet 1:(\top\wedge\Diamond q)\bigr).$$

Further applications of $\vee$ and t keep this interpolant intact. Note that the application of d'_t does not require to continue proof search for the right branch. Instead, Step 4b prescribes that $A_p(0,\varnothing;\varphi,\overline{p},\Diamond\Diamond(p\vee q))\equiv\bigl(\bullet:\overline{p}\ \ovee\ \bullet:\Diamond\top\bigr)\owedge$

$$\Bigl(\bigl(\bullet:(\overline{p}\wedge\Diamond\top\wedge\Diamond(\top\wedge\Diamond\top))\bigr)\ \ovee\ \bigl(\bullet:(\top\wedge\Diamond\top\wedge\Diamond(q\wedge\Diamond\top))\bigr)\ \ovee$$
$$\bigl(\bullet:(\top\wedge\Diamond q\wedge\Diamond(\top\wedge\Diamond\top))\bigr)\ \ovee\ \bigl(\bullet:(\top\wedge\Diamond\top\wedge\Diamond(\top\wedge\Diamond q))\bigr)\ \ovee\ \bullet:\Box\bot\Bigr).$$

Simplifying, we finally obtain

$$A_p(0,\varnothing;\varphi)\equiv\bullet:\Bigl((\overline{p}\vee\Diamond\top)\wedge\bigl((\overline{p}\wedge\Diamond\top)\vee\Diamond q\vee\Diamond\Diamond q\vee\Box\bot\bigr)\Bigr)\equiv\bullet:(\overline{p}\vee\Diamond\Diamond q). \tag{8}$$

To check that $\overline{p}\vee\Diamond\Diamond q$ is a uniform Lyndon interpolant for φ w.r.t. literal p, it is sufficient to verify that (8) is a BLU interpolant for $\mathcal{G}$ by checking the conditions in Definition 18. We only check BLUIP(iii) as the least trivial. If $\mathcal{M},\mathcal{I}\not\models\bullet:(\overline{p}\vee\Diamond\Diamond q)$ for an interpretation $\mathcal{I}$ into a K5-model $\mathcal{M}=(W,R,V)$, then, by Definitions 14 and 11, $\mathcal{M},\rho\not\models\overline{p}\vee\Diamond\Diamond q$ for the root ρ of $\mathcal{M}$. For $\ell=p$, we have an ℓ-bisimulation $(\mathcal{M}',\mathcal{I})\leq_\ell(\mathcal{M},\mathcal{I})$ for $\mathcal{M}'=(W,R,V')$ with $V'(p)=\{\rho\}$ and $V'(r)=V(r)$ for $r\neq p$ since **literals**$_p$ allows to turn p from true to false. It is easy to see that $\mathcal{M}',\rho\not\models\overline{p}\vee\Diamond\Diamond(p\vee q)$. Thus, $\mathcal{M}',\mathcal{I}\not\models\bullet:\varphi$.

We have the following properties of the algorithm (we provide a proof in [14]).

Lemma 23. *All recursive calls $A_\ell(t, \Sigma_c; \mathcal{G})$ in a proof search tree of $A_\ell(0, \varnothing; \varphi)$ have the following properties:*

1. *The algorithm is terminating.*
2. *When Step 4b is applied, $t = 0$ and every branch going up from it consists of Steps 2–3 followed by either final Step 1 or continuation via Step 4d.*
3. *After Step 4d is applied, every branch going up from it consists of Steps 2 followed by a call $A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\Theta]_{\bullet\mathsf{d}}, [[\Phi]]_{\mathsf{d}})$ of one of the following types:*
   (a) sufficient and final when calculated via Step 1;
   (b) sufficient and propositionally saturated when calculated via Step 3, with every branch going up from there consisting of more Steps 2–3, followed by either final Step 1 or continuation via Step 4d;
   (c) insufficient and saturated when calculated via Step 4c.

Theorem 24. *Logic* K5 *has the BLUIP and, hence, the ULIP.*

Proof. It is sufficient to prove that, once the algorithm starts on $A_\ell(0, \varnothing; \varphi)$, then every sufficient call $A_\ell(t, \Sigma_c; \mathcal{G})$ in the proof search returns a BLU interpolant for a K5-sequent $\mathcal{G}$. Because the induction on the proof-search is quite technical and involves multiple cases, we demonstrate only a few representative cases and omitting simple ones, e.g., BLUIP(i), altogether. We present more cases in the Appendix of [14].

BLUIP(ii) We show that $\mathcal{M}, \mathcal{I} \vDash A_\ell(t, \Sigma_c; \mathcal{G})$ implies $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$ for any interpretation $\mathcal{I}$ of $\mathcal{G}$ into any K5-model $\mathcal{M} = (W, R, V)$. The hardest among Steps 1–3 is **Step 3 using row 5** in Table 4. Let $\mathcal{G} = \mathcal{G}', \Box\varphi$ and $\mathcal{M}, \mathcal{I} \vDash A_\ell(t, \Sigma_c; \mathcal{G}', \Box\varphi)$ for

$$A_\ell(t, \Sigma_c; \quad \mathcal{G}', \Box\varphi) \quad = \quad \bigotimes_{i=1}^{h} \left(\bullet : \Box\delta_i \otimes \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau} \right), \tag{9}$$

i.e., for each $1 \leq i \leq h$ either $\mathcal{M}, \rho \vDash \Box\delta_i$ or $\mathcal{M}, \mathcal{I}(\tau) \vDash \gamma_{i,\tau}$ for some $\tau \in \mathcal{G}$. For an arbitrary v such that $\rho R v$ and the the smallest j such that $\bullet j \notin \mathcal{G}$, clearly $\mathcal{I}_v = \mathcal{I} \sqcup \{(\bullet j, v)\}$ is an interpretation of $\mathcal{G}', \Box\varphi, [\varphi]_{\bullet j}$ into $\mathcal{M}$. Since $\mathcal{M}, \mathcal{I}_v(\bullet j) \vDash \delta_i$ whenever $\mathcal{M}, \rho \vDash \Box\delta_i$, it follows that for each $1 \leq i \leq h$ either $\mathcal{M}, \mathcal{I}_v(\bullet j) \vDash \delta_i$ or $\mathcal{M}, \mathcal{I}_v(\tau) \vDash \gamma_{i,\tau}$ for some $\tau \in \mathcal{G}$, i.e., $\mathcal{M}, \mathcal{I}_v \vDash A_\ell\big(t, \Sigma_c; \mathcal{G}', \Box\varphi, [\varphi]_{\bullet j}\big)$ for

$$A_\ell\big(t, \Sigma_c; \quad \mathcal{G}', \Box\varphi, [\varphi]_{\bullet j}\big) \quad \equiv \quad \bigotimes_{i=1}^{h} \left(\bullet j : \delta_i \otimes \bigovee_{\tau \in \mathcal{G}} \tau : \gamma_{i,\tau} \right). \tag{10}$$

By IH, $\mathcal{M}, \mathcal{I}_v \vDash \mathcal{G}', \Box\varphi, [\varphi]_{\bullet j}$ whenever $\rho R v$. If $\mathcal{M}, \rho \vDash \Box\varphi$, then $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$. Otherwise, $\mathcal{M}, \mathcal{I}_v(\bullet j) \nvDash \varphi$ for some v with $\rho R v$. For it, $\mathcal{M}, \mathcal{I}_v \vDash \mathcal{G}'$, hence, $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$.

The only other case we consider (here) is **Step 4d**. Let $\mathcal{M}, \mathcal{I} \vDash A_\ell(t, \Sigma_c; \mathcal{G})$ for $A_\ell(t, \Sigma_c; \mathcal{G})$ from (5), i.e., for some $1 \leq i \leq h$ we have $\mathcal{M}, \rho \vDash \Diamond\delta_i$, and $\mathcal{M}, \mathcal{I}(\bullet 1) \vDash \Diamond\delta'_i$, and $\mathcal{M}, \mathcal{I}(\tau) \vDash \gamma_{i,\tau}$ for all $\tau \in \mathcal{G}$. In particular, $\mathcal{M}, v \vDash \delta_i$ for some $\rho R v$ and $\mathcal{M}, u \vDash \delta'_i$ for some $\mathcal{I}(\bullet 1) R u$. Let $\mathcal{M}'$ be obtained by copying u into u' away from the root in $\mathcal{M}$ and let $\mathcal{J} = \mathcal{I} \sqcup \{(\bullet \mathsf{d}, v), (\mathsf{d}, u')\}$ be a well-defined interpretation. $\mathcal{M}', \mathcal{J} \vDash A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\{\psi \mid \Diamond\psi \in \mathcal{G}\}]_{\bullet\mathsf{d}}, [[\{\chi \mid \Diamond\chi \in \mathcal{G}_c\}]]_{\mathsf{d}})$, as (6) is true for $\mathcal{M}'$ and $\mathcal{J}$. By IH, $\mathcal{M}', \mathcal{J} \vDash \mathcal{G}, [\{\psi \mid \Diamond\psi \in \mathcal{G}\}]_{\bullet\mathsf{d}}, [[\{\chi \mid \Diamond\chi \in \mathcal{G}_c\}]]_{\mathsf{d}}$. If $\mathcal{M}', v \vDash \psi$ for some $\Diamond\psi \in \mathcal{G}$ or $\mathcal{M}', u' \vDash \chi$ for some $\Diamond\chi \in \mathcal{G}_c$, then $\mathcal{M}', \mathcal{J} \vDash \mathcal{G}$ because of $\Diamond\psi$ or $\Diamond\chi$ respectively. Otherwise, also $\mathcal{M}', \mathcal{J} \vDash \mathcal{G}$. Since we have $(\mathcal{M}, \mathcal{I}) \sim (\mathcal{M}', \mathcal{J})$ by Lemma 21, we have $\mathcal{M}, \mathcal{I} \vDash \mathcal{G}$ by Lemma 17(2) in all cases.

BLUIP(iii) We show the following statement by induction restricted to sufficient calls: if $\mathcal{M}, \mathcal{I} \nvDash A_\ell(t, \Sigma_c; \mathcal{G})$, then $\mathcal{M}', \mathcal{J}' \nvDash \mathcal{G}$ for some interpretation $\mathcal{J}'$ of $\mathcal{G}$ into another K5-model $\mathcal{M}'$ such that $(\mathcal{M}', \mathcal{J}') \leq_\ell (\mathcal{M}, \mathcal{I})$. Here we only consider **Step** 4 as the other steps are sufficiently similar to K and S5 covered in [12,13]. Among the four subcases, Step 4a is tedious but conceptually transparent. Step 4c is trivial because the induction statement is only for sufficient calls while Step 4c calls are insufficient by Lemma 23. Out of remaining two steps we only have space for **Step** 4d, which is conceptually the most interesting because its recursive call may be insufficient, precluding the use of IH for it. Let $\mathcal{M}, \mathcal{I} \nvDash A_\ell(t, \Sigma_c; \mathcal{G})$ for $A_\ell(t, \Sigma_c; \mathcal{G})$ from (5).

We first modify $\mathcal{M}$ and $\mathcal{I}$ to obtain an injective interpretation $\mathcal{I}'$ into a K5-model $\mathcal{N}' = (W', R', V')$ such that $W' \setminus \mathit{Range}(\mathcal{I}')$ is not empty and partitioned into pairs (v, u) with $\mathcal{I}'(\bullet) R v$ and not $\mathcal{I}'(\bullet) R u$. To this end we employ copying as per Definition 20, constructing a sequence of interpretations $\mathcal{I}_i$ from $\mathcal{G}$ into models $\mathcal{N}_i = (W_i, R_i, V_i)$ starting from $\mathcal{N}_0 = \mathcal{M}$ and $\mathcal{I}_0 = \mathcal{I}$ as follows:

1. If $\mathcal{I}_i(\tau_1) = \mathcal{I}_i(\tau_2)$ for $\tau_1 \neq \tau_2$, obtain $\mathcal{N}_{i+1}$ by copying $\mathcal{I}_i(\tau_2)$ to a new world w and redirect τ_2 to this new world, i.e., $\mathcal{I}_{i+1} = \mathcal{I}_i \sqcup \{(\tau_2, w)\} \setminus \{(\tau_2, \mathcal{I}_i(\tau_2))\}$.
2. If $\mathcal{I}_{K-1}$ is injective but $W_{K-1} \setminus \mathit{Range}(\mathcal{I}_{K-1}) = \varnothing$, obtain $\mathcal{N}_K$ by copying $\mathcal{I}_{K-1}(\bullet 1)$ to a new world y. Set $\mathcal{I}_K = \mathcal{I}_{K-1}$. Now $W_K \setminus \mathit{Range}(\mathcal{I}_K) \neq \varnothing$.
3. Finally, define the two sets $Y = \{y \in W_K \setminus \mathit{Range}(\mathcal{I}_K) \mid \mathcal{I}_K(\bullet) R_K y\}$ and $Z = \{z \in W_K \setminus \mathit{Range}(\mathcal{I}_K) \mid \text{not } \mathcal{I}_K(\bullet) R_K z\}$ and obtain $\mathcal{N}'$ by copying:
   - for each $y \in Y$, copy $\mathcal{I}_K(\bullet 1)$ away from the root to a new world y_2;
   - for each $z \in Z$, copy $\mathcal{I}_K(\bullet 1)$ to a new world z_1.

   Then $\mathcal{I}' = \mathcal{I}_K$ is an injective interpretation of $\mathcal{G}$ into $\mathcal{N}'$.

Note that $W' \setminus \mathit{Range}(\mathcal{I}') = Y \sqcup Z \sqcup \{y_2 \mid y \in Y\} \sqcup \{z_1 \mid z \in Z\} \neq \varnothing$. Further, $\mathcal{I}'(\bullet) R' y$ for all $y \in Y$, and not $\mathcal{I}'(\bullet) R' y_2$ for all $y \in Y$, and $\mathcal{I}'(\bullet) R' z_1$ for all $z \in Z$, and not $\mathcal{I}'(\bullet) R' z$ for all $z \in Z$. Thus, we obtain the requisite partition $P = \{(y, y_2) \mid y \in Y\} \sqcup \{(z_1, z) \mid z \in Z\} \neq \varnothing$ of the non-empty $W' \setminus \mathit{Range}(\mathcal{I}')$.

It is clear that $(\mathcal{N}', \mathcal{I}') \sim (\mathcal{M}, \mathcal{I})$. So $\mathcal{N}', \mathcal{I}' \nvDash A_\ell(t, \Sigma_c; \mathcal{G})$ by Lemma 17, i.e., for each $1 \leq i \leq h$ we have $\mathcal{N}', \rho \nvDash \Diamond\delta_i$ for $\rho = \mathcal{I}'(\bullet)$, or $\mathcal{N}', \mathcal{I}'(\bullet 1) \nvDash \Diamond\delta'_i$, or $\mathcal{N}', \mathcal{I}'(\tau) \nvDash \gamma_{i,\tau}$ for some $\tau \in \mathcal{G}$. Thus, for any $(v, u) \in P$ and each $1 \leq i \leq h$, we have $\mathcal{N}', v \nvDash \delta_i$, or $\mathcal{N}', u \nvDash \delta'_i$, or $\mathcal{N}', \mathcal{I}'(\tau) \nvDash \gamma_{i,\tau}$ for some $\tau \in \mathcal{G}$. Hence, (6) is false under injective interpretation $\mathcal{J}_{v,u} = \mathcal{I}' \sqcup \{(\bullet \mathsf{d}, v), (\mathsf{d}, u)\}$ into $\mathcal{N}'$, i.e., abbreviating $\Theta = \{\psi \mid \Diamond\psi \in \mathcal{G}\}$ and $\Phi = \{\chi \mid \Diamond\chi \in \mathcal{G}_c\}$, we get $\mathcal{N}', \mathcal{J}_{v,u} \nvDash A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\Theta]_{\bullet\mathsf{d}}, [[\Phi]]_{\mathsf{d}})$.

Ordinarily, here we would use IH, but this is only possible for sufficient calls, which, alas, is not guaranteed for (6). What is known by Lemma 23(3) is that every branch going up from (6) leads to a call of the form

$$A_\ell(1, \Box\Diamond\mathcal{G}_c; \quad \mathcal{G}, [\Theta_j]_{\bullet\mathsf{d}}, [[\Phi_j]]_{\mathsf{d}}), \tag{11}$$

where $\Theta_j \supseteq \Theta$ and $\Phi_j \supseteq \Phi$, that returns multiformula $\mho_j$ and is either sufficient or insufficient but saturated. Let Ξ denote the multiset of these multiformulas $\mho_j$ returned by all these calls. Since Step 2 is the only one used between that call and all the calls comprising (11), it is clear that (6) is their conjunction, i.e., $A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\Theta]_{\bullet\mathsf{d}}, [[\Phi]]_{\mathsf{d}}) \equiv \bigotimes_{\mho_j \in \Xi} \mho_j$. Collecting all this together, we conclude that for each pair $(v, u) \in P$ there is some $\mho_{v,u} \in \Xi$ such that

$$\mathcal{N}', \mathcal{J}_{v,u} \nvDash \mho_{v,u}. \tag{12}$$

We distinguish between two cases. First, suppose for at least one pair $(v, u) \in P$ there is a sufficient $\mho_{v,u} = A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\Theta_{v,u}]_{\bullet\mathsf{d}}, [[\Phi_{v,u}]]_{\mathsf{d}})$ satisfying (12). By IH for this $\mho_{v,u}$ there is an interpretation $\mathcal{J}_0'$ into a $\mathsf{K5}$-model $\mathcal{M}'$ such that $(\mathcal{M}', \mathcal{J}_0') \leq_\ell (\mathcal{N}', \mathcal{J}_{v,u})$ and $\mathcal{M}', \mathcal{J}_0' \nvDash \mathcal{G}, [\Theta_{v,u}]_{\bullet\mathsf{d}}, [[\Phi_{v,u}]]_{\mathsf{d}}$. Thus, $\mathcal{M}', \mathcal{J}' \nvDash \mathcal{G}$ for $\mathcal{J}' = \mathcal{J}_0' \restriction Lab(\mathcal{G})$. Finally, by restricting to labels of $\mathcal{G}$, we can see that

$$(\mathcal{M}', \mathcal{J}') \quad \leq_\ell \quad (\mathcal{N}', \mathcal{I}') \quad \sim \quad (\mathcal{M}, \mathcal{I}). \tag{13}$$

Otherwise, (12) does not hold for any pair $(v, u) \in P$ and any sufficient $\mho_{v,u} \in \Xi$. In this case, $\mathcal{N}', \mathcal{J}_{v,u} \nvDash \bigotimes_{\mho_j \in \Xi} \mho_j$ guarantees the existence of an insufficient $\mho_{v,u} \in \Xi$ for each pair $(v, u) \in P$ such that (12) holds. Since all these $\mho_{v,u}$ are insufficient, we cannot use IH. Instead, we construct $\mathcal{M}'$ and $\mathcal{J}'$ directly by changing ℓ from true to false if needed based on $\mathcal{G}$ within $Range(\mathcal{I}')$ and based on $\mho_{v,u}$'s outside of this range. Thanks to $\mathcal{I}'$ being injective, we do not need to worry about conflicting requirements from different components of $\mathcal{G}$. Similarly, P being a partition prevents conflicts outside $Range(\mathcal{I}')$. Let $\mathcal{M}' = (W', R', U')$ be $\mathcal{N}'$ with V' changed into U'. We define $V' \downarrow_\ell T$ as the valuation that makes ℓ false in all worlds from $T \subseteq W'$, i.e., $(V' \downarrow_\ell T)(q) = V'(q)$ for all $q \notin \{\ell, \overline{\ell}\}$, while

$$(V' \downarrow_\ell T)(p) = \begin{cases} V'(p) \setminus T & \text{if } \ell = p, \\ V'(p) \cup T & \text{if } \ell = \overline{p} \end{cases}$$

for $p \in \{\ell, \overline{\ell}\}$. Using this notation, we define $U' = V' \downarrow_\ell T_{\mathcal{G}}$ where

$$T_{\mathcal{G}} = \{\mathcal{I}'(\sigma) \mid \sigma : \ell \in \mathcal{G}\} \sqcup \{v \mid (v, u) \in P \text{ and } \bullet\mathsf{d} : \ell \in \mho_{v,u}\} \sqcup \\ \{u \mid (v, u) \in P \text{ and } \mathsf{d} : \ell \in \mho_{v,u}\}. \tag{14}$$

Finally, $\mathcal{J}' = \mathcal{I}'$. It is clear that (13) holds for these $\mathcal{M}'$ and $\mathcal{J}'$.

It remains to show that $\mathcal{M}', \mathcal{J}' \nvDash \mathcal{G}$. This is done by mutual induction on the construction of formula φ for the following three induction statements

$$\sigma : \varphi \in \mathcal{G} \Longrightarrow \mathcal{M}', \mathcal{I}'(\sigma) \nvDash \varphi, \tag{15}$$

$$\bullet\mathsf{d} : \varphi \in \mho_{v,u} \Longrightarrow \mathcal{M}', v \nvDash \varphi, \tag{16}$$

$$\mathsf{d} : \varphi \in \mho_{v,u} \Longrightarrow \mathcal{M}', u \nvDash \varphi. \tag{17}$$

Case $\varphi = \ell' \in \mathsf{Lit} \setminus \{\ell, \overline{\ell}\}$**.** By Lemma 23(3), all $\mho_{v,u}$ are computed by Step 4c due to their insufficiency, i.e., $\mho_{v,u} = \mathsf{LitDis}_\ell(\mathcal{G}, [\Theta_{v,u}]_{\bullet\mathsf{d}}, [[\Phi_{v,u}]]_{\mathsf{d}})$. (16) and (17) follow from (12) and (2) because $\mathcal{M}'$ agrees with $\mathcal{N}'$ on $\ell' \notin \{\ell, \overline{\ell}\}$. Similarly, since $\mathcal{J}_{v,u}$ agrees with $\mathcal{J}' = \mathcal{I}'$ on $Lab(\mathcal{G})$, (15) follows by using $\mho_{v,u}$ for any $(v,u) \in P \neq \varnothing$.

Case $\varphi = \overline{\ell}$ is analogous to the previous one. The only difference is the reason why $\mathcal{M}'$ agrees with $\mathcal{N}'$ on $\overline{\ell}$. Here, $\sigma : \overline{\ell} \in \mathcal{G}$ implies $\sigma : \ell \notin \mathcal{G}$ because $\mathcal{G}$ was processed by Step 4d not Step 1. Therefore, $\mathcal{I}'(\sigma) \notin T_{\mathcal{G}}$ by the injectivity of $\mathcal{I}'$, and $\overline{\ell}$ was not made true in $\mathcal{I}'(\sigma)$, ensuring (15). The argument for (16) and (17) is similar, except $\bullet\mathsf{d}/\mathsf{d} : \overline{\ell}$ is taken from $\mho_{v,u}$ processed by Step 4c not Step 1.

Case $\varphi = \ell$**.** All of (15)–(17) follow from (14).

Cases $\varphi = \varphi_1 \wedge \varphi_2$ **and** $\varphi = \varphi_1 \vee \varphi_2$ are standard and follow by IH due to saturation of $\mathcal{G}$ for (15) and $\mho_{v,u}$ for (16) and (17).

Case $\varphi = \Box\xi$**.** If $\sigma : \Box\xi \in \mathcal{G}$, then by saturation of $\mathcal{G}$, there is a τ such that $\tau : \xi \in \mathcal{G}$ and $\mathcal{I}'(\sigma) R' \mathcal{I}'(\tau)$: if $\sigma = \bullet$, then $\tau = \bullet j$ for some j, while if $\sigma \neq \bullet$, then $\tau \neq \bullet$. By IH(15), $\mathcal{M}', \mathcal{I}'(\tau) \nvDash \xi$, and $\mathcal{M}', \mathcal{I}'(\sigma) \nvDash \Box\xi$.

If $\bullet\mathsf{d}/\mathsf{d} : \Box\xi \in \mho_{v,u}$, then $\Box\xi \in \Box\Diamond\mathcal{G}_c$ by conditions of Step 4c due to (11), i.e., $\Box\xi \in \mathcal{G}_c$. By saturation of $\mathcal{G}$, there is a $\tau \neq \bullet$ such that $\tau : \xi \in \mathcal{G}$. Since v, u, and $\mathcal{I}'(\tau)$ are all in the cluster C of $\mathcal{M}'$, we have $vR'\mathcal{I}'(\tau)$ and $uR'\mathcal{I}'(\tau)$. It remains to use IH(16) and IH(17).

Case $\varphi = \Diamond\xi$**.** First consider $\sigma = \bullet$ and $\bullet : \Diamond\xi \in \mathcal{G}$. Since $\mathcal{I}'(\bullet) = \rho$ is the root, $\rho R' w$ implies either $w = \mathcal{I}'(\bullet j)$ for some j or $w \notin Range(\mathcal{I}')$. In the former case, $\bullet j : \xi \in \mathcal{G}$ by saturation of $\mathcal{G}$, so $\mathcal{M}', w \nvDash \xi$ by IH(15). In the latter case, $(w,u) \in P$ for some u. Recall for $A_\ell(1, \Box\Diamond\mathcal{G}_c; \mathcal{G}, [\Theta_{w,u}]_{\bullet\mathsf{d}}, [[\Phi_{w,u}]]_{\mathsf{d}})$ that we have $\Theta_{w,u} \supseteq \Theta = \{\psi \mid \Diamond\psi \in \mathcal{G}\} \ni \xi$. Hence, $\bullet\mathsf{d} : \xi \in \mho_{w,u}$ and $\mathcal{M}', w \nvDash \xi$ by IH(16). Since $\mathcal{M}', w \nvDash \xi$ for all $\mathcal{I}'(\bullet) = \rho R' w$, we conclude $\mathcal{M}', \mathcal{I}'(\bullet) \nvDash \Diamond\xi$.

If $\sigma \neq \bullet$ and $\sigma : \Diamond\xi \in \mathcal{G}$, the argument is similar. But additionally we may have $w = \mathcal{I}'(k)$ for some k or $(v,w) \in P$ for some v. In the former case, $k : \xi \in \mathcal{G}$ by saturation of $\mathcal{G}$, so $\mathcal{M}', w \nvDash \xi$ by IH(15). In the latter case, $\Phi_{v,w} \supseteq \Phi = \{\chi \mid \Diamond\chi \in \mathcal{G}_c\} \ni \xi$. Hence, $\mathsf{d} : \xi \in \mho_{v,w}$ and $\mathcal{M}', w \nvDash \xi$ by IH(17). Since $\mathcal{M}', w \nvDash \xi$ for all $\mathcal{I}'(\sigma) R' w$, we conclude $\mathcal{M}', \mathcal{I}'(\sigma) \nvDash \Diamond\xi$.

If $\bullet\mathsf{d}/\mathsf{d} : \Diamond\xi \in \mho_{v,u}$, then, similar to the analogous subcase of $\Box\xi$, conditions of Step 4c imply that $\Diamond\xi \in \mathcal{G}_c$, i.e., $\tau_0 : \Diamond\xi \in \mathcal{G}$ for some $\tau_0 \neq \bullet$. Then $\tau : \xi \in \mathcal{G}$ for all $\tau \neq \bullet$ by saturation of $\mathcal{G}$. Thus, $\mathcal{M}', \mathcal{I}'(\tau) \nvDash \xi$ for all $\tau \neq \bullet$ by IH(15). For each $y \notin Range(\mathcal{I}')$ such that $\rho R' y$, there is x such that $(y,x) \in P$ and $\bullet\mathsf{d} : \xi \in \mho_{y,x}$ because $\Theta_{y,x} \supseteq \Theta \ni \xi$. Hence, $\mathcal{M}', y \nvDash \xi$ by IH(16). Finally, for each $x \notin Range(\mathcal{I}')$ such that not $\rho R' x$, there is y such that $(y, x) \in P$ and $\mathsf{d} : \xi \in \mho_{y,x}$ because $\Phi_{y,x} \supseteq \Phi \ni \xi$. Hence, $\mathcal{M}', x \nvDash \xi$ by IH(17). We have shown that $\mathcal{M}', w \nvDash \xi$ whenever $vR'w$ $(uR'w)$. Thus, $\mathcal{M}', v \nvDash \Diamond\xi$ and $\mathcal{M}', u \nvDash \Diamond\xi$. □

5 Conclusion

We presented layered sequent calculi for several extensions of modal logic K5: namely, K5 itself, KD5, K45, KD45, KB5, and S5. By leveraging the simplicity of Kripke models for these logics, we were able to formulate these calculi in a modular way and obtain optimal complexity upper bounds for proof search. We used the calculus for K5 to obtain the first syntactic (and, hence, constructive) proof of the uniform Lyndon interpolation property for K5.

Due to the proof being technically involved, space considerations prevented us from extending the syntactic proof of ULIP to KD5, K45, KD45, KB5, and S5. For S5, layered sequents coincide with hypersequents, and we plan to upgrade the hypersequent-based syntactic proof of UIP from [11] to ULIP (see also [13]). As for KD5, K45, KD45, and KB5, the idea is to modify the method presented here for K5 by using the layered sequent calculus for the respective logic and making other necessary modifications, e.g., to rule dd, to fit the specific structure of the layers. We conjecture that the proof for K45, KD45, and KB5 would be similar to that for S5, whereas KD5 would more closely resemble K5.

Acknowledgments. Iris van der Giessen and Raheleh Jalali are grateful for the productive and exciting four-week research visit to the Embedded Computing Systems Group at TU Wien. The authors thank the anonymous reviewers for their useful comments.

References

1. Akbar Tabatabai, A., Iemhoff, R., Jalali, R.: Uniform Lyndon interpolation for basic non-normal modal and conditional logics. Eprint 2208.05202, arXiv (2022). https://doi.org/10.48550/arXiv.2208.05202
2. Akbar Tabatabai, A., Iemhoff, R., Jalali, R.: Uniform Lyndon interpolation for intuitionistic monotone modal logic. Eprint 2208.04607, arXiv (2022). https://doi.org/10.48550/arXiv.2208.04607
3. Akbar Tabatabai, A., Jalali, R.: Universal proof theory: semi-analytic rules and uniform interpolation. E-print 1808.06258, arXiv (2018). https://doi.org/10.48550/arXiv.1808.06258
4. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In: Hodges, W., Hyland, M., Steinhorn, C., Truss, J. (eds.) Logic: From Foundations to Applications: European Logic Colloquium, pp. 1–32. Clarendon Press (1996)
5. Bílková, M.: Interpolation in modal logics. Ph.D. thesis, Charles University Prague (2006). https://dspace.cuni.cz/handle/20.500.11956/15732
6. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001). https://doi.org/10.1017/CBO9781107050884

7. Brünnler, K.: Deep sequent systems for modal logic. Arch. Math. Logic **48**, 551–577 (2009). https://doi.org/10.1007/s00153-009-0137-3
8. Fitting, M., Kuznets, R.: Modal interpolation via nested sequents. Ann. Pure Appl. Logic **166**, 274–305 (2015). https://doi.org/10.1016/j.apal.2014.11.002
9. Ghilardi, S.: An algebraic theory of normal forms. Ann. Pure Appl. Logic **71**, 189–245 (1995). https://doi.org/10.1016/0168-0072(93)E0084-2
10. Ghilardi, S., Zawadowski, M.: Undefinability of propositional quantifiers in the modal system S4. Stud. Logica. **55**, 259–271 (1995). https://doi.org/10.1007/BF01061237
11. van der Giessen, I.: Uniform Interpolation and Admissible Rules. Proof-theoretic investigations into (intuitionistic) modal logics. Ph.D. thesis, Utrecht University (2022). https://doi.org/10.33540/1486
12. van der Giessen, I., Jalali, R., Kuznets, R.: Uniform interpolation via nested sequents. In: Silva, A., Wassermann, R., de Queiroz, R. (eds.) WoLLIC 2021. LNCS, vol. 13038, pp. 337–354. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88853-4_21
13. van der Giessen, I., Jalali, R., Kuznets, R.: Uniform interpolation via nested sequents and hypersequents. E-print 2105.10930, arXiv (2021). https://doi.org/10.48550/arXiv.2105.10930
14. van der Giessen, I., Jalali, R., Kuznets, R.: Extensions of K5: proof theory and uniform Lyndon interpolation. E-print 2307.11727, arXiv (2023). https://doi.org/10.48550/arXiv.2307.11727
15. Halpern, J.Y., Rêgo, L.C.: Characterizing the NP-PSPACE gap in the satisfiability problem for modal logic. J. Log. Comput. **17**, 795–806 (2007). https://doi.org/10.1093/logcom/exm029
16. Iemhoff, R.: Uniform interpolation and the existence of sequent calculi. Ann. Pure Appl. Logic **170**, 102711 (2019). https://doi.org/10.1016/j.apal.2019.05.008
17. Koopmann, P.: Practical Uniform Interpolation for Expressive Description Logics. Ph.D. thesis, University of Manchester (2015). https://www.research.manchester.ac.uk/portal/files/54574261/FULL_TEXT.PDF
18. Kurahashi, T.: Uniform Lyndon interpolation property in propositional modal logics. Arch. Math. Logic **59**, 659–678 (2020). https://doi.org/10.1007/s00153-020-00713-y
19. Kuznets, R.: Craig interpolation via hypersequents. In: Probst, D., Schuster, P. (eds.) Concepts of Proof in Mathematics, Philosophy, and Computer Science. Ontos Mathematical Logic, vol. 6, pp. 193–214. De Gruyter (2016). https://doi.org/10.1515/9781501502620-012
20. Kuznets, R.: Proving Craig and Lyndon interpolation using labelled sequent calculi. In: Michael, L., Kakas, A. (eds.) JELIA 2016. LNCS (LNAI), vol. 10021, pp. 320–335. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48758-8_21
21. Kuznets, R.: Multicomponent proof-theoretic method for proving interpolation properties. Ann. Pure Appl. Logic **169**, 1369–1418 (2018). https://doi.org/10.1016/j.apal.2018.08.007
22. Kuznets, R., Lellmann, B.: Grafting hypersequents onto nested sequents. Logic J. IGPL **24**, 375–423 (2016). https://doi.org/10.1093/jigpal/jzw005
23. 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

24. Kuznets, R., Lellmann, B.: Interpolation for intermediate logics via injective nested sequents. J. Log. Comput. **31**, 797–831 (2021). https://doi.org/10.1093/logcom/exab015
25. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: Walsh, T. (ed.) IJCAI 2011, vol. 2, pp. 989–995. AAAI Press (2011). https://www.ijcai.org/Proceedings/11/Papers/170.pdf
26. Minc, G.E.: On some calculi of modal logic. In: Orevkov, V.P. (ed.) The Calculi of Symbolic Logic. I, Proceedings of the Steklov Institute of Mathematics, vol. 98 (1968), pp. 97–124. AMS (1971)
27. Nagle, M.C.: The decidability of normal K5 logics. J. Symb. Log. **46**, 319–328 (1981). https://doi.org/10.2307/2273624
28. Nagle, M.C., Thomason, S.K.: The extensions of the modal logic K5. J. Symb. Log. **50**, 102–109 (1985). https://doi.org/10.2307/2273793
29. Pietruszczak, A., Klonowski, M., Petrukhin, Y.: Simplified Kripke-style semantics for some normal modal logics. Stud. Logica. **108**(3), 451–476 (2019). https://doi.org/10.1007/s11225-019-09849-2
30. 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
31. Pottinger, G.: Uniform, cut-free formulations of T, S_4, and S_5. J. Symb. Logic **48**, 900 (1983). https://doi.org/10.2307/2273495
32. Shavrukov, V.Y.: Subalgebras of diagonalizable algebras of theories containing arithmetic, Dissertationes Mathematicae, vol. 323. Institute of Mathematics, Polish Academy of Sciences (1993). http://matwbn.icm.edu.pl/ksiazki/rm/rm323/rm32301.pdf
33. Shvarts, G.F.: Gentzen style systems for K45 and K45D. In: Meyer, A.R., Taitslin, M.A. (eds.) Logic at Botik 1989. LNCS, vol. 363, pp. 245–256. Springer, Heidelberg (1989). https://doi.org/10.1007/3-540-51237-3_20
34. Takano, M.: A modified subformula property for the modal logics K5 and K5D. Bull. Sect. Logic **30**(2), 115–122 (2001)
35. Visser, A.: Uniform interpolation and layered bisimulation. In: Hájek, P. (ed.) Gödel 1996: Logical Foundations of Mathematics, Computer Science and Physics – Kurt Gödel's Legacy, Lecture Notes in Logic, vol. 6, pp. 139–164. ASL (1996). https://doi.org/10.1017/9781316716939.010

On Intuitionistic Diamonds (and Lack Thereof)

Anupam Das and Sonia Marin(✉)

University of Birmingham, Birmingham, UK
{a.das,s.marin}@bham.ac.uk

Abstract. A variety of intuitionistic versions of modal logic K have been proposed in the literature. An apparent misconception is that all these logics coincide on their $\Box$-only (or $\Diamond$-free) fragment, suggesting some robustness of '$\Box$-only intuitionistic modal logic'. However in this work we show that this is not true, by consideration of negative translations from classical modal logic: Fischer Servi's IK proves strictly more $\Diamond$-free theorems than Fitch's CK, and indeed iK, the minimal $\Box$-normal intuitionistic modal logic.

On the other hand we show that the smallest extension of iK by a normal $\Diamond$ is in fact conservative over iK (over $\Diamond$-free formulas). To this end, we develop a novel proof calculus based on nested sequents for intuitionistic propositional logic due to Fitting. Along the way we establish a number of new metalogical results.

Keywords: Modal logic · Intuitionistic logic · Negative translation · Proof theory · Nested sequents · Cut-elimination

1 Introduction

Usual (propositional) modal logic extends the language of classical propositional logic (CPL) by two modalities, $\Box$ and $\Diamond$, informally representing 'necessity' and 'possibility', resp. This informality is made precise by relational semantics. This semantics gives rise to the 'standard translation', allowing us to distill the normal modal logic K as a well-behaved fragment of the first-order logic (FOL).

Notably, over classical logic, $\Box$ and $\Diamond$ are De Morgan dual, just like $\forall$ and $\exists$: we have $\Diamond A = \neg\Box\neg A$. However, in light of the association with FOL, one would naturally expect an intuitionistic counterpart of modal logic not to satisfy any such reduction. The pursuit of a reasonable definition for an 'intuitionistic' modal logic goes back decades, including works such as [7–9,14] as early as the 1950s-60s, more developments [13,25,29,32] in the 1970s, and a growing interest [6,12,17,26,28,30,31,34,35] in the 1980s. See [33] or [20] for a survey.

The smallest such logic that is typically considered is iK, obtained by simply extending intuitionistic propositional logic (IPL) by the axiom k_1 and rules mp, nec from Fig. 1, but not including any axioms involving $\Diamond$, e.g. [6,36]. It

$$\begin{array}{ll} k_1 : \Box(A \to B) \to (\Box A \to \Box B) & \\ k_2 : \Box(A \to B) \to (\Diamond A \to \Diamond B) & nec\,\dfrac{A}{\Box A} \\ k_3 : \Diamond(A \vee B) \to (\Diamond A \vee \Diamond B) & \\ k_4 : (\Diamond A \to \Box B) \to \Box(A \to B) & mp\,\dfrac{A \quad A \to B}{B} \\ k_5 : \Diamond\bot \to \bot & \end{array}$$

Fig. 1. Axioms and rules for intuitionistic modal logics.

seems that Fitch [14] was the first one to propose a way to treat $\Diamond$ in an intuitionistic setting by considering a version of *CK*, extending *iK* with k_2. *CK* enjoys a rather natural proof-theoretic formulation [35] that simply adapts the sequent calculus for *K* according to the usual intuitionistic restriction: each sequent may have just one formula on the RHS. What is more, cut-elimination for this simple calculus is just a specialisation of the classical case.

IK, which includes all axioms and rules in Fig. 1, was introduced by [28] and is equivalent to the logic proposed by [31], or even to [12] in the context of intuitionistic tense logic. In [33] Simpson gives logical arguments in favour of *IK*, namely as a logic that corresponds to intuitionistic FOL along the same standard translation that lifts *K* to classical FOL. The price to pay, however, is steep: there is no known cut-free sequent calculus complete for *IK*. On the other hand, Simpson demonstrates how the relational semantics of classical modal logic may be leveraged to recover a labelled sequent calculus. The cut-elimination theorem, this time, specialises the cut-elimination theorem for intuitionistic FOL.

Contribution. An apparently widespread perception about intuitionistic modal logics is that *iK* and *IK* (and so all logics in between) coincide on their '$\Box$-only' (i.e. $\Diamond$-free) fragments. We show that this is not true by giving an explicit separation of *IK* from *iK* (also *CK*) by a $\Diamond$-free formula, and go on to initiate a comparison of the various logics by their $\Diamond$-free fragments. For the first separation, we show *IK* validates a form of Gödel-Gentzen translation from *K*, but that *CK* does not; the simplest such separation arising from this is given by $\neg\neg\Box\bot \to \Box\bot$. An important question at this point is whether it is even possible to conservatively extend *iK* by a normal $\Diamond$, i.e. is $CK + k_3 + k_5$ $\Diamond$-free conservative over *CK*? We answer this positively by designing a new system for the logic based on Fitting's nested sequents for *IPL* [16] and proving a cut-elimination result. Our results are summarised in Fig. 2.

Some of the ideas behind this work were announced and discussed on *The Proof Theory Blog* in 2022 [11] (but have not been peer-reviewed before). We shall reference that discussion further in Sect. 4.

2 Preliminaries

Let us fix a countable set of *propositional variables*, written p, q etc. When working in predicate logic, we shall simultaneously construe these as unary predicate symbols, and further fix a (infix) binary relation symbol R.

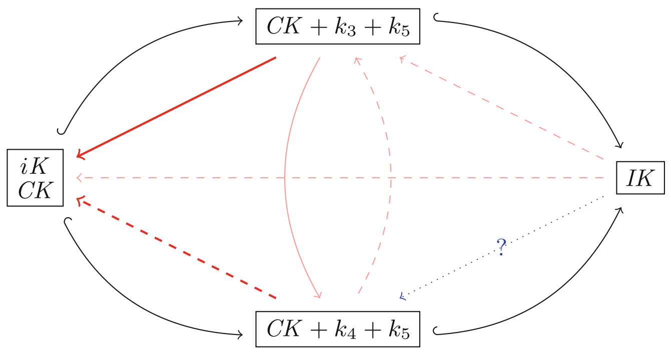


Fig. 2. Comparison of $\Diamond$-free fragments. Solid arrows denote inclusion, dashed arrows denote non-inclusion. All new results of this work are in red, where faded arrows are consequences of the non-faded ones. The dotted blue ? arrow is apparently open. (Color figure online)

Throughout this paper we shall work with *(modal propositional) formulas*, written A, B etc., generated by:

$$A \quad ::= \quad \bot \quad | \quad p \quad | \quad (A \vee B) \quad | \quad (A \wedge B) \quad | \quad (A \to B) \quad | \quad \Diamond A \quad | \quad \Box A$$

We may write $\neg A := A \to \bot$, and frequently omit brackets to aid legibility when it is unambiguous. We write, say, $A \to B \to C$ for $A \to (B \to C)$.

Due to space constraints, we shall not cover any formal semantics in this work; however it is insightful to recall how modal formulas may be viewed as a fragment of first-order predicate logic. The *standard translation* is a certain action of modal formulas on first-order variables given by a predicate formula:

Definition 1 (Standard translation). *For modal formulas A we define the predicate formula $A(x)$ by:*

$$\begin{array}{rcl} \bot(x) & := & \bot \\ p(x) & := & px \\ (A \vee B)(x) & := & A(x) \vee B(x) \\ (A \wedge B)(x) & := & A(x) \wedge B(x) \end{array} \qquad \begin{array}{rcl} (A \to B)(x) & := & A(x) \to B(x) \\ (\Diamond A)(x) & := & \exists y (xRy \wedge A(y)) \\ (\Box A)(x) & := & \forall y (xRy \to A(y)) \end{array}$$

For the reader familiar with the usual relational semantics of modal logic, note that the formula $A(x)$ simply describes the evaluation of the modal formula A at a 'world' x, within predicate logic. From this point of view we have:

Definition 2. *K is the set of modal formulas A s.t. $A(x)$ is classically valid.*

2.1 Some Axiomatisations and Characterisations

The intuitionistic modal logics we consider will always be extensions of intuitionistic propositional logic (IPL) by some of the axioms and rules in Fig. 1. Let us first point out the following well-known axiomatisation:

Proposition 3 (see, e.g., [4,5]). *The $\Diamond$-free fragment of K is axiomatised by classical propositional logic (CPL), k_1, mp and nec.*

In classical modal logic it suffices at this point to set $\Diamond A \leftrightarrow \neg\Box\neg A$ in order to recover the full axiomatisation of K, but this will not (in general) be the case for intuitionistic modal logics we are concerned with.

$$
id\ \frac{}{A \Rightarrow A} \qquad w\ \frac{\Gamma \Rightarrow A}{\Gamma, \Gamma' \Rightarrow A} \qquad \bot\text{-}l\ \frac{}{\bot \Rightarrow A} \qquad \Box\ \frac{\Gamma \Rightarrow A}{\Box\Gamma \Rightarrow \Box A} \qquad \Diamond\ \frac{\Gamma, A \Rightarrow B}{\Box\Gamma, \Diamond A \Rightarrow \Diamond B}
$$

$$
\vee\text{-}l\ \frac{\Gamma, A \Rightarrow C \quad \Gamma, B \Rightarrow C}{\Gamma, A \vee B \Rightarrow C} \qquad \wedge\text{-}l\ \frac{\Gamma, A_i \Rightarrow B}{\Gamma, A_0 \wedge A_1 \Rightarrow B} \qquad \to\text{-}l\ \frac{\Gamma \Rightarrow A \quad \Gamma, B \Rightarrow C}{\Gamma, A \to B \Rightarrow C}
$$

$$
\vee\text{-}r\ \frac{\Gamma \Rightarrow A_i}{\Gamma \Rightarrow A_0 \vee A_1} \qquad \wedge\text{-}r\ \frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B} \qquad \to\text{-}r\ \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \to B}
$$

Fig. 3. The cut-free sequent calculus LCK, obtained from the calculus for K by requiring exactly one formula on the RHS.

Definition 4. *We define the following intuitionistic modal logics:*

- *iK extends IPL by k_1 and is closed under mp and nec;*
- *CK extends IPL by k_1, k_2 and is closed under mp and nec;*
- *IK extends IPL by all the axioms k_1-k_5 and is closed under mp and nec.*

iK was studied in, e.g., [6] and [36]. The logic $CK + k_5$ was considered in [35], while the restriction to *CK* itself was given a categorical treatment in [3] and further in [23]. *IK* was first defined in [30] and [28], and investigated in details in [33]. Note that it is clear from the definitions that $iK \subseteq CK \subseteq IK$.

Since we do not work with formal semantics, we shall introduce certain proof theoretic characterisations of the logics above in order to more easily reason about (non-)provability. At the same time, these characterisations will expose some naturality underlying the logics *iK*, *CK* and *IK*.

First, let us point out that classical modal logic K has a simple sequent calculus, extending the usual propositional fragment of Gentzen's LK by the modal rules (see, e.g., [15]):

$$
\Diamond\ \frac{\Gamma, A \Rightarrow \Delta}{\Box\Gamma, \Diamond A \Rightarrow \Diamond\Delta} \qquad \Box\ \frac{\Gamma \Rightarrow \Delta, A}{\Box\Gamma \Rightarrow \Diamond\Delta, \Box A}
$$

Here Γ and Δ are sets of formulas (*cedents*) and $\Rightarrow$ is just a syntactic delimiter. A *sequent* $\Gamma \Rightarrow \Delta$ is understood logically as $\bigwedge \Gamma \to \bigvee \Delta$, its *formula translation*. Note in particular here the symmetry of the two rules, underpinned by the De Morgan duality between $\Diamond$ and $\Box$ in classical modal logic.

The characteristic property of the logic *CK* is that it is obtained from the sequent calculus for K by imposing the usual intuitionistic restriction: each sequent must have exactly one formula on the RHS. Formally, writing LCK for the (cut-free) sequent calculus given in Fig. 3, we have the well-known result:

Theorem 5 (e.g., implied by [35]). LCK *is sound and complete for* CK.

This has an entirely syntactic proof, simulating the axiomatisation of CK using a 'cut' rule and proving cut-elimination (for the completeness direction). An immediate (and well-known) consequence of this result is the following, justifying the leftmost node of Fig. 2:

Corollary 6. *CK is conservative over iK, over $\Diamond$-free formulas.*

Proof (idea). By the subformula property of LCK only $\Diamond$-free formulas appear in any proof with $\Diamond$-free conclusion. It is easily verified that any inference step whose premisses and conclusion are $\Diamond$-free are already derivable in iK.

Let us now turn to IK. One of the principal motivations behind IK is its compatibility with the standard translation, analogous to classical K:

Theorem 7 (Intuitionistic standard translation, [33]). *IK is the set of modal formulas whose standard translations are intuitionistically valid.*

This result corresponds to Simpson's 'Requirement 6' in his PhD thesis [33]. Note here the analogy to K's relationship with classical predicate logic, cf. Definition 2. The proof of the above theorem is a priori nontrivial and is beyond the scope of this work. Importantly, this result induces a proof-theoretic characterisation of IK similar to that of CK, only beginning from a different underlying calculus. Namely, IK can be obtained from the 'labelled' calculus for K (e.g. [24]) by requiring that each sequent has exactly one formula on the RHS.

Remark 8. Before closing this section it is worthwhile to mention that several other logics intermediate to CK and IK have been studied. One notable choice is Wijesekara's $CK + k_5$, sometimes called WK (e.g. in [10]). Wijesekera used a minor adaptation of LCK to allow *empty* RHS (as well as singleton), resulting in a calculus that is sound and (cut-free) complete for WK [35]. We shall return to this idea later but for now let us point out that a similar argument to Corollary 6 above indeed shows that even WK is $\Diamond$-free conservative over iK. This will be subsumed by our later result for $CK + k_3 + k_5$.

3 Separating *CK* and *IK* over the $\Diamond$-Free Fragment

In this section we shall justify the main subject matter of this work: the comparison of $\Diamond$-free fragments of intuitionistic modal logics. That such an investigation is even nontrivial is surprising: for decades now numerous papers have claimed that iK, CK, IK all coincide on their $\Diamond$-free fragments.[1] In this section we show that this is not the case.

[1] It is not the purpose of this paper to enumerate all such cases in the literature (nor do we believe it is fruitful to do so), but we point the reader to the blog post [11] for more background underlying this perception.

3.1 The Gödel-Gentzen Negative Translation

Gödel and Gentzen (independently) introduced certain double negation translations for embedding classical first-order predicate logic into its intuitionistic counterpart [18,19]. Inspired by the 'standard translation' of Definition 1, we duly adapt this translation to the language of modal logic:

Definition 9 (Gödel-Gentzen negative translation). *For each modal formula A we define another modal formula A^N as follows:*

$$\begin{aligned} \bot^N &:= \bot \\ p^N &:= \neg\neg p \\ (A \vee B)^N &:= \neg(\neg A^N \wedge \neg B^N) \\ (A \wedge B)^N &:= A^N \wedge B^N \end{aligned} \qquad \begin{aligned} (A \to B)^N &:= A^N \to B^N \\ \Diamond A^N &:= \neg\Box\neg A^N \\ \Box A^N &:= \Box A^N \end{aligned}$$

Note that the image of $\cdot^N$ is $\{\vee, \Diamond\}$-free: it is formed from only the 'negative' connectives $\bot, \wedge, \to, \Box$. For the reader familiar with the usual Gödel-Gentzen translation $\cdot^N$ on first-order predicate formulas, note that our translation above is justified by the standard translation from Definition 1: $A^N(x)$ is the same as $A(x)^N$, up to double negations in front of atomic relational formulas xRy. Nonetheless due to this slight difference, and for self-containment of the exposition, we better give the necessary characterisations explicitly.

3.2 *IK* Validates Gödel-Gentzen

Lemma 10 (Negativity). *IK proves the following:*

$$\begin{array}{ccc} \neg\neg\bot \to \bot & \neg\neg(A \wedge B) \to \neg\neg A \wedge \neg\neg B & \\ \neg\neg\neg A \to \neg A & \neg\neg(A \to B) \to \neg\neg A \to \neg\neg B & \neg\neg\Box A \to \Box\neg\neg A \end{array}$$

Proof. The non-modal cases are already theorems of *IPL*, so it remains to check the final $\Box$ case:

$$\begin{array}{ll} A \to \neg\neg A & \text{IPL} \\ \Box(A \to \neg\neg A) & \text{necessitation} \\ \Box A \to \Box\neg\neg A & \text{by } k_1 \\ \Box A \to \Diamond\neg A \to \Diamond\bot & \text{by } k_2 \\ \Box A \to \neg\Diamond\neg A & \text{by } k_5 \\ \neg\neg\Box A \to \neg\Diamond\neg A & \because \neg\Box A \to \neg\neg\neg\Box A \\ \neg\neg\Box A \to \Diamond\neg A \to \Box\bot & \text{by } \textit{ex falso quodlibet}, \bot \to \Box\bot \\ \neg\neg\Box A \to \Box\neg\neg A & \text{by } k_4 \end{array}$$

Let us point out that k_3 was not used in the argument above. We shall keep track of k_3 (non-)use during this section and state stronger results later. From here by structural induction on formulas, using the above Lemma, we have:

Lemma 11 (Double-negation elimination). *$IK \vdash \neg\neg A^N \to A^N$.*

Theorem 12. *If $K \vdash A$ then $IK \vdash A^N$.*

Proof (sketch). Referring to Proposition 3, simply take an axiomatic K proof of A and replace every formula by its image under $\cdot^N$. Any non-constructive reasoning is justified by appealing to Lemma 11 above.[2]

Let us point out that no modal reasoning was used to justify Lemma 11 and Theorem 12, further to what we used for Lemma 10. Thus it is immediate that $CK + k_4 + k_5$ also validates the Gödel-Gentzen translation:

Corollary 13. *If $K \vdash A$ then $CK + k_4 + k_5 \vdash A^N$.*

Example 14. Instantiating the $\Box$-case of the proof of Lemma 10 by $A = \bot$, and since $IPL \vdash \neg\neg\bot \to \bot$, we have that $CK + k_4 + k_5 \vdash \neg\neg\Box\bot \to \Box\bot$.

3.3 *CK* Does *not* validate Gödel-Gentzen

On the other hand, it is easy to show that CK does *not* validate the Gödel-Gentzen translation. In particular the simplest such separation is given by:

Proposition 15. *$CK \nvdash \neg\neg\Box\bot \to \Box\bot$.*

Proof. By case analysis on cut-free bottom-up proof search in LCK. The only applicable rule is $\to$ -r, requiring us to prove $\neg\neg\Box\bot \Rightarrow \Box\bot$. At this stage there are two possible choices:

- weaken $\neg\neg\Box\bot$ on the LHS: this would require us to prove $\Rightarrow \Box\bot$, which is not even classically valid.
- apply $\to$ -l on $\neg\neg\Box\bot$ on the LHS:[3] this requires us to prove $\Rightarrow \neg\Box\bot$ (the left premiss) which is, again, not even classically valid.

Recalling Lemma 10 for IK, what breaks down here for CK is the negativity of the $\Box$, i.e. $\neg\neg\Box A \to \Box\neg\neg A$. Its underivability in CK is immediate from Proposition 15 above, cf. Example 14. In particular we have:

Corollary 16. *$CK + k_4 + k_5$ (and so also IK) proves strictly more $\Diamond$-free theorems than CK (and so also iK).*

[2] Note that a common axiomatisation of CPL simply extends IPL by $\neg\neg A \to A$.

[3] Recall that $\neg A := A \to \bot$.

4 Perspectives

4.1 On Other Separations and $\Diamond$-Free Axiomatisations

Despite the separation in the preceding section, iK and CK are known to validate some other double-negation translations, see e.g. [22]. Of course none of these translations rely on negativity of the $\Box$, i.e. $\neg\neg\Box A \to \Box\neg\neg A$. Our separation was announced (but not peer-review published) in a post on *The Proof Theory Blog* in August 2022 [11]. The discussion therein covered several other separating formulas too. In particular, Alex Simpson reported such a separation $C = (\neg\Box\bot \to \Box\bot) \to \Box\bot$ privately communicated to him in 1996 by Carsten Grefe. Let us point out that this latter separation is already a consequence of Proposition 15, as even IPL already proves $C \to \neg\neg\Box\bot \to \Box\bot$: it is an instance of the IPL theorem $((\neg A \to A) \to A) \to \neg\neg A \to A$ by $A = \Box\bot$.

In the same discussion it was mentioned that the $\Diamond$-free fragment of IK was not finitely $\Diamond$-free axiomatisable. We could not find this result in the literature, nor could we easily verify it independently. While its status is beyond the scope of this work, let us make an observation:

Proposition 17. *We have:*

1. *The $\Diamond$-free fragment of $CK + k_4 + k_5$ is finitely $\Diamond$-free axiomatised.*
2. *The $\{\vee, \Diamond\}$-free fragment of IK is finitely $\{\vee, \Diamond\}$-free axiomatised and coincides with that of $CK + k_4 + k_5$.*

Proof (sketch). Replacing $\Diamond\cdot$ by $\neg\Box\neg\cdot$ and $\cdot \vee \cdot$ by $\neg(\neg\cdot \wedge \neg\cdot)$ in the axioms k_1-k_5 yields theorems of $CK + k_4 + k_5$. Both results follow from here by carrying out the same replacement everywhere in an axiomatic proof, construing the modified versions of k_1-k_5 as the underlying axiomatisation.

Note that an immediate consequence of the result above is that, if indeed the $\Diamond$-free fragment of IK is not finitely axiomatised, then it is separated from the $\Diamond$-free fragment of $CK + k_4 + k_5$, and any such separation must make crucial use of $\vee$, cf. the blue arrow in Fig. 2.

4.2 On $\Diamond$-Normality and the Problem of $CK + k_3 + k_5$

The $\Diamond$-free separation of iK and IK forces us to question some of the 'canonical' aspects of '$\Box$-only intuitionistic modal logic' iK. Above all, it is not clear whether fixing iK (or the $\Diamond$-free fragment of CK) forces, say, *ab*normality of the $\Diamond$; equivalently, does normality of the $\Diamond$, i.e. $k_3 + k_5$, force more $\Diamond$-free theorems over iK (or CK)? Let us point out that in the post [11] there was significant discussion about the status of $CK + k_3 + k_5$, with no definitive resolution about its $\Diamond$-free fragment with respect to iK, CK, IK. The remainder of this paper is devoted to a resolution of this question; namely, $CK + k_3 + k_5$ is indeed $\Diamond$-free conservative over iK, cf. Fig. 2.

Before turning to that, let us briefly discuss why the status of $CK + k_3 + k_5$ is somewhat nontrivial. Recalling Remark 8, it would be natural to further generalise the calculus LCK to a ‘multi-succedent’ version, allowing *any number* of formulas on the RHS, not just 1 (or 0 for WK). The RHS singleton restriction now only applies to the $\Box$ and $\rightarrow$ -r rules. The idea is that, while 0 formulas on the RHS corresponds to k_5, many could correspond to k_3. Indeed this seems promising in light of the following (cut-free) multi-succedent proofs of those axioms:

$$k_3 : \quad {\scriptstyle \rightarrow\text{-}r}\dfrac{{\scriptstyle \vee\text{-}r}\dfrac{{\scriptstyle \Diamond}\dfrac{{\scriptstyle IPL}\dfrac{}{A \vee B \Rightarrow A, B}}{\Diamond(A \vee B) \Rightarrow \Diamond A, \Diamond B}}{\Diamond(A \vee B) \Rightarrow \Diamond A \vee \Diamond B}}{\Rightarrow \Diamond(A \vee B) \rightarrow (\Diamond A \vee \Diamond B)} \qquad\qquad k_5 : \quad {\scriptstyle \rightarrow\text{-}r}\dfrac{{\scriptstyle \bot\text{-}r}\dfrac{{\scriptstyle \Diamond}\dfrac{{\scriptstyle \bot\text{-}l}\dfrac{}{\bot \Rightarrow}}{\Diamond\bot \Rightarrow}}{\Diamond\bot \Rightarrow \bot}}{\Rightarrow \Diamond\bot \rightarrow \bot}$$

The calculus is hence readily seen to be sound for $CK + k_3 + k_5$. However it does not enjoy cut-elimination, due to issues with commutative cases arising from the single succedent restriction on the $\Box$ rule and the $\rightarrow$ -r rule. In particular, while $CK + k_3 + k_5 \vdash \Diamond(A \vee (B \rightarrow C)) \rightarrow (\Diamond A \vee (\Box B \rightarrow \Diamond C))$, e.g. by the proof,

$$ {\scriptstyle cut}\dfrac{{\scriptstyle \Diamond}\dfrac{{\scriptstyle \vee-l}\dfrac{{\scriptstyle id}\dfrac{}{A \Rightarrow A} \qquad {\scriptstyle id}\dfrac{}{B \rightarrow C \Rightarrow B \rightarrow C}}{A \vee (B \rightarrow C) \Rightarrow A, B \rightarrow C}}{\Diamond(A \vee (B \rightarrow C)) \Rightarrow \Diamond A, \Diamond(B \rightarrow C)} \qquad {\scriptstyle \rightarrow-r}\dfrac{{\scriptstyle \Diamond}\dfrac{{\scriptstyle \rightarrow-l}\dfrac{{\scriptstyle id}\dfrac{}{B \Rightarrow B} \qquad {\scriptstyle id}\dfrac{}{C \Rightarrow C}}{B \rightarrow C, B \Rightarrow C}}{\Diamond(B \rightarrow C), \Box B \Rightarrow \Diamond C}}{\Diamond(B \rightarrow C) \Rightarrow \Box B \rightarrow \Diamond C}}{\Diamond(A \vee (B \rightarrow C)) \Rightarrow \Diamond A, \Box B \rightarrow \Diamond C}$$

note that it has no cut-free such proof, by consideration of rule applications.

5 Nested Sequent Calculus for $CK + k_3 + k_5$

In this section we will introduce a *nested sequent* calculus $\mathsf{nJ}_{\Diamond,\Box}$ for $CK + k_3 + k_5$, by extending Fitting's calculus for IPL [16] by natural modal rules. We prove a cut-elimination result for $\mathsf{nJ}_{\Diamond,\Box}$, which will imply the $\Diamond$-free conservativity of $CK + k_3 + k_5$ over CK. We shall mostly follow the notation employed by Fitting, but deviate in minor conventions to facilitate our ultimate cut-elimination result. All results are self-contained.

A *(nested) sequent*, written S etc., is an expression $\Gamma \Rightarrow X$ where Γ is a set of formulas and X is a set of formulas and nested sequents. We interpret sequents by a formula translation: $fm(\Gamma \Rightarrow \Delta, X) \quad := \quad \bigwedge \Gamma \rightarrow \left(\bigvee \Delta \vee \bigvee_{S \in X} fm(S)\right)$.

A *(nested sequent) context*, written $S[\,]$, is defined as expected. Note that it is implicit in this notation that the context hole must only occur where a nested sequent may be placed to produce a correct nested sequent, i.e., for $S[\,]$ a context and S' a nested sequent, $S[S']$ is always a nested sequent.

Example 18 (Contexts). $A \Rightarrow B, (C, D \Rightarrow E, [\,])$ is a context, but $A, [\,] \Rightarrow B, C$ and $A \Rightarrow B, (C, [\,] \Rightarrow D)$ are not.

We may also write contexts for sets (of nested sequents and formulas), e.g. $X[\,]$, etc., where again $X[S]$ must always be a correct set of nested sequents and formulas. A consequence of the definition of nested sequent is that we can safely substitute sets in place of context hole, i.e. if Y is a set of nested sequents and formulas then ($X[Y]$ and) $S[Y]$ is a (set of) nested sequent(s and formulas).

5.1 System $\mathsf{nJ}_{\Diamond,\Box}$

The system nJ is given by the structural rules and (left and right) logical rules from Fig. 4. It is equivalent to the nested calculus given by Fitting in [16], but we shall not use this fact: its soundness and completeness for *IPL* will be a consequence of later results. To define its extension by modalities, we must first generalise the usual notion of a modality distributing over a sequent:

Structural rules

$$id\ \frac{}{S[\Gamma, A \Rightarrow X[A]]} \qquad w\text{-}l\ \frac{S[\Gamma \Rightarrow X]}{S[\Gamma, A \Rightarrow X]} \qquad w\text{-}r\ \frac{S[\Gamma \Rightarrow X]}{S[\Gamma \Rightarrow X, S']}$$

$$\Rightarrow\ \frac{S[\Gamma \Rightarrow X[\Delta, \Sigma \Rightarrow Y]]}{S[\Gamma, \Delta \Rightarrow X[\Sigma \Rightarrow Y]]} \qquad \Rightarrow\text{-}e\ \frac{S[\Rightarrow X]}{S[X]}$$

Left logical rules

$$\bot\text{-}l\ \frac{}{S[\Gamma, \bot \Rightarrow X]} \qquad \vee\text{-}l\ \frac{S[\Gamma, A \Rightarrow X] \quad S[\Gamma, B \Rightarrow X]}{S[\Gamma, A \vee B \Rightarrow X]}$$

$$\wedge\text{-}l\ \frac{S[\Gamma, A, B \Rightarrow X]}{S[\Gamma, A \wedge B \Rightarrow X]} \qquad \to\text{-}l\ \frac{S[\Gamma, A \to B \Rightarrow X, A] \quad S[\Gamma, B \Rightarrow X]}{S[\Gamma, A \to B \Rightarrow X]}$$

Right logical rules

$$\vee\text{-}r\ \frac{S[\Gamma \Rightarrow X, A, B]}{S[\Gamma \Rightarrow X, A \vee B]} \qquad \wedge\text{-}r\ \frac{S[\Gamma \Rightarrow X, A] \quad S[\Gamma \Rightarrow X, B]}{S[\Gamma \Rightarrow X, A \wedge B]} \qquad \to\text{-}r\ \frac{S[\Gamma \Rightarrow X, (A \Rightarrow B)]}{S[\Gamma \Rightarrow X, A \to B]}$$

Modal rules

$$\Diamond\ \frac{S[\Gamma, A \Rightarrow X]}{S^\circ[\Box\Gamma, \Diamond A \Rightarrow X^\circ]} \qquad \Box\ \frac{S[\Gamma \Rightarrow A]}{S^\circ[\Box\Gamma \Rightarrow \Box A]}\ S \text{ is right-,-free}$$

Fig. 4. System $\mathsf{nJ}_{\Diamond,\Box}$.

Definition 19 (Promotion). *For sets X define X° by:*

$$\varnothing^\circ := \varnothing \qquad A^\circ := \Diamond A \qquad (X,Y)^\circ := X^\circ, Y^\circ \qquad (\Gamma \Rightarrow X)^\circ := \Box\Gamma \Rightarrow X^\circ$$

For (set-)contexts $X[]$, we define $X^\circ[]$ the same way and by setting $[]^\circ := []$.

Remark 20 (Promotion and $\Diamond$-normality). The intention is that X° is a consequence of $\Diamond fm(X)$. The $\varnothing$ case is justified by k_5, while the ‘,’ case is justified by k_3. The ‘$\Rightarrow$’ case is justified by the ‘Fischer Servi’ property: $\Diamond(A \to B) \to \Box A \to \Diamond B$. This is a consequence already of CK:

$$
{\scriptstyle 2\to}\,\dfrac{{\scriptstyle \Diamond}\,\dfrac{{\scriptstyle IPL}\,\overline{\overline{A \to B, A \Rightarrow B}}}{\Diamond(A \to B), \Box A \Rightarrow \Diamond B}}{\Rightarrow \Diamond(A \to B) \to \Box A \to \Diamond B}
$$

A *right-,* is a comma ‘,’ on the RHS of some $\Rightarrow$ (immediately, not hereditarily). A sequent (or context) is *right-,-free* if it has no right-,.

Definition 21. *The system* $\mathsf{nJ}_{\Diamond,\Box}$ *consists of all the rules in Fig. 4.*

Example 22. Recall the formula $\Diamond(A \vee (B \to C)) \to (\Diamond A \vee (\Box B \to \Diamond C))$ from Subsect. 4.2, which is a consequence of $CK + k_3 + k_5$ but has no cut-free proof in the ‘multi-succedent’ version of LCK. We here give a $\mathsf{nJ}_{\Diamond,\Box}$ proof of it:

$$
{\scriptstyle \to\text{-}r}\,\dfrac{{\scriptstyle \vee\text{-}r}\,\dfrac{{\scriptstyle \to\text{-}r}\,\dfrac{{\scriptstyle \Diamond}\,\dfrac{{\scriptstyle \vee\text{-}l}\,\dfrac{{\scriptstyle id}\,\dfrac{}{\Rightarrow \textcolor{red}{A} \Rightarrow \textcolor{red}{A}, (B \Rightarrow C)} \qquad {\scriptstyle \Rightarrow}\,\dfrac{{\scriptstyle \to\text{-}l}\,\dfrac{{\scriptstyle id}\,\dfrac{}{\Rightarrow\Rightarrow A, (B \to C, \textcolor{red}{B} \Rightarrow C, \textcolor{red}{B})} \quad {\scriptstyle id}\,\dfrac{}{\Rightarrow\Rightarrow A, (\textcolor{red}{C}, B \Rightarrow \textcolor{red}{C})}}{\Rightarrow\Rightarrow A, (B \to C, B \Rightarrow C)}}{\Rightarrow \textcolor{red}{B \to C} \Rightarrow A, (B \Rightarrow C)}}{\Rightarrow A \textcolor{red}{\vee} (B \to C) \Rightarrow A, (B \Rightarrow C)}}{\Rightarrow \textcolor{red}{\Diamond}(A \vee (B \to C)) \Rightarrow \Diamond A, (\Box B \Rightarrow \Diamond C)}}{\Rightarrow \Diamond(A \vee (B \to C)) \Rightarrow \Diamond A, \Box B \textcolor{red}{\to} \Diamond C}}{\Rightarrow \Diamond(A \vee (B \to C)) \Rightarrow \Diamond A \textcolor{red}{\vee} (\Box B \to \Diamond C)}}{\Rightarrow \Diamond(A \vee (B \to C)) \textcolor{red}{\to} (\Diamond A \vee (\Box B \to \Diamond C))}
$$

We have coloured red the ‘principal’ part of an inference step. Note at the top the necessity of applying the $\Rightarrow$ rule before $\to$ -l, bottom-up, in order to prove $\Rightarrow B \to C \Rightarrow A, (B \Rightarrow C)$.

The main result of this section is:

Theorem 23 (Soundness and completeness). $\mathsf{nJ}_{\Diamond,\Box} \vdash \Rightarrow A$ *if and only if* $CK + k_3 + k_5 \vdash A$.

To show the completeness (if) direction we will need to first give a simulation using a ‘cut’ rule, then prove cut-elimination. To avoid case explosion later in the presence of modal rules, it will facilitate our ultimate cut-elimination argument to consider a ‘context-joining’ cut, à la Tait. For this, we first need to generalise the usual notion of sequent union:

Definition 24 (Context joining). *For contexts $S[]$, $S'[]$ define $S[] \cdot S'[]$ by:*

- $[] \cdot S[] := S[]$;
- $(\Gamma \Rightarrow X, S[]) \cdot (\Gamma' \Rightarrow X', S'[]) := \Gamma, \Gamma' \Rightarrow X, X', (S[] \cdot S'[])$

Note that, by a basic induction on the structure of contexts, we have that $\cdot$ is associative, commutative and idempotent. We shall sometimes write simply $(S \cdot S')[]$ for $(S[] \cdot S'[])$, as abuse of notation. We shall also sometimes extend this notation to set-contexts, $X[] \cdot X'[]$, by adding the clause $(X, Y[]) \cdot (X', Y'[]) := X, X', (Y[] \cdot Y'[])$. From here the *cut* rule is defined as:

$$cut\ \frac{S[\Gamma \Rightarrow X, A] \quad S'[\Gamma', A \Rightarrow X']}{(S \cdot S')[\Gamma, \Gamma' \Rightarrow X, X']} \tag{1}$$

5.2 Metalogical Results

By induction on the structure of $\mathsf{nJ}_{\Diamond,\Box} + cut$ proofs it is routine to establish the 'only if' direction of our main result Theorem 23:

Proposition 25 (Soundness). *If $\mathsf{nJ}_{\Diamond,\Box} + cut \vdash S$ then $CK + k_3 + k_5 \vdash fm(S)$.*

The most interesting case is the $\Diamond$ rule, which is justified by Remark 20. Among the non-modal rules the most interesting cases are the 'switch' rule $\Rightarrow$ and the branching rules, which make use of the following lemma:

Lemma 26. *The following are intuitionistically valid:*

$$((A \to B) \vee C) \to (A \to (B \vee C)) \qquad (A \to (B \wedge C)) \leftrightarrow ((A \to B) \wedge (A \to C))$$
$$((A \vee B) \to C) \leftrightarrow ((A \to C) \wedge (B \to C)) \qquad (A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$$

Let us write $\Rightarrow^n$ for $\overbrace{\Rightarrow \cdots \Rightarrow}^{n}$. Note that, if S is a nested sequent, then so is $\Rightarrow^n S$, for all $n \geq 0$. We have a routine (cut-free) simulation of CK in $\mathsf{nJ}_{\Diamond,\Box}$:

Lemma 27 (Simulation of LCK). *If $\mathsf{LCK} \vdash \Gamma \Rightarrow A$ then $\mathsf{nJ}_{\Diamond,\Box} \vdash \Rightarrow^n \Gamma \Rightarrow A$ for all $n \geq 0$.*

Proof (sketch). The proof is by straightforward induction on the structure of a (cut-free) LCK proof of $\Gamma \Rightarrow A$. Almost all rules of LCK are essentially special cases of their analogues in $\mathsf{nJ}_{\Diamond,\Box}$; the only exception is the right implication rule, which is simulated as follows:[4]

$$\to\text{-}r\ \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \to B} \quad \rightsquigarrow \quad \to\text{-}r\ \frac{\Rightarrow\ \dfrac{\Rightarrow^{n+1} \Gamma, A \Rightarrow B}{\Rightarrow^n \Gamma \Rightarrow A \Rightarrow B}}{\Rightarrow^n \Gamma \Rightarrow A \to B}$$

[4] Note here the necessity of proving the statement for all $n \geq 0$ as inductive invariant.

Proposition 28 (Cut-completeness with cut). *If $CK + k_3 + k_5 \vdash A$ then $\mathsf{nJ}_{\Diamond,\Box} + cut \vdash \Rightarrow A$.*

Proof (sketch). By induction on an axiomatic $CK + k_3 + k_5$ proof of A. In light of Lemma 27 above, and the presence of *cut*, it suffices to prove k_3 and k_5:

$$\mathord{\to}\text{-}r\,\dfrac{\vee\text{-}r\,\dfrac{\Diamond\,\dfrac{\vee\text{-}l\,\dfrac{\mathit{id}\,\dfrac{}{\Rightarrow A \Rightarrow A, B} \qquad \mathit{id}\,\dfrac{}{\Rightarrow B \Rightarrow A, B}}{\Rightarrow A \vee B \Rightarrow A, B}}{\Rightarrow \Diamond(A \vee B) \Rightarrow \Diamond A, \Diamond B}}{\Rightarrow \Diamond(A \vee B) \Rightarrow \Diamond A \vee \Diamond B}}{\Rightarrow \Diamond(A \vee B) \to (\Diamond A \vee \Diamond B)} \qquad\qquad \mathord{\to}\text{-}r\,\dfrac{w\text{-}r\,\dfrac{\Diamond\,\dfrac{\bot\text{-}l\,\dfrac{}{\Rightarrow \bot \Rightarrow}}{\Rightarrow \Diamond\bot \Rightarrow}}{\Rightarrow \Diamond\bot \Rightarrow \bot}}{\Rightarrow \Diamond\bot \to \bot}$$

6 Cut-Elimination Argument

The goal of this section is to prove:

Theorem 29 (Cut-elimination). *If $\mathsf{nJ}_{\Diamond,\Box} + cut \vdash S$ then also $\mathsf{nJ}_{\Diamond,\Box} \vdash S$.*

From here note that our main result follows immediately:

Proof (of Theorem 23). Immediate from Theorem 29 above, Soundness (Proposition 25) and Completeness with cut (Proposition 28).

The *size* of a proof is its number of inference steps. The *degree* of a cut is the number of symbols in its cut-formula, i.e. the formula A distinguished in (1). Our ultimate argument for cut-elimination is based on a typical double induction:

Proof (of Theorem 29, sketch). We proceed by induction on the multiset of cut-degrees in a proof. We start with a(ny) topmost cut, employing a subinduction on the size of the subproof rooting it, permuting the cut upwards in order to apply the subinductive hypothesis. At key cases the multiset of cut-degrees will decrease and we instead apply the main inductive hypothesis on the entire proof; sometimes we may need to first apply the subinductive hypothesis. In terms of the permutation strategy, we always permute cuts over non-modal rules (on either side) maximally, so that our modal cut-reductions only apply when the inference step immediately above each side of a cut is modal.

The next subsection is devoted to describing some of the cut-reductions. Before that let us give the desired consequence of cut-elimination for $\mathsf{nJ}_{\Diamond,\Box}$, namely the classification of the $\Diamond$-free fragment of $CK + k_3 + k_5$, cf. Fig. 2:

Corollary 30. *$CK + k_3 + k_5$ is conservative over iK, over $\Diamond$-free formulas.*

Proof (sketch). If $CK + k_3 + k_5$ proves a $\Diamond$-free formula A, then there is a $\mathsf{nJ}_{\Diamond,\Box}$ proof P of $\Rightarrow A$ by Theorem 23. By the subformula property, P must be $\Diamond$-free itself, so the only modal rule occurring in P is the $\Box$-rule, whose formula translation is derivable already in iK. (Note that the formula translation of $\Diamond$-free nested sequents is always $\Diamond$-free). All other rules are derivable already in *IPL*.

6.1 Cut-Reduction Cases (Non-modal)

To facilitate the description of the cut-reduction cases we will need to 'bootstrap' $\mathsf{nJ}_{\Diamond,\Box}$ somewhat. We say a rule r is *size-preserving admissible* for a system L if, whenever there is a proof in $\mathsf{L}+r$ of S, there is a proof in L of S of the same or smaller size.

Proposition 31. *The following rules are size-preserving admissible for* $\mathsf{nJ}_{\Diamond,\Box}$*:*

$$,\frac{S[R[X],Y]}{S[R[X,Y]]} \qquad (2) \qquad\qquad \Rightarrow\text{-}i\,\frac{S[X]}{S[\Rightarrow X]} \qquad (3)$$

Thanks to the way we have presented our rules, almost all cut-reduction cases are 'the same' as those for usual sequent calculi for intuitionistic and/or modal logic, only under a sequent context. We highlight here some cases that need special attention.

For key cases, when the cut-formula is principal for a logical rule on both sides of a cut, the corresponding reduction is almost always the same as that for the usual (multi-succedent) sequent calculus for *IPL*, only under a sequent context. The only exception is for $\to$, since its right-introduction rule is different from that of the sequent calculus. The key case for $\to$ is:

$$cut\,\dfrac{\to\text{-}r\,\dfrac{S[\Gamma \Rightarrow X,(A\Rightarrow B)]}{S[\Gamma\Rightarrow X, A\to B]} \qquad \to\text{-}l\,\dfrac{S'[\Gamma',A\to B\Rightarrow X',A] \quad S'[\Gamma',B\Rightarrow X']}{S'[\Gamma',A\to B\Rightarrow X']}}{(S\cdot S')[\Gamma,\Gamma'\Rightarrow X,X']} \qquad \rightsquigarrow$$

$$cut\,\dfrac{cut\,\dfrac{cut\,\dfrac{\Rightarrow\text{-}l\,\dfrac{S[\Gamma\Rightarrow X,(A\Rightarrow B)]}{S[\Gamma\Rightarrow X,A\to B]} \quad S'[\Gamma',A\to B\Rightarrow X',A]}{(S\cdot S')[\Gamma,\Gamma'\Rightarrow X,X',A]} \qquad \Rightarrow\text{-}e\,\dfrac{\Rightarrow\,\dfrac{S[\Gamma\Rightarrow X,(A\Rightarrow B)]}{S[\Gamma,A\Rightarrow X,(\Rightarrow B)]}}{S[\Gamma,A\Rightarrow X,B]}}{(S\cdot S')[\Gamma,\Gamma'\Rightarrow X,X',B]} \qquad S'[\Gamma',B\Rightarrow X']}{(S\cdot S')[\Gamma,\Gamma'\Rightarrow X,X']}$$

Referring to our cut-elimination argument, note we must apply the subinductive hypothesis to the topmost cut before calling the main inductive hypothesis.

Any cut immediately preceded by an identity step (on either side) can be reduced to an identity step, eliminating the cut. Also all commutations of a cut above a logical rule are routine, as the $\Rightarrow$-depth of the cut-formula is not affected.

Almost all permutations when a cut is preceded by a structural step are routine. The only exception is a permutation over a $\Rightarrow$ step. Before we can present this we need to set up some notation. First, let us write $\Rightarrow^{X[\,]}$ for $\Rightarrow^{d}$ where d is the $\Rightarrow$-depth of the hole $[\,]$ in $X[\,]$. I.e.,

$$\begin{aligned} \Rightarrow^{[\,]} &:= \\ \Rightarrow^{X,S[\,]} &:= \Rightarrow^{S[\,]} \\ \Rightarrow^{\Gamma\Rightarrow X[\,]} &:= \Rightarrow\Rightarrow^{X[\,]} \end{aligned}$$

We shall sometimes write $\Rightarrow^X$ for $\Rightarrow^{X[\,]}$, as abuse of notation. By a straightforward induction on the structure of set-contexts we have that $\Rightarrow^X [\,] \cdot X[\,] = X[\,]$. Now we can give the critical $\Rightarrow$-permutation by:

$$\dfrac{S[\Gamma \Rightarrow X, A] \qquad \dfrac{S'[\Gamma' \Rightarrow X'[\Delta, A, \Sigma \Rightarrow Y]]}{S'[\Gamma', \Delta, A \Rightarrow X'[\Sigma \Rightarrow Y]]}\Rightarrow}{(S \cdot S')[\Gamma, \Gamma', \Delta \Rightarrow X, X'[\Sigma \Rightarrow Y]]}\mathit{cut}$$

$$\rightsquigarrow \quad \dfrac{\dfrac{\dfrac{S[\Gamma \Rightarrow X, A]}{S[\Gamma \Rightarrow X, (\Rightarrow^{X'} A)]}\Rightarrow\text{-}i^* \qquad S'[\Gamma' \Rightarrow X'[\Delta, A, \Sigma \Rightarrow Y]]}{(S \cdot S')[\Gamma, \Gamma' \Rightarrow X, X'[\Delta, \Sigma \Rightarrow Y]]}\mathit{cut}}{(S \cdot S')[\Gamma, \Gamma', \Delta \Rightarrow X, X'[\Sigma \Rightarrow Y]]}\Rightarrow$$

Note the importance here of size-preserving admissibility of $\Rightarrow$ -i, Proposition 31, in order to appeal to the subinductive hypothesis.

6.2 Cut-Reduction Cases (Modal)

Defining the modal cut-reductions is facilitated by the observation that $(S_0^\circ \cdot S_1^\circ)[] = (S_0 \cdot S_1)^\circ[]$, proved again by a straightforward induction on the structure of sequent-contexts. The case analysis for modal cut-reductions is routine but lengthy; all reductions allow immediate appeal to the (sub)inductive hypothesis:

– ($\Diamond$-$\Diamond$) If a cut is preceded on both sides by a $\Diamond$ step, then the cut-formula on the right must be the distinguished $\Diamond$-formula of the $\Diamond$ rule in Fig. 4. We employ a case analysis on the relative location of the distinguished $\Diamond$ formula and the cut formula on the left, but each situation is handled similarly. If, e.g., the distinguished $\Diamond$ formula and cut formula occur in parallel in the sequent context we have the following reduction:

$$\dfrac{\dfrac{S_0[\Gamma, A \Rightarrow X_0][\Delta_0 \Rightarrow Y_0, B]}{S_0^\circ[\Box\Gamma, \Diamond A \Rightarrow X_0^\circ][\Box\Delta_0 \Rightarrow Y_0^\circ, \Diamond B]}\Diamond \qquad \dfrac{S_1[X_1][\Delta_1, B \Rightarrow Y_1]}{S_1^\circ[X_1^\circ][\Box\Delta_1, \Diamond B \Rightarrow Y_1^\circ]}\Diamond}{(S_0^\circ \cdot S_1^\circ)[(\Box\Gamma, \Diamond A \Rightarrow X_0^\circ), X_1^\circ][\Box\Delta_0, \Box\Delta_1 \Rightarrow Y_0^\circ, Y_1^\circ]}\mathit{cut}$$

$$\rightsquigarrow \quad \dfrac{\dfrac{S_0[\Gamma, A \Rightarrow X_0][\Delta_0 \Rightarrow Y_0, B] \qquad S_1[X_1][\Delta_1, B \Rightarrow Y_1]}{(S_0 \cdot S_1)[(\Gamma, A \Rightarrow X_0), X_1][\Delta_0, \Delta_1 \Rightarrow Y_0, Y_1]}\mathit{cut}}{(S_0^\circ \cdot S_1^\circ)[(\Box\Gamma, \Diamond A \Rightarrow X_0^\circ), X_1^\circ][\Box\Delta_0, \Box\Delta_1 \Rightarrow Y_0^\circ, Y_1^\circ]}\Diamond$$

– ($\Diamond$-$\Box$) It is not possible for a cut to be preceded by a $\Diamond$ step on the left and a $\Box$ step on the right, since the former has only $\Diamond$ formulas in positive positions and the latter has only $\Box$ formulas in negative positions.
– ($\Box$-$\Diamond$) If a cut is preceded by a $\Box$ rule on the left and a $\Diamond$ rule on the right then the cut-formula must be a $\Box$ formula, and so cannot be the distinguished $\Diamond$ formula of the $\Diamond$ step. We again employ a case analysis on the relative location of the distinguished $\Diamond$ formula and cut formula on the right, but

each situation is handled similarly. If, e.g., the distinguished $\Diamond$ formula occurs (relatively) deeper than the cut formula, we have the following reduction:

$$\dfrac{\Box\,\dfrac{S_0[\Gamma_0 \Rightarrow A]}{S_0^\circ[\Box\Gamma_0 \Rightarrow \Box A]} \qquad \Diamond\,\dfrac{S_1[\Gamma_1, A \Rightarrow X[\Delta, B \Rightarrow Y]]}{S_1^\circ[\Box\Gamma_1, \Box A \Rightarrow X^\circ[\Box\Delta, \Diamond B \Rightarrow Y^\circ]]}}{(S_0^\circ \cdot S_1^\circ)[\Box\Gamma_0, \Box\Gamma_1 \Rightarrow X^\circ[\Box\Delta, \Diamond B \Rightarrow Y^\circ]]}\,cut$$

$$\rightsquigarrow \quad \Diamond\,\dfrac{cut\,\dfrac{S_0[\Gamma_0 \Rightarrow A] \quad S_1[\Gamma_1, A \Rightarrow X[\Delta, B \Rightarrow Y]]}{(S_0 \cdot S_1)[\Gamma_0, \Gamma_1 \Rightarrow X[\Delta, B \Rightarrow Y]]}}{(S_0 \cdot S_1)^\circ[\Box\Gamma_0, \Box\Gamma_1 \Rightarrow X^\circ[\Box\Delta, \Diamond B \Rightarrow Y^\circ]]}$$

– ($\Box$-$\Box$) If a cut is preceded on both sides by a $\Box$ rule, then the only possible reduction, due to right-,-freeness in the right premiss, is:

$$\dfrac{\Box\,\dfrac{S_0[\Gamma_0 \Rightarrow A]}{S_0^\circ[\Box\Gamma_0 \Rightarrow \Box A]} \qquad \Box\,\dfrac{S_1[A, \Gamma_1 \Rightarrow R[\Delta \Rightarrow B]]}{S_1^\circ[\Box A, \Box\Gamma_1 \Rightarrow R^\circ[\Box\Delta \Rightarrow \Box B]]}}{(S_0^\circ \cdot S_1^\circ)[\Box\Gamma_0, \Box\Gamma_1 \Rightarrow R^\circ[\Box\Delta \Rightarrow \Box B]]}\,cut$$

$$\rightsquigarrow \quad \Box\,\dfrac{cut\,\dfrac{S_0[\Gamma_0 \Rightarrow A] \quad S_1[A, \Gamma_1 \Rightarrow R[\Delta \Rightarrow B]]}{(S_0 \cdot S_1)[\Gamma_0, \Gamma_1 \Rightarrow R[\Delta \Rightarrow B]]}}{(S_0 \cdot S_1)^\circ[\Box\Gamma_0, \Box\Gamma_1 \Rightarrow R^\circ[\Box\Delta \Rightarrow \Box B]]}$$

7 Conclusions

We showed that iK and CK are separated from IK by their $\Diamond$-free theorems, and have moreover initiated a comparison of intuitionistic modal logics by their $\Diamond$-free fragments. In particular, we have verified using proof theoretic techniques that the extension of iK by a normal $\Diamond$ is indeed conservative over iK, over $\Diamond$-free formulas. Again, our results are summarised in Fig. 2.

Our nested sequent system $\mathsf{nJ}_{\Diamond,\Box}$ is based on Fitting's for IPL in [16], but let us point out that he did not give a cut-elimination result. Naturally our cut-elimination result Theorem 29 also implies cut-elimination for the nested calculus nJ for IPL. Let us emphasise that, just as iK, CK, IK are proof-theoretically natural by the characterisations in Subsect. 2.1, so too is $CK + k_3 + k_5$: it is just the extension of the calculus nJ for IPL by modal rules.

From here it would be fruitful to understand how to adequately extend (birelational) semantics for CK to $CK + k_3 + k_5$. This could also yield an alternative (and perhaps simpler) proof of completeness of $\mathsf{nJ}_{\Diamond,\Box}$ for $CK + k_3 + k_5$.[5] We have also not addressed the decidability of logics in this work, but let us point out that we believe that $CK + k_3 + k_5$ might be proved decidable by eliminating $\Rightarrow$-e in $\mathsf{nJ}_{\Diamond,\Box}$ and employing a basic loop checking argument.

There has been significant work on computational interpretations of CK e.g. [1–3,21,27]. However, one shortfall of CK here is that its interpretations

[5] We are aware of ongoing work by Nicola Olivetti and Han Gao investigating this.

do not lift to K along the Gödel-Gentzen translation; while alternative double-negation translations are available, cf. [22], these do not seem robust against modest extensions, e.g. when including a global modality $\Box^*$. On the other hand the fact that IK validates Gödel-Gentzen, Theorem 12, suggests that it is better designed for computational interpretations, in particular for interpreting classical modal logic K. Under the standard translation, it would be interesting to classify the Curry-Howard interpretation of IK as a suitable fragment of *dependent type theory*. Let us point out that Simpson already gives a termination and confluence proof for a version of intuitionistic natural deduction specialised to IK in his thesis [33].

Acknowledgements. The authors would like to thank *The Proof Theory Blog* community for all the feedback from their post [11]. In particular this work would not have been possible without several insightful interactions with Alex Simpson, Reuben Rowe, Nicola Olivetti, Tiziano Dalmonte, Dale Miller, Dominik Kirst, Iris van der Giessen, and Marianna Girlando. We thank Nicola Olivetti in particular for encouraging us to publish these results.

This (alphabetically) first author was supported by a UKRI Future Leaders Fellowship, 'Structure vs Invariants in Proofs', project reference MR/S035540/1.

References

1. Acclavio, M., Catta, D., Straßburger, L.: Game semantics for constructive modal logic. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 428–445. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_25
2. Arisaka, R., Das, A., Straßburger, L.: On nested sequents for constructive modal logic. Log. Methods Comput. Sci. (2015)
3. Bellin, G., De Paiva, V., Ritter, E.: Extended curry-howard correspondence for a basic constructive modal logic. In: Proceedings of Methods for Modalities, vol. 2 (2001)
4. Blackburn, P., van Benthem, J.F., Wolter, F.: Handbook of Modal Logic. Elsevier, Amsterdam (2006)
5. Blackburn, P., De Rijke, M., Venema, Y.: Modal Logic, vol. 53. Cambridge University Press, Cambridge (2001)
6. Božić, M., Došen, K.: Models for normal intuitionistic modal logics. Stud. Logica **43**(3), 217–245 (1984)
7. Bull, R.A.: A modal extension of intuitionist logic. Notre Dame J. Form. Log. **6**(2), 142–146 (1965). https://doi.org/10.1305/ndjfl/1093958154
8. Bull, R.A.: MIPC as the formalisation of an intuitionist concept of modality. J. Symb. Log. **31**(4), 609–616 (1966)
9. Curry, H.B.: The elimination theorem when modality is present1. J. Symb. Log. **17**(4), 249–265 (1952)
10. Dalmonte, T.: Wijesekera-style constructive modal logics. In: Fernández-Duque, D., Palmigiano, A., Pinchinat, S. (eds.) Advances in Modal Logic, AiML 2022, Rennes, France, 22–25 August 2022, pp. 281–304. College Publications (2022)
11. Das, A., Marin, S.: Brouwer meets Kripke: constructivising modal logic (2022). Post on The Proof Theory Blog. https://prooftheory.blog/2022/08/19/brouwer-meets-kripke-constructivising-modal-logic/. Accessed 24 May 2023

12. Ewald, W.B.: Intuitionistic tense and modal logic. J. Symb. Log. **51**(1), 166–179 (1986)
13. Fischer-Servi, G.: On modal logic with an intuitionistic base. Stud. Logica **36**, 141–149 (1977). https://doi.org/10.1007/bf02121259
14. Fitch, F.B.: Intuitionistic modal logic with quantifiers. Port. Math. **7**(2), 113–118 (1948)
15. Fitting, M.: Modal proof theory. Handbook of Modal Logic, pp. 85–136 (2006)
16. Fitting, M.: Nested sequents for intuitionistic logics. Notre Dame J. Form. Log. **55**(1) (2014)
17. Font, J.M.: Modality and possibility in some intuitionistic modal logics. Notre Dame J. Form. Log. **27**(4), 533–546 (1986)
18. Gentzen, G.: Die Widerspruchsfreiheit der reinen Zahlentheorie. Math. Ann. **112**(1), 493–565 (1936)
19. Gödel, K.: Zur intuitionistischen Arithmetik und Zahlentheorie. Ergebnisse eines mathematischen Kolloquiums **4**, 34–38 (1933)
20. Kavvos, G.A.: The many worlds of modal λ-calculi: I. curry-howard for necessity, possibility and time. CoRR abs/1605.08106 (2016). http://arxiv.org/abs/1605.08106
21. Kavvos, G.A.: Dual-context calculi for modal logic. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, 20–23 June 2017, pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005089
22. Litak, T., Polzer, M., Rabenstein, U.: Negative translations and normal modality. In: Miller, D. (ed.) 2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017). Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, vol. 84, pp. 27:1–27:18. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017). https://doi.org/10.4230/LIPIcs.FSCD.2017.27. http://drops.dagstuhl.de/opus/volltexte/2017/7741
23. Mendler, M., de Paiva, V.: Constructive CK for contexts. In: Context Representation and Reasoning (CRR-2005), vol. 13 (2005)
24. Negri, S.: Proof analysis in modal logic. J. Philos. Log. **34**, 507–544 (2005)
25. Ono, H.: On some intuitionistic modal logics. Publ. Res. Inst. Math. Sci. **13**(3), 687–722 (1977)
26. Ono, H., Suzuki, N.Y.: Relations between intuitionistic modal logics and intermediate predicate logics. Rep. Math. Logic **22**, 65–87 (1988)
27. Pfenning, F., Davies, R.: A judgmental reconstruction of modal logic. Math. Struct. Comput. Sci. **11**(4), 511–540 (2001). Notes to an invited talk at the Workshop on Intuitionistic Modal Logics and Applications (IMLA'99)
28. Plotkin, G., Stirling, C.: A framework for intuitionistic modal logics. In: Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK), pp. 399–406 (1986)
29. Satre, T.W.: Natural deduction rules for modal logics. Notre Dame J. Form. Log. **13**(4), 461–475 (1972)
30. Servi, G.F.: Semantics for a class of intuitionistic modal calculi. In: Dalla Chiara, M.L. (ed.) Italian Studies in the Philosophy of Science. Boston Studies in the Philosophy of Science, vol. 47, pp. 59–72. Springer, Dordrecht (1980). https://doi.org/10.1007/978-94-009-8937-5_5
31. Servi, G.F.: Axiomatizations for some intuitionistic modal logics. Rendiconti del Seminario Matematico dell' Università Politecnica di Torino **42**(3), 179–194 (1984)
32. Siemens, D.F.: Fitch-style rules for many modal logics. Notre Dame J. Form. Log. **18**(4), 631–636 (1977). https://doi.org/10.1305/ndjfl/1093888133

33. Simpson, A.: The proof theory and semantics of intuitionistic modal logic. Ph.D. thesis, University of Edinburgh (1994)
34. Suzuki, N.Y.: An algebraic approach to intuitionistic modal logics in connection with intermediate predicate logics. Stud. Logica **48**(2), 141–155 (1989)
35. Wijesekera, D.: Constructive modal logics I. Ann. Pure Appl. Logic **50**(3), 271–301 (1990)
36. Wolter, F., Zakharyaschev, M.: On the relation between intuitionistic and classical modal logics. Algebra Logic **36**, 73–92 (1997). https://doi.org/10.1007/BF02672476

CoNP Complexity for Combinations of Non-normal Modal Logics

Tiziano Dalmonte[1](✉) and Andrea Mazzullo[2]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
tiziano.dalmonte@unibz.it
[2] University of Trento, Trento, Italy
andrea.mazzullo@unitn.it

Abstract. We study the complexity of the validity/derivability problem for combinations of non-normal modal logics in the form of logic fusions, possibly extended with simple interaction axioms. We first present cut-free sequent calculi for these logic combinations. Then, we introduce hypersequent calculi with invertible rules, and show that they allow for a coNP proof search procedure. In the last part of the paper, we consider the case of combinations of logics sharing a universal modality. Using the hypersequent calculi, we show that these logics remain coNP-complete, and also provide an equivalent axiomatisation for them.

Keywords: Non-normal modal logics · Combination of logics · Fusion · Universal modality · Complexity · Hypersequent calculus

1 Introduction

Modal logics that combine different modalities have widespread diffusion. On the one hand, modal logics designed for applications usually contain multiple operators, possibly with interactions among them. On the other hand, non-standard modal logics, such as intuitionistic or description modal logics, have been connected with classical logics with combined modalities [18,19,46,47], an observation that allowed for a fruitful transfer of results among the different formalisms.

Concerning logics designed for applications, several systems contain modalities that display a non-normal behaviour, as they do not satisfy some principles that are validated by any normal operator. Significant examples are epistemic logics without omniscience [4], deontic logics [1], agency and ability logics [6,14,26], coalition logics [37,43]. At the same time, the recent introduction of non-normal systems based on intuitionistic or description logic [9,10,12,40,41] naturally raises the question of their connections with classical systems with combined non-normal modalities.

Multimodal logics obtained as combinations of normal systems have been extensively studied, with a specific focus on fusions and products [19,20,45], and the transfer of properties from the single systems to their combinations.

Concerning fusions of normal logics, it is known for instance that decidability, interpolation [45] and semantic completeness [17,30] are always preserved, whereas the complexity of the satisfiability/validity problem is not: while fusions of PSpace logics generally remain PSpace, the same does not hold for fusions of systems with coNP validity (respectively, NP satisfiability) problem, as witnessed by the PSpace bimodal logics $\mathsf{S5}_2$, $\mathsf{KD45}_2$, $\mathsf{K4.3}_2$ and $\mathsf{S4.3}_2$ [25,42], in contrast with their coNP monomodal counterparts[1] (see [19] for an overview on transfer results).

Although most studies focus on combinations of normal modal logics, similar questions have been also addressed for fusions of non-normal systems. In particular, decidability [3,23] and superamalgamation [21,22] (an algebraic property corresponding to a form of interpolation) are known to be preserved, while completeness is not [15,16]. By contrast, less is understood about the transfer of complexity results, which is the topic of the present work.

Non-normal modal logics (NNMLs in the following) are good examples of coNP modal logics. These logics are defined by extending classical propositional logic with the congruence rule $A \leftrightarrow B / \Box A \leftrightarrow \Box B$ and combinations of standard modal axioms (cf. Sec. 2). As shown by Vardi [44], in this family of logics, the complexity of the validity problem strictly depends on the presence of the agglomeration axiom $\Box A \wedge \Box B \to \Box(A \wedge B)$: the logics with this axiom are in PSpace, whereas the logics without it are coNP-complete.[2] Differently from the coNP normal systems mentioned above, the same complexity bounds hold for the multi-modal formulations of these logics where all modalities are of the *same* kind [44]. For this reason, combinations of NNMLs are promising in terms of preservation of coNP complexity.

In this paper, we investigate the complexity of the validity problem for some kinds of combinations of coNP NNMLs. In particular, we consider all coNP NNMLs of the classical cube [7,34] as well as their coNP extensions with non-iterative modal axioms. We first consider the fusions of NNMLs, roughly corresponding to the disjoint union of the modal axiomatisations of the combined systems, as well as their extensions with interaction axioms of the form $\Box_i A \to \Box_j A$ (that correspond, for instance, to the well-known principles of 'ought implies can' and 'does implies can' of deontic and agency logics (see e.g. [1,6,14])). In the last part of the paper we also consider the case of combinations of NNMLs sharing a universal modality. While most studies on property transfers are based on algebraic or model-theoretical techniques, we adopt here a proof-theoretical approach. We first present cut-free sequent calculi for these logic combinations. Then we present a reformulation of the calculi in terms of hypersequents where

[1] In the following, when mentioning the complexity of a logic, we always refer to the complexity of its *validity* problem. Dual results immediately follow for the corresponding *satisfiability* problem: in particular, coNP-complete logics have an NP-complete satisfiability problem. If not differently specified, the complexity bounds are tight: by coNP logic, respectively PSpace logic, we mean that the logic is coNP-complete, respectively PSpace-complete.

[2] More precisely, Vardi [44] shows that the satisfiability problems for these logics are NP-complete.

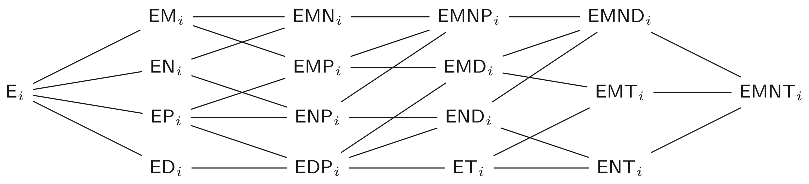


Fig. 1. Diagram of non-normal monomodal logics.

all the rules are invertible, and show that they provide a coNP decision procedure for validity in the logics. In the last part of the paper, we consider the case of combinations of logics sharing a universal modality. Using the hypersequent calculi, we show that these logics remain coNP-complete.

2 Non-normal Modal Logics and Their Combinations

Given a set of unary modalities $\{\Box_1, ..., \Box_n\}$, we denote $\mathcal{L}[\Box_1, ..., \Box_n]$ the propositional modal language based on a set $Atm = \{p_1, p_2, p_3, ...\}$ of countably many propositional variables, containing the Boolean operators $\bot$, $\rightarrow$, and the modalities $\Box_1, ..., \Box_n$. We consider $\top, \neg, \wedge, \vee, \Diamond_i$ to be defined as usual.

Non-normal monomodal logics are defined in a language $\mathcal{L}[\Box_i]$, for some $i \in \mathbb{N}$, by extending any axiomatisation of classical propositional logic (containing modus ponens), formulated in $\mathcal{L}[\Box_i]$, with the rule RE_i below, and a combination of the following axioms:

$$RE_i \ \frac{A \leftrightarrow B}{\Box_i A \leftrightarrow \Box_i B} \qquad \begin{array}{ll} M_i & \Box_i(A \wedge B) \rightarrow \Box_i A \\ N_i & \Box_i \top \end{array} \qquad \begin{array}{ll} T_i & \Box_i A \rightarrow A \\ D_i & \Box_i A \rightarrow \neg\Box_i\neg A \\ P_i & \neg\Box_i\bot \end{array}$$

The minimal non-normal monomodal logic defined in $\mathcal{L}[\Box_i]$, denoted by E_i, only contains RE_i (that is, it does not contain any additional modal axiom). Given a list of modal axioms Σ_i in $\mathcal{L}[\Box_i]$ (without repetitions), the other non-normal monomodal systems are denoted by $\mathsf{E\Sigma}_i$. We call *monotonic* any system $\mathsf{E\Sigma}_i$ such that $M_i \in \Sigma_i$. Moreover, we use L_i to denote any logic defined in $\mathcal{L}[\Box_i]$.

We consider the standard notion of derivability in axiomatic modal systems: a rule $B_1, ..., B_n/A$ is derivable in a logic L_i if there is a finite sequence of formulas ending with A in which every formula is an (instance of an) axiom of L_i, or it belongs to $\{B_1, ..., B_n\}$, or it is obtained from previous formulas by the application of a rule of L_i. A formula A is derivable in L_i, written $\vdash_{\mathsf{L}_i} A$, if the rule $\emptyset/A$ is derivable in L_i. Finally, a formula A is (locally) derivable from a set of formulas Φ in L_i, written $\Phi \vdash_{\mathsf{L}_i} A$, if there is a finite set $\{B_1, ..., B_n\} \subseteq \Phi$ such that $\vdash_{\mathsf{L}_i} B_1 \wedge ... \wedge B_n \rightarrow A$. We recall that the axioms M_i and N_i are respectively equivalent to the monotonicity rule $A \rightarrow B/\Box_i A \rightarrow \Box_i B$ and to the necessitation rule $A/\Box_i A$. Note also that the axioms P_i and D_i are equivalent in *normal* modal logics (i.e., modal logics extending K_i), but are not equivalent in non-normal ones. In particular, the following derivability relations hold: $\vdash_{\mathsf{ET}_i} P_i$, $\vdash_{\mathsf{ET}_i} D_i$, $\vdash_{\mathsf{EMD}_i} P_i$, $\vdash_{\mathsf{END}_i} P_i$. By virtue of these relations, the considered family contains 17 distinct monomodal logics, displayed in Fig. 1.

In this paper, we study multimodal logics obtained by combining non-normal monomodal logics in the following way. First, let $\mathsf{L}_1, ..., \mathsf{L}_n$ be n non-normal monomodal logics respectively formulated in the languages $\mathcal{L}[\Box_1]$, ..., $\mathcal{L}[\Box_n]$ sharing the same propositional variables and Boolean operators, but with distinct modalities $\Box_1$, ..., $\Box_n$. Moreover, let $\mathcal{I}$ be an *acyclic* set of pairs (i, j) with $1 \leq i, j \leq n$ (that is, there is no chain (i, j_1), (j_1, j_2), ..., (j_k, i)).

Definition 1. *The* combination $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ *is the smallest multimodal logic in the language* $\mathcal{L}[\Box_1, ..., \Box_n]$ *that contains* $\mathsf{L}_1 \cup ... \cup \mathsf{L}_n$ *as well as the interaction axioms* $\Box_i A \rightarrow \Box_j A$*, for all* $(i, j) \in \mathcal{I}$*, and is closed under the rules of* L_1, ..., L_n *(that is, modus ponens and* RE_1, ..., RE_n*).*

Note that $\langle \mathsf{L}_1 ... \mathsf{L}_n \emptyset \rangle$ corresponds to the *fusion* of $\mathsf{L}_1, ..., \mathsf{L}_n$ [45]. The reason for restricting to acyclic sets $\mathcal{I}$ is that in presence of cycles (i, j_1), (j_1, j_2), ..., (j_k, i), the modalities $\Box_i$, $\Box_{j_1}$, ..., $\Box_{j_k}$ become all indistinguishable. In the following, for every logic $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$, we denote $\mathcal{I}^*$ the transitive closure of $\mathcal{I}$.

The standard semantics of non-normal monomodal logics is given in terms of so-called neighbourhood models. Dealing with multimodal logics, we consider here models endowed with n neighbourhood functions, one for each modality.

Definition 2. *A* n-neighbourhood model *is a tuple* $\mathcal{M} = (\mathcal{W}, \mathcal{N}_1, ..., \mathcal{N}_n, \mathcal{V})$, *where* $\mathcal{W}$ *is a non-empty set of* worlds, $\mathcal{V}$: *Atm* $\longrightarrow \mathcal{P}(\mathcal{W})$ *is a* valuation function*, and each* $\mathcal{N}_i$ *is a* neighbourhood function $\mathcal{W} \longrightarrow \mathcal{P}(\mathcal{P}(\mathcal{W}))$ *possibly satisfying the following conditions for all* $w \in \mathcal{W}$*, where* $\alpha, \beta \subseteq \mathcal{W}$*:*

$(M_i\text{-}c)$ *if* $\alpha \in \mathcal{N}_i(w)$ *and* $\alpha \subseteq \beta$, *then* $\beta \in \mathcal{N}_i(w)$;	$(N_i\text{-}c)$ $\mathcal{W} \in \mathcal{N}_i(w)$;
$(T_i\text{-}c)$ *if* $\alpha \in \mathcal{N}_i(w)$, *then* $w \in \alpha$;	$(P_i\text{-}c)$ $\emptyset \notin \mathcal{N}_i(w)$;
$(D_i\text{-}c)$ *if* $\alpha \in \mathcal{N}_i(w)$, *then* $\mathcal{W} \setminus \alpha \notin \mathcal{N}_i(w)$;	$(Int_{ij}\text{-}c)$ $\mathcal{N}_i(w) \subseteq \mathcal{N}_j(w)$.

Given a monomodal logic $\mathsf{E}\Sigma_i$ *and a neighbourhood function* $\mathcal{N}_i$*, we say that* $\mathcal{N}_i$ *is a* $\mathsf{E}\Sigma_i$-function *if it satisfies Condition* $(\sigma_i\text{-}c)$*, for every* $\sigma_i \in \Sigma_i$*. Moreover, we say that a model* $\mathcal{M} = (\mathcal{W}, \mathcal{N}_1, ..., \mathcal{N}_n, \mathcal{V})$ *is a model for a multimodal logic* $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$*, or it is a* $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$*-model, if* $\mathcal{N}_i$ *is a* L_i*-function for all* $1 \leq i \leq n$*, and* $\mathcal{M}$ *satisfies* $(Int_{ij}\text{-}c)$ *for all* $(i, j) \in \mathcal{I}$*.*

The relation $\mathcal{M}, w \Vdash A$ *is defined as usual for propositional variables and Boolean connectives, while for* $\Box_i$ *it is as follows, where* $[\![A]\!]_{\mathcal{M}} = \{v \mid \mathcal{M}, v \Vdash A\}$*:*

$$\mathcal{M}, w \Vdash \Box_i A \quad \textit{iff} \quad [\![A]\!]_{\mathcal{M}} \in \mathcal{N}_i(w).$$

We consider the usual notions of *validity in a model* $\mathcal{M}$ and *validity in a class of models* $\mathcal{C}$: $\mathcal{M} \models A$ iff $\mathcal{M}, w \Vdash A$, for all w of $\mathcal{M}$; and $\mathcal{C} \models A$ iff $\mathcal{M} \models A$, for all $\mathcal{M} \in \mathcal{C}$, respectively. In the following, we omit to specify $\mathcal{M}$, and simply write $w \Vdash A$ or $[\![A]\!]$, when it is clear from the context.

In this paper, we study the complexity of the *validity problem* for the logics $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$, that is, the problem of deciding, given a formula A of $\mathcal{L}[\Box_1, ..., \Box_n]$, whether A is valid in the class of all $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-models. Due to the following completeness result, the validity problem for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ is equivalent to the *derivability problem* for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$, that is, the problem of deciding whether A is derivable in the axiomatic system $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ (Definition 1).

$$(\mathsf{init})\ \Gamma, p \Rightarrow p, \Delta \qquad (\bot_{\mathsf{L}})\ \Gamma, \bot \Rightarrow \Delta \qquad (\rightarrow_{\mathsf{L}})\ \frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma, B \Rightarrow \Delta}{\Gamma, A \rightarrow B \Rightarrow \Delta} \qquad (\rightarrow_{\mathsf{R}})\ \frac{\Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow A \rightarrow B, \Delta}$$

$$(\mathsf{e}_i)\ \frac{A \Rightarrow B \qquad B \Rightarrow A}{\Gamma, \Box_i A \Rightarrow \Box_i B, \Delta} \qquad (\mathsf{m}_i)\ \frac{A \Rightarrow B}{\Gamma, \Box_i A \Rightarrow \Box_i B, \Delta} \qquad (\mathsf{n}_i)\ \frac{\Rightarrow A}{\Gamma \Rightarrow \Box_i A, \Delta}$$

$$(\mathsf{p}_i)\ \frac{A \Rightarrow}{\Gamma, \Box_i A \Rightarrow \Delta} \qquad (\mathsf{d}_i)\ \frac{A, B \Rightarrow \qquad \Rightarrow A, B}{\Gamma, \Box_i A, \Box_i B \Rightarrow \Delta} \qquad (\mathsf{d}'_i)\ \frac{A \Rightarrow \qquad \Rightarrow A}{\Gamma, \Box_i A \Rightarrow \Delta}$$

$$(\mathsf{md}_i)\ \frac{A, B \Rightarrow}{\Gamma, \Box_i A, \Box_i B \Rightarrow \Delta} \qquad (\mathsf{t}_i)\ \frac{\Gamma, \Box_i A, A \Rightarrow \Delta}{\Gamma, \Box_i A \Rightarrow \Delta} \qquad (\mathsf{e}_{ij})\ \frac{A \Rightarrow B \qquad B \Rightarrow A}{\Gamma, \Box_i A \Rightarrow \Box_j B, \Delta}$$

$$(\mathsf{m}_{ij})\ \frac{A \Rightarrow B}{\Gamma, \Box_i A \Rightarrow \Box_j B, \Delta} \qquad (\mathsf{d}_{ij})\ \frac{A, B \Rightarrow \qquad \Rightarrow A, B}{\Gamma, \Box_i A, \Box_j B \Rightarrow \Delta} \qquad (\mathsf{md}_{ij})\ \frac{A, B \Rightarrow}{\Gamma, \Box_i A, \Box_j B \Rightarrow \Delta}$$

Fig. 2. Sequent rules.

Theorem 1. *A formula A of $\mathcal{L}[\Box_1, ..., \Box_n]$ is derivable in $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ if and only if it is valid in the class of all $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-models.*

Proof. Soundness is routine by showing that all axioms and rules are, respectively, valid and validity preserving in the corresponding models. For completeness, we adapt the standard proof for non-normal monomodal logics (cf. [7]). As usual, we call $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-maximal consistent (or maxcons) any set Φ of formulas of $\mathcal{L}[\Box_1, ..., \Box_n]$ such that $\Phi \nvdash_{\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle} \bot$, and for all $A \in \mathcal{L}[\Box_1, ..., \Box_n]$, $A \notin \Phi$ implies $\Phi \cup \{A\} \vdash_{\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle} \bot$. Moreover, we denote $[A]$ the class of $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-maxcons sets s.t. $A \in \Phi$. The usual properties of maxcons sets hold, in particular: if $\Phi \nvdash_{\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle} \bot$, then there is Ψ $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-maxcons s.t. $\Phi \subseteq \Psi$. We define the canonical model for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ as $\mathcal{M} = (\mathcal{W}, \mathcal{N}_1, ..., \mathcal{N}_n, \mathcal{V})$, where $\mathcal{W}$ is the class of all $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-maxcons sets, and for all $p \in Atm$, $\mathcal{V}(p) = [p]$. Moreover, for all $1 \leq i \leq n$ and all $\Phi \in \mathcal{W}$, we define $\alpha \in \mathcal{N}_i(\Phi)$ iff $\alpha = [A]$ for some $\Box_j A \in \Phi$ s.t. $j = i$ or $(j, i) \in \mathcal{I}^*$, or $\alpha \supseteq [B]$ for some $\Box_k B \in \Phi$ s.t. $k = i$ or $(k, i) \in \mathcal{I}^*$, and $M_i \in \mathsf{L}_i$, or $M_k \in \mathsf{L}_k$, or $M_u \in \mathsf{L}_u$ for some u s.t. $(k, u), (u, i) \in \mathcal{I}^*$. We can show that $\mathcal{M}$ is a $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$-model, and that for all $A \in \mathcal{L}[\Box_1, ..., \Box_n]$, $[\![A]\!] = [A]$. Then the completeness of $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ follows in the usual way. $\Box$

3 Sequent Calculi

In this section, we present sequent calculi for all the considered combinations of NNMLs. We show that the calculi are sound and cut-free complete with respect to the corresponding axiomatic systems.

In the following, we use capital Greek letters $\Gamma, \Delta, \Pi, \Theta$ to denote possibly empty *multisets* of formulas. As usual, we call *sequent* any pair $\Gamma \Rightarrow \Delta$ of finite multisets of formulas. Sequents are interpreted in the language of the logic by the *formula interpretation* $\iota(\Gamma \Rightarrow \Delta) = \bigwedge \Gamma \rightarrow \bigvee \Delta$, if $\Gamma \neq \emptyset$, and $\iota(\Gamma \Rightarrow \Delta) = \bigvee \Delta$, if $\Gamma = \emptyset$, where $\bigvee \emptyset = \bot$.

$\mathbb{S}.\mathsf{E}_i$:	$\{\mathsf{e}_i\}$	$\mathbb{S}.\mathsf{ENP}_i$:	$\{\mathsf{e}_i, \mathsf{n}_i, \mathsf{p}_i\}$	$\mathbb{S}.\mathsf{EMN}_i$:	$\{\mathsf{m}_i, \mathsf{n}_i\}$
$\mathbb{S}.\mathsf{EP}_i$:	$\{\mathsf{e}_i, \mathsf{p}_i\}$	$\mathbb{S}.\mathsf{END}_i$:	$\{\mathsf{e}_i, \mathsf{n}_i, \mathsf{p}_i, \mathsf{d}_i\}$	$\mathbb{S}.\mathsf{EMT}_i$:	$\{\mathsf{m}_i, \mathsf{t}_i\}$
$\mathbb{S}.\mathsf{ED}_i$:	$\{\mathsf{e}_i, \mathsf{d}_i, \mathsf{d}'_i\}$	$\mathbb{S}.\mathsf{ENT}_i$:	$\{\mathsf{e}_i, \mathsf{n}_i, \mathsf{t}_i\}$	$\mathbb{S}.\mathsf{EMNP}_i$:	$\{\mathsf{m}_i, \mathsf{n}_i, \mathsf{p}_i\}$
$\mathbb{S}.\mathsf{EDP}_i$:	$\{\mathsf{e}_i, \mathsf{d}_i, \mathsf{p}_i\}$	$\mathbb{S}.\mathsf{EM}_i$:	$\{\mathsf{m}_i\}$	$\mathbb{S}.\mathsf{EMND}_i$:	$\{\mathsf{m}_i, \mathsf{n}_i, \mathsf{p}_i, \mathsf{md}_i\}$
$\mathbb{S}.\mathsf{ET}_i$:	$\{\mathsf{e}_i, \mathsf{t}_i\}$	$\mathbb{S}.\mathsf{EMP}_i$:	$\{\mathsf{m}_i, \mathsf{p}_i\}$	$\mathbb{S}.\mathsf{EMNT}_i$:	$\{\mathsf{m}_i, \mathsf{n}_i, \mathsf{t}_i\}$
$\mathbb{S}.\mathsf{EN}_i$:	$\{\mathsf{e}_i, \mathsf{n}_i\}$	$\mathbb{S}.\mathsf{EMD}_i$:	$\{\mathsf{m}_i, \mathsf{p}_i, \mathsf{md}_i\}$		

Fig. 3. Modal rules of sequent calculi for non-normal monomodal logics.

Sequent calculi for non-normal monomodal logics are studied in [27,28,31,34,36].[3] For each logic L_i, the corresponding sequent calculus $\mathbb{S}.\mathsf{L}_i$ contains the propositional rules init, $\bot_\mathsf{L}$, $\rightarrow_\mathsf{L}$, $\rightarrow_\mathsf{R}$ and suitable modal rules from Fig. 2, as summarised in Fig. 3.

Concerning the other rules in Fig. 2, note that the order of the indexes i, j is relevant for e_{ij} and m_{ij} ($\Box_i A$ is in Γ while $\Box_j B$ is in Δ), while it is not relevant for d_{ij} and md_{ij} (both $\Box_i A$ and $\Box_j B$ are in Γ). Accordingly, we assume $\mathsf{d}_{ij} = \mathsf{d}_{ji}$ and $\mathsf{md}_{ij} = \mathsf{md}_{ji}$, whereas $\mathsf{e}_{ij} \neq \mathsf{e}_{ji}$ and $\mathsf{m}_{ij} \neq \mathsf{m}_{ji}$. The sequent calculi for the combinations of NNMLs are defined as follows.

Definition 3. *The sequent calculus $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ contains, for all $1 \leq i \leq n$, all the rules of $\mathbb{S}.\mathsf{L}_i$ different from d'_i, as well as the following rules:*

- e_{ij}, *if $(i,j) \in \mathcal{I}^*$, and $\mathsf{m}_i \notin \mathbb{S}.\mathsf{L}_i$, and $\mathsf{m}_j \notin \mathbb{S}.\mathsf{L}_j$, and there is no k such that $(i,k), (k,j) \in \mathcal{I}^*$ and $\mathsf{m}_k \in \mathbb{S}.\mathsf{L}_k$;*
- m_{ij}, *if $(i,j) \in \mathcal{I}^*$, and $\mathsf{m}_i \in \mathbb{S}.\mathsf{L}_i$ or $\mathsf{m}_j \in \mathbb{S}.\mathsf{L}_j$ or there is k s.t. $\mathsf{m}_k \in \mathbb{S}.\mathsf{L}_k$ and $(i,k), (k,j) \in \mathcal{I}^*$;*
- n_i, *if there is j such that $(j,i) \in \mathcal{I}^*$ and $\mathsf{n}_j \in \mathbb{S}.\mathsf{L}_j$;*
- d_i, *if there is j such that $(i,j) \in \mathcal{I}^*$, and $\mathsf{e}_{ij} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and $\mathsf{d}_j \in \mathbb{S}.\mathsf{L}_j$;*
- md_i, *if there is j such that $(i,j) \in \mathcal{I}^*$, and $\mathsf{m}_{ij} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and $\mathsf{d}_j \in \mathbb{S}.\mathsf{L}_j$ or $\mathsf{md}_j \in \mathbb{S}.\mathsf{L}_j$;*
- d_{ij}, *if there is k such that (1) $(i,k) \in \mathcal{I}^*$, and (2) $(j,k) \in \mathcal{I}^*$ or $k = j$, and (3) $\mathsf{d}_k \in \mathbb{S}.\mathsf{L}_k$, and (4) $\mathsf{e}_{ik}, \mathsf{e}_{jk} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$,*
- md_{ij}, *if there is k s.t. (1) $(i,k) \in \mathcal{I}^*$, (2) $(j,k) \in \mathcal{I}^*$ or $k = j$, (3) $\mathsf{d}_k \in \mathbb{S}.\mathsf{L}_k$ or $\mathsf{md}_k \in \mathbb{S}.\mathsf{L}_k$, and (4) $\mathsf{m}_{ik} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{m}_{jk} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$;*
- p_i, *if there is j such that $j = i$ or $(i,j) \in \mathcal{I}^*$, and $\mathsf{p}_j \in \mathbb{S}.\mathsf{L}_j$ or there is k such that $\mathsf{n}_k \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and $\mathsf{d}_{jk} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{md}_{jk} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$;*
- d'_i, *if $\mathsf{p}_i \notin \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and there is j s.t. $j = i$ or $(i,j) \in \mathcal{I}^*$, and $\mathsf{d}'_j \in \mathbb{S}.\mathsf{L}_j$.*
- t_i, *if there is j such that $(i,j) \in \mathcal{I}^*$ and $\mathsf{t}_j \in \mathbb{S}.\mathsf{L}_j$.*

The rules listed in Definition 3 are necessary in order to ensure cut-free completeness of the sequent calculi in presence of interactions. Two examples of calculi resulting from the definition are as follows:

[3] Here we only consider pure Gentzen-style sequent calculi for NNMLs. Other sequent calculi for NNMLs have been defined in the literature in terms of labelled sequent calculi [13,24,35], nested or hypersequent calculi [11,33,34], and display calculi [8].

$$\begin{aligned}
\mathbb{S}\langle \mathsf{EN}_1, \mathsf{ET}_2, \mathsf{EM}_3\{(1,2),(2,3)\}\rangle &= \{\mathsf{e}_1, \mathsf{n}_1, \mathsf{e}_2, \mathsf{t}_2, \mathsf{m}_3, \mathsf{m}_{1,2}, \mathsf{m}_{1,3}, \mathsf{m}_{2,3}, \mathsf{t}_1,\\
&\qquad \mathsf{n}_2, \mathsf{n}_3\};\\
\mathbb{S}\langle \mathsf{EN}_1, \mathsf{EM}_2, \mathsf{ED}_3\{(1,3),(2,3)\}\rangle &= \{\mathsf{e}_1, \mathsf{n}_1, \mathsf{m}_2, \mathsf{e}_3, \mathsf{d}_3, \mathsf{e}_{1,3}, \mathsf{m}_{2,3}, \mathsf{n}_3, \mathsf{d}_{1,3},\\
&\qquad \mathsf{d}_1, \mathsf{md}_{2,3}, \mathsf{p}_3, \mathsf{p}_1, \mathsf{p}_2\}.
\end{aligned}$$

As usual, initial sequents are formulated only for propositional variables but can be extended to arbitrary formulas. We say that a rule is *admissible* in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ if whenever the premisses are derivable in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, the conclusion is also derivable, and that a single-premiss rule is *height-preserving admissible* in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ (hp-admissible for short) if whenever the premiss is derivable, the conclusion is derivable with a derivation of at most the same height. Moreover, we say that a rule $\mathcal{S}_1, ..., \mathcal{S}_n / \mathcal{S}'$ is *height-preserving invertible* in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ (hp-invertible) if the rule $\mathcal{S}'/\mathcal{S}_i$ is hp-admissible for all premisses $\mathcal{S}_i$. One can show that the propositional rules of $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ are hp-invertible, by contrast the modal rules are not (with the exception of t_i). As an easy example, consider the sequents $p \Rightarrow q$ and $\Box_i p \Rightarrow \Box_i q, \Box_i(p \vee r)$, respectively premiss and conclusion of an instance of m_i, where the conclusion is derivable and the premiss is not.

Proposition 1. *In every calculus* $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$*, the following structural rules* Lwk*,* Rwk*,* Lctr *and* Rctr *are hp-admissible, and the following rule* cut *is admissible:*

$$\mathsf{Lwk}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta} \qquad \mathsf{Rwk}\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow A, \Delta} \qquad \mathsf{Lctr}\ \frac{\Gamma, A, A \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta} \qquad \mathsf{Rctr}\ \frac{\Gamma \Rightarrow A, A, \Delta}{\Gamma \Rightarrow A, \Delta}$$

$$\mathsf{cut}\ \frac{\Gamma \Rightarrow A, \Delta \qquad \Pi, A \Rightarrow \Theta}{\Gamma, \Pi \Rightarrow \Delta, \Theta}$$

Proof. Hp-admissibility of Lwk, Rwk, Lctr and Rctr is proved as usual by mutual induction on the height of the derivation of their premisses (with d'_i ensuring that contraction is admissible also in the calculi with d_i). Admissibility of cut is proved by induction on the lexicographically ordered pairs (c, h), where c is the weight of the cut formula, and $h = h_1 + h_2$ is the cut height, where h_1 and h_2 are the heights of the derivations of the premisses of cut. The proof is standard and distinguishes some cases according to whether the cut formula is or not principal in the last rules applied in the derivation of the premisses of cut. Here we only show two representative cases, where the cut formula is principal in the last rule applied in the derivation of both premisses of cut.

$(\mathsf{e}_{iu} - \mathsf{md}_{uj})$ The derivation on the left is converted into the one on the right:

$$\frac{\mathsf{e}_{iu}\ \dfrac{A \Rightarrow B \qquad B \Rightarrow A}{\Gamma, \Box_i A \Rightarrow \Box_u B, \Delta} \qquad \dfrac{B, C \Rightarrow}{\Pi, \Box_u B, \Box_j C \Rightarrow \Theta}\ \mathsf{md}_{uj}}{\Gamma, \Pi, \Box_i A, \Box_j C \Rightarrow \Delta, \Theta}\ \mathsf{cut} \quad \rightsquigarrow \quad \frac{\dfrac{A \Rightarrow B \qquad B, C \Rightarrow}{A, C \Rightarrow}\ \mathsf{cut}}{\Gamma, \Pi, \Box_i A, \Box_j C \Rightarrow \Delta, \Theta}\ \mathsf{md}_{ij}$$

where the application of cut has a lower height, and $\mathsf{md}_{ij} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ by Definition 3. Indeed, $\mathsf{e}_{iu} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ implies $(i, u) \in \mathcal{I}^*$. Moreover, since $\mathsf{md}_{uj} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, following Definition 3 there are three possibilities: (1) $(u, j) \in \mathcal{I}^*$, and $\mathsf{m}_{uj} \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and $\mathsf{d}_j \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{md}_j \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$;

or (2) $(j,u) \in \mathcal{I}^*$, and $\mathsf{m}_{ju} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, and $\mathsf{d}_u \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ or $\mathsf{md}_u \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$; or (3) there is k such that $(u,k),(j,k) \in \mathcal{I}^*$, and $\mathsf{m}_{uk} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ or $\mathsf{m}_{jk} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, and $\mathsf{d}_k \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ or $\mathsf{md}_k \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. If (1), then $(i,j) \in \mathcal{I}^*$ and $\mathsf{m}_{ij} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. If (2), then $(i,u),(j,u) \in \mathcal{I}^*$. If (3), then $(i,k),(j,k) \in \mathcal{I}^*$. In all these cases, by Definition 3, $\mathsf{md}_{ij} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$.

$(\mathsf{m}_{ij} - \mathsf{p}_j)$ The derivation on the left is converted into the one on the right:

$$\dfrac{\mathsf{m}_{ij}\,\dfrac{A \Rightarrow B}{\Gamma, \Box_i A \Rightarrow \Box_j B, \Delta} \qquad \dfrac{B \Rightarrow}{\Pi, \Box_j B \Rightarrow \Theta}\,\mathsf{p}_j}{\Gamma, \Pi, \Box_i A \Rightarrow \Delta, \Theta}\,\mathsf{cut} \quad \rightsquigarrow \quad \dfrac{\dfrac{A \Rightarrow B \qquad B \Rightarrow}{A \Rightarrow}\,\mathsf{cut}}{\Gamma, \Pi, \Box_i A \Rightarrow \Delta, \Theta}\,\mathsf{p}_i$$

where the application of cut has a lower height, and $\mathsf{p}_i \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ by Definition 3. Indeed, $\mathsf{m}_{ij} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ implies $(i,j) \in \mathcal{I}^*$. Moreover, since $\mathsf{p}_j \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ we have three possibilities: (1) $\mathsf{p}_j \in \mathbb{S}.\mathsf{L}_j$; or (2) there is k such that $(j,k) \in \mathcal{I}^*$ and $\mathsf{p}_k \in \mathbb{S}.\mathsf{L}_k$; or (3) there are k, u such that $(j,k),(k.u) \in \mathcal{I}^*$, $\mathsf{n}_u \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, and $\mathsf{d}_{ku} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ or $\mathsf{md}_{ku} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. If (1), then by Definition 3, $\mathsf{p}_i \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. If (2) or (3), then $(i,k) \in \mathcal{I}^*$, and in both cases by Definition 3, $\mathsf{p}_i \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. □

Theorem 2. *$\Gamma \Rightarrow \Delta$ is derivable in $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ if and only if $\bigwedge \Gamma \rightarrow \bigvee \Delta$ is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$*

Proof. ($\Rightarrow$) For each rule $\mathcal{S}/\mathcal{S}'$ or $\mathcal{S}_1, \mathcal{S}_2/\mathcal{S}'$ of $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, we need to show that the corresponding rule $\iota(\mathcal{S})/\iota(\mathcal{S}')$ or $\iota(\mathcal{S}_1), \iota(\mathcal{S}_2)/\iota(\mathcal{S}')$ is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. We consider as an example the rule md_{ij}, and write $\vdash$ for $\vdash_{\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle}$. First, it is easy to see that $\vdash \Box_i A \rightarrow \Box_j A$ for all $(i,j) \in \mathcal{I}^*$. Now suppose that $\vdash A \wedge B \rightarrow \bot$, hence $\vdash A \rightarrow \neg B$. By Definition 3, there is k such that $(i,k) \in \mathcal{I}^*$ or $k = i$, $(j,k) \in \mathcal{I}^*$ or $k = j$, $\mathsf{d}_k \in \mathbb{S}.\mathsf{L}_k$ or $\mathsf{md}_k \in \mathbb{S}.\mathsf{L}_k$, and $\mathsf{m}_{ik} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ or $\mathsf{m}_{jk} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. Then, by def. of monomodal calculi, $D_k \in \mathsf{L}_k$. Suppose that $\mathsf{m}_{ik} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$. One can show that the rule $C \rightarrow D/\Box_i C \rightarrow \Box_k D$ is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ for any C, D. Then since $\vdash A \rightarrow \neg B$, we have $\vdash \Box_i A \rightarrow \Box_k \neg B$. Moreover, we have $\vdash \Box_j B \rightarrow \Box_k B$. Then by D_k, $\vdash \Box_i A \wedge \Box_j B \rightarrow \bot$, thus $\vdash \bigwedge \Gamma \wedge \Box_i A \wedge \Box_j B \rightarrow \bigvee \Delta$ for all Γ, Δ. If $\mathsf{m}_{jk} \in \mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ the proof is analogous. ($\Leftarrow$) By showing that all axioms and rules of $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ are derivable, respectively admissible, in $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, with modus ponens simulated by cut in the usual way. □

In this paper, we provide a proof of coNP-complexity for the validity problem for the logics $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ following a strategy based on a reformulation of the calculi $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ in terms of hypersequents, as explained in the next section. Alternatively, it could be possible to devise a strategy directly based on the calculi $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ only.[4] To this goal, two key observations are in order. First, it is easy to see that in any proof tree $\mathcal{T}$ for $\Gamma \Rightarrow \Delta$ in $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, every branch of $\mathcal{T}$ has polynomial length with respect to the length n of $\Gamma \Rightarrow \Delta$. Second, for every non-invertible modal rule, at most quadratically many premisses (w.r.t. n) are possible. This would allow one to obtain certificates for non-derivability in $\mathbb{S}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ verifiable in polynomial time by a deterministic Turing machine. We leave as future work further investigation in this direction.

[4] We thank one reviewer for suggesting us this possibility.

4 Invertible Calculi and CoNP Complexity

In this section, we present a proof of coNP complexity for the logics $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ based on a reformulation of the sequent calculi $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ where all the rules are invertible. In particular, in order to make the modal rules invertible, we rewrite all the rules using hypersequents, following the strategy of [11]. We show that the hypersequent calculi $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ provide a coNP decision procedure for the validity problem in $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$. Specifically, we present a coNP proof search algorithm in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ that explicitly constructs a derivation for every valid hypersequent/formula. Moreover, we show that from every failed derivation one can extract a countermodel of the input hypersequent: this means that we can construct a countermodel of every non-valid formula.

A *hypersequent* $\mathcal{H}$ [2] is a finite multiset of sequents, and is written $\Gamma_1 \Rightarrow \Delta_1 \mid ... \mid \Gamma_k \Rightarrow \Delta_k$, where $\Gamma_1 \Rightarrow \Delta_1, ..., \Gamma_k \Rightarrow \Delta_k$ are called the *components* of $\mathcal{H}$. The hypersequent rules for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ are direct reformulation of the sequent rules, and are displayed in Fig. 4. Essentially, backward applications of the hypersequent modal rules introduce a new component which coincides with the premiss of the corresponding sequent rule. In this way, all information contained in the conclusion is preserved into the premisses, thus making alternative rule applications still possible in bottom-up proof search. Concerning the propositional rules, we consider a cumulative formulation of them where the principal formulas are kept into the premisses. As we will see, this allows us to easily extract countermodels from failed proofs.

Differently from sequents, hypersequents cannot be interpreted as formulas of $\mathcal{L}[\Box_1, ..., \Box_n]$ (we will come back to this problem in the next section). Hypersequents are evaluated on n-neighbourhood models as: $\mathcal{M}, w \Vdash \Gamma \Rightarrow \Delta$ if and only if $\mathcal{M}, w \Vdash \iota(\Gamma \Rightarrow \Delta)$; $\mathcal{M} \models \Gamma \Rightarrow \Delta$ if and only if $\mathcal{M}, w \Vdash \Gamma \Rightarrow \Delta$, for all w of $\mathcal{M}$; and $\mathcal{M} \models \Gamma_1 \Rightarrow \Delta_1 \mid ... \mid \Gamma_k \Rightarrow \Delta_k$ if and only if $\mathcal{M} \models \Gamma_\ell \Rightarrow \Delta_\ell$, for some $1 \leq \ell \leq k$.

Definition 4. *The hypersequent calculus $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ is defined as $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ (Definition 3), with the difference that the rules are formulated in their hypersequent version (Fig. 4).*

We first show that the calculi are sound and complete with respect to the corresponding logics. Since hypersequents do not have a formula interpretation, we consider a semantic proof of soundness.

Proposition 2. *If $\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$, then $\mathcal{H}$ is valid in every n-neighbourhood model for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$.*

Proof. It is immediate to see that the initial hypersequents init and $\bot_\mathsf{L}$ are valid in every model. We need to show that all rules of $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ are validity preserving in every model for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$. We consider as an example the rule md_{ij}: Suppose that $\mathcal{M} \models \mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta \mid A, B \Rightarrow$. If $\mathcal{M} \models \mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta$ we are done. Otherwise $\mathcal{M} \models A, B \Rightarrow$, that is, $[\![A]\!] \subseteq [\![\neg B]\!]$. As a consequence of Definition 3, md_{ij} belongs to $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$ in two cases: (1)

$$(\mathsf{init})\ \mathcal{H} \mid \Gamma, p \Rightarrow p, \Delta \qquad (\bot_\mathsf{L})\ \mathcal{H} \mid \Gamma, \bot \Rightarrow \Delta$$

$$(\rightarrow_\mathsf{L})\ \frac{\mathcal{H} \mid \Gamma, A \rightarrow B \Rightarrow A, \Delta \quad \mathcal{H} \mid \Gamma, A \rightarrow B, B \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, A \rightarrow B \Rightarrow \Delta} \quad (\rightarrow_\mathsf{R})\ \frac{\mathcal{H} \mid \Gamma, A \Rightarrow B, A \rightarrow B, \Delta}{\mathcal{H} \mid \Gamma \Rightarrow A \rightarrow B, \Delta}$$

$$(\mathsf{e}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_i B, \Delta \mid A \Rightarrow B \qquad \mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_i B, \Delta \mid B \Rightarrow A}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_i B, \Delta}$$

$$(\mathsf{m}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_i B, \Delta \mid A \Rightarrow B}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_i B, \Delta} \qquad (\mathsf{n}_i)\ \frac{\mathcal{H} \mid \Gamma \Rightarrow \Box_i A, \Delta \mid \Rightarrow A}{\mathcal{H} \mid \Gamma \Rightarrow \Box_i A, \Delta}$$

$$(\mathsf{p}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta \mid A \Rightarrow}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta} \quad (\mathsf{d}'_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta \mid A \Rightarrow \quad \mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta \mid \Rightarrow A}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta}$$

$$(\mathsf{d}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A, \Box_i B \Rightarrow \Delta \mid A, B \Rightarrow \qquad \mathcal{H} \mid \Gamma, \Box_i A, \Box_i B \Rightarrow \Delta \mid \Rightarrow A, B}{\mathcal{H} \mid \Gamma, \Box_i A, \Box_i B \Rightarrow \Delta}$$

$$(\mathsf{md}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A, \Box_i B \Rightarrow \Delta \mid A, B \Rightarrow}{\mathcal{H} \mid \Gamma, \Box_i A, \Box_i B \Rightarrow \Delta} \qquad (\mathsf{t}_i)\ \frac{\mathcal{H} \mid \Gamma, \Box_i A, A \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Delta}$$

$$(\mathsf{e}_{ij})\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_j B, \Delta \mid A \Rightarrow B \qquad \mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_j B, \Delta \mid B \Rightarrow A}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_j B, \Delta}$$

$$(\mathsf{m}_{ij})\ \frac{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_j B, \Delta \mid A \Rightarrow B}{\mathcal{H} \mid \Gamma, \Box_i A \Rightarrow \Box_j B, \Delta} \qquad (\mathsf{md}_{ij})\ \frac{\mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta \mid A, B \Rightarrow}{\mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta}$$

$$(\mathsf{d}_{ij})\ \frac{\mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta \mid A, B \Rightarrow \qquad \mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta \mid \Rightarrow A, B}{\mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta}$$

Fig. 4. Hypersequent rules.

$(i, j) \in \mathcal{I}^*$ and $\mathcal{M}$ satisfies (D_j-c) and (M_i-c) or (M_j-c), or (2) there is k such that $(i, k), (j, k) \in \mathcal{I}^*$ and $\mathcal{M}$ satisfies (D_k-c) and (M_i-c) or (M_j-c) or (M_k-c). If (1), then suppose $w \Vdash \Box_i A$, that is $[\![A]\!] \in \mathcal{N}_i(w)$. If ($M_i$-c), then $[\![\neg B]\!] \in \mathcal{N}_i(w)$, and by ($Int_{ij}$-c), $[\![\neg B]\!] \in \mathcal{N}_j(w)$. Otherwise by ($Int_{ij}$-c), $[\![A]\!] \in \mathcal{N}_j(w)$, and by ($M_j$-c), $[\![\neg B]\!] \in \mathcal{N}_j(w)$. Thus by ($D_j$-c), $[\![B]\!] \notin \mathcal{N}_j(w)$. If (2), let us assume (M_k-c), the other cases being similar. Suppose $w \Vdash \Box_i A \wedge \Box_j B$. Then $[\![A]\!] \in \mathcal{N}_i(w)$ and $[\![B]\!] \in \mathcal{N}_j(w)$. By ($Int_{ik}$-c) and ($Int_{jk}$-c), $[\![A]\!], [\![B]\!] \in \mathcal{N}_k(w)$, thus $[\![B]\!], [\![\neg B]\!] \in \mathcal{N}_k(w)$, against ($D_k$-c). Thus in both cases $w \not\Vdash \Box_i A \wedge \Box_j B$. Since this holds for every w, we have $\mathcal{M} \models \Box_i A, \Box_j B \Rightarrow$, hence $\mathcal{M} \models \mathcal{H} \mid \Gamma, \Box_i A, \Box_j B \Rightarrow \Delta$. □

To prove completeness, we consider here a simple proof that relies on the cut-free completeness of the sequent calculi, although a direct proof of cut elimination analogous to the one in the previous section could be given. The proof is based on the following observation, which can be easily proved by induction on the height of the derivation of the premiss of the rules.

Lemma 1. *The rules of external weakening and external contraction are height-preserving admissible in* $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$*:*

$$\mathsf{Ewk}\ \frac{\mathcal{H}}{\mathcal{H} \mid \Gamma \Rightarrow \Delta} \qquad \mathsf{Ectr}\ \frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid \Gamma \Rightarrow \Delta}$$

Proposition 3. *If $\Gamma \Rightarrow \Delta$ is derivable in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, then $\Gamma \Rightarrow \Delta$ is derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$.*

Proof. By induction on the height of the derivation of $\Gamma \Rightarrow \Delta$ in $\mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, considering the last rule applied in the derivation. For initial sequents and propositional rules the proof is immediate. For modal rules, suppose that $\Gamma \Rightarrow \Delta$ is obtained from $\mathcal{S}_1$ and (possibly) $\mathcal{S}_2$ by the application of the sequent rule R. Then by i.h., $\mathcal{S}_1$ and $\mathcal{S}_2$ are derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and by Ewk, $\Gamma \Rightarrow \Delta \mid \mathcal{S}_1$ and $\Gamma \Rightarrow \Delta \mid \mathcal{S}_2$ are derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. Then by the hypersequent version of the rule R, $\Gamma \Rightarrow \Delta$ is derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. □

Another immediate consequence of the height-preserving admissibility of external weakening is that all the rules of $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ are height-preserving invertible in the calculi. It follows that one single proof search is sufficient to establish whether a hypersequent is derivable or not. However, as a difference with sequent rules, backward applications of the hypersequent rules increase the complexity of the hypersequents, thus proof search in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ does not terminate *per se.* In order to retrieve termination but also obtain an optimal proof search, following [11] (cf. also [32]), we consider a proof search strategy based on the following loop checking condition and on a fixed order of rule applications.

Definition 5. *An application of a hypersequent rule with premisses $\mathcal{G}_1$, ..., $\mathcal{G}_n$ and conclusion $\mathcal{H}$ satisfies the* local loop checking condition *(LLCC) if for each premiss $\mathcal{G}_i$, there exists a component $\Gamma \Rightarrow \Delta$ in $\mathcal{G}_i$ such that for no component $\Pi \Rightarrow \Theta$ of the conclusion $\mathcal{H}$ we have $\mathsf{set}(\Gamma) \subseteq \mathsf{set}(\Pi)$ and $\mathsf{set}(\Delta) \subseteq \mathsf{set}(\Theta)$. Moreover, having fixed an enumeration $R_1, ..., R_m$ of the rules of $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, we say that the backward application of a rule R_i with conclusion $\mathcal{H}$ satisfies the* priority order *(PO) if there is no R_j backward applicable to $\mathcal{H}$ with $j < i$.*

Bottom-up proof search with LLCC and PO is described by Algorithm 1. We now show that bottom-up proof search with LLCC and PO is complete, and that it provides a coNP procedure for deciding derivability in $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$.

Proposition 4. *If $\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, then it is derivable with a derivation in which all rule applications satisfy the LLCC and the PO.*

Proof. First, we show by induction on the height n of the derivation $\mathcal{D}$ of $\mathcal{H}$ in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ that if $\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, then it is derivable respecting the LLCC: If $n = 0$, then $\mathcal{H}$ is an initial hypersequent and $\mathcal{D}$ trivially satisfies LLCC. For $n+1$, let R be the last rule applied in $\mathcal{D}$. If R satisfies the LLCC, then we apply the i.h. to its premisses and are done. Otherwise, there is a premiss $\mathcal{G}_i$ of R such that for all components $\Gamma \Rightarrow \Delta$ in $\mathcal{G}_i$, there is $\Pi \Rightarrow \Theta$ in $\mathcal{H}$ s.t. $\mathsf{set}(\Gamma) \subseteq \mathsf{set}(\Pi)$ and $\mathsf{set}(\Delta) \subseteq \mathsf{set}(\Theta)$. Then $\mathcal{H}$ can be obtained from $\mathcal{G}_i$ by means of height-preserving applications of the structural rules. Again, by applying the i.h. we obtain a derivation of $\mathcal{H}$ where every rule application satisfies the LLCC. Moreover, given the invertibility of the rules, any derivation can be transformed into one satisfying PO by rearranging the order of the rule applications. □

Proposition 5. *For every logic $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$, Algorithm 1 runs in* coNP.

Proof. The algorithm is presented in the form of a non-deterministic Turing machine with only universal states (that is, states that are accepting if every transition leads to some accepting state), thus in order to prove that it runs in coNP, we need to show that every computation takes polynomial time. Let $\mathcal{H}$ be the input hypersequent and n be the size of $\mathcal{H}$ defined as the sum of the lengths of the formulas occurring in it. Since every backward application of a rule introduces a formula or a component, the number of possible rule applications, whence the number of computation steps, is bounded by the maximal length of the hypersequents that can be generated by the procedure. Given that all formulas occurring in a hypersequent are subformulas of some formulas occurring in $\mathcal{H}$, and that the LLCC avoids multiple occurrences of the same formulas in the same components, every component has length at most $\mathcal{O}(n)$. Moreover, new components are generated by a modal formula or a pair of modal formulas. Because of the LLCC, no matter in which component their occur, the same formula or pair of formulas cannot generate more than one component. Then the number of components is bounded by $\mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n^2)$. It follows that every hypersequent has a maximal length of $\mathcal{O}(n^3)$. Finally, checking that a premiss does not violate the LLCC takes polynomial time in the length of the conclusion. Thus the whole execution takes polynomial time. $\square$

Algorithm 1: Decision procedure for derivability in $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I} \rangle$.

```
Input: A hypersequent $\mathcal{H}$ and the code of a calculus $\mathbb{H}\langle L_1 ... L_n \mathcal{I} \rangle$.
Output: derivable, if $\mathcal{H}$ is derivable; a hypersequent otherwise.
1  if there is a component $\Gamma \Rightarrow \Delta$ in $\mathcal{H}$ with $\bot \in \Gamma$ or $\Gamma \cap \Delta \neq \emptyset$ then
2      return derivable and halt;
3  else if there is a rule backward applicable to $\mathcal{H}$ respecting the LLCC then
4      pick the first applicable rule according to PO;
5      universally choose a premiss $\mathcal{G}$ of this rule application;
6      check that the premiss does not violate the LLCC;
7      check recursively whether $\mathcal{G}$ is derivable, output the answer and halt;
8  else
9      return $\mathcal{H}$ and halt;
10 end
```

In order for the procedure to succeed, it is necessary that all executions terminate on an initial hypersequent, hence a single failed execution is sufficient to ensure the non-derivability of the input hypersequent. In this latter case, the procedure constructs a hypersequent which is not initial and it is such that no rule is backward applicable to it without violating the LLCC. We call such a hypersequent *saturated.* We now show that from a saturated hypersequent we can extract a countermodel of the input hypersequent.

Definition 6. *Let $\mathcal{H} = \Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_k \Rightarrow \Delta_k$ be a saturated hypersequent returned by Algorithm 1 on input $\mathcal{G}$ and $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. For all formulas B occurring in $\mathcal{H}$ and all $1 \leq i \leq n$, we define*

$$\lfloor B \rfloor_i = \{\ell \mid B \in \Gamma_\ell\};$$
$$\lceil B \rceil_i = \begin{cases} \mathcal{W} \setminus \{\ell \mid B \in \Delta_\ell\}, & \text{if } \mathsf{L}_i \text{ is not monotonic;} \\ \mathcal{W}, & \text{if } \mathsf{L}_i \text{ is monotonic;} \end{cases}$$
$$\eta_i = \begin{cases} \{\mathcal{W}\}, & \text{if there is } j \text{ such that } j = i \text{ or } (i,j) \in \mathcal{I}^*, \text{ and } N_j \in \mathsf{L}_j; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Then the model $\mathcal{M} = (\mathcal{W}, \mathcal{N}_1, ..., \mathcal{N}_n, \mathcal{V})$ is defined with $\mathcal{W} = \{\ell \mid \Gamma_\ell \Rightarrow \Delta_\ell \in \mathcal{H}\}$; for all $p \in Atm$, $\mathcal{V}(p) = \{\ell \mid p \in \Gamma_\ell\}$; and for all $1 \leq i \leq n$ and all $1 \leq \ell \leq k$,

$$\mathcal{N}_i(\ell) = \eta_i \cup \{\alpha \subseteq \mathcal{W} \mid \text{there is } \Box_j B \in \Gamma_\ell \text{ such that } j = i \text{ or } (j,i) \in \mathcal{I}^*, \\ \text{and } \lfloor B \rfloor_j \subseteq \alpha \subseteq \lceil B \rceil_j\}.$$

Proposition 6. *Let $\mathcal{H}$ be a saturated hypersequent returned by Algorithm 1 on input $\mathcal{G}$ and $\mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, and $\mathcal{M}$ be the model defined on the basis of $\mathcal{H}$ as in Definition 6. Then for all formulas B and all worlds ℓ of $\mathcal{M}$, it holds:*

- *if $B \in \Gamma_\ell$, then $\mathcal{M}, \ell \Vdash B$;*
- *if $B \in \Delta_\ell$, then $\mathcal{M}, \ell \nVdash B$.*

Moreover, $\mathcal{M}$ is a $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$-model.

Proof. The first claim is proved by induction on the construction of B. For $B = p$, $B = \bot$ and $B = C \wedge D$ the proof is standard. Suppose $B = \Box_i C \in \Gamma_\ell$. By i.h., $\lfloor C \rfloor_i \subseteq [\![C]\!] \subseteq \lceil C \rceil_i$. Then by definition, $[\![C]\!] \in \mathcal{N}_i(\ell)$, thus $\mathcal{M}, \ell \Vdash \Box_i C$. Now suppose $B = \Box_i C \in \Delta_\ell$. If there is no $\Box_i D \in \Gamma_\ell$ or $\Box_j D \in \Gamma_\ell$ with $(j,i) \in \mathcal{I}^*$, then if $\eta_i = \emptyset$, then $\mathcal{N}_i(\ell) = \emptyset$, hence $\mathcal{M}, \ell \nVdash \Box_i C$. If instead $\eta_i = \{\mathcal{W}\}$, then $\mathcal{N}_i(\ell) = \{\mathcal{W}\}$, moreover by Definition 3, $\mathsf{n}_i \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, hence by Definition 4, $\mathsf{n}_i \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. Thus, since $\mathcal{H}$ is saturated, there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ where $C \in \Delta_m$, then by i.h., $\mathcal{M}, m \nVdash C$, hence $[\![C]\!] \neq \mathcal{W}$, thus $[\![C]\!] \notin \mathcal{N}_i(\ell)$, hence $\mathcal{M}, \ell \nVdash \Box_i C$. Otherwise let $\Box_j D \in \Gamma_\ell$ with $j = i$ or $(j,i) \in \mathcal{I}^*$. If L_i is monotonic, then by the rule m_{ji} there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ such that $D \in \Gamma_m$ and $C \in \Delta_m$, while if L_i is not monotonic, then by the rule e_{ji} there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ such that $D \in \Gamma_m$ and $C \in \Delta_m$, or $C \in \Gamma_m$ and $D \in \Delta_m$. In the first case, by i.h., $\lfloor D \rfloor_j \not\subseteq [\![C]\!]$, and in the second case, $\lfloor D \rfloor_j \not\subseteq [\![C]\!]$ or $[\![C]\!] \not\subseteq \lceil D \rceil_j$. Since this holds for all $\Box_j D \in \Gamma_\ell$ with $j = i$ or $(j,i) \in \mathcal{I}^*$, $[\![C]\!] \notin \mathcal{N}_i(\ell)$, thus $\mathcal{M}, \ell \nVdash \Box_i C$.

We now prove that $\mathcal{M}$ is a $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$-model. From the definition of $\mathcal{N}_i$ it follows immediately that (Int_{ij}-c) is satisfied for all $(i,j) \in \mathcal{I}^*$, that (M_i-c) is satisfied if $M_i \in \mathsf{L}_i$, and that (N_i-c) is satisfied if $N_i \in \mathsf{L}_i$. We show (D_i-c) as an example for the other conditions: Suppose that $D_i \in \mathsf{L}_i$ and, by contradiction, $\alpha \in \mathcal{N}_i(\ell)$ and $\mathcal{W} \setminus \alpha \in \mathcal{N}_i(\ell)$. By def. of the monomodal calculi, $\mathsf{d}_i \in \mathbb{S}.\mathsf{L}_i$ or $\mathsf{md}_i \in \mathbb{S}.\mathsf{L}_i$. Moreover, by def. of $\mathcal{N}_i$, there is $\Box_j B \in \Gamma_\ell$ s.t. $j = i$ or $(j,i) \in \mathcal{I}^*$, and $\lfloor B \rfloor_j \subseteq \alpha \subseteq \lceil B \rceil_j$, and either there is $\Box_u C \in \Gamma_\ell$ s.t. $u = i$ or $(u,i) \in \mathcal{I}^*$, and $\lfloor C \rfloor_u \subseteq \mathcal{W} \setminus \alpha \subseteq \lceil C \rceil_u$, which implies $\lfloor B \rfloor_j \cap \lfloor C \rfloor_u = \emptyset$ and $\mathcal{W} \setminus \lceil B \rceil_j \cap \mathcal{W} \setminus \lceil C \rceil_u =$

$$\mathcal{U}_{\mathsf{L}}\ \frac{\mathcal{H} \mid \Gamma, \mathcal{U}A \Rightarrow \Delta \mid \Sigma, A \Rightarrow \Pi}{\mathcal{H} \mid \Gamma, \mathcal{U}A \Rightarrow \Delta \mid \Sigma \Rightarrow \Pi} \quad \mathcal{U}_{\mathsf{R}}\ \frac{\mathcal{H} \mid \Gamma \Rightarrow \mathcal{U}A, \Delta \mid \Rightarrow A}{\mathcal{H} \mid \Gamma \Rightarrow \mathcal{U}A, \Delta} \quad \mathcal{U}_{\mathsf{t}}\ \frac{\mathcal{H} \mid \Gamma, \mathcal{U}A, A \Rightarrow \Delta}{\mathcal{H} \mid \Gamma, \mathcal{U}A \Rightarrow \Delta}$$

Fig. 5. Hypersequent rules for universal modality.

$\emptyset$, or $\mathcal{W} \setminus \alpha = \mathcal{W}$ and $\eta_i = \{\mathcal{W}\}$. There are four possible cases. (1) If $j = u$ and $B = C$, then by Definition 3, $\mathsf{d}'_j \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{p}_j \in \mathbb{S}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, hence by Definition 4, $\mathsf{d}'_j \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{p}_j \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. Thus by saturation of $\mathcal{H}$, there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ s.t. $B \in \Gamma_m$ or $B \in \Delta_m$. Then $m \in \lfloor B \rfloor_j$ or $m \in \mathcal{W} \setminus \lceil B \rceil_j$. Since $\lfloor B \rfloor_j = \lfloor C \rfloor_u$ and $\lceil B \rceil_j = \lceil C \rceil_u$, this gives a contradiction. (2) If $j = u$ and $B \neq C$, by Definition 3 and 4 we have $\mathsf{d}_j \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{md}_j \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. (3) If $j \neq u$, by Definition 3 and 4, $\mathsf{d}_{ju} \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ or $\mathsf{md}_{ju} \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. In both cases, by saturation there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ s.t. $B, C \in \Gamma_m$ or $B, C \in \Delta_m$, which implies $m \in \lfloor B \rfloor_j \cap \lfloor C \rfloor_j$ or $m \in \mathcal{W} \setminus \lceil B \rceil_j \cap \mathcal{W} \setminus \lceil C \rceil_j$, giving a contradiction. (4) $\mathcal{W} \setminus \alpha = \mathcal{W}$ and $\eta_i = \{\mathcal{W}\}$, that is $\alpha = \emptyset$. By Definition 3 and 4, $\mathsf{p}_j \in \mathbb{H}\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$. Thus there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ s.t. $B \in \Gamma_m$, then $\lfloor B \rfloor_j \neq \emptyset$, then $\alpha \neq \emptyset$, giving a contradiction. It follows that $\alpha \notin \mathcal{N}_i(\ell)$ or $\mathcal{W} \setminus \alpha \notin \mathcal{N}_i(\ell)$. □

Note that the model $\mathcal{M}$ of Proposition 6 is also a countermodel for the input hypersequent $\mathcal{G}$. Indeed, since backward rule applications never delete formulas or components, for all components $\Gamma \Rightarrow \Delta$ in $\mathcal{G}$, there is $\Pi \Rightarrow \Theta$ in $\mathcal{H}$ such that $\mathsf{set}(\Gamma) \subseteq \mathsf{set}(\Pi)$ and $\mathsf{set}(\Delta) \subseteq \mathsf{set}(\Theta)$. Thus the world corresponding to $\Pi \Rightarrow \Theta$ in $\mathcal{M}$ falsifies also $\Gamma \Rightarrow \Delta$. In the light of this model extraction, Algorithm 1 can be easily reformulated in order to provide a NP decision procedure for the satisfiability problem in $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$, with the algorithm taking as input hypersequents of the form $A \Rightarrow$. On the basis of the above results, we can conclude the following.

Theorem 3. *The validity problem for $\langle \mathsf{L}_1 ... \mathsf{L}_n \mathcal{I}\rangle$ is* coNP*-complete.*

5 Adding the Universal Modality

As we have seen, hypersequents cannot be interpreted in the language of NNMLs. The reason is that the hypersequent construct "|" semantically corresponds to a disjunction of validities of sequents. In order to make the hypersequent calculi fully internal, we now extend the language with a universal modality $\mathcal{U}$, and add to the calculi suitable hypersequent rules for it. This operation allows us to treat another kind of logic combinations, namely the combination of NNMLs whose common language also contains $\mathcal{U}$ (together with the propositional variables and the Boolean connectives). Differently from the combinations introduced in Sect. 2, we define these logic combinations not based on the axiomatic systems, but based on the hypersequent calculi. We show that this extension of the calculi still provides a coNP proof search procedure, and also allows one to extract suitable countermodels. Based on the hypersequent calculi and the formula interpretation of the hypersequents, we also provide an axiomatisation for the resulting logics.

Let $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ be the language containing the modalities $\Box_1$, ..., $\Box_n$ as well as $\mathcal{U}$. Hypersequents are now interpreted in $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ by considering the standard formula interpretation of hypersequent calculi for $\mathsf{S5}$ [2,38]:

$$\iota(\Gamma_1 \Rightarrow \Delta_1 \mid ... \mid \Gamma_n \Rightarrow \Delta_n) = \mathcal{U}(\bigwedge \Gamma_1 \rightarrow \bigvee \Delta_1) \vee ... \vee \mathcal{U}(\bigwedge \Gamma_n \rightarrow \bigvee \Delta_n).$$

Moreover, let $\mathsf{L}_1, ..., \mathsf{L}_n$ be n non-normal monomodal logics respectively formulated in the languages $\mathcal{L}[\Box_1]$, ..., $\mathcal{L}[\Box_n]$, with $\Box_1$, ..., $\Box_n$ all distinct but sharing the same propositional variables, Boolean operators, and universal modality $\mathcal{U}$.

Definition 7. *For every calculus $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ from Sect. 4, the corresponding calculus $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ in $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ contains the rules of $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$, plus the rules $\mathcal{U}_{\mathsf{L}}$, $\mathcal{U}_{\mathsf{R}}$ and $\mathcal{U}_{\mathsf{t}}$ in Fig. 5. Moreover, we call $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-model any $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$-model (Definition 2), where $\mathcal{U}$ is interpreted as $\mathcal{M}, w \Vdash \mathcal{U}A$ if and only if $\mathcal{M}, v \Vdash A$ for all worlds v of $\mathcal{M}$.*

The rules for $\mathcal{U}$ are taken from [38] (see also [39] for similar rules, while different hypersequent rules for $\mathsf{S5}$ can be found in [29] and references therein). We start by showing that some of the results proved for $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ immediately extend to $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$.

Proposition 7. *If $\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, then $\mathcal{H}$ is valid in every $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-model.*

Proof. By extending the proof of Proposition 2. We consider as an example the rule $\mathcal{U}_{\mathsf{L}}$: Suppose that $\mathcal{M} \models \mathcal{H} \mid \Gamma, \mathcal{U}A \Rightarrow \Delta \mid \Sigma, A \Rightarrow \Pi$. If $\mathcal{M} \models \mathcal{H} \mid \Gamma, \mathcal{U}A \Rightarrow \Delta$ we are done. Otherwise $\mathcal{M} \models \Sigma, A \Rightarrow \Pi$, and since $\mathcal{M} \models \mathcal{U}A$ or $\mathcal{M} \models \neg\mathcal{U}A$, from $\mathcal{M} \not\models \Gamma, \mathcal{U}A \Rightarrow \Delta$ we get $\mathcal{M} \models \mathcal{U}A$. Then $\mathcal{M} \models \Sigma \Rightarrow \Pi$. $\Box$

Proposition 8. *Algorithm 1 on inputs $\mathcal{H}$ in $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ and $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ runs in* coNP.

Proof. The proof is exactly as the one of Proposition 5, observing that every formula $\mathcal{U}A$ can generate at most one component (cf. [32]). Note that LLCC and Algorithm 1 remain well-defined on the new inputs. $\Box$

Proposition 9. *Let $\mathcal{H} = \Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_k \Rightarrow \Delta_k$ be a saturated hypersequent returned by Algorithm 1 on input $\mathcal{G}$ and $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, and $\mathcal{M} = (\mathcal{W}, \mathcal{N}_1, ..., \mathcal{N}_n, \mathcal{V})$ be the model defined on the basis of $\mathcal{G}$ as in Definition 6. Then for all formulas B of $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ and all $\ell \in \mathcal{W}$, it holds: if $B \in \Gamma_\ell$, then $\mathcal{M}, \ell \Vdash B$, and if $B \in \Delta_\ell$, then $\mathcal{M}, \ell \nVdash B$. Moreover, $\mathcal{M}$ is a $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-model.*

Proof. The proof extends the one of Proposition 6 with the case $B = \mathcal{U}C$, which is standard: If $\mathcal{U}C \in \Gamma_\ell$, then by $\mathcal{U}_{\mathsf{L}}$ and $\mathcal{U}_{\mathsf{t}}$, $C \in \Gamma_m$ for all $m \in \mathcal{W}$, then by i.h., $\mathcal{M}, m \Vdash C$ for all $m \in \mathcal{W}$, that is $\mathcal{M}, \ell \Vdash \mathcal{U}C$. If $\mathcal{U}C \in \Delta_\ell$, then by $\mathcal{U}_{\mathsf{R}}$ there is $\Gamma_m \Rightarrow \Delta_m$ in $\mathcal{H}$ with $C \in \Delta_m$. By i.h., $\mathcal{M}, m \nVdash C$, thus $\mathcal{M}, \ell \nVdash \mathcal{U}C$. $\Box$

As before, on the basis of Proposition 9, we can obtain from the algorithm a NP decision procedure for satisfiability of $\mathcal{L}[\Box_1, ..., \Box_n]^{\mathcal{U}}$ formulas in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-models. As a further consequence, Proposition 9 entails that the calculi $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ are complete with respect to the corresponding models. Indeed, if the proof search procedure fails on input $\mathcal{H}$, then it constructs a saturated hypersequent $\mathcal{G}$ that extends $\mathcal{H}$. From Proposition 9 we get a $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-countermodel of $\mathcal{G}$, whence of $\mathcal{H}$, which means that $\mathcal{H}$ is not $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-valid.

Theorem 4. *$\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ with LLCC and PO if and only if $\mathcal{H}$ is valid in every $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-model.*

We now take advantage of the completeness of the calculi $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ and of the formula interpretation of hypersequents to provide an axiomatisation for the corresponding logics.

Definition 8. *A logic $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ is axiomatically defined as the corresponding logic $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ (Definition 1), but, for each $1 \leq i \leq n$, replacing RE_i, M_i, N_i, D_i and P_i with the corresponding axiom $E_i^{\mathcal{U}}$, $M_i^{\mathcal{U}}$, $N_i^{\mathcal{U}}$, $D_i^{\mathcal{U}}$ and $P_i^{\mathcal{U}}$ below, and adding $K_{\mathcal{U}}$, $T_{\mathcal{U}}$, $5_{\mathcal{U}}$ and $RN_{\mathcal{U}}$ (*S5 *axioms for $\mathcal{U}$):*

$E_i^{\mathcal{U}}$	$\mathcal{U}(A \to B) \wedge \mathcal{U}(B \to A) \to \mathcal{U}(\Box_i A \to \Box_i B)$	$K_{\mathcal{U}}$	$\mathcal{U}(A \to B) \wedge \mathcal{U}A \to \mathcal{U}B$
$M_i^{\mathcal{U}}$	$\mathcal{U}(A \to B) \to \mathcal{U}(\Box_i A \to \Box_i B)$	$T_{\mathcal{U}}$	$\mathcal{U}A \to A$
$N_i^{\mathcal{U}}$	$\mathcal{U}A \to \mathcal{U}\Box_i A$	$5_{\mathcal{U}}$	$\mathcal{U}A \vee \mathcal{U}\neg\mathcal{U}A$
$D_i^{\mathcal{U}}$	$\mathcal{U}(A \to B) \wedge \mathcal{U}(B \to A) \to \mathcal{U}(\Box_i A \to \neg\Box_i \neg B)$	$RN_{\mathcal{U}}$	$\dfrac{A}{\mathcal{U}A}$
$P_i^{\mathcal{U}}$	$\mathcal{U}\neg A \to \mathcal{U}\neg\Box_i A$		

T_i is the only axiom that does not change. $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ is an extension of $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ as RE_i is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ for all $1 \leq i \leq n$, and M_i, N_i, D_i or P_i is derivable if, respectively, $M_i^{\mathcal{U}}$, $N_i^{\mathcal{U}}$, $D_i^{\mathcal{U}}$ or $P_i^{\mathcal{U}}$ belongs to $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$. Consider as an example M_i: From $A \wedge B \to A$, by $RN_{\mathcal{U}}$, $\mathcal{U}(A \wedge B \to A)$, then by $M_i^{\mathcal{U}}$, $\mathcal{U}(\Box_i(A \wedge B) \to \Box_i A)$, thus by $T_{\mathcal{U}}$, $\Box_i(A \wedge B) \to \Box_i A$. We now show that each logic $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ is equivalent to the corresponding calculus $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$.

Proposition 10. *If A is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, then $\Rightarrow A$ is derivable in $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, and if $\mathcal{H}$ is derivable in $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, then $\iota(\mathcal{H})$ is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$.*

Proof. For the first claim, one can show that the axioms of $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ are derivable in $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$. For the second claim, we prove that for every rule $\mathcal{H}/\mathcal{H}'$ or $\mathcal{H}_1, \mathcal{H}_2/\mathcal{H}'$ of $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, the corresponding rule $\iota(\mathcal{H})/\iota(\mathcal{H}')$ or $\iota(\mathcal{H}_1), \iota(\mathcal{H}_2)/\iota(\mathcal{H}')$ is derivable in $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$. The proof follows the lines of the proof of Theorem 2 ($\Rightarrow$), considering that depending on the logics, additional axioms such as $\mathcal{U}(A \to B) \wedge \mathcal{U}(B \to A) \to \mathcal{U}(\Box_i A \to \neg\Box_j \neg B)$ can be derivable.

Finally, considering the properties of the calculi $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ and their equivalence with the systems $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$, we can conclude the following.

Theorem 5. *$\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ is sound and complete with respect to the class of all $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$-models. Moreover, the validity problem for $\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ is* coNP*-complete.*

6 Conclusion

We have proved that the validity/derivability problem for fusions of standard coNP NNMLs, as well as for their extensions with interaction axioms of the form $\Box_i A \to \Box_j A$, remains coNP-complete, and that the same result holds for combinations of logics sharing also a universal modality. In this respect, combinations of NNMLs display a different behaviour than combinations of standard coNP normal logics such as S5, KD45, K4.3 and S4.3, whose fusions are instead PSpace.

As we have seen, fully invertible hypersequent calculi offer a good point of view on the problem, as they allow one to decompose its global complexity into the one of the single rule applications. As a further advantage, the hypersequent calculi $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle$ allow one to explicitly construct derivations of valid hypersequents/formulas, as well as to construct countermodels of non-valid hypersequents/formulas. Furthermore, after the integration of the rules for $\mathcal{U}$ from [38], the calculi $\mathbb{H}\langle \mathsf{L}_1...\mathsf{L}_n\mathcal{I}\rangle^{\mathcal{U}}$ directly construct countermodels where both $\mathcal{U}$ and the neighbourhood functions behave correctly. This can be compared with alternative techniques such as the submodel generation [5] that might be non-trivial to apply in presence of the neighbourhood functions.

On the other hand, the definition of cut-free calculi for the logics with interaction axioms requires an intricate combinatorial analysis, in future work we would like to study calculi that allow for a modular definition of the logic combinations. We would also like to study logics with iterative axioms such as 4, 5, B, as well as product-like combinations for NNMLs.

Acknowledgements. We thank Alessandro Gianola and Anton Gnatenko for helpful discussions and the anonymous reviewers for detailed comments that helped us to improve the paper. This research has been partially supported by the project D2G2 funded through the Call for International Cooperation Projects Germany-South Tyrol by the Province of Bolzano and DFG (DFG grant n. 500249124). Andrea Mazzullo acknowledges the support of the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU.

References

1. Anglberger, A.J., Gratzl, N., Roy, O.: Obligation, free choice, and the logic of weakest permissions. Rev. Symbolic Logic **8**(4), 807–827 (2015)
2. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In: Logic: From Foundations to Applications, pp. 1–32. Oxford Science Publications (1996)
3. Baader, F., Ghilardi, S., Tinelli, C.: A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. Inf. Comput. **204**(10), 1413–1452 (2006)
4. Balbiani, P., Fernández-Duque, D., Lorini, E.: The dynamics of epistemic attitudes in resource-bounded agents. Stud. Logica. **107**(3), 457–488 (2019)
5. Blackburn, P., De Rijke, M., Venema, Y.: Modal Logic, vol. 53. Cambridge University Press, Cambridge (2001)

6. Brown, M.A.: On the logic of ability. J. Philos. Log. **17**(1), 1–26 (1988)
7. Chellas, B.F.: Modal Logic: An Introduction. Cambridge University Press, Cambridge (1980)
8. Chen, J., Greco, G., Palmigiano, A., Tzimoulis, A.: Non-normal modal logics and conditional logics: semantic analysis and proof theory. Inf. Comput. **287**, 104756 (2022)
9. Dalmonte, T.: Wijesekera-style constructive modal logics. In: Advances in Modal Logic, vol. 14, pp. 281–304. College Publications (2022)
10. Dalmonte, T., Grellois, C., Olivetti, N.: Intuitionistic non-normal modal logics: a general framework. J. Philos. Log. **49**(5), 833–882 (2020)
11. Dalmonte, T., Lellmann, B., Olivetti, N., Pimentel, E.: Hypersequent calculi for non-normal modal and deontic logics: countermodels and optimal complexity. J. Log. Comput. **31**(1), 67–111 (2021)
12. Dalmonte, T., Mazzullo, A., Ozaki, A., Troquard, N.: Non-normal modal description logics. In: JELIA 2023 (2023, to appear)
13. Dalmonte, T., Olivetti, N., Negri, S.: Non-normal modal logics: bi-neighbourhood semantics and its labelled calculi. In: Advances in Modal Logic, vol. 12, pp. 159–178. College Publications (2018)
14. Elgesem, D.: The modal logic of agency. Nord. J. Philos. Log. **2**(2), 1–46 (1997)
15. Fajardo, R., Finger, M.: How not to combine modal logics. In: Proceedings of IICAI 2005, pp. 1629–1647. IICAI (2005)
16. Fine, K., Schurz, G.: Transfer theorems for multimodal logics. In: Logic and Reality: Essays on the Legacy of Arthur Prior, pp. 169–213. Oxford University Press (1996)
17. Finger, M., Weiss, M.A.: The unrestricted combination of temporal logic systems. Log. J. IGPL **10**(2), 165–189 (2002)
18. Fischer Servi, G.: Axiomatizations for some intuitionistic modal logics. Rendiconti del Seminario Matematico - PoliTO **42**(3), 179–194 (1984)
19. Gabbay, D.M., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-Dimensional Modal Logics: Theory and Applications. Elsevier Science B.V. (2003)
20. Gabbay, D.M., Shehtman, V.B.: Products of modal logics, Part 1. Log. J. IGPL **6**(1), 73–146 (1998)
21. Ghilardi, S., Gianola, A.: Interpolation, amalgamation and combination (the non-disjoint signatures case). In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS (LNAI), vol. 10483, pp. 316–332. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66167-4_18
22. Ghilardi, S., Gianola, A.: Modularity results for interpolation, amalgamation and superamalgamation. Ann. Pure Appl. Logic **169**(8), 731–754 (2018)
23. Ghilardi, S., Santocanale, L.: Algebraic and model theoretic techniques for fusion decidability in modal logics. In: Vardi, M.Y., Voronkov, A. (eds.) LPAR 2003. LNCS (LNAI), vol. 2850, pp. 152–166. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39813-4_10
24. Gilbert, D.R., Maffezioli, P.: Modular sequent calculi for classical modal logics. Stud. Logica. **103**(1), 175–217 (2015)
25. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artif. Intell. **54**(3), 319–379 (1992)
26. Horty, J.F., Belnap, N.: The deliberative stit: a study of action, omission, ability, and obligation. J. Philos. Log. **24**(6), 583–644 (1995)
27. Indrzejczak, A.: Sequent calculi for monotonic modal logics. Bull. Sect. Logic **34**(3), 151–164 (2005)
28. Indrzejczak, A.: Admissibility of cut in congruent modal logics. Logic Log. Philos. **20**(3), 189–203 (2011)

29. Indrzejczak, A.: Sequents and Trees. Birkhäuser Cham (2021)
30. Kracht, M., Wolter, F.: Properties of independently axiomatizable bimodal logics. J. Symbolic Logic **56**(4), 1469–1485 (1991)
31. Lavendhomme, R., Lucas, T.: Sequent calculi and decision procedures for weak modal systems. Stud. Logica. **66**(1), 121–145 (2000)
32. Lellmann, B.: Hypersequent rules with restricted contexts for propositional modal logics. Theoret. Comput. Sci. **656**, 76–105 (2016)
33. Lellmann, B., Pimentel, E.: Proof search in nested sequent calculi. In: Davis, M., Fehnker, A., McIver, A., Voronkov, A. (eds.) LPAR 2015. LNCS, vol. 9450, pp. 558–574. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48899-7_39
34. Lellmann, B., Pimentel, E.: Modularisation of sequent calculi for normal and non-normal modalities. ACM Trans. Comput. Log. **20**(2), 7:1–7:46 (2019)
35. Negri, S.: Proof theory for non-normal modal logics: the neighbourhood formalism and basic results. IfCoLog J. Log. Their Appl. **4**(4), 1241–1286 (2017)
36. Orlandelli, E.: Sequent calculi and interpolation for non-normal modal and deontic logics. Logic Log. Philos. **30**(1), 139–183 (2020)
37. Pauly, M.: A modal logic for coalitional power in games. J. Log. Comput. **12**(1), 149–166 (2002)
38. Poggiolesi, F.: A cut-free simple sequent calculus for modal logic S5. Rev. Symbolic Logic **1**(1), 3–15 (2008)
39. Restall, G.: Proofnets for S5: sequents and circuits for modal logic. In: Logic Colloquium 2005, vol. 28, pp. 151–172. Cambridge University Press (2007)
40. Seylan, I., Erdur, R.C.: A tableau decision procedure for $\mathcal{ALC}$ with monotonic modal operators and constant domains. ENTCS **231**, 113–130 (2009)
41. Seylan, I., Jamroga, W.: Description logic for coalitions. In: AAMAS 2009, pp. 425–432. IFAAMAS (2009)
42. Spaan, E.: Complexity of modal logics. Ph.D. thesis (1993)
43. Troquard, N.: Reasoning about coalitional agency and ability in the logics of "bringing-it-about." Auton. Agents Multi-Agent Syst. **28**, 381–407 (2014)
44. Vardi, M.Y.: On the complexity of epistemic reasoning. In: Proceedings LICS 1989, pp. 243–252. IEEE Computer Society (1989)
45. Wolter, F.: Fusions of modal logics revisited. In: Advances in Modal Logic, vol. 1, pp. 361–379. CSLI Publications (1996)
46. Wolter, F., Zakharyaschev, M.: The relation between intuitionistic and classical modal logics. Algebra Logic **36**(2), 73–92 (1997)
47. Wolter, F., Zakharyaschev, M.: Intuitionistic modal logics as fragments of classical bimodal logics. In: Logic at Work, pp. 168–186. Springer (1999)

Resolution Calculi for Non-normal Modal Logics

Dirk Pattinson[1](✉), Nicola Olivetti[2], and Cláudia Nalon[3]

[1] School of Computing, The Australian National University, Canberra, Australia
dirk.pattinson@anu.edu.au

[2] Aix Marseille University, CNRS, LIS, Marseille, France
nicola.olivetti@lis-lab.fr

[3] Department of Computer Science, University of Brasília, Brasília, Brazil
nalon@unb.br

Abstract. We present resolution calculi for the cube of classical non-normal modal logics. The calculi are based on a simple clausal form that comprises both local and global clauses. Any formula can be efficiently transformed into a small set of clauses. The calculi contain uniform rules and provide a decision procedure for all logics. Their completeness is based on a new and crucial notion of inconsistency predicate, needed to ensure the usual closure properties of maximal consistent sets. As far as we know the calculi presented here are the first resolution calculi for this class of logics.

Keywords: Modal Logic · Automated Reasoning · Resolution

1 Introduction

Non-normal modal logics (NNMLs) have been studied since the seminal work by Kripke in the 1960s, and then developed prominently by Montague, Segeberg, Scott, and Chellas in the 1970s. They are called *non-normal* as they do not satisfy all axioms of minimal normal modal logic **K**. NNMLs are used in a variety of contexts. In epistemic reasoning they offer a simple (preliminary) solution to the problem of logical omniscience. In deontic logic, they allow to avoid some well-known paradoxes of classical deontic logic, and enable us to represent conflicting obligations. Multi-agent non-normal modalities have been used to capture notions of agency and ability, where $\Box\phi$ is read as "the agent can bring about ϕ", for a formula ϕ [12]. Moreover, the non-normal monotonic logic **EM** coincides with the 2-agent case of Pauly's coalition logic with determinacy. Finally NNMLs are the formalism of choice to express normality and typicality, or truth in most of the cases, as a modality [43].

In this paper we consider the classical cube of NNMLs. It comprises the minimal modal logic **E**, the smallest modal logic closed under congruence (only), and extensions of **E** with one or more of the axioms C, M and N. This results in a

C. Nalon was partially supported by FAPDF 11/2021, DPG/UnB 004/2022.

cube of 8 systems, where the stronger one (defined by all three axioms M, N, and C) is just the normal modal logic **K**. NNMLs have a well-understood semantics defined in terms of neighbourhood models [7]. In these models, each world w is associated with a set of neighbourhoods $N(w)$, where each neighbourhood is a set of worlds itself. If we accept the traditional interpretation of a proposition as the set of worlds in which it holds (its truth set), we can think of $N(w)$ as a set of propositions associated with w, i.e. precisely those propositions that are necessary, known, obligatory, ... at the world w. The classical cube arises by imposing closure properties on the set of neighbourhoods (or propositions) associated with a world, and captured syntactically by the axioms.

From an automated reasoning and proof theoretic view, NNMLs are not as well studied as normal modal logics. Cut-free Gentzen calculi for NNML have been studied in [22,23,25,41,42]. Labelled calculi of different kinds have been proposed in [10,15,37], where the neighbourhood semantics is represented syntactically through two different labels, for worlds and neighbourhoods. Situated between these two approaches, there are calculi that augment sequents with additional structure, but without fully representing the neighbourhood semantics: linear nested sequents with an additional nesting operator [26] and structured hypersequents [9]. All these calculi have different purposes and properties. Cut-free Gentzen calculi typically provide a straightforward decision procedure, in some cases of optimal complexity, and help to prove interpolation [42]. Labelled calculi, and also the approach taken in [9], allow us to extract countermodels of unprovable sequents. The structured calculi of [26] provide a uniform and modular formulation of NNML when extended with axioms of the standard modal cube. An algorithmic alternative to deduction has been proposed in [16], where the satisfiability problem in NNML is reduced to a set of SAT problems. This essentially implements the proof of the complexity bound for these logics given by Vardi [52].

This paper presents a different approach to reasoning in NNMLs and introduces resolution calculi for all logics in the NNML cube. Resolution methods usually rely on normal forms, which not only helps in the design of the inference rules, but also allow for simple implementations. Moreover, although the complexity of the method is high – proofs might be exponential in the size of the input for some problems [21] – resolution for classical logics is widely implemented [11,17,27,28,47,49,50] with excellent performance in practice [48]. Resolution calculi have been designed for several modal logics, including the normal modal logic **K** and its extensions in the modal cube, either as direct method or using translations into more expressive logics, e.g. as in [1–6,8,13,14,29–31,36] and [38–40]. Recent evaluations [18,31–35,44] show that resolution-based provers for **K** also perform well when compared with tableaux, SAT, and translation based procedures for modal logics [11,17–20,24,47,49–51].

To the best of our knowledge, ours are the first resolution calculi for NNMLs. We use a very simple, congruential translation of formulae into sets of local and global clauses, where the latter are required to hold at any point in the model. Completeness is established via canonical models, and the main conceptual novelty is the analysis of maximally consistent sets using *inconsistency predicates*.

As we demonstrate by example, our modal resolution calculus does not derive the modal literal $\neg l$ from a set $\mathcal{C}$ of clauses if $\mathcal{C} \cup \{l\}$ is inconsistent. Rather, it derives a (set of) literals e such that $\{e, l\}$ are inconsistent over $\mathcal{C}$. This allows us to show that maximally consistent sets are negation complete and disjunction complete. Also, inconsistency predicates allow us to lift statements of global satisfiability of clauses to resolution derivability, which in turn establishes premisses of resolution rules that we need to establish completeness.

The paper is structured as follows. In the next section we present the language of NNMLs and their axiomatisations. We then present the calculi for each modal logic in the NNML cube in Sect. 3, together with results for termination and soundness. Completeness is shown in Sect. 4. The completeness results show that proof systems for stronger logics are obtained modularly by adding rules to the weaker systems. We conclude in Sect. 5.

2 Syntax, Semantics, and Axiomatisation

Definition 1. We fix a countable set $\mathcal{V}$ of propositional variables. The *language* $\mathcal{L}$ of the basic unimodal logic is given by the grammar $\mathcal{L} \ni \phi, \psi := p \mid \neg\phi \mid \Box\phi \mid \phi \vee \psi$ where $p \in \mathcal{V}$.

Other connectives $\top, \bot, \wedge, \rightarrow$ and $\Diamond$ are defined in the standard way, and we use the usual operator precedence $\wedge, \vee, \rightarrow, \leftrightarrow$ from strongest to weakest. We denote the set of subformulae of $\phi \in \mathcal{L}$ and their negations by $\mathsf{subf}(\phi)$, where leading double negations are eliminated.

Terminology 2. Variables and their negations are called *propositional literals*, and *modal literals* are of the form $\Box p$ or $\neg\Box p$ where $p \in \mathcal{V}$ is a propositional variable. A *literal* is either a propositional or a modal literal. We write $\mathsf{Lit}(\mathcal{V})$ for the set of literals with variables in $\mathcal{V}$.

Formulae are interpreted with respect to *neighbourhood models.*

Definition 3. A *neighbourhood frame* is a pair (W, N) where W is a set (of worlds) and $N : W \rightarrow \mathcal{P}(\mathcal{P}(W))$ is a (neighbourhood) function, where $\mathcal{P}(S)$ denotes the powerset of S. A *neighbourhood model* is a neighbourhood frame endowed with a valuation, that is, a triple (W, N, θ) where (W, N) is a neighbourhood frame and $\theta : \mathcal{V} \rightarrow \mathcal{P}(W)$ is a (valuation) function.

Definition 4. Truth of a formula $\phi \in \mathcal{L}$ at a world $w \in W$ of a neighbourhood model $M = (W, N, \theta)$ is given inductively by:

$$\begin{aligned} M, w \models p &\iff w \in \theta(p) \\ M, w \models \phi \vee \psi &\iff M, w \models \phi \text{ or } M, w \models \psi \\ M, w \models \neg\phi &\iff M, w \not\models \phi \\ M, w \models \Box\phi &\iff [\![\phi]\!]_M \in N(w) \end{aligned}$$

where $[\![\phi]\!]_M = \{w \in W \mid M, w \models \phi\}$ is the *truth set* of ϕ.

Table 1. Axioms and frame properties, where (W, N) is a frame, $\alpha, \beta \subseteq W$, $w \in W$.

Axiom	Frame Property
C: $(\Box\phi \wedge \Box\psi) \to \Box(\phi \wedge \psi)$	Closed under intersection: $\alpha \in N(w) \wedge \beta \in N(w) \to \alpha \cap \beta \in N(w)$
M: $\Box(\phi \wedge \psi) \to \Box\phi$	Supplemented: $\alpha \in N(w) \wedge \alpha \subseteq \beta \to \beta \in N(w)$
N: $\Box\top$	Contains the unit: $W \in N(w)$

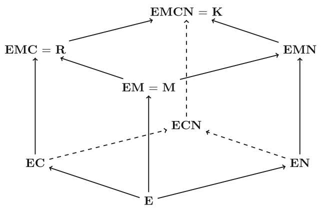


Fig. 1. The classical modal cube. Arrows indicate proper inclusion.

A formula $\phi \in \mathcal{L}$ is *satisfiable* in a neighbourhood model $M = (W, N, \theta)$ if there is $w \in W$ such that $M, w \models \phi$. A set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, $n \in \mathbb{N}$, is *satisfiable* if and only if there is a neighbourhood model (W, N, θ) and a world $w \in W$ such that $M, w \models \gamma_i$, for all $1 \leq i \leq n$. A formula ϕ is *satisfiable in a class $\mathcal{C}$ of neighbourhood models* if there exists $M \in \mathcal{C}$ such that ϕ is satisfiable in M. We denote by $\mathcal{E}$ the class of all neighbourhood models.

The axiomatisation for the minimal logic **E** comprises the axiomatisation of classical propositional logic and the rule RE: from $\phi \leftrightarrow \psi$ derive $\Box\phi \leftrightarrow \Box\psi$. We also consider the extensions of **E** with the axioms given in Table 1. Neighbourhood models modularly characterise the classical cube of NNMLs given in Fig. 1 in the sense that a formula ϕ is a theorem of **E** if and only if it is valid in the class $\mathcal{E}$ of all neighbourhood models [7]. Furthermore, ϕ is a theorem of $\mathbf{E}\Sigma$ with $\Sigma \subseteq \{\text{C,M,N}\}$ if and only if it is valid in the class of neighbourhood models that satisfy each of the additional axioms, whose corresponding frame conditions are given in Table 1. That is, the following holds [7, Theorem 7.5].

Theorem 5. *The logic* **E** *(resp.* **EC**, **EM**, **EN**, **EMC**, **ECN**, **EMN**, **EMCN***) is characterised by the class $\mathcal{E}$ (resp. $\mathcal{EC}$, $\mathcal{EM}$, $\mathcal{EN}$, $\mathcal{EMC}$, $\mathcal{ECN}$, $\mathcal{EMN}$, $\mathcal{EMCN}$) of neighbourhood models.*

We also note that axioms M and N are, respectively, equivalent to the rules RM ($\phi \to \psi$ / $\Box\phi \to \Box\psi$) and RN (ϕ / $\Box\phi$), and that the axiom K ($\Box(\phi \to \psi) \to \Box\phi \to \Box\psi$) is derivable from M and C. As a consequence, the top system **EMCN** is equivalent to **K**, the weakest normal modal logic [7, Theorem 8.9]. Monotonicity and aggregation correspond to *regularity*, that is, the system with both M and C is equivalent to the regular system **R** [7, Theorem 8.11].

We conclude this section by providing the well-known results about the complexity of the satisfiability problem for the logics here considered [52].

Theorem 6. *Let* $\mathbf{E}\Sigma$ *with* $\Sigma \subseteq \{\text{M,N}\}$. *The satisfiability problem for* $\mathbf{E}\Sigma$ *is in* NP *and the satisfiability problem for* $\mathbf{EC}\Sigma$ *is in* PSPACE.

3 Resolution Calculi

Our resolution calculi operates over sets of formulae in a specific normal form: disjunctions of (propositional or modal) literals. Formulae can be transformed into this form by means of renaming [45] which creates new propositions together with their definitions in the resulting formula. The idea here is simple. To translate the formula $\Box\phi$, say, to clausal form, we stipulate $\Box\phi$ to be equivalent to $\Box p$, and additionally p to be equivalent to ϕ – but the latter has to be true in *every* world of a neighbourhood model. Hence $\Box\phi$ is satisfiable if and only if the formulae $\Box p$ and $\mathsf{G}(p \leftrightarrow \phi)$ are satisfiable. Here $\mathsf{G}(\cdot)$ is a global modality that stipulates that a formula is true at every world in a model. For a neighbourhood model (W, N, θ), $w \in W$, and a formula $\phi \in \mathcal{L}$, we have that $M, w \models \mathsf{G}(\phi) \iff M, w' \models \phi$, for all $w' \in W$, where $M, w \models \phi$ is as in Definition 4. Alternatively (and equivalently), $M, w \models \mathsf{G}(\phi) \iff [\![\phi]\!] = W$.

A *clause* is a formula in one of the following forms:

- local clauses: $\bigvee_i l_i$, where the l_i are propositional or modal literals; or
- global clauses: $\mathsf{G}(\bigvee_i l_i)$, where the l_i are propositional or modal literals.

We often think of a clause as a set of literals and sometimes use set notation, that is, we identify $l_1 \vee \ldots \vee l_n$ with the set $\{l_1, \ldots, l_n\}$, for $n \in \mathbb{N}$. This allows us to also use set theoretic notation on clauses. For instance, for a literal l and clause γ, we may write $l \in \gamma$ and say that l is an element of γ. Similarly, $\gamma_1 \subseteq \gamma_2$ means that all literals of γ_1 are literals of γ_2.

It is easy to see that every formula can be represented as a set of clauses. As most logics in the cube are non-monotonic, we only replace the argument of $\Box$ with an equivalent formula. As a consequence, the rewriting steps and introduction of new variables by renaming consistently use bi-implications ($\leftrightarrow$). For a fixed formula $\phi \in \mathcal{L}$, we let $\eta = \eta_\phi : \mathsf{subf}(\phi) \longrightarrow \mathcal{V} \setminus \mathcal{V}(\phi)$ be an injective renaming function that associates a fresh propositional variable to every (possibly negated) subformula of ϕ.

Proposition 7. *A formula* ϕ *is satisfiable if, and only if,* $\{\eta(\phi)\} \cup \mathsf{R}(\mathsf{G}(\eta(\phi) \leftrightarrow \phi))$ *is satisfiable, where* R *is defined as follows and* $t, p \in \mathcal{V}$:

$$\begin{aligned}
\mathsf{R}(\mathsf{G}(t \leftrightarrow p)) &= \{\mathsf{G}(\neg t \vee p), \mathsf{G}(t \vee \neg p)\}\\
\mathsf{R}(\mathsf{G}(t \leftrightarrow \neg\psi)) &= \{\mathsf{G}(\neg t \vee \neg\eta(\psi)), \mathsf{G}(t \vee \eta(\psi))\} \cup \mathsf{R}(\mathsf{G}(\eta(\psi) \leftrightarrow \psi))\\
\mathsf{R}(\mathsf{G}(t \leftrightarrow \psi \vee \psi')) &= \{\mathsf{G}(\neg t \vee \eta(\psi) \vee \eta(\psi')), \mathsf{G}(t \vee \neg\eta(\psi)), \mathsf{G}(t \vee \neg\eta(\psi'))\}\\
&\quad \cup \mathsf{R}(\mathsf{G}(\eta(\psi) \leftrightarrow \psi)) \cup \mathsf{R}(\mathsf{G}(\eta(\psi') \leftrightarrow \psi'))\\
\mathsf{R}(\mathsf{G}(t \leftrightarrow \Box\psi)) &= \{\mathsf{G}(\neg t \vee \Box\eta(\psi)), \mathsf{G}(t \vee \neg\Box\eta(\psi))\} \cup \mathsf{R}(\mathsf{G}(\eta(\psi) \leftrightarrow \psi))
\end{aligned}$$

Moreover, the size of $\{\eta(\phi)\} \cup \{\mathsf{R}(\mathsf{G}(\eta(\phi) \leftrightarrow \phi))\}$ *is linear on the size of* ϕ.

The proof is standard. We can transform a model that satisfies ϕ into a model where $\eta(\phi)$ has exactly the same truth set as ϕ by just changing the valuation of the renaming symbol. Conversely, models that satisfy the transformation are automatically models of ϕ. The number of recursive calls is proportional to the number of subformulae of ϕ, hence the linear complexity bound.

The inference rules for the modal logic **E** and its extensions are given in Table 2. In the table, C and D are clauses, l are literals and p are propositional variables, possibly subscripted or primed. Inference rules are presented using standard notation with premisses and conclusion, called the *resolvent* separated by a horizontal line. Every inference rule except G2L has a local and a global variant, expressed by a leading L (resp. G) in its name. The second letter of the rule name indicates the logic axiomatised by the rule, so that e.g. GMRES is sound for the monotone modal logic **EM**. In the following, we give the intuition for the global inference rules that can be readily translated to their local variants. We consider the following four groups of inference rules.

- *Inference rules for all classical modal logics*: The rule GRES is a syntactical variation of the propositional resolution rule [46], the only differences being that reasoning is carried out within the global modality and that l occurring in the premisses may be a modal literal. The rule G2L asserts that local satisfiability is a consequence of its global counterpart. The rule GERES expresses that $\Box p$ and $\neg\Box p'$ are inconsistent whenever p and p' are globally equivalent, i.e. have the same truth set. By virtue of the side condition, we have three non-redundant instances: (1) $\mathsf{G}(C) = \mathsf{G}(\neg p \vee p')$ and $\mathsf{G}(C') = \mathsf{G}(p \vee \neg p')$, which means that p and p' are semantically equivalent; (2) $\mathsf{G}(C) = \mathsf{G}(\neg p)$ and $\mathsf{G}(C') = \mathsf{G}(\neg p')$, in which case p and p' are globally false and so semantically equivalent; or (3) $\mathsf{G}(C) = \mathsf{G}(p')$ and $\mathsf{G}(C') = \mathsf{G}(p)$, where p and p' are semantically equivalent as they are both globally true. All other instances are already contradictory or can be reduced to the above by means of GRES.

- *Inference rules for classical modal logics with aggregation* that validate the axiom C. The rules GCRES1 and GCRES2 are sound in classical modal logics containing the axiom C. They are similar to the rule GERES, but the side conditions for clauses C_i ensure that $(p_1 \wedge \ldots \wedge p_n \leftrightarrow p)$ is globally true.

- *Inference rules for monotone classical modal logics* that validate the axiom M: The rule GMRES is sound in logics that are monotone. This rule is a weaker version of GERES where congruence is required. For monotone logics, the rule RM (from $\phi \rightarrow \psi$ derive $\Box\phi \rightarrow \Box\psi$) holds. The side condition gives three concrete instances: (1) $C = \mathsf{G}(\neg p \vee p')$, thus, from $\Box p$ in the first premiss we have that $\Box p'$ holds, which contradicts with $\neg\Box p'$ in the second premiss; (2) $C = \mathsf{G}(\neg p)$, that is, p is globally false and, *ex falso sequitur quodlibet*, we again have that $\Box p'$ holds, which contradicts the modal literal in the second premiss; or (3) $C = \mathsf{G}(p')$, from which we can derive $\neg\Box p$, using the contrapositive of RM, which contradicts with the modal literal in the first premiss.

- *Inference rules for classical modal logics with the unit* that validate the axiom N: The rule GNRES is sound for these logics, as the premiss $\mathsf{G}(p)$ says that $\neg\Box p$

Table 2. Inference Rules

$$
\begin{array}{ccccc}
\mathsf{LRES} & \mathsf{GRES} & \mathsf{G2L} & \mathsf{LERES} & \mathsf{GERES} \\
\begin{array}{c} (D \lor l) \\ (D' \lor \neg l) \\ \hline (D \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D \lor l) \\ \mathsf{G}(D' \lor \neg l) \\ \hline \mathsf{G}(D \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D) \\ \hline D \end{array} &
\begin{array}{c} (D \lor \Box p) \\ (D' \lor \neg \Box p') \\ \mathsf{G}(C) \\ \mathsf{G}(C') \\ \hline (D \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D \lor \Box p) \\ \mathsf{G}(D' \lor \neg \Box p') \\ \mathsf{G}(C) \\ \mathsf{G}(C') \\ \hline \mathsf{G}(D \lor D') \end{array}
\end{array}
$$

where $C \subseteq (\neg p \lor p')$ and $C' \subseteq (p \lor \neg p')$

$$
\begin{array}{cccc}
\mathsf{LMRES} & \mathsf{GMRES} & \mathsf{LNRES} & \mathsf{GNRES} \\
\begin{array}{c} (D \lor \Box p) \\ (D' \lor \neg \Box p') \\ \mathsf{G}(C) \\ \hline (D \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D \lor \Box p) \\ \mathsf{G}(D' \lor \neg \Box p') \\ \mathsf{G}(C) \\ \hline \mathsf{G}(D \lor D') \end{array} &
\begin{array}{c} (D \lor \neg \Box p) \\ \mathsf{G}(p) \\ \hline D \end{array} &
\begin{array}{c} \mathsf{G}(D \lor \neg \Box p) \\ \mathsf{G}(p) \\ \hline \mathsf{G}(D) \end{array}
\end{array}
$$

where $C \subseteq (\neg p \lor p')$

$$
\begin{array}{cc}
\mathsf{LCRES1} & \mathsf{GCRES1} \\
\begin{array}{c} (D_1 \lor \Box p_1) \\ \ldots \\ (D_n \lor \Box p_n) \\ (D' \lor \neg \Box p) \\ \mathsf{G}(\neg p_1 \lor \ldots \lor \neg p_n \lor p) \\ \mathsf{G}(C_1) \\ \ldots \\ \mathsf{G}(C_n) \\ \hline (D_1 \lor \ldots \lor D_n \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D_1 \lor \Box p_1) \\ \ldots \\ \mathsf{G}(D_n \lor \Box p_n) \\ \mathsf{G}(D' \lor \neg \Box p) \\ \mathsf{G}(\neg p_1 \lor \ldots \lor \neg p_n \lor p) \\ \mathsf{G}(C_1) \\ \ldots \\ \mathsf{G}(C_n) \\ \hline \mathsf{G}(D_1 \lor \ldots \lor D_n \lor D') \end{array}
\end{array}
$$

where $C_i \subseteq (\neg p \lor p_i)$ and $p_i \in C_i$

$$
\begin{array}{cc}
\mathsf{LCRES2} & \mathsf{GCRES2} \\
\begin{array}{c} (D_1 \lor \Box p_1) \\ \ldots \\ (D_n \lor \Box p_n) \\ (D' \lor \neg \Box p) \\ \mathsf{G}(\neg p_1 \lor \ldots \lor \neg p_n) \\ \mathsf{G}(\neg p) \\ \hline (D_1 \lor \ldots \lor D_n \lor D') \end{array} &
\begin{array}{c} \mathsf{G}(D_1 \lor \Box p_1) \\ \ldots \\ \mathsf{G}(D_n \lor \Box p_n) \\ \mathsf{G}(D' \lor \neg \Box p) \\ \mathsf{G}(\neg p_1 \lor \ldots \lor \neg p_n) \\ \mathsf{G}(\neg p) \\ \hline \mathsf{G}(D_1 \lor \ldots \lor D_n \lor D') \end{array}
\end{array}
$$

(or its global occurrence) cannot be satisfied, therefore it must be the case that the resolvent $\mathsf{G}(D)$ is satisfied.

The basic resolution calculus, $\mathsf{RES}_{\mathbf{E}}$, comprises the inference rules LRES, GRES, G2L, LERES and GERES. For the extensions of **E**, the calculi can be obtained in a modular way, that is, by just adding the rules that are sound with respect to the axioms for the logic. However, it is easy to see that, for

Table 3. Inference rules corresponding to each logic

Calculus	Inference Rules
$\mathsf{RES}_{\mathbf{E}}$	LRES, GRES, G2L, LERES, GERES
$\mathsf{RES}_{\mathbf{EC}}$	LRES, GRES, G2L, LCRES1, GCRES1, LCRES2, GCRES2
$\mathsf{RES}_{\mathbf{EM}}$	LRES, GRES, G2L, LMRES, GMRES
$\mathsf{RES}_{\mathbf{EN}}$	LRES, GRES, G2L, LERES, GERES, LNRES, GNRES
$\mathsf{RES}_{\mathbf{EMC}}$	LRES, GRES, G2L, LCRES1, GCRES1, LCRES2, GCRES2, LMRES, GMRES
$\mathsf{RES}_{\mathbf{ECN}}$	LRES, GRES, G2L, LCRES1, GCRES1, LCRES2, GCRES2, LNRES, GNRES
$\mathsf{RES}_{\mathbf{EMN}}$	LRES, GRES, G2L, LMRES, GMRES, LNRES, GNRES
$\mathsf{RES}_{\mathbf{EMCN}}$	LRES, GRES, G2L,LCRES1, GCRES1, LCRES2, GCRES2, LMRES, GMRES, LNRES, GNRES

instance, when considering monotone logics, whenever LERES or GERES can be applied, the rules LMRES or GMRES can also be applied, generating exactly the same resolvent. Thus, LERES and GERES are both redundant in the calculi for monotone logics. In Table 3 we give the rules for the calculus for each considered logic, but where redundant inference rules are suppressed. We denote by $\mathsf{RES}_{\mathsf{L}}$ the resolution calculus for a particular logic **L**.

The following definitions are needed before we establish our main results.

Definition 8. Let $\mathcal{C}$ be a finite set of clauses and $\mathsf{L} = \mathbf{E}\Sigma$ with $\Sigma \subseteq \{\text{C,M,N}\}$. A derivation from $\mathcal{C}$ in $\mathsf{RES}_{\mathsf{L}}$ is a sequence of sets of clauses $\mathcal{C}_0, \mathcal{C}_1, \ldots$ where $\mathcal{C}_0 = \mathcal{C}$ and for every $i \in \mathbb{N}$, $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{D\}$ where the resolvent D was obtained from $\mathcal{C}_i$ by applying the rules of $\mathsf{RES}_{\mathsf{L}}$ given in Table 3. We require that $D \notin \mathcal{C}_i$ and that D is not a tautology (that is, a clause containing l and $\neg l$).

Definition 9. Let $\mathcal{C}$ be a finite set of clauses and $\mathcal{C}_0, \mathcal{C}_1, \ldots$ a derivation from $\mathcal{C}$ in $\mathsf{RES}_{\mathsf{L}}$ where $\mathsf{L} = \mathbf{E}\Sigma$ with $\Sigma \subseteq \{\text{C,M,N}\}$. If there is $k \in \mathbb{N}$ such that $\epsilon \in \mathcal{C}_k$, then $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k$ is a *refutation* of $\mathcal{C}$. If there is $k \in \mathbb{N}$ such that any resolvent D obtained from $\mathcal{C}_k$ by applying the rules of $\mathsf{RES}_{\mathsf{L}}$ given in Table 3 to $\mathcal{C}_k$ is such that $D \in \mathcal{C}_k$, then $\mathcal{C}_k$ is *saturated*, and $\mathcal{C}_k$ is the *saturation* of $\mathcal{C}$.

The following two theorems establish termination and soundness of the calculi.

Theorem 10. *Let* $\mathsf{L} = \mathbf{E}\Sigma$ *with* $\Sigma \subseteq \{\text{C,M,N}\}$, $\mathcal{C}$ *be a finite set of clauses and* $\mathcal{C}_0, \mathcal{C}_1, \ldots$ *be a derivation from* $\mathcal{C}$ *in* $\mathsf{RES}_{\mathsf{L}}$. *Then there is* $k \in \mathbb{N}$ *such that* $\mathcal{C}_k$ *is saturated, or* $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k$ *is a refutation.*

As there is a finite number of literals in $\mathcal{C}$ and no inference rule introduces new literals, there is also an upper bound on the number of clauses that can be generated by $\mathsf{RES}_{\mathsf{L}}$. Hence either the empty clause is generated at some $\mathcal{C}_k$ or no new clauses can be generated. Thus, any derivation in $\mathsf{RES}_{\mathsf{L}}$ terminates.

Theorem 11. *Let* $\mathsf{L} = \mathbf{E}\Sigma$ *with* $\Sigma \subseteq \{\text{C,M,N}\}$. *Then* $\mathsf{RES}_{\mathsf{L}}$ *is sound.*

The proof is by induction on the number of steps of a derivation: as every step of a derivation is satisfiability preserving, as argued above, then all derivations from satisfiable sets of clauses only generate satisfiable sets of clauses.

We present two examples before establishing completeness in the next section.

Example 12. We show that $\Box(p \vee q) \rightarrow \Box(p \vee \neg\Box(a \vee \neg a) \vee q)$ is valid in the logic **EN** by using the calculus $\mathsf{RES}_{\mathbf{EN}}$. For the refutation, we negate the formula and obtain $\phi = \Box(p \vee q) \wedge \neg\Box(p \vee \neg\Box(a \vee \neg a) \vee q)$. We show next the relevant clauses resulting from the transformation, where we have that $\phi_1 = \Box(p \vee q)$, $\phi_2 = \neg\Box(p \vee \neg\Box(a \vee \neg a) \vee q)$, and $\phi_3 = (p \vee \neg\Box(a \vee \neg a) \vee q)$:

1. t_ϕ	8. $\mathsf{G}(\neg t_{\phi_2} \vee \neg\Box t_{\phi_3})$
2. $\mathsf{G}(\neg t_\phi \vee t_{\phi_1})$	9. $\mathsf{G}(\neg t_{\phi_3} \vee t_p \vee t_q \vee \neg\Box t_{a\vee\neg a})$
3. $\mathsf{G}(\neg t_\phi \vee t_{\phi_2})$	10. $\mathsf{G}(t_{\phi_3} \vee \neg t_p)$
4. $\mathsf{G}(\neg t_{\phi_1} \vee \Box t_{p\vee q})$	11. $\mathsf{G}(t_{\phi_3} \vee \neg t_q)$
5. $\mathsf{G}(\neg t_{p\vee q} \vee t_p \vee t_q)$	12. $\mathsf{G}(t_{a\vee\neg a} \vee \neg t_a)$
6. $\mathsf{G}(t_{p\vee q} \vee \neg t_p)$	13. $\mathsf{G}(t_{a\vee\neg a} \vee \neg t_{\neg a})$
7. $\mathsf{G}(t_{p\vee q} \vee \neg t_q)$	14. $\mathsf{G}(t_{\neg a} \vee t_a)$

The steps of the refutation are as follows:

15. $\mathsf{G}(t_{a\vee\neg a} \vee t_a)$	$[\mathsf{GRES}, 13, 14]$	21. $\mathsf{G}(t_{\phi_3} \vee \neg t_{p\vee q})$	$[\mathsf{GRES}, 20, 10]$
16. $\mathsf{G}(t_{a\vee\neg a})$	$[\mathsf{GRES}, 15, 12]$	22. $\mathsf{G}(\neg t_{\phi_1} \vee \neg t_{\phi_2})$	$[\mathsf{GERES}, 4, 8, 19, 21]$
17. $\mathsf{G}(\neg t_{\phi_3} \vee t_p \vee t_q)$	$[\mathsf{GNRES}, 16, 9]$	23. $\mathsf{G}(\neg t_\phi \vee \neg t_{\phi_1})$	$[\mathsf{GRES}, 22, 3]$
18. $\mathsf{G}(\neg t_{\phi_3} \vee t_{p\vee q} \vee t_p)$	$[\mathsf{GRES}, 17, 7]$	24. $\mathsf{G}(\neg t_\phi)$	$[\mathsf{GRES}, 23, 2]$
19. $\mathsf{G}(\neg t_{\phi_3} \vee t_{p\vee q})$	$[\mathsf{GRES}, 18, 6]$	25. $\neg t_\phi$	$[\mathsf{G2L}, 24]$
20. $\mathsf{G}(t_{\phi_3} \vee \neg t_{p\vee q} \vee t_p)$	$[\mathsf{GRES}, 11, 5]$	26. ϵ	$[\mathsf{LRES}, 25, 1]$

Example 13. We now show that $\phi = \Box p \wedge \Box q \rightarrow \Box(p \wedge q)$ is valid in **EC**. The transformation of $\neg\phi$ produces, among others, Clauses (1)–(7). The refutation is refreshingly short: it is obtained in two steps after an application of $\mathsf{GCRES1}$:

1. t_ϕ	6. $\mathsf{G}(\neg t_{p\wedge q} \vee t_q)$	
2. $\mathsf{G}(\neg t_\phi \vee \Box t_p)$	7. $\mathsf{G}(t_{p\wedge q} \vee \neg t_p \vee \neg t_q)$	
3. $\mathsf{G}(\neg t_\phi \vee \Box t_q)$	8. $\mathsf{G}(\neg t_\phi)$	$[\mathsf{GCRES1}, 2, 3, 4, 5, 6, 7]$
4. $\mathsf{G}(\neg t_\phi \vee \neg\Box t_{p\wedge q})$	9. $\neg t_\phi$	$[\mathsf{G2L}, 8]$
5. $\mathsf{G}(\neg t_{p\wedge q} \vee t_p)$	10. ϵ	$[\mathsf{LRES}, 9, 1]$

4 Completeness

We prove completeness by means of a canonical model construction. Our maximally consistent sets comprise both local and global clauses. The proof of the truth lemma hinges on the fact that maximally consistent sets are negation complete, that is, they contain either a literal or its negation. In completeness proofs of Hilbert systems, the argument is as follows. If M is a maximally consistent set, and neither $\phi \in M$ nor $\neg\phi \in M$, then both $M \cup \{\phi\}$ and $M \cup \{\neg\phi\}$ are inconsistent, that is, $M \cup \{\phi\} \vdash\bot$ and $M \cup \{\neg\phi\} \vdash\bot$. Hence $M \vdash \neg\phi$ and $M \vdash \phi$ which contradicts the consistency of M, so that our supposition that neither $\phi \in M$ nor $\neg\phi \in M$ must have been false.

However, this argument is not available for resolution calculi, where we take a set $\mathcal{C}$ of local or global clauses to be consistent if $\mathcal{C} \nvdash \epsilon$. In the simplest calculus, $\mathsf{RES}_{\mathbf{E}}$, consider the set $\mathcal{C} = \{\mathsf{G}(\neg p \vee q), \mathsf{G}(\neg q \vee p), \neg\Box q\}$. Then clearly $\mathcal{C} \cup \{\Box p\} \vdash \epsilon$, but it is patently false that $\mathcal{C} \vdash \neg\Box p$.

However, something nearly as useful eventuates: We have that $\mathcal{C} \vdash \neg\Box q$, and $\Box p$ and $\neg\Box q$ together are inconsistent over $\mathcal{C}$ (using a single application of LERES). That is, while we cannot derive $\neg\Box p$, at least we can derive a literal, here $\neg\Box q$, that is inconsistent with $\Box p$ over $\mathcal{C}$. This is captured in the notion of inconsistency predicate, where, in full generality, we need to consider the inconsistency of n-element sets to accommodate instances of LNRES (where we are going to designate singleton sets as inconsistent) and the LCRES rules (where inconsistent sets can contain any finite number of elements). We formulate this for an arbitrary resolution calculus.

Definition 14. A *modal resolution calculus* is a relation $\vdash$ between clause sets and clauses that is closed under propositional resolution. That is, $\mathcal{C} \vdash D \vee l$ and $\mathcal{C} \vdash D' \vee \neg l$ then $\mathcal{C} \vdash D \vee D'$, for all local clauses D and literals l. Let $\vdash$ be a modal resolution calculus and $\mathcal{C}$ be a set of global clauses. An *inconsistency predicate* for $\mathcal{C}$ and $\vdash$ is a subset $\mathsf{P} \subseteq \mathcal{P}(\mathsf{Lit}(\mathcal{V}))$ such that the following three conditions hold:

1. Every element $I = \{l_1, \ldots, l_n\} \in \mathsf{P}$ is inconsistent over $\mathcal{C}$, that is, there are global clauses $\Gamma_1, \ldots, \Gamma_n$ such that $\{\Gamma_1, \ldots, \Gamma_k, l_1, \ldots, l_n\} \vdash \epsilon$ and $\mathcal{C} \vdash \Gamma_i$ for all $1 \leq i \leq k$.
2. The set P is closed under cut, that is $A \cup B \in \mathsf{P}$ whenever $A \cup \{l\} \in \mathsf{P}$ and $B \cup \{\neg l\} \in \mathsf{P}$.
3. Propositional literals are only inconsistent with their negations, i,e. $A = \{p, \neg p\}$ whenever $p \in A \in \mathsf{P}$ for a propositional variable $p \in \mathcal{V}$.

The formulation of inconsistency predicate instantiates to all modal calculi in the paper, where for a calculus RES, we say that $\mathcal{C} \vdash D$ if D is in the saturation of $\mathcal{C}$. We think of an element $\{l_1, \ldots, l_n\}$ of an inconsistency predicate not as a clause, but rather as a conjunction of singleton clauses (that is inconsistent as per the first requirement). The second requirement formalises the semantically sound condition $\bigcap_i a_i \cap \bigcap_j b_j = \emptyset$ whenever $x \cap \bigcap_i a_i = \emptyset = (W \setminus x) \cap \bigcap_j b_j$ for subsets $x, a_i, b_j \subseteq W$ of a set W. We require that, in the formulation of the condition, that $A \cup B$ is inconsistent, i.e., $\mathcal{C}$ proves a sufficient number of global clauses Γ that, together with $A \cup B$, allows us to derive the empty clause ϵ.

As an example, and a stepping stone to prove the completeness of classical modal logic, we have the following:

Lemma 15. *Let $\vdash$ be the calculus for classical modal logic and let $\mathcal{C}$ be a set of global clauses. Then the set P_E containing*

- *the set* $\{l, \neg l\}$ *for every (propositional or modal) literal* $l \in \mathsf{Lit}(\mathcal{V})$*, and*
- *the set* $\{\Box p, \neg\Box q\}$ *for every pair* $p, q \in \mathcal{V}$ *of propositions such that* $\mathcal{C} \vdash \mathsf{G}(C)$ *and* $\mathcal{C} \vdash \mathsf{G}(C')$ *for sub-clauses* $C \subseteq (\neg p \vee q)$ *and* $C' \subseteq (\neg q \vee p)$*.*

is an inconsistency predicate for $\vdash$ *and* $\mathcal{C}$*.*

Proof (Sketch). The inconsistency requirement is clear, as every element of an inconsistency predicate is an instance of a resolution rule. For cut closure, apply GRES to premisses of a rule inducing a cut.

The following definition is an adaptation of the deduction theorem to modal resolution calculi. The reader is encouraged to instantiate this to the case of the modal logic **E** (and the inconsistency predicate of Lemma 15), as we do in the example following the definition.

Definition 16. An inconsistency predicate P is *compatible* with a modal resolution calculus $\vdash$ if for every local clause D and every (propositional or modal) literal l with $\mathcal{C} \cup \{l\} \vdash D$, either $D = l$ or there is $n \geq 0$ and $D_1, \ldots, D_n$, $E_1, \ldots, E_n$ such that

- $D = D_1 \vee \cdots \vee D_n$
- $\mathcal{C} \vdash E_i \vee D_i$ for all $1 \leq i \leq n$
- $\{l, e_1, \ldots, e_n\} \in \mathsf{P}$ for all $e_1, \ldots, e_n$ with $e_i \in E_i$.

For the case of classical modal logic, the definition of compatibility takes the following form.

Example 17. If $\vdash$ is the resolution calculus for the classical modal logic **E**, the inconsistency predicate P_E from Lemma 15 is binary. As a consequence, the above definition can only be instantiated with $n = 1$. Hence P_E is compatible, if for all literals l and all local clauses D with $\mathcal{C} \cup \{l\} \vdash D$ either $D = l$ or there is a local clause E such that $\mathcal{C} \vdash E \vee D$ and $\{l, e\} \in \mathsf{P}_E$ for all $e \in E$.

As a second example, and to make further progress to completeness of the resolution calculus $\vdash$ for classical modal logic, we establish that the inconsistency predicate P_E from Lemma 15 is indeed compatible.

Lemma 18. *The inconsistency relation* P_E *from Lemma 15 is compatible with the resolution calculus* $\vdash$ *for classical modal logic.*

The proof proceeds by induction on the derivation of $\mathcal{C} \cup \{l\}$ and is omitted.

Finally, we can reap some of the benefits of our work, and take the next step towards showing that maximally consistent sets are negation complete, i.e. for every literal l, they contain either l or $\neg l$.

Lemma 19. *Let* $\mathcal{C}$ *be a set of local or global clauses,* l *be a literal and* P *be a compatible inconsistency predicate. If* $\mathcal{C} \cup \{l\} \vdash \epsilon$ *and* $\mathcal{C} \cup \{\neg l\} \vdash \epsilon$*, then* $\mathcal{C} \vdash \epsilon$*.*

Proof. We demonstrate the proof for the special case of a binary inconsistency relation P, i.e. every set $A \in \mathsf{P}$ has two elements. As $\mathcal{C} \cup \{l\} \vdash \epsilon$, we have a local clause E such that $\mathcal{C} \vdash E$, and $\{e, l\} \in \mathsf{P}$ for all $e \in E$ by compatibility. Similarly, as $\mathcal{C} \cup \{\neg l\} \vdash \epsilon$, we have a local clause E' with $\{\neg l, e'\} \in \mathsf{P}$ for all $e' \in E'$. If either $E = \epsilon$ or $E' = \epsilon$ we are done. If not, we have $\{e, e'\} \in \mathsf{P}$ for all $e \in E$ and $e' \in E'$ as P is cut closed. This allows us to construct a resolution proof of ϵ from $\mathcal{C} \vdash E$ and $\mathcal{C} \vdash E'$ as P is an inconsistency predicate.

Remark 20. For classical modal logic, we have shown that $\mathcal{C} \cup \{l\} \vdash D$, then either $D = l$ or $\mathcal{C} \vdash E \vee D$ where $\{l, e\} \in \mathsf{P}$ for all $e \in E$, where P is the inconsistency predicate from Lemma 15.

One might hypothesise whether E can always be chosen to be a singleton, or at least a sub-singleton. We show, by means of example, that neither is the case. First, we cannot always choose E as singleton: For $\mathcal{C} = \{p\}$ and $l = q$, we have that $\mathcal{C} \cup \{l\} \vdash p$ but we do not have $\mathcal{C} \vdash E \vee p$ for any singleton clause E (here, $E = \epsilon$ satisfies the condition).

We also cannot always choose E to be a sub-singleton clause. For example, put $\mathcal{C} = \{\neg\Box q \vee \neg\Box p \vee D, \mathsf{G}(\neg p \vee q), \mathsf{G}(p \vee \neg q)\}$. Then $\mathcal{C} \cup \{\Box p\} \vdash D$, but there is no sub-singleton clause E so that $\mathcal{C} \vdash E \vee D$.

We have now collected all the preliminaries to define and investigate maximally consistent sets, i.e. the worlds of the canonical model.

Definition 21. Let $\mathcal{C}$ be a set of global clauses. A *local extension* of $\mathcal{C}$ is a set M of clauses that extends $\mathcal{C}$ by local clauses only. That is, a local extension of $\mathcal{C}$ is a set M of clauses that satisfies $\{\Gamma \in M \mid \Gamma \text{ global}\} = \mathcal{C}$.

A local extension of $\mathcal{C}$ is *maximally consistent* if M is consistent ($M \not\vdash \epsilon$) and every other consistent local extension of M' of Γ that encompasses M ($M' \supseteq M$) satisfies $M = M'$.

Calculi with a compatible inconsistency relation are negation complete.

Lemma 22. *Let $\vdash$ be a modal calculus with a compatible inconsistency relation, and let M be a maximally consistent local extension of a set $\mathcal{C}$ of global clauses. Then, for every (propositional or modal) literal l, we have $l \in M$ or $\neg l \in M$.*

Proof. If neither $l \in M$ nor $\neg l \in M$, then $M \cup \{l\} \vdash \epsilon$ and $M \cup \{\neg l\} \vdash \epsilon$. Applying Lemma 19 now contradicts the consistency of M.

As we have insisted that resolution calculi are closed under propositional resolution, they are also disjunction complete:

Corollary 23. *Let $\vdash$ be a modal resolution calculus with a compatible inconsistency relation, and let M be a maximally consistent local extension of a set $\mathcal{C}$ of global clauses. If $l_1 \vee \cdots \vee l_n \in M$, then there exists $1 \leq i \leq n$ such that $l_i \in M$.*

Proof. If neither $l_i \in M$, then all $\neg l_i \in M$ and we conclude inconsistency of M.

Compatible inconsistency predicates allow us to assert properties relative to derivations of a clause with the help of an additional singleton clause. The following lemma generalises this to a finite number of singleton clauses, but requires that the singleton clauses be *propositional*. This allows us to harness the fact that propositional literals are only inconsistent with their negation, which is enough to establish the hypotheses of the form $\mathsf{G}(C)$ where $C \subseteq D$ is a sub-clause of a propositional clause D.

Lemma 24. *Let $\vdash$ be a modal resolution calculus with compatible inconsistency predicate. Moreover, suppose that $\mathcal{C}$ is a set of global clauses, $l_1, \ldots, l_n$ are propositional literals and D is a (local) clause such that $l_i \notin D$ for all $i = 1, \ldots, n$, and $\mathcal{C} \cup \{l_1, \ldots, l_n\} \vdash D$. Then there is a sub-clause $E_0 \subseteq \neg l_1 \vee \cdots \vee \neg l_n$ such that $\mathcal{C} \vdash E \vee D$.*

Proof. By induction on the number n of literals, where $n = 0$ is evident. If $\mathcal{C} \cup \{l_1, \ldots, l_{n+1}\} \vdash D$, we have that $\mathcal{C} \cup \{l_1, \ldots, l_n\} \vdash E_0 \vee D$ where $\{e, l_{n+1}\} \in \mathsf{P}$, for all $e \in E_0$. This implies that either $E_0 = \epsilon$ or $E_0 = \neg l_{n+1}$. The claim follows by applying the inductive hypothesis.

The above lemma *fails* without assuming that the l_i are propositional literals, as illustrated by the example at the beginning of this section.

In the proof of the truth lemma, we need to show derivability of premisses (of modal rules) based on the truth set of formulae in maximally consistent sets. The following corollary establishes this for local clauses, which we will then lift to global derivability.

Corollary 25. *Consider a modal resolution calculus with a compatible inconsistency predicate, and let $\mathcal{C}$ be a set of global clauses, and let $D = l_1 \vee \cdots \vee l_n$ be a propositional clause such that all maximally consistent local extensions M of $\mathcal{C}$ contain at least one l_i $(i = 1, \ldots, n)$. Then there exists a sub-clause $D_0 \subseteq D$ such that $\mathcal{C} \vdash D_0$.*

The next property is obviously present in the calculus RE and its extensions.

Definition 26. A modal resolution calculus has the *global lifting property* if, for any set $\mathcal{C}$ of *global* clauses, and a *local* clause D, we have that $\mathcal{C} \vdash \mathsf{G}(D)$ whenever $\mathcal{C} \vdash D$.

For our calculi, this essentially means that rules with a global clause as a conclusion only have global clauses as premisses.

Lemma 27. *The calculus $\mathsf{RES}_{\mathbf{E}}$, as well as all other calculi discussed in this paper, has the global lifting property.*

We finally turn to canonical models, where we isolate the construction that is identical for all of the logics that we treat here.

Definition 28 (Canonical Model). Let $\mathcal{C}$ be a set of global clauses. The *$\mathcal{C}$-canonical model*, or the *canonical model based on $\mathcal{C}$*, is the triple (W, N, θ) where

- W is the set of all maximally consistent local extensions of $\mathcal{C}$
- $\theta(p) = \{M \in W \mid p \in M\}$
- $N(M) = \{\theta(p) \mid \Box p \in M\}$.

Here, consistent and maximally consistent refers to consistency in the modal resolution calculus $\mathsf{RES}_{\mathbf{E}}$ for classical modal logic.

This gives the truth lemma for classical modal logic.

Lemma 29 (Truth Lemma). *For the calculus* RE*, let* (W, N, θ) *be the* $\mathcal{C}$*-canonical model for some set* $\mathcal{C}$ *of global clauses. Then, for* $M \in W$*,* $M \models \Gamma$ *whenever* $\Gamma \in M$*, for all local clauses* Γ*.*

Proof. By disjunction completeness, it suffices to show the claim for singleton clauses. The propositional cases and $\Box p \in M$ are easy. For the only interesting case assume $\neg\Box p \in M$, and assume for a contradiction that $\theta(p) \in N(M)$. By construction, there must be a variable $q \in \mathcal{V}$ with $\Box q \in M$ and $\theta(p) = \theta(q)$. That is $p \in M' \iff q \in M'$ for all maximally consistent local extensions M' of $\mathcal{C}$. By Corollary 25 and Lemma 27 we obtain the premisses of the modal rule that proves $M \vdash \epsilon$, contradiction.

Remark 30. In the proof of the truth lemma, the modal rule was only used in a very specific form, i.e. $D = D' = \epsilon$ in definition of the modal rule. The more general form of the rule is needed to establish Lemma 18. The reader is also invited to convince themselves that completeness fails without the more general form, for example to show that $\mathcal{C} = \{\mathsf{G}(\neg p \vee q), \mathsf{G}(\neg q \vee p), \mathsf{G}(\neg q \vee r), \mathsf{G}(\neg r \vee q), \neg\Box p \vee \neg\Box q, \Box r\}$ is inconsistent.

We have used the rule GRES in the proof of Lemma 18. The rule GERES is hidden in the proof of Lemma 27. The reader is invited to convince themselves that GERES is needed to show the inconsistency of $\{\mathsf{G}(\neg p \vee q \vee \Box r), \mathsf{G}(p \vee \neg q), \mathsf{G}(\neg\Box s), \mathsf{G}(s), \mathsf{G}(r), \mathsf{G}(\neg\Box q)\}$.

Corollary 31. *Let* $\mathcal{C}$ *be a set of local or global clauses. If* $\mathcal{C}$ *is unsatisfiable in the class of neighbourhood models, then* $\mathcal{C} \vdash \epsilon$*.*

4.1 Monotone Modal Logic

To show completeness for the resolution calculus for monotone modal logic, we follow the same approach, and start with a compatible inconsistency predicate.

Lemma 32. *Let* $\vdash$ *be the calculus for monotone modal logic and let* $\mathcal{C}$ *be a set of global clauses. Then the set* P_M *containing*

- *the set* $\{l, \neg l\}$ *for every (propositional or modal) literal* $l \in \mathsf{Lit}(\mathcal{V})$*, and*
- *the set* $\{\Box p, \neg\Box q\}$ *for every pair* $p, q \in \mathcal{V}$ *of propositions such that* $\mathcal{C} \vdash \mathsf{G}(C)$ *for a sub-clauses* $C \subseteq \neg p \vee q$*.*

is a compatible inconsistency predicate for $\vdash$ *and* $\mathcal{C}$*.*

The proof is very similar to that of classical modal logic (Lemma 15 and Lemma 18). The canonical model construction is an adaptation of the construction for **E** where the construction ensures that the set of neighbourhoods is upward closed.

Definition 33. Let $\mathcal{C}$ be a set of global clauses. The *$\mathcal{C}$-canonical model* for the calculus $\mathsf{RES}_{\mathbf{EM}}$ is the triple (W, N, θ) where W and θ are the same as for classical modal logic (Definition 28) and the neighbourhood function N is defined by

$$N(M) = \{\alpha \subseteq W \mid \theta(p) \subseteq \alpha \text{ for some } \Box p \in M\}.$$

where $M \in W$ is a maximally consistent, local extension of $\mathcal{C}$.

It is obvious that canonical models for $\mathsf{RES}_{\mathbf{EM}}$ are monotone by construction, but we need to re-establish the truth lemma for the calculus $\mathsf{RES}_{\mathbf{EM}}$ as the construction of the model has changed.

Lemma 34 (Truth Lemma for EM). *For the calculus $\mathsf{RES}_{\mathbf{EM}}$, let (W, N, θ) be the $\mathcal{C}$-canonical model for some set $\mathcal{C}$ of global clauses. Then, for $M \in W$, $M \models \Gamma$ whenever $\Gamma \in M$, for all local clauses Γ.*

The proof is in fact a simplification of the corresponding proof for classical modal logic, and we obtain completeness similar to Corollary 31.

Corollary 35. *Monotone modal logic is complete, i.e. any consistent set $\mathcal{C}$ of local or global clauses satisfies $\mathcal{C} \vdash \epsilon$ whenever $\mathcal{C}$ is unsatisfiable in the class of monotone neighbourhood models.*

4.2 Logics with Unit

We now adapt the construction to also incorporate logics with unit, i.e. the modal logics **EN** and **EMN** that – in addition to the frame conditions for **E** and **EM** – additionally require that the entire set of worlds is always a neighbourhood of any world. To show completeness for these logics, we need to provide a compatible inconsistency relation, which – in contrast to the logics **E** and **EM** – will no longer be binary.

Lemma 36. *Let $\vdash$ be the calculus $\mathsf{RES}_{\mathbf{EN}}$ (resp. $\mathsf{RES}_{\mathbf{EMN}}$) and let $\mathcal{C}$ be a set of global clauses. Let $U = \{\neg\Box p \mid \mathcal{C} \vdash \mathsf{G}(p)\}$. Then the set $\mathsf{P} \cup U$ is compatible inconsistency predicate for $\vdash$ and $\mathcal{C}$, where P is the inconsistency relation for the calculus $\mathsf{RES}_{\mathbf{E}}$ (resp. $\mathsf{RES}_{\mathbf{EM}}$).*

Proof. The inconsistency requirement follows as the predicate closely resembles the modal rules of the calculus. To see cut closure, suppose that $\{\neg\Box p\}$ and $\{\neg\Box q, \Box p\} \in \mathsf{P} \cup U$. Then the premisses that derive inconsistency of both sets can be combined to derive inconsistency of the cut $\{\neg\Box q\}$. For compatibility, we additionally need to consider the case $n = 0$ from Example 17, and extend the inductive proof of Lemma 18, where LNRES as last applied rule precisely induces this case.

This allows us to show completeness, again with a slight variation of the canonical model construction. The definition of the canonical model just adds the entire set of worlds to all neighbourhoods.

Definition 37. The canonical model for the logic **EN** and **EMN** is the triple (W, N, θ) where W and N are as for the logic **E** (or **EM**) and $N(w) = N_0(w) \cup \{W\}$, where N_0 is the neighbourhood function of the canonical model for the logic **E** (resp. **EM**).

The truth lemma follows as before, where we apply the rule LNRES to show inconsistency in case $W \in N(\theta)$.

Lemma 38 (Truth Lemma for EN and EMN). *Let (W, N, θ) be the canonical model for the logic* **EN** *or* **EMN***, respectively, over a set $\mathcal{C}$ of global clauses. Then, for $M \in W$, $M \models \Gamma$ whenever $\Gamma \in M$, for all local clauses Γ.*

Proof. In addition to the cases for **E** and **EM**, consider, for a contradiction, that $\neg\Box p \in M$ and $M \models \Box p$ where $\theta(p) = W$. In this case, $\mathcal{C} \vdash \mathsf{G}(p)$ whence $M \vdash \epsilon$, contradicting consistency of M using LNRES.

Completeness for **EN** and **EMN** follows as before.

Corollary 39. *The calculi $\mathsf{RES}_{\mathbf{EN}}$ and $\mathsf{RES}_{\mathbf{EMN}}$ are complete, i.e. $\mathcal{C} \vdash \epsilon$ whenever $\mathcal{C}$ is inconsistent, for any set $\mathcal{C}$ of global clauses.*

4.3 Logics with Aggregation

We now turn to completeness for logics that additionally satisfy aggregation, i.e. the axiom C from Table 1. Our proof strategy is entirely similar to that of the previous cases, and we start with a compatible inconsistency relation. The format of the LCRES-rules is precisely chosen for the inconsistency relation below to be closed under cut which necessitates to generalise the C-axiom from binary conjunctions to arbitrary finite conjunctions.

Lemma 40. *Let P be the inconsistency relation for the calculi $\mathsf{RES}_{\mathbf{E}}$, $\mathsf{RES}_{\mathbf{EM}}$, $\mathsf{RES}_{\mathbf{EN}}$ or $\mathsf{RES}_{\mathbf{EMN}}$, and let*

$$\begin{aligned} U = &\{\{\neg\Box p_0, \Box p_1, \dots, \Box p_n\} \mid \mathcal{C} \vdash \mathsf{G}(C_i) \textit{ for } i = 0, \dots, n \textit{ and clauses} \\ &\qquad C_0 \subseteq \neg p_0 \vee p_1 \vee \cdots \vee p_n, C_i \subseteq \neg p_0 \vee p_i \textit{ for } i = 1, \dots, n\}. \end{aligned}$$

Then $\mathsf{P} \cup U$ is a compatible inconsistency relation for a set $\mathcal{C}$ of global clauses and the calculus $\mathsf{RES}_{\mathbf{EC}}$, $\mathsf{RES}_{\mathbf{EMC}}$, $\mathsf{RES}_{\mathbf{ECN}}$ or $\mathsf{RES}_{\mathbf{EMCN}}$, respectively.

The proof is as before, noting that the inconsistency predicate is again modelled on the shape of the modal rules. The canonical model now takes the following form, where we distinguish between the different logics.

Definition 41. Let $\mathcal{C}$ be a set of global clauses. The *canonical model* for $\mathcal{C}$ and the logics **EC**, **ECN**, **EMC** or **EMCN**, respectively, is the triple (W, N, θ) where W and θ are as before (Definition 28) and N is given by

$$\begin{aligned} N_{\mathbf{EC}}(M) &= \{\theta(p_1) \cap \cdots \cap \theta(p_n) \mid \Box p_1, \dots, \Box p_n \in M\} && \text{for } \mathbf{EC} \\ N_{\mathbf{ECN}}(M) &= N_{\mathbf{EC}}(M) \cup W && \text{for } \mathbf{ECN} \\ N_{\mathbf{EMC}}(M) &= \{\alpha \subseteq W \mid \beta \subseteq \alpha \text{ for some } \beta \in N_{\mathbf{EC}}(M)\} && \text{for } \mathbf{EMC} \\ N_{\mathbf{EMCN}}(M) &= N_{\mathbf{EMC}}(M) \cup \{W\} && \text{for } \mathbf{EMCN} \end{aligned}$$

for a maximally consistent local extension $M \in W$ of $\mathcal{C}$.

As before, we have a truth lemma that gives completeness.

Lemma 42. *Let* RES *be one of* $\mathsf{RES}_{\mathbf{EC}}$, $\mathsf{RES}_{\mathbf{ECN}}$, $\mathsf{RES}_{\mathbf{EMC}}$ *or* $\mathsf{RES}_{\mathbf{EMCN}}$, *let* (W, N, θ) *be the canonical model for* RES, *and let* $\mathcal{C}$ *be a set of global clauses. Then* $M \models D$ *whenever* $D \in M$, *for all local clauses* D *and all maximally* RES*-consistent local extensions* M *of* $\mathcal{C}$.

Proof. The interesting case here is **EC** as the others are extensions of **EC** that we have previously discussed. Again, we just consider $\neg\Box p \in M$ and assume for a contradiction that $M \models \Box p$. Then there are $p_1, \dots, p_n$ such that $\theta(p) = \theta(p_1) \cap \cdots \cap \theta(p_n)$ and $\Box p_1, \dots, \Box p_n \in M$. From the former we conclude the premiss of $\mathsf{LCRES1}$ or $\mathsf{LCRES2}$ depending on the sub-clauses we derive through Corollary 25 and arrive at a contradiction to the consistency of M.

Completeness now follows as in the other cases we have discussed before.

Corollary 43 (Completeness). *The calculi* $\mathsf{RES}_{\mathbf{EC}}$, $\mathsf{RES}_{\mathbf{ECN}}$, $\mathsf{RES}_{\mathbf{EMC}}$ *and* $\mathsf{RES}_{\mathbf{EMCN}}$ *are complete with respect to the classes of models* $\mathcal{EC}$, $\mathcal{ECN}$, $\mathcal{EMC}$ *and* $\mathcal{EMCN}$, *respectively.*

5 Conclusion and Future Work

We have presented the first resolution calculi for the cube of classical non-normal modal logics. The calculi manipulate sets of modal clauses of a very simple form. Their completeness is based on the notion of inconsistency predicate. Moreover, we have seen that resolution calculi appear to be modular, i.e. rules can just be combined to obtain a stronger calculus. Is this a coincidence? Are there general principles that enable this compositionality? This is what we are going to explore in a follow up paper. Also, the shape of our calculi, i.e. the modal resolution rules, when compared to the Hilbert axioms, insinuate that there might be a more principled way of synthesising resolution systems from Hilbert axioms. We aim to investigate this as a next step.

References

1. Abadi, M., Manna, Z.: Modal theorem proving. In: Siekmann, J.H. (ed.) CADE 1986. LNCS, vol. 230, pp. 172–189. Springer, Heidelberg (1986). https://doi.org/10.1007/3-540-16780-3_89
2. Areces, C., de Nivelle, H., de Rijke, M.: Prefixed resolution: a resolution method for modal and description logics. In: CADE 1999. LNCS (LNAI), vol. 1632, pp. 187–201. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48660-7_13
3. Auffray, Y.: Linear strategy for propositional modal resolution. Inf. Process. Lett. **28**(2), 87–92 (1988)
4. del Cerro, L.F.: Resolution modal logics. In: Proceedings of Advanced NATO Study Institute on Logics and Models for Verification and Specification of Concurrent Systems, La Colle-sur-Loup, France, pp. 46–78 (1984)
5. del Cerro, L.F.: A simple deduction method for modal logic. Inf. Process. Lett. **14**(2), 49–51 (1982)
6. Chan, M.C.: The recursive resolution method for modal logic. N. Gener. Comput. **5**, 155–183 (1987)
7. Chellas, B.F.: Modal Logic. Cambridge (1980)
8. Cialdea, M.: Resolution for some first-order modal systems. Theor. Comput. Sci. **85**, 213–229 (1991)
9. Dalmonte, T., Lellmann, B., Olivetti, N., Pimentel, E.: Hypersequent calculi for non-normal modal and deontic logics: countermodels and optimal complexity. J. Log. Comput. **31**(1), 67–111 (2021)
10. Dalmonte, T., Olivetti, N., Negri, S.: Non-normal modal logics: bi-neighbourhood semantics and its labelled calculi. In: Bezhanishvili, G., D'Agostino, G., Metcalfe, G., Studer, T. (eds.) Proceedings of the AiML 2018 (2018)
11. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 388–397. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_24
12. Elgesem, D.: The modal logic of agency. Nord. J. Philos. Log. **2**, 1–46 (1997)
13. Enjalbert, P., del Cerro, L.F.: Modal resolution in clausal form. Theor. Comput. Sci. **65**, 1–33 (1989)
14. Fitting, M.C.: Destructive Modal Resolution. CUNY Technical Report (1989)
15. Gilbert, D.R., Maffezioli, P.: Modular sequent calculi for classical modal logics. Stud. Logica. **103**(1), 175–217 (2015)
16. Giunchiglia, E., Tacchella, A., Giunchiglia, F.: SAT-based decision procedures for classical modal logics. J. Autom. Reason. **28**(2), 143–171 (2002)
17. Gleißner, T., Steen, A.: Leo-III (2022). https://doi.org/10.5281/zenodo.4435994. Accessed 24 July 2023
18. Goré, R., Kikkert, C.: CEGAR-tableaux: improved modal satisfiability via modal clause-learning and SAT. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 74–91. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_5
19. Goré, R., Olesen, K., Thomson, J.: Implementing tableau calculi using BDDs: BDDTab system description. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 337–343. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08587-6_25
20. Götzmann, D., Kaminski, M., Smolka, G.: Spartacus: a tableau prover for hybrid logic. Electron. Notes Theor. Comput. Sci. **262** (2010)

21. Haken, A.: The intractability of resolution. Theor. Comput. Sci. **39**(2–3), 297–308 (1985)
22. Indrzejczak, A.: Sequent calculi for monotonic modal logics. Bull. Section Logic **34**(3), 151–164 (2005)
23. Indrzejczak, A.: Admissibility of cut in congruent modal logics. Logic Log. Philos. **21**, 189–203 (2011)
24. Kaminski, M., Tebbi, T.: InKreSAT: modal reasoning via incremental reduction to SAT. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 436–442. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_31
25. Lavendhomme, R., Lucas, T.: Sequent calculi and decision procedures for weak modal systems. Stud. Logica. **65**, 121–145 (2000)
26. Lellmann, B., Pimentel, E.: Modularisation of sequent calculi for normal and non-normal modalities. ACM Trans. Comput. Logic **20**(2), 7:1–7:46 (2019)
27. McCune, W.W.: OTTER Users' Guide, Version 3.3 (2003). Argonne National Laboratory
28. McCune, W.W.: Prover9 and mace4 (2010). http://www.cs.unm.edu/~mccune/prover9/. Accessed 24 July 2023
29. Mints, G.: Gentzen-type systems and resolution rules part I propositional logic. In: Martin-Löf, P., Mints, G. (eds.) COLOG 1988. LNCS, vol. 417, pp. 198–231. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52335-9_55
30. Nalon, C., Dixon, C.: Clausal resolution for normal modal logics. J. Algorithms **62**, 117–134 (2007)
31. Nalon, C., Dixon, C., Hustadt, U.: Modal resolution: proofs, layers, and refinements. ACM Trans. Comput. Logic **20**(4), 23:1–23:38 (2019)
32. Nalon, C., Hustadt, U., Dixon, C.: KSP: a resolution-based prover for multimodal K. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 406–415. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_28
33. Nalon, C., Hustadt, U., Dixon, C.: KSP: a resolution-based prover for multimodal K, abridged report. In: Sierra, C. (ed.) Proceedings of the IJCAI 2017, pp. 4919–4923. IJCAI/AAAI Press (2017)
34. Nalon, C., Hustadt, U., Dixon, C.: KSP a resolution-based theorem prover for K_n: architecture, refinements, strategies and experiments. J. Autom. Reason. **64**(3), 461–484 (2020)
35. Nalon, C., Hustadt, U., Papacchini, F., Dixon, C.: Local reductions for the modal cube. In: Proceedings of the IJCAR 2022 (2022)
36. Nalon, C., Marcos, J., Dixon, C.: Clausal resolution for modal logics of confluence. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 322–336. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08587-6_24
37. Negri, S.: Proof theory for non-normal modal logics: the neighbourhood formalism and basic results. IfCoLog J. Appl. Log. **4**(4), 1241–1286 (2017)
38. de Nivelle, H., Schmidt, R.A., Hustadt, U.: Resolution-based methods for modal logics. Logic J. IGPL **8**(3), 265–292 (2000)
39. Ohlbach, H.J.: Semantics-based translation methods for modal logics. J. Log. Comput. **1**(5), 691–746 (1990)
40. Ohlbach, H.J., Schmidt, R.A., Hustadt, U.: Translating graded modalities into predicate logics. In: Wansing, H. (ed.) Proof Theory of Modal Logic, Applied Logic Series, vol. 2, pp. 253–291. Kluwer Academic Publishers (1996)
41. Orlandelli, E.: Proof analysis in deontic logics. In: Cariani, F., Grossi, D., Meheus, J., Parent, X. (eds.) DEON 2014. LNCS (LNAI), vol. 8554, pp. 139–148. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08615-6_11

42. Orlandelli, E.: Sequent calculi and interpolation for non-normal modal and deontic logics. Logic Log. Philos. **30**(1), 139–183 (2020)
43. Pacuit, E.: Neighborhood Semantics for Modal Logic. Springer, Heidelberg (2017)
44. Papacchini, F., Nalon, C., Hustadt, U., Dixon, C.: Efficient local reductions to basic modal logic. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 76–92. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_5
45. Plaisted, D.A., Greenbaum, S.A.: A structure-preserving clause form translation. J. Log. Comput. **2**, 293–304 (1986)
46. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
47. Schulz, S.: E 2.6 (2022). http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/Download.html. Accessed 24 July 2023
48. Sutcliff, G. (ed.): Proceedings of the 11th IJCAR ATP System Competition (CASC-J11) (2022). https://www.tptp.org/CASC/J11/. Accessed 24 July 2023
49. The SPASS Team: Spass 3.9 (2016). http://www.spass-prover.org/. Accessed 24 July 2023
50. The Vampire Team: Vampire 4.7 (2022). https://github.com/vprover/vampire/releases. Accessed 24 July 2023
51. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006). https://doi.org/10.1007/11814771_26
52. Vardi, M.Y.: On the complexity of epistemic reasoning. In: Proceedings of the LICS 1989, pp. 243–252. IEEE Computer Society (1989)

Canonicity of Proofs in Constructive Modal Logic

Matteo Acclavio[1(✉)], Davide Catta[2], and Federico Olimpieri[3]

[1] University of Southern Denmark, Odense, Denmark
acclavio@imada.sdu.dk
[2] Università degli studi di Napoli, Federico II, Naples, Italy
[3] University of Leeds, Leeds, UK

Abstract. In this paper we investigate the Curry-Howard correspondence for constructive modal logic in light of the gap between the proof equivalences enforced by the lambda calculi from the literature and by the recently defined winning strategies for this logic.

We define a new lambda-calculus for a minimal constructive modal logic by enriching the calculus from the literature with additional reduction rules and we prove normalization and confluence for our calculus. We then provide a typing system in the style of focused proof systems allowing us to provide a unique proof for each term in normal form, and we use this result to show a one-to-one correspondence between terms in normal form and winning innocent strategies.

Keywords: Constructive Modal Logic · Lambda Calculus · Game Semantics

1 Introduction

Proof theory is the branch of mathematical logic whose aim is studying the properties of logical arguments (i.e., proofs) as well as the structure of proofs and their invariants. For this purpose, the most used representations of proofs are based on tree-like data structures inductively defined using inference rules of a proof system.[1] *Natural deduction* and *sequent calculus* are among the most used proof systems due to their intuitive representation. Both these proof systems were originally devised by Gentzen in order to prove the consistency of first-order arithmetic. Their versatility resulted in their employment for a wide variety of logics.

The first author is supported by Villum Fonden, grant no. 50079. The second author is supported by the PRIN project RIPER (No. 20203FFYLK) The third author is supported by the US Air Force Office for Scientific Research under award number FA9550-21-1-0007.

[1] It is worth noting that some proof systems (in the sense of [13]) allows to represent proofs using structures such as infinite trees (for non-well-founded proof systems, see, e.g., [16]), graphs (see proof nets [23,24], combinatorial proofs [28] or proof diagrams [3]) or structures defined in a compositional way (see open deduction [25] and deep inference [51]).

R. Ramanayake and J. Urban (Eds.): TABLEAUX 2023, LNAI 14278, pp. 342–363, 2023.
https://doi.org/10.1007/978-3-031-43513-3_19

However, having formalisms able to represent proofs is not enough to define "what is a proof" since different derivations, or derivations in different proof systems, could represent the same abstract object. A notion of *proof identity* is therefore required to define a proof as a proper mathematical entity [19]. Such a notion of identity is provided by delineating the conditions under which two distinct formal representations of a proof represent the same logical argument. The definition of these conditions are often driven by semantic considerations (by performing specific transformations on two derivations, they can be transformed to the same object) or intuitive ones (two derivations only differ for the order in which the same rules are applied to the same formulas).

Natural deduction is often considered a satisfactory formalism since it allows to define a more canonical representation of proofs with respect to sequent calculus: sequent calculus derivations differing because of some rules permutations are represented (*via* a standard translation) by the same natural deduction derivation. Moreover, natural deduction provides a one-to-one correspondence between derivations and lambda-terms, called the *Curry-Howard correspondence* [49].

Constructive Modal Logic. Classical modal logics are obtained by extending *classical logic* with unary operators, called *modalities*, that qualify the truth of a judgment. The most used modalities are the $\Box$ (called *box*) and its dual operator $\Diamond$ (called *diamond*) which are usually interpreted as *necessity* and *possibility.* According to the interpretation of such modalities, modal logics find applications, for example, in knowledge representation [52], artificial intelligence [41] and the formal verification of computer programs [20,37,46]. The work of Fitch [22] initiated the investigation of the proof theory of modal logics extending intuitionistic logic, leading to numerous results on the topic [21,27,36,40,47].

In particular, the Curry-Howard correspondence has been extended to various constructive modal logics [7,10,17,32,33,45]. Intuitionistic logic can be extended with modalities in different ways (for an overview see [48]): while in classical logic axioms involving only $\Box$ provide also description of the behavior of $\Diamond$, for intuitionistic logic this is no more the case since the duality of the two modalities does not hold anymore. This leads to different approaches. *Constructive modal logics* consider minimal sets of axioms to guarantee the definition of the behaviors of the $\Box$ and $\Diamond$ modalities. A second approach, referred to as *intuitionistic modal logic*, considers additional axioms in order to validate the Gödel-Gentzen translation [15]. In this work we consider a minimal fragment of the constructive modal logic CK only containing the implication $\to$ and the modality $\Box$. This fragment is enough to define types for a λ-calculus with a Let constructor [7] which can be interpreted as an explicit substitution and, for this reason, we more concisely denote by $N\,[M_1,\dots M_n/x_1,\dots,x_n]_{\blacksquare}$ instead of $\mathsf{Let}\ M_1,\dots M_n\ \mathsf{be}\ x_1,\dots,x_n\ \mathsf{in}\ N$.

Recent works on the the proof equivalence of constructive modal logics [6] expose a complexity gap between the proof equivalences induced by the natural deduction [10] and winning innocent strategies [5] for this logic. This discrepancy cannot be observed in intuitionistic propositional logic where there are one-to-one correspondences between natural deduction derivations, lambda terms and innocent winning strategies. In particular, in the logic CK we observe sequent calculus proofs which correspond to the same winning strategy but which cannot be represented by the same natural deduction derivation in the systems provided in [10,32] (or equivalently corresponding to different modal λ-terms). By means of example, consider the terms $x\,[z/x]_{\blacksquare}$ and $x\,[z,w/x,y]_{\blacksquare}$ and their (unique) typing derivations shown in Fig. 1 (see Fig. 3 for the typing system). Intuitively, the two terms $x\,[z/x]_{\blacksquare}$ and $x\,[z,w/x,y]_{\blacksquare}$ should be semantically

$$\dfrac{\dfrac{}{z:\Box a, w:\Box b \vdash z:\Box a}\,\mathsf{Id} \qquad \dfrac{}{x:a, y:b \vdash x:a}\,\mathsf{Id}}{z:\Box a, w:\Box b \vdash x\,[z/x]_{\blacksquare}:\Box a}\,\Box\text{-subst}$$

$$\dfrac{\dfrac{}{z:\Box a, w:\Box b \vdash z:\Box a}\,\mathsf{Id} \qquad \dfrac{}{z:\Box a, w:\Box b \vdash w:\Box b}\,\mathsf{Id} \qquad \dfrac{}{x:a, y:b \vdash x:a}\,\mathsf{Id}}{z:\Box a, w:\Box b \vdash x\,[z,w/x,y]_{\blacksquare}:\Box a}\,\Box\text{-subst}$$

Fig. 1. The typing derivations of the modal λ-terms $x\,[z/x]_{\blacksquare}$ and $x\,[z,w/x,y]_{\blacksquare}$.

equivalent since the explicit substitution of the variable y in the term x is vacuous. Said differently, if we explicit the substitution encoded by the constructor Let, both terms $x\,[z/x]_{\blacksquare}$ and $x\,[z,w/x,y]_{\blacksquare}$ should reduce to the term z.

In fact, this undesirable behavior disappear when considering the Winning Innocent Strategies for CK defined in [5]. In this syntax, both the above natural deduction derivations correspond to the same strategy below.

$$\mathcal{S} = \{\epsilon, a^{\circ}, a^{\circ}a^{\bullet}\} \quad \text{over the arena} \quad [\![\Box a, \Box b \vdash \Box a]\!] \;=\; \begin{tikzcd} \Box \arrow[rr, bend left] \arrow[rrd] \arrow[d, squiggly] & \Box \arrow[r] \arrow[rd] \arrow[d, squiggly] & \Box \arrow[d, squiggly] \\ b \arrow[rru] \arrow[rr, bend left] & a^{\bullet} \arrow[ru] \arrow[r] & a^{\circ} \end{tikzcd} \tag{1}$$

Contribution. In this paper we define a new modal λ-calculus for CK by considering additional rewriting rules that allow us to retrieve a one-to-one correspondence between terms in normal form and winning innocent strategies, that is, providing more canonical representatives for proofs with respect to natural deduction and modal λ-terms defined in the literature. From the technical point-of-view, we obtain this result by extending the operational semantics of the modal λ-calculus with the appropriate new reduction rules for the explicit substitution encoded by the Let, dealing with contraction and weakening operating on the variables bound by the Let. We call this set of rules the κ-reduction, which we show to be strongly normalizing using elementary combinatorial methods. In order to deal with the interaction of the η-reduction with β-reduction, we define a restricted η-reduction following an approach similar to the one used in [18,31,43]. We prove strong normalization and confluence for our new operational semantics.

After proving confluence and strong normalization for our modal λ-calculus, we provide a canonical typing system inspired by focused sequent calculi (see, e.g., [8]) providing a unique typing derivation for each term in normal form. We conclude by establishing a one-to-one correspondence between the winning strategies defined in [5] and proofs of this calculi, therefore with terms in normal form.

Related Work. To the best of our knowledge, the first paper proposing a Curry-Howard correspondence for the logic CK is [10]. In this work, the authors provide a natural deduction system for the logic CK by enriching the standard system for intuitionistic propositional logic with a generalized elimination rule capable of taking into account the behavior of the $\Box$-modality. At the level of lambda calculus, they enrich the syntax of terms by adding a new constructor Let defined as follows:

$$\mathsf{Let}\, x_1, \dots x_n \,\mathsf{be}\, N_1, \dots, N_n \,\mathsf{in}\, M \quad \text{(which we denote } M\left[N_1, \dots, N_n / x_1, \dots, x_n\right]_{\blacksquare}) \tag{2}$$

providing a notation which can be interpreted as an explicit substitution of the variable x_i with the term N_i for all occurrences of $x_1 \dots, x_n$ inside a term M. For this calculus, the authors only consider the usual η and β reductions plus the following reduction:

$$\mathsf{Let}\, y \,\mathsf{be}\, P \,\mathsf{in}\, (\mathsf{Let}\, x \,\mathsf{be}\, N \,\mathsf{in}\, M) \rightsquigarrow \mathsf{Let}\, x \,\mathsf{be}\, (\mathsf{Let}\, y \,\mathsf{be}\, P \,\mathsf{in}\, N) \,\mathsf{in}\, (\mathsf{Let}\, x \,\mathsf{be}\, x \,\mathsf{in}\, M)$$
$$\big(\text{in our syntax this reduction is written as} \quad M\left[N/x\right]_{\blacksquare}\left[P/y\right]_{\blacksquare} \rightsquigarrow M\left[x/x\right]_{\blacksquare}\left[N\left[P/y\right]_{\blacksquare}/x\right]_{\blacksquare}\big)$$

In [32] the author considers the usual η and β reduction with an the following additional β-reduction rule specifically designed to handle the explicit substitution construct.

$$M\left[\vec{P}, R\left[\vec{N}/\vec{z}\right]_{\blacksquare}, \vec{Q}/\vec{x}, y, \vec{w}\right]_{\blacksquare} \rightsquigarrow_{\beta_2} M\left\{R/y\right\}\left[\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}\right]_{\blacksquare} \tag{3}$$

In the same paper, the author provides a detailed proof of strong normalization and confluence for modal lambda terms with respect to the standard η and β reduction, plus this new β_2 reduction. However, also this calculus does not manage to fix the aforementioned problem with canonicity.

An alternative natural deduction system (and λ-calculus) is proposed in [33], where the symmetry between elimination and introduction rules typical of natural deduction is restored. However, this result requires to define a sequent calculus where sequents have a more complex structure (dual-contexts), and lacks an in-depth study of the operational semantics because the η-expansion is not considered in the calculus.

Outline of the Paper. In Sect. 2 we recall the definition of the fragment of the logic CK we consider in this paper, as well as the main results on the proof theory for this logic, its natural deduction and lambda calculus. In Sect. 3 we define the modal λ-calculus we consider in this paper, proving its strong normalization and confluence properties. In Sect. 4 we provide a typing system in the style of focused sequent calculi, where we are able to narrow the proof search of the type assignment of our normal terms to a single derivation. In Sect. 5 we recall the definition of the game semantics for the logic we consider and we prove the one-to-one correspondence between terms in normal form and winning strategies.

For reason of space, we omit in the paper the proofs of those technical lemmas that are not particularly interesting (mostly by induction and case analysis). These proofs can be found in the extended version of this paper [4].

2 Preliminaries

In this section we recall the definition of the (fragment of the) constructive modal logic CK we consider in this paper, and we recall the definition and some terminology for modal λ-terms. We are interested in a minimal constructive modal logic whose *formulas* are defined from a countable set of propositional variables $\mathcal{A} = \{a, b, c, \dots\}$ using the following grammar:

$$A \coloneqq a \mid (A \to A) \mid \Box A \tag{4}$$

We say that a formula is *modality-free* if it contains no occurrences of the modality $\Box$. A formula is a $\to$*-formula* if it is of the form $A \to B$. In the following we use Krivine's convention [38] and write $(A_1, \dots, A_n) \to C$ as a shortcut for $(A_1 \to (\cdots \to (A_n \to C)\cdots))$ A *sequent* is an expression $\Gamma \vdash C$ where Γ is a finite (possibly empty) list of formulas and C is a formula. If $\Gamma = A_1, \dots, A_n$ and σ a permutation over $\{1, \dots, n\}$, then we may write $\sigma(\Gamma)$ to denote $A_{\sigma(1)}, \dots, A_{\sigma(n)}$.

In this paper we consider the logic CK defined by extending the conjunction-free and disjunction-free fragment of intuitionistic propositional logic with the modality $\Box$ whose behavior is defined by the *necessitation rule* and the axiom K_1 below.

$$\mathsf{Nec} \coloneqq \text{if } A \text{ is provable, then also } \Box A \text{ is} \qquad \mathsf{K}_1 \coloneqq \Box(A \to B) \to (\Box A \to \Box B)$$

The sequent calculus SCK, whose rules are provided in Fig. 2, is a sound and complete proof system for the logic CK. This system have been extracted from the one presented in [39] and satisfies cut-elimination.

2.1 A Lambda Calculus for CK

The set of (untyped) *modal λ-terms* is defined inductively from a countable set of *variables* $\mathcal{V} = \{x, y, \dots\}$ using the following grammar:

$$M, N \coloneqq x \mid \lambda x.M \mid (MN) \mid M\left[\vec{N}/\vec{x}\right]_{\blacksquare} \quad \text{where} \quad \begin{cases} \vec{N} = N_1, \dots, N_n \text{ is a list of terms and} \\ \vec{x} = x_1, \dots x_n \text{ is a list of distinct variables.} \end{cases}$$

modulo the standard α-equivalence (denoted $=_\alpha$, see [9]) and modulo the equivalence generated by the following permutations (for any σ permutation over the set $\{1, \dots, n\}$) over the order of substitutions in the $[\cdot/\cdot]_{\blacksquare}$ constructor:

$$\left[\vec{N}/\vec{x}\right]_{\blacksquare} \coloneqq [N_1, \dots, N_n/x_1, \cdots, x_n]_{\blacksquare} = \left[N_{\sigma(1)}, \dots, N_{\sigma(n)}/x_{\sigma(1)}, \dots, x_{\sigma(n)}\right]_{\blacksquare} \eqqcolon \left[\sigma(\vec{N})/\sigma(\vec{x})\right]_{\blacksquare}$$
$$\text{for any } \sigma \text{ permutation over } \{1, \dots, n\}.$$

$$\mathsf{ax}\,\frac{}{a \vdash a} \qquad \mathsf{ex}\,\frac{\Gamma \vdash C}{\sigma(\Gamma) \vdash C} \qquad \to^{\mathsf{R}}\frac{\Gamma, A \vdash C}{\Gamma \vdash A \to C} \qquad \to^{\mathsf{L}}\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, \Delta, A \to B \vdash C}$$

$$\mathsf{K}^{\Box}\,\frac{\Gamma \vdash A}{\Box\Gamma \vdash \Box A} \qquad \mathsf{W}\,\frac{\Gamma \vdash C}{\Gamma, A \vdash C} \qquad \mathsf{C}\,\frac{\Gamma, A, A \vdash C}{\Gamma, A \vdash C} \qquad \mathsf{cut}\,\frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}$$

Fig. 2. Sequent calculus rules of the sequent system SCK, where σ is a permutation over $\{1, \ldots, n\}$

$$\mathsf{Id}\,\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\, i \in \{1,\ldots,n\} \qquad \mathsf{Abs}\,\frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x.M : A \to C} \qquad \mathsf{App}\,\frac{\Gamma \vdash N : A \quad \Gamma \vdash M : A \to C}{\Gamma \vdash MN : C}$$

$$\Box\text{-}\mathsf{subst}\,\frac{\Gamma \vdash N_1 : \Box A_1 \quad \cdots \quad \Gamma \vdash N_n : \Box A_n \quad x_1 : A_1, \ldots, x_n : A_n \vdash M : C}{\Gamma \vdash M\,[N_1, \ldots, N_n / x_1, \ldots, x_n]_{\blacksquare} : \Box C}\, x_{1,\ldots,x_n} \text{ do not occur in } \Gamma$$

Fig. 3. Typing rules in the natural deduction system $\mathsf{ND}_{\mathsf{CK}}$ for modal λ-terms.

As usual, application associates to the left, and has higher precedence than abstraction. For example, $\lambda xyz.xyz := \lambda x.(\lambda y.(\lambda z.((xy)z)))$. A modal λ-term is a *(explicit) substitution* if it is of the form $M\left[\vec{N}/\vec{x}\right]_{\blacksquare}$, an *application* if of the form MN, and a λ-*abstraction* if of the form $\lambda x.M$.

The set of *subterms* of a term M (denoted $\mathsf{SUB}(M)$) is defined as follows:

$$\mathsf{Sub}(x) = \{x\} \quad , \quad \mathsf{Sub}(\lambda x.M) = \mathsf{Sub}(M) \cup \{\lambda x.M\} \quad , \quad \mathsf{Sub}(MN) = \mathsf{Sub}(M) \cup \mathsf{Sub}(N) \cup \{MN\}\ ,$$
$$\mathsf{Sub}(M\,[N_1, \ldots, N_n/x_1, \ldots, x_n]_{\blacksquare}) = \mathsf{Sub}(M) \cup \left(\bigcup_{i \in \{1,\ldots,n\}} \mathsf{Sub}(N_i)\right) \cup \{M\,[N_1, \ldots, N_n/x_1, \ldots, x_n]_{\blacksquare}\}\ .$$

Its *length* $|M|$ and its set of *free variables* $\mathtt{FV}(M)$ are defined as:

$$|M| = \begin{cases} 0 & \text{if } M = x \\ |N| + 1 & \text{if } M = \lambda x.N \\ \max\{|N|, |P|\} + 1 & \text{if } M = NP \\ \max\{|N|, |P_1|, \ldots, |P_n|\} + 1 & \text{if } M = N\left[\vec{P}/\vec{x}\right]_{\blacksquare} \end{cases} \qquad \mathtt{FV}(M) = \begin{cases} \{x\} & \text{if } M = x \\ \mathtt{FV}(N) \setminus \{x\} & \text{if } M = \lambda x.N \\ \mathtt{FV}(N) \cup \mathtt{FV}(P) & \text{if } M = NP \\ \bigcup_i \mathtt{FV}(P_i) & \text{if } M = N\left[\vec{P}/\vec{x}\right]_{\blacksquare} \end{cases}$$

We denote $|M|_x$ the number of the occurrences of the free variable x in a term M and we may write $|M|_x = 0$ if $x \notin \mathtt{FV}(M)$ and we say that a term M is *linear* in the variables $x_1, \ldots, x_n$ if $|M|_{x_i} = 1$ for all $i \in \{1, \ldots, n\}$. We denote by $M\,\{N_1, \ldots, N_n/x_1, \ldots, x_n\}$ the result of the standard capture avoiding substitution of the occurrences of the variable $x_1, \ldots, x_n$ in M with the term $N_1, \ldots, N_n$ respectively (see, e.g., [50]).

A *variable declaration* is an expression $x : A$ where x is a variable and A is a *type*, that is, a formula as defined in Equation (4). A *(typing) context* is a finite list $\Gamma := x_1 : A_1, \ldots, x_n : A_n$ of distinct variable declarations. Given a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, we say that a variable x *appears* in Γ if $x = x_i$ for a $i \in \{1, \ldots, n\}$ and we denote by $\Gamma, y : B$ the context $x_1 : A_1, \ldots, x_n : A_n, y : B$ implicitly assuming that y does not appear in Γ. A *type assignment* is an expression of the form $\Gamma \vdash M : A$ where Γ is a context, M a modal λ-term and A a type.

Definition 1. *Let $\Gamma \vdash M : A$ be an type assignment. A* typing derivation *(or* derivation *for short) of $\Gamma \vdash M : A$ in $\mathsf{ND}_{\mathsf{CK}}$ is a finite tree of type assignment constructed using the rules in Fig. 3 in such a way it has root $\Gamma \vdash M : A$ and each leaf is the conclusion of a Id-rule. A type assignment is* derivable *(in $\mathsf{ND}_{\mathsf{CK}}$) if there is a derivation with conclusion the given type assignment.*

We denote by Λ (resp. by $\Lambda^{\blacksquare}$ and Λ^{λ}) the set of modal λ-terms (resp. the set of substitutions and λ-abstractions in Λ) admitting a derivable type assignment in $\mathsf{ND}_{\mathsf{CK}}$.

3 A New Modal Lambda Calculus

In this section we define a new modal lambda calculus by enriching the operational semantics of the previous calculi with additional reduction rules aiming at recovering canonicity, proving confluence and strong normalization properties.

To define our term rewriting rules, we require special care when they are applied in a proper sub-term. This is due to the fact that the explicit substitution encoded by $[\cdot/\cdot]_{\blacksquare}$ could capture free variables. For this reason, we introduce the notion of *term with a hole* as a term of the form $\mathbf{C}[\circ]$ containing a single occurrence of a special variable $\circ$. More precisely, the set CwH of terms with a hole and the two sets CwH_{η_1} and CwH_{η_2} of specific terms with a hole are defined by the following grammars:

$$\begin{array}{ll}
\mathsf{CwH} & : \mathbf{C}[\circ] := \circ \mid \lambda x.\mathbf{C}[\circ] \mid M\mathbf{C}[\circ] \mid \mathbf{C}[\circ]M \mid \mathbf{C}[\circ][\vec{M}/\vec{x}] \quad \mid M\left[\vec{N}_1, \mathbf{C}[\circ], \vec{N}_2/\vec{x}_1, x, \vec{x}_2\right]_{\blacksquare} \\
\mathsf{CwH}_{\eta_1} & : \mathbf{E}[\circ] := \circ \mid \lambda x.\mathbf{E}[\circ] \mid M\mathbf{E}[\circ] \mid \mathbf{E}'[\circ]M \mid \mathbf{E}[\circ]\left[\vec{M}/\vec{x}\right]_{\blacksquare} \mid M\left[\vec{N}_1, \mathbf{E}, \vec{N}_2/\vec{x}_1, x, \vec{x}_2\right]_{\blacksquare} \\
\mathsf{CwH}_{\eta_2} & : \mathbf{D}[\circ] := \circ \mid \lambda x.\mathbf{D}[\circ] \mid M\mathbf{D}[\circ] \mid \mathbf{D}[\circ]M \mid \mathbf{D}[\circ]\left[\vec{M}/\vec{x}\right]_{\blacksquare} \mid M\left[\vec{N}_1, \mathbf{D}'[\circ], \vec{N}_2/\vec{x}_1, x, \vec{x}_2\right]_{\blacksquare} \\
 & \qquad\qquad\qquad\text{with } \mathbf{E}'[\circ] \neq [\circ] \neq \mathbf{D}'[\circ]
\end{array}$$

We denote by $\mathbf{C}[M]$ the term obtained by replacing the hole $\circ$ in $\mathbf{C}[\circ]$ with the term M. By means of example, if $\mathbf{C}[\circ] = \circ$ then $\mathbf{C}[M] = M$ and if $\mathbf{E}[\circ] = (\lambda x.xN)[\circ/x]_{\blacksquare}$ then $\mathbf{E}[M] = (\lambda x.xN)[M/x]_{\blacksquare}$. The reduction relations of our calculus are provided in Fig. 4, where the ground steps and the rules for extending them to specific contexts are provided.

Remark 1. The term constructor Let (i.e., $[\cdot/\cdot]_{\blacksquare}$ from Equation (2)) plays no role in the standard η and β reduction rules from the literature, where it behaves as a black-box during reduction. The inertness of this constructor with respect to normalization is indeed what makes the lambda calculus in [10,32] unable to identify terms whose expected behavior is the same as, for example, the following pairs of terms:

$$x[v/x]_{\blacksquare} \quad\text{and}\quad x[v,w/x,y]_{\blacksquare} \qquad\Big|\qquad xyz[v,v/y,z]_{\blacksquare} \quad\text{and}\quad xyy[v/y]_{\blacksquare} \tag{5}$$

Our operational semantics extends the one provided in [32]. The novelty of our approach is the definition of the κ-reduction and the restriction of the η-reduction. The former is needed to being able to identify modal λ-terms with the

Ground Steps:

$$
\begin{array}{rcll}
(\lambda x.M)N & \rightsquigarrow_{\beta_1} & M\{N/x\} & \\
M\left[\vec{P}, R\left[\vec{N}/\vec{z}\right]_{\blacksquare}, \vec{Q}/\vec{x}, y, \vec{w}\right]_{\blacksquare} & \rightsquigarrow_{\beta_2} & M\{R/y\}\left[\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}\right]_{\blacksquare} & \\
M & \rightsquigarrow_{\eta_1} & \lambda x.Mx & \text{if } \Gamma \vdash M : A \to B,\ x \notin \mathtt{FV}(M) \text{ and } M \notin \Lambda^{\lambda} \\
M & \rightsquigarrow_{\eta_2} & x\,[M/x]_{\blacksquare} & \text{if } \Gamma \vdash M : \Box A,\ x \notin \mathtt{FV}(M) \text{ and } M \notin \Lambda^{\blacksquare} \\
M\left[\vec{P}, N, \vec{Q}/\vec{x}, y, \vec{z}\right]_{\blacksquare} & \rightsquigarrow_{\kappa_1} & M\left[\vec{P}, \vec{Q}/\vec{x}, \vec{z}\right]_{\blacksquare} & \text{if } |M|_y = 0 \\
M\left[\vec{P}, N, N, \vec{Q}/\vec{x}, y_1, y_2, \vec{z}\right]_{\blacksquare} & \rightsquigarrow_{\kappa_2} & M\{v, v/y_1, y_2\}\left[\vec{P}, N, \vec{Q}/\vec{x}, v, \vec{z}\right]_{\blacksquare} & \text{with } v \text{ fresh}
\end{array}
$$

Reduction Steps in Contexts:

$$
\frac{M \rightsquigarrow_{\beta_i} N}{\mathbf{C}[M] \rightsquigarrow_{\beta} \mathbf{C}[N]}\, i\in\{1,2\} \qquad \frac{M \rightsquigarrow_{\kappa_i} N}{\mathbf{C}[M] \rightsquigarrow_{\kappa} \mathbf{C}[N]}\, i\in\{1,2\} \qquad \frac{M \rightsquigarrow_{\eta_1} N}{\mathbf{E}[M] \rightsquigarrow_{\eta} \mathbf{E}[N]} \qquad \frac{M \rightsquigarrow_{\eta_2} N}{\mathbf{D}[M] \rightsquigarrow_{\eta} \mathbf{D}[N]}
$$

with $\mathbf{C}[\circ] \in \mathsf{CwH}$ and $\mathbf{E}[\circ] \in \mathsf{CwH}_{\eta_1}$ and $\mathbf{D}[\circ] \in \mathsf{CwH}_{\eta_2}$

Fig. 4. Definition of the ground steps of the reduction relations, and the rules for their extension to terms with holes.

same expected computational meaning, as the ones in Eq. (5). The latter is carefully defined to avoid η-redexes that would make the reduction non-terminating, using a well-known technique in term rewriting theory (see, e.g., [31,43]).

The need of these restrictions can be observed in the two following (unrestricted) η-reduction chains, which are both forbidden by our restricted rule from Fig. 4.

$$
\begin{array}{c}
M \rightsquigarrow_{\eta} \lambda x.Mx \rightsquigarrow_{\eta} \lambda x.(\lambda y.My)x \rightsquigarrow_{\eta} \dots \\
\text{whenever } \Gamma \vdash M : A \to B
\end{array}
\quad \text{and} \quad
\begin{array}{c}
M \rightsquigarrow_{\eta} x\,[M/x]_{\blacksquare} \rightsquigarrow_{\eta} x\,[y\,[M/y]_{\blacksquare}/x]_{\blacksquare} \rightsquigarrow_{\eta} \dots \\
\text{whenever } \Gamma \vdash M : \Box A
\end{array}
$$

Moreover, our definition rules out interactions between the η and β reductions which could lead to infinite chains, as the ones shown below.

$$
\begin{array}{ccccccl}
\lambda x.M & \rightsquigarrow_{\eta} & \lambda y.(\lambda x.M)y & \rightsquigarrow_{\beta} & \lambda y.(M\{x/y\}) & =_{\alpha} \lambda x.M & \text{or} \\
x\,[M/x]_{\blacksquare} & \rightsquigarrow_{\eta} & x\,[y\,[M/y]_{\blacksquare}/x]_{\blacksquare} & \rightsquigarrow_{\beta} & y\,[M/y]_{\blacksquare} & =_{\alpha} x\,[M/x]_{\blacksquare} & .
\end{array}
$$

Definition 2. *We define the following reduction relations:*

$$
\rightsquigarrow_{\beta\eta} = \rightsquigarrow_{\beta} \cup \rightsquigarrow_{\eta} \qquad \rightsquigarrow_{\beta\kappa} = \rightsquigarrow_{\eta} \cup \rightsquigarrow_{\kappa} \qquad \rightsquigarrow_{\beta\eta\kappa} = \rightsquigarrow_{\beta} \cup \rightsquigarrow_{\eta} \cup \rightsquigarrow_{\kappa} \tag{6}
$$

For any $\xi \in \{\beta, \eta, \kappa, \beta\eta, \beta\kappa, \beta\eta\kappa\}$, we denote by $\rightsquigarrow_{\xi}^{+}$ its transitive closure, *by $\rightsquigarrow_{\xi}^{=}$ its* reflexive closure, *by $\rightsquigarrow_{\xi}^{*}$ its* reflexive and transitive closure, *and by $\equiv_{\xi}$ the* equivalence relation *it enforces over terms, that is, its reflexive, symmetric and transitive closure. Given a term M, we denote by $\mathsf{nf}_{\xi}(M)$ the set of its $\rightsquigarrow_{\xi}$-*normal form. *A term M is* strongly normalizable *for $\rightsquigarrow_{\xi}$ if it admits no infinite $\rightsquigarrow_{\xi}$-chains A reduction $\rightsquigarrow_{\xi}$ is* strongly normalizing *if every term M is strongly normalizable for it. A reduction $\rightsquigarrow_{\xi}$ is* confluent *if given $M \rightsquigarrow_{\xi}^{*} N_1$ and $M \rightsquigarrow_{\xi}^{*} N_2$ there exists a term N such that $N_1 \rightsquigarrow_{\xi}^{*} N$ and $N_2 \rightsquigarrow_{\xi}^{*} N$.*

The *substitution* lemma and *subject reduction* theorem holds for the reduction $\rightsquigarrow_{\beta\eta\kappa}$.

Lemma 1. *[Substitution Lemma] Let* $\Gamma, x : B \vdash M : C$ *and* $\Gamma \vdash N : B$ *be derivable type assignments. Then* $\Gamma, x : B \vdash M\{N/x\} : C$ *is a derivable type assignment.*

Theorem 1. *Let* $\Gamma \vdash M : C$ *be derivable. If* $M \rightsquigarrow_{\beta\eta\kappa} N$*, then* $\Gamma \vdash N : C$*.*

Proof. Because of Lemma 1, it suffices to check the cases when M reduces to N in one ground step of $\rightsquigarrow_{\beta\eta\kappa}$:

- if $M \rightsquigarrow_{\beta_1} N$, then $M = (\lambda x.P)Q$ and $N = P\{Q/x\}$. The case where $M \rightsquigarrow_{\beta_2} N$ uses a similar argument. The result follows the fact that if $\Gamma, x : B \vdash M : C$ and $\Gamma \vdash N : B$ are derivable type assignment, then $\Gamma, x : B \vdash M\{N/x\} : C$ by Lemma 1.
- if $M \rightsquigarrow_{\eta_1} N$, then $C = A \to B$ and $N = \lambda x.Mx$. The result follows by applying the rule Abs. The case where $M \rightsquigarrow_{\eta_2} N$ uses a similar argument;
- if $M \rightsquigarrow_{\kappa_1} N_1$, then $M = M'[P_1, \ldots, P_k, N, P_{k+1}, \ldots, P_n/x_1, \ldots, x_k, x, x_{k+1}, \ldots, x_n]_{\blacksquare}$ such that x is not free in M, $C = \Box B$, and $N_1 = M'\left[\vec{P}, \vec{Q}/\vec{x}, \vec{y}\right]_{\blacksquare}$. Then there are derivations for $\Gamma \vdash P_i : A_i$ for all $i \in \{1, \ldots, n\}$ (for some A_i) and a derivation for $x_1 : A_1, \ldots, x_k : A_k, x : A, x_{k+1} : A_{k+1} \ldots, x_n : A_n \vdash M' : B$. Therefore we have a derivation for $x_1 : A_1, \ldots, x_n : A_n \vdash M' : B$ since weakening is admissible (that is, whenever $\Gamma, x : A \vdash M : C$ is derivable and x does not occur free in M, then $\Gamma \vdash M : C$ is also derivable[2]. Then we have a derivation of $\Gamma \vdash N : C$ with bottom-most rule a $\Box$-subst with right-most premise $x_1 : A_1, \ldots, x_n : A_n \vdash M' : B$. and a premise $\Gamma \vdash P_i : A_i$ for each $i \in \{1, \ldots, n\}$;
- if $M \rightsquigarrow_{\kappa_2} N_1$, then we conclude similarly to the previous point since we have

$$M = M'\left[\vec{P}, N, N, \vec{Q}/\vec{x}, y_1, y_2, \vec{z}\right]_{\blacksquare} \quad \text{and} \quad N_1 = M\{y, y/y_1, y_2\}\left[\vec{P}, N, \vec{Q}/\vec{x}, y, \vec{z}\right]_{\blacksquare}.$$

We can prove local confluence of $\rightsquigarrow_{\beta\eta\kappa}$ by case analysis of the critical pairs using the following lemma.

Lemma 2. *Let* P, P' *and* Q *modal* λ*-terms. If* $P \rightsquigarrow_{\beta\eta\kappa} P'$*, then* $P\{Q/x\} \rightsquigarrow^*_{\beta\eta\kappa} P'\{Q/x\}$*. Moreover, there is a* N_Q *such that* $Q\{P/x\} \rightsquigarrow^*_{\beta\eta\kappa} N_Q$ *and* $Q\{P'/x\} \rightsquigarrow^*_{\beta\eta\kappa} N_Q$*.*

Proposition 1. *The reduction* $\rightsquigarrow_{\beta\eta\kappa}$ *is locally confluent.*

Proof. We show that if there are M, N_1 and N_2 with $N_1 \neq N_2$ such that $M \rightsquigarrow_{\beta\eta\kappa} N_1$ and $M \rightsquigarrow_{\beta\eta\kappa} N_2$, then there exists N such that $N_1 \rightsquigarrow^*_{\beta\eta\kappa} N$ and $\rightsquigarrow^*_{\beta\eta\kappa} N$. Without loss of generality we have the following cases:

[2] The admissibility of weakening is easily proven by induction on the size of a derivation.

1. if $M \rightsquigarrow_{\beta_1} N_1$ with $M = (\lambda x.P)Q$ and $N_1 = P\{Q/x\}$, then N_2 can only be obtained by applying $\rightsquigarrow_{\beta\eta\kappa}$ the subterms P and Q of M. We conclude by Lemma 2;
2. if $M \rightsquigarrow_{\beta_2} N_1$ with $M = M'\left[\vec{P}, R\left[\vec{N}/\vec{z}\right]_{\blacksquare}, \vec{Q}/\vec{x}, y, \vec{w}\right]_{\blacksquare}$ and with $N_1 = M'\{R/y\}\left[\vec{P}, \vec{N}, \vec{Q}/\vec{x}, \vec{z}, \vec{w}\right]_{\blacksquare}$, then N_2 must be a term obtained by applying $\rightsquigarrow_{\beta\eta\kappa}$ on R or on one of the terms in $\vec{P}$, $\vec{N}$ or $\vec{Q}$. We conclude again by Lemma 2;
3. if $M \rightsquigarrow_{\eta_1} N_1$, then $\Gamma \vdash M : A \to B$ and $N_1 = \lambda x.Mx$. Therefore, for any N_2 such that $M \rightsquigarrow_{\beta\eta\kappa} N_2$ we have that $\Gamma \vdash N_2 : A \to B$ (by subject reduction). Then
   - either N_2 is not an abstraction and we conclude by letting $N = \lambda x.N_2 x$.
   - otherwise $N_2 = \lambda y.M'$ and we conclude since $N_1 \rightsquigarrow_{\eta_1} \lambda x.N_2 x \rightsquigarrow_{\beta_1} N_2$.
4. if $M \rightsquigarrow_{\eta_2} N_1$ with $\Gamma \vdash M : \Box A$ and $N_1 = x\,[M/x]_{\blacksquare}$, then we conclude with a similar argument with respect to the previous point by letting $N = x\,[N_2/x]_{\blacksquare}$.
5. if $M \rightsquigarrow_{\kappa} N_1$, then either $M = M'\left[\vec{P}, N, \vec{Q}/\vec{x}, x, \vec{y}\right]_{\blacksquare}$ reduces via $\rightsquigarrow_{\kappa_1}$ to $N_1 = M'\left[\vec{P}, \vec{Q}/\vec{x}, \vec{y}\right]_{\blacksquare}$, or $M = M'\left[\vec{P}, N, N, \vec{Q}/\vec{x}, y_1, y_2, \vec{z}\right]_{\blacksquare}$ reduces via $\rightsquigarrow_{\kappa_2}$ to $N_1 = M\{y, y/y_1, y_2\}\left[\vec{P}, N, \vec{Q}/\vec{x}, y, \vec{z}\right]_{\blacksquare}$. In both cases we conclude with an argument similar to the one in Case (2).

In order to prove the termination of $\rightsquigarrow_{\beta\eta\kappa}$, we define the following measures.

Definition 3. *Let M be a modal λ-term. We define the following multisets of derivable type assignments:*

$$\mathsf{Est}_1(M) = \left\{B \to C \mid P \in \mathsf{Sub}(M) \setminus \Lambda^{\lambda} \text{ such that } M \neq PQ \text{ and } \Gamma \vdash P : B \to C\right\}$$
$$\mathsf{Est}_2(M) = \left\{\Box B \mid P \in \mathsf{Sub}(M) \setminus \Lambda^{\blacksquare} \text{ such that } M \neq Q\left[\vec{N}_1, P, \vec{N}_2/\vec{x}_1, x, \vec{x}_2\right]_{\blacksquare} \text{ and } \Gamma \vdash P : \Box B\right\}$$

We then define $\|M\|_\eta := \|M\|_\eta^1 + \|M\|_\eta^2$ with

$$\|M\|_\eta^1 := \sum_{A \in \mathsf{Est}_1(M)} \|A\|_\eta^1 \quad \text{and} \quad \|M\|_\eta^2 := \sum_{A \in \mathsf{Est}_2(M)} \|A\|_\eta^2$$

$$\text{where} \quad \begin{array}{lll} \|a\|_\eta^1 = 0 & \|A \to B\|_\eta^1 = \|A\|_\eta^1 + \|B\|_\eta^1 + 1 & \|\Box A\|_\eta^1 = \|A\|_\eta^1 \\ \|a\|_\eta^2 = 0 & \|A \to B\|_\eta^2 = \|A\|_\eta^2 + \|B\|_\eta^2 & \|\Box A\|_\eta^2 = \|A\|_\eta^2 + 1 \end{array}$$

We also define $\|M\|_\kappa$ as the size of substitution subterms of M as follows:

$$\|x\|_\kappa = 0 \qquad \|\lambda x M\|_\kappa = \|M\|_\kappa \qquad \|MN\|_\kappa = \|M\|_\kappa + \|N\|_\kappa$$
$$\|M\,[N_1, \ldots, N_n/x_1, \ldots, x_n]_{\blacksquare}\,\|_\kappa = \|M\|_\kappa + \|N\|_\kappa + n$$

Example 1. Intuitively, the measure $\|\cdot\|_\eta$ does not take into account all the subterms of M, but only the ones on which we can apply the restricted $\rightsquigarrow_\eta$. For an example, consider the modal λ-term $M = (\lambda z^{a \to a}.z)y$ with $\|M\|_\eta = 3$ because all four subterms of M are of type a $\to$-formula, but the subterm $\lambda z.z$ is an abstraction, therefore no $\rightsquigarrow_\eta$ can be applied on it. If $M \rightsquigarrow_\eta N$, because of the restrictions on $\rightsquigarrow_\eta$, we have that

- either $N = (\lambda z.z)(\lambda v.yv)$ with $\|N\|_\eta = 2$ because no $\rightsquigarrow_\eta$ can be applied to the subterms y and $\lambda z.z$ (they occur on the left of an application) or $\lambda v.yv$ (it is an abstraction), but only to the subterms z and the whole term N;
- or $N = \lambda v^a.((\lambda z.z)y)v$ with $\|N\|_\eta = 2$ because $\rightsquigarrow_\eta$ can only be applied to the subterms z and y.

Lemma 3. *Let M and N be modal λ-terms. If $M \rightsquigarrow_\eta N$, either $\|N\|_\eta < \|M\|_\eta$ or there is N' such that $N \rightsquigarrow_\eta N'$ and $\|N'\|_\eta < \|M\|_\eta$.*

Lemma 4. *The following commutations between $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$ and $\rightsquigarrow_\kappa$ hold:*

- *if $M \rightsquigarrow_\kappa N \rightsquigarrow_\beta N'$, then there is M' such that $M \rightsquigarrow_\beta M'$ and $M' \rightsquigarrow_\kappa^* N'$;*
- *if $M \rightsquigarrow_\eta N \rightsquigarrow_\kappa N'$, then there is M' such that $M \rightsquigarrow_\kappa M'$ and $M' \rightsquigarrow_\eta^* N'$;*
- *if $M \rightsquigarrow_\beta N \rightsquigarrow_\eta N'$, then there is M' such that $M \rightsquigarrow_\eta M'$ and $M' \rightsquigarrow_\beta^* N'$.*

Theorem 2. *The reduction relation $\rightsquigarrow_{\beta\eta\kappa}$ is strongly normalizing and confluent.*

Proof. After Proposition 1, it suffices to prove that $\rightsquigarrow_{\beta\eta\kappa}$ is strongly normalizing to conclude by Newman's lemma that $\rightsquigarrow_{\beta\eta\kappa}$ is also confluent.

To prove strong normalization we use the fact that the reductions $\rightsquigarrow_\beta$, $\rightsquigarrow_\eta$ and $\rightsquigarrow_\kappa$ are strongly normalizing: for $\rightsquigarrow_\beta$ the proof can be found in [32], for $\rightsquigarrow_\eta$ the proof is by induction on $\|\cdot\|_\eta$ using Lemma 3, and for $\rightsquigarrow_\kappa$ it follows the fact that, by definition of $\|\cdot\|_\kappa$, we have that $\|M\|_\kappa > \|N\|_\kappa$ whenever $M \rightsquigarrow_\kappa N$. To conclude that $\rightsquigarrow_{\beta\eta\kappa}$ also is strongly normalizing, the standard result (see, e.g., [50]) in rewriting theory ensuring that given two strongly normalizing reduction relations $\rightsquigarrow_1$ and $\rightsquigarrow_2$ with $\rightsquigarrow_1$ confluent, if $M \rightsquigarrow_2 N$ implies the existence of a reduction $\mathsf{nf}_1(M) \rightsquigarrow_2^+ \mathsf{nf}_1(N)$ for any M and N, , then $\rightsquigarrow_1 \cup \rightsquigarrow_2$ is strongly normalizing. In our case, the fact that $M \rightsquigarrow_2 N$ implies $\mathsf{nf}_1(M) \rightsquigarrow_2^+ \mathsf{nf}_1(N)$ is a corollary of Lemma 4.

Definition 4. *The set $\widehat{\Lambda}$ is the set of modal λ-terms defined inductively as follows:*

- *if x is a variable, $T_1, \dots, T_n \in \widehat{\Lambda}$, and there are derivations for the types assignments $\Gamma \vdash x : (A_1, \dots, A_n) \to C$ with C atomic and $\Gamma \vdash T_i : A_i$ for all $i \in \{1, \dots, n\}$, then $xT_1 \cdots T_n \in \widehat{\Lambda}$. Variables are the special case with $n = 0$;*
- *if $T \in \widehat{\Lambda}$ and there is a derivation of $\Gamma, x : A \vdash T : C$, then $\lambda x^A.T \in \widehat{\Lambda}$;*
- *if $M \in \widehat{\Lambda}$, $\mathtt{FV}(M) = \{x_1, \dots, x_n\}$ and the type assignment $x_1 : B_1, \dots, x_n : B_n \vdash M : C$ is derivable, and if there are n distinct terms $T_1, \dots, T_n \in \Lambda$ of the shape $T_i = y_i U_{i1} \cdots U_{ik_i}$ with $U_{ij} \in \widehat{\Lambda}$ for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, k_i\}$, such that the type assignment $\Gamma \vdash T_i : \Box B_i$ is derivable for all $i \in \{1, \dots, n\}$, then $M\,[T_1, \dots, T_n / x_1, \dots, x_n]_{\blacksquare} \in \widehat{\Lambda}$.*

Proposition 2. *The set $\widehat{\Lambda}$ is the set of modal λ-terms in $\beta\eta\kappa$-normal form $\mathsf{nf}_{\beta\eta\kappa}(\Lambda)$.*

Proof. By definition, every $\widehat{\Lambda} \subseteq \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$ is $\rightsquigarrow_{\beta\eta\kappa}$-normal. To prove the converse we proceed by induction on the structure of $M \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$:

– if $M = x$, then $M \in \widehat{\Lambda}$ by definition;
– if $M = \lambda x.M' \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$, then also $M' \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$. By inductive hypothesis, this implies $M' \in \widehat{\Lambda}$. Therefore $\lambda x.M' \in \widehat{\Lambda}$;
– if $M = PQ \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$, then both P and Q are in $\mathsf{nf}_{\beta\eta\kappa}(\Lambda)$ and there is a derivable type assignment $\Gamma \vdash M : C$, and derivable type assignments $\Gamma \vdash P : A \to C$ and $\Gamma \vdash Q : A$. We have that no $\rightsquigarrow_\eta$-rule can be applied to C because $M \in \mathsf{nf}_\eta(\Lambda)$; thus C must be atomic. We know that P cannot be in Λ^λ since $M \in \mathsf{nf}_\beta(\Lambda)$ and P cannot be in $\Lambda^{\blacksquare}$ because $\Gamma \vdash P : A \to C$ is derivable. Then by inductive hypothesis we have that $P = xT_1, \dots T_n$ for some $T_1, \dots, T_n \in \widehat{\Lambda}$. We conclude that $PQ \in \widehat{\Lambda}$;
– if $M = P\,[Q_1, \dots, Q_n/x_1, \dots, x_n]_{\blacksquare} \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$, then there is a derivable type assignment $x_1 : B_1, \dots, x_n : B_n \vdash P : C$ and derivable type assignments $\Gamma \vdash Q_i : \Box B_i$ for all $i \in \{1, \dots, n\}$. Since $M \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$, then no $\rightsquigarrow_{\beta\eta\kappa}$-rule can be applied to M, nor to P; thus $P \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$. Similarly, since $M \in \mathsf{nf}_{\beta\eta\kappa}(\Lambda)$, then $Q_i \notin \Lambda^{\blacksquare}$ (otherwise we could apply $\rightsquigarrow^2_\beta$), $Q_i \in \mathsf{nf}_{\beta\kappa}(\Lambda)$ (since no $\rightsquigarrow_{\beta\kappa}$-rule can be applied to Q_i) and Q_i cannot be in $\mathsf{nf}_\eta(\Lambda)$ (because $Q_i : \Box B_i$ and otherwise $\rightsquigarrow_\eta$-steps could be applied on M) for all $i \in \{1, \dots, n\}$. We conclude that $M \in \widehat{\Lambda}$.

$$\mathsf{ax}\,\frac{}{\Gamma, x : c \vdash x : c} \qquad \mathsf{ex}\,\frac{\Gamma \vdash M : C}{\sigma(\Gamma) \vdash M : C}\,{*} \qquad \mathsf{K}^{\Box}\,\frac{x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Delta, y_1 : \Box A_1, \dots, y_n : \Box A_n \vdash M\,[x_1, \dots, x_n/y_1, \dots, y_n]_{\blacksquare} : \Box C}\,\star$$

$$\to^{\mathsf{ax}}_{\mathsf{L}}\,\frac{\{\Gamma, y : B \vdash N_i : A_i\}_{i\in\{1,\dots,n\}}}{\Gamma, y : \underbrace{(A_1, \dots, A_n) \to c}_{B} \vdash yN_1 \cdots N_n : c}\,\S \qquad \to^{*}_{\mathsf{R}}\,\frac{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash M : C}{\Gamma \vdash \lambda x_1^{A_1} \cdots x_n^{A_n}.M : (A_1, \dots A_n) \to C}$$

$$\to^{\mathsf{K}}_{\mathsf{L}}\,\frac{\left\{\Gamma, \Delta \vdash T_{i,j} : A_{i,j}\right\}_{i\in\{1,\dots,n\}, j\in\{1,\dots,k_i\}} \qquad \Gamma, \Delta, x_1 : \Box B_1, \dots, x_n : \Box B_n \vdash M\,[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_{\blacksquare} : \Box C}{\Gamma, \underbrace{f_1 : (A_{1,1}, \dots, A_{1,k_1}) \to \Box B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \to \Box B_n}_{\Delta} \vdash M\,[N_1, \dots, N_n, \vec{z}/y_1, \dots y_n, \vec{w}]_{\blacksquare} : \Box C}\,{\dagger,\S}$$

$*$:= σ permutation over $\{1, \dots, n\}$ $\qquad$ $\star$:= $FV(M) = \{x_1, \dots x_n\}$ and $y_1, \dots, y_n$ fresh
$\S$:= each $N_i = f_i T_{i,1} \cdots T_{i,k_i}$ for $i \in \{1, \dots, n\}$ $\qquad$ $\dagger$:= Γ contains no formula of the shape $(A_1 \cdots A_n) \to \Box B$

Fig. 5. Typing rules of the typing system CK^{F}.

4 A Canonical Type System for CK

In this section we present an alternative typing system for modal λ-terms where each term in $\widehat{\Lambda}$ admits exactly one typing derivation. The rules of this system (we call CK^{F}) are provided in Fig. 5 and are conceived to reduce the non-determinism of the typing process, following the same approach used in designing focused sequent calculi [8,12,42]. Derivations and derivability in CK^{F} are defined analogously to Definition 1, using rules in CK^{F} instead of rules in $\mathsf{ND}_{\mathsf{CK}}$. We remark that the structural rules of weakening and contraction are admissible in the system.

We can now prove a result of *canonicity* of CK^{F} with respect to typing derivations of modal λ-terms in $\mathsf{nf}_{\beta\eta\kappa}(\Lambda)$.

Theorem 3. *Let $T \in \widehat{\Lambda}$ and $\Gamma \vdash T : A$ be a derivable type assignment. Then there is a unique (up to* ex*-rules) derivation of $\Gamma \vdash T : A$ in* CK^F*.*

Proof. The proof of this theorem follows from the correspondence between the inductive definition of terms in $\widehat{\Lambda}$ (Definition 4) and the shape of the typing rules of CK^F. Details are provided the extended version of this paper [4].

5 Game Semantics for CK

In this section we recall definitions and results on the winning innocent strategies for the logic CK defined in [5]. For this purpose, we first recall the construction extending Hyland-Ong arenas [29,44] for intuitionistic propositional formulas to represent formulas containing modalities, and then we recall the characterization of the winning innocent strategies representing proofs in CK. We conclude by proving the full-completeness result between for those strategies by showing a one-to-one correspondence between strategies for type assignments of terms in normal forms and their (unique) typing derivations in CK^F.

5.1 Arenas with Modalities

We recall the definition of arenas with modalities from [5] extending the encoding of arenas from [26,30]. For this purpose, we assume the reader familiar with the definition of *two-color directed graph* (or *2-dag's* for short), i.e., directed acyclic graphs with two disjoint sets of directed edges $\to$ and $\rightsquigarrow$ (details can be found in [5,26]).

Definition 5. *The* arena *of a formula F is the 2-dag $[\![F]\!]$ with vertices are labeled by elements in $\mathcal{L} = \mathcal{A} \cup \{\Box\}$ inductively defined as follows:*

$$[\![a]\!] = a \qquad [\![A \to B]\!] = [\![A]\!] \mathrel{-\!\!\triangleright} [\![B]\!] \qquad [\![\Box A]\!] = \Box \mathrel{\sim\!\!\triangleright} [\![A]\!] \tag{7}$$

where a and $\Box$ denote the graphs consisting of a single vertex labeled by a and $\Box$ respectively, and where the binary operation $\mathrel{-\!\!\triangleright}$ and $\mathrel{\sim\!\!\triangleright}$ on 2-dag's are defined as follows:

$$\mathcal{G} \mathrel{-\!\!\triangleright} \mathcal{H} = \langle\, V_\mathcal{G} \uplus V_\mathcal{H} \,,\, \overset{\mathcal{G}\uplus\mathcal{H}}{\to} \cup (\vec{R}_\mathcal{G} \curvearrowright \vec{R}_\mathcal{H}) \,,\, \overset{\mathcal{G}\uplus\mathcal{H}}{\rightsquigarrow} \,\rangle \quad \textit{and} \quad \mathcal{G} \mathrel{\sim\!\!\triangleright} \mathcal{H} = \langle\, V_\mathcal{G} \uplus V_\mathcal{H} \,,\, \overset{\mathcal{G}\uplus\mathcal{H}}{\to} \,,\, \overset{\mathcal{G}\uplus\mathcal{H}}{\rightsquigarrow} \cup (\vec{R}_\mathcal{G} \curvearrowright \vec{R}_\mathcal{H}) \,\rangle \quad \textit{with}$$

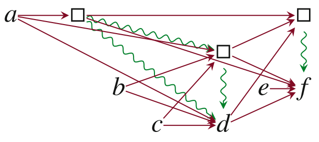
$$\begin{aligned} V_\mathcal{G} \uplus V_\mathcal{H} &= \{(v_i, i) \mid i \in \{0,1\} \textit{ and } v_0 \in V_\mathcal{G} \textit{ and } v_1 \in V_\mathcal{H}\} \quad \textit{and} \quad \ell((v_i,i)) = \ell(v_i) \\ \overset{\mathcal{G}\uplus\mathcal{H}}{\curvearrowright} &= \left\{((v_i,i),(w_i,i)) \mid i \in \{0,1\} \textit{ and } (v_0,w_0) \in \overset{\mathcal{G}}{\curvearrowright} \textit{ and } (v_1,w_1) \in \overset{\mathcal{H}}{\curvearrowright}\right\} \quad \textit{for each} \curvearrowright \in \{\to, \rightsquigarrow\} \\ (\vec{R}_\mathcal{G} \curvearrowright \vec{R}_\mathcal{H}) &= \{\, ((v,0),(w,1)) \big| v \in \vec{R}_\mathcal{G}, w \in \vec{R}_\mathcal{H} \,\} \quad \textit{where} \quad \vec{R}_X \coloneqq \{v \in V_X \mid v \overset{X}{\to} w \textit{ for no } w \in V_X\} \end{aligned}$$

The arena of a sequent $A_1, \ldots, A_n \vdash C$ *is the arena* A *of* $[\![(A_1, \ldots, A_n) \to C]\!]$*.*

Remark 2. By construction, an arena $\mathcal{G}$ of a formula or a sequent $\Gamma \vdash C$ always admits a unique non $\Box$-labeled vertex in $\vec{R}_\mathcal{G}$, i.e., a unique vertex v with $\ell(v) \neq \Box$ such that there is no $w \in V_\mathcal{G}$ such that $v \overset{\mathcal{G}}{\to} w$.

We draw 2-dag's by representing a vertex v by its label $\ell(v)$. If v and w are vertices of an 2-dag, then we draw if $v \rightarrow w$ and if $v \rightsquigarrow w$. By means of example, consider the arena below.

$$[\![(a \to \Box(b \to (c \to \Box d))) \to \Box(e \to f)]\!] \quad =$$

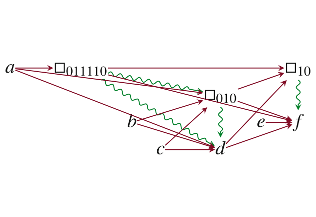


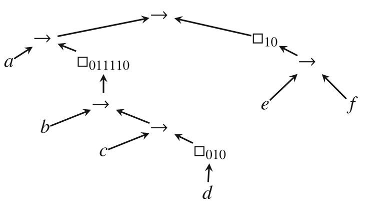
(8)

Remark 3. All arenas of the form $\left[\!\left[(A_{\sigma(1)}, \dots, A_{\sigma(n)}) \to C\right]\!\right]$ have the same representation for any σ permutation over $\{1, \dots, n\}$. More in general, it can be shown that the arena of any two equivalent formulas modulo Currying $A \to (B \to C) \sim B \to (A \to C)$ can be depicted by the same arena. However, whenever there may be ambiguity because of the presence of two vertices with the same label, we may represent the vertex $v = ((\cdots(v', i_1)\cdots), i_n)$ (where $i_1, \dots, i_n \in \{0, 1\}$) by $\ell(v)_{i_1,\dots,i_n}$ instead of simply $\ell(v) = \ell(v')$ (see Example 2).

Definition 6. *Let $[\![F]\!]$ be an arena and v one of its vertices. The* depth *of v is the number $d(v)$ of vertices in a $\rightarrow$-path from v to a vertex in $\overrightarrow{R}_{[\![F]\!]}$* [3]*. The* address *of v is defined as the unique sequence of modal vertices $\mathsf{add}(v) = m_1, \dots, m_h$ in $V_{[\![F]\!]}$ corresponding to the sequence of modalities in the path in the formula tree of F connecting the node of v to the root. If $\mathsf{add}(v) = m_1, \dots, m_h$, we denote by $\mathsf{add}^k(v) = m_k$ its k^{th} element and we call the* height *of v (denoted h_v) the number of elements in $\mathsf{add}(v)$.*

Example 2. Below an alternative representation of its arena of the formula $\big(a \to \Box(b \to (c \to \Box d))\big) \to \Box(e \to f)$ in Equation (8) where the ambiguity of the vertex representation is avoided by the use of indices, the corresponding formula-tree, and the complete list of the addresses of all vertices in this arena.





$\mathsf{add}(a)$	$=\epsilon$
$\mathsf{add}(\Box_{011110})$	$=\epsilon$
$\mathsf{add}(b)$	$=\Box_{011110}$
$\mathsf{add}(c)$	$=\Box_{011110}$
$\mathsf{add}(\Box_{010})$	$=\Box_{011110}$
$\mathsf{add}(d)$	$=\Box_{011110}\Box_{010}$
$\mathsf{add}(\Box_{10})$	$=\epsilon$
$\mathsf{add}(e)$	$=\Box_{10}$
$\mathsf{add}(f)$	$=\Box_{10}$

5.2 Games and Winning Innocent Strategies

In this subsection, we briefly recall the definitions of games and winning strategies from [5] required to make the paper self-contained. Note that differently from the previous works, we here include the additional information of the *pointer*

[3] As proven in [6,26], arenas are *stratified*, that is, all the $\rightarrow$-path from a vertex v to any vertex in $\overrightarrow{R}_{[\![F]\!]}$ have the same length. Therefore the number $d(v)$ is well-defined.

function in the definition of views. This information is crucial for the results in Sect. 4 where we provide a one-to-one correspondence between our winning strategies and modal λ-terms.

Definition 7. *Let* A *be an arena. We call a* move *an occurrence of a vertex* v *of* A *with* $\ell(v) \neq \Box$. *The* polarity *of a move* v *is the parity of* $d(v)$*: a move is a* $\circ$-move *(resp. a* $\bullet$-move*) if* $d(v)$ *is even (resp. odd).*

A pointed sequence *in* A *is a pair* $\mathsf{p} = \langle \mathsf{s}, f \rangle$ *where* $\mathsf{s} = \mathsf{s}_0, \dots, \mathsf{s}_n$ *is a finite sequences of moves in* A *and a* pointer function $f \colon \{1, \dots, n\} \to \{0, \dots, n-1\}$ *such that* $f(i) < i$ *and* $\mathsf{s}_i \xrightarrow{\mathsf{A}} \mathsf{s}_{f(i)}$. *The* length *of* p *(denoted* $|\mathsf{p}|$*) is defined as the length of* s*, that is,* $|\mathsf{p}| = n + 1$. *Note that we also use* ϵ *to denote the* empty pointed sequence $\langle \epsilon, \emptyset \rangle$.

Remark 4. It follows by definition of view that the player $\circ$ (resp. $\bullet$) can only play vertices whose $d(v)$ is even (resp. odd). For this reason, for each $v \in V_{\mathcal{G}}$ we write v° (resp. $v^{\bullet}$) if the parity of $d(v)$ even (resp. odd).

Note that the parity of a modality in the address of a move may not be the same as the parity of the move itself. By means of example, consider the vertex c in Example 2 which belongs in the scope of two modalities $\Box_{011110}$ and $\Box_{010}$ with odd parity.

Given two pointed sequences $\mathsf{p} = \langle \mathsf{s}, f \rangle$ and $\mathsf{p}' = \langle \mathsf{s}', f' \rangle$ in A, we write $\mathsf{p} \sqsubseteq \mathsf{p}'$ whenever s is a prefix of s' (thus $|\mathsf{s}| \leq |\mathsf{s}'|$) and $f(i) = f'(i)$ for all $i \in \{1, \dots, |\mathsf{p}'|\}$ and we say that p is a *predecessor* of p' if $\mathsf{p} \sqsubset \mathsf{p}'$ and $|\mathsf{p}| = |\mathsf{p}'| - 1$.

Definition 8. *Let* A *be an arena. A* play *on* A *is a pointed sequence* $\mathsf{p} = \langle \mathsf{s}, f \rangle$ *such that, either* $\mathsf{s} = \epsilon$*, or* s_i *and* s_{i+1} *have opposite polarities for all* $i \in \{0, \dots, |\mathsf{p}| - 1\}$.

The game of A *(denoted* $\mathcal{G}_{\mathsf{A}}$*) is the set of prefix-closed sets of plays over* A.

A view *is a play* $\mathsf{p} = \langle \mathsf{s}, f \rangle$ *such that either* $\mathsf{p} = \epsilon$ *or the following properties hold:*

- p *is* $\circ$-shortsighted *:* $f(2k) = 2k - 1$ *for every* $2k \in \{2, \dots, |\mathsf{p}|\}$*;*
- p *is* $\bullet$-uniform *:* $\ell(\mathsf{s}_{2k+1}) = \ell(\mathsf{s}_{2k})$ *for every* $2k + 1 \in \{0, \dots, |\mathsf{p}|\}$.

A winning innocent strategy *(or* WIS *for short) for the game* $\mathcal{G}_{\mathsf{A}}$ *is a finite non-empty prefix-closed set* $\mathcal{S}$ *of views in* $\mathcal{G}_{\mathsf{A}}$ *such that:*

- $\mathcal{S}$ *is* $\circ$-complete*: if* $\mathsf{p} \in \mathcal{S}$ *and* p *as odd length,*
  then every successor of p *(in* $\mathcal{G}_{\mathsf{A}}$*) is also in* $\mathcal{S}$ *;*
- p *is* $\bullet$-total*: if* $\mathsf{p} \in \mathcal{S}$ *and* p *has even length,*
  then exactly one successor of p *(in* $\mathcal{G}_{\mathsf{A}}$*) is in* $\mathcal{S}$ *;*

A view is maximal *in* $\mathcal{S}$ *if it is not prefix of any other view in* $\mathcal{S}$. $\mathcal{S}$ *is* trivial *if* $\mathcal{S} = \{\epsilon\}$. *We say that* $\mathcal{S}$ *is a* WIS *for a sequent* $A_1, \dots, A_n \vdash C$ *if* $\mathcal{S}$ *is a* WIS *for* $[\![A_1, \dots, A_n \vdash C]\!]$.

The definition of WIS above is a reformulation of the one in the literature of game semantics for intuitionistic propositional logic [14,26,29]. In presence of modalities, this definition requires to be refined to guarantee the possibility of gather modalities in *batches* corresponding to the modalities introduced by a

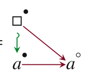
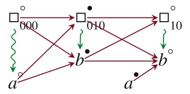
Arena	$[\![(\Box a) \to a]\!] =$ 	$[\![(\Box a \to \Box b) \to \Box(a \to b)]\!] =$ 
WIS	$\mathcal{S}_1 = \{\epsilon, a^\circ, a^\circ a^\bullet\}$	$\mathcal{S}_2 = \{\epsilon, b^\circ, b^\circ b^\bullet, b^\circ b^\bullet a^\circ, b^\circ b^\bullet a^\circ a^\bullet\}$
(failed) Derivation	$\dfrac{\dfrac{FAIL}{\Box a \vdash a}}{\vdash \Box a \to a}\,{\to}^{\mathsf{R}}$	$\dfrac{\dfrac{\dfrac{\dfrac{FAIL}{\vdash a}}{\vdash \Box^\circ a}\,\mathsf{K}^\Box \quad \dfrac{\dfrac{\dfrac{\dfrac{}{b \vdash b}\,\mathsf{ax}}{b, a \vdash b}\,\mathsf{W}}{b \vdash a \to b}\,{\to}^{\mathsf{L}}}{\Box^\bullet b \vdash \Box^\circ(a \to b)}\,\mathsf{K}^\Box}{\Box^\circ a \to \Box^\bullet b \vdash \Box^\circ(a \to b)}\,{\to}^{\mathsf{L}}}{\vdash (\Box^\circ a \to \Box^\bullet b) \to \Box^\circ(a \to b)}\,{\to}^{\mathsf{R}}$

Fig. 6. Examples of WISs for arenas not corresponding to proofs.

single application of the $\mathsf{K}^\Box$ (see Fig. 2). By means of example, consider the following arenas and their corresponding WISs, which cannot represent valid proofs in CK because of the impossibility of applying rules handling the modalities in a correct way.

Example 3. Consider the formulas $F_1 = (\Box a) \to a$ and $F_2 = (\Box a \to \Box b) \to \Box(a \to b)$ and their arenas in Fig. 6. The set of views $\mathcal{S}_1$ and $\mathcal{S}_2$ are WISs for F_1 and F_2 respectively. However, these formulas are not provable in SCK because the proof search fails (see Fig. 6). In particular, in the first case, no $\mathsf{K}^\Box$ can be applied because only there is a mismatch between the modalities on the left-hand side and on the right-hand side of the sequent; in the second case the problem is more subtle and, intuitively, is related to the fact that each $\mathsf{K}^\Box$ can remove only a single $\Box^\circ$ at a time, corresponding to the modality of the unique formula on the right-hand side of the sequent.

Therefore, in order to capture provability in CK, the notion of winning strategies has to be refined as follows.

Definition 9. *Let* $\mathsf{p} = (\mathsf{s}, f)$ *be a view in a strategy* $\mathcal{S}$ *on an arena* A*, and let* $h_\mathsf{p} = 1 + \max\{h_v \mid v \in \mathsf{p}\}$*. We define the* batched view *of* p *as the* $h_\mathsf{p} \times n$ *matrix* $\mathcal{F}(\mathsf{p}) = \big(\mathcal{F}(\mathsf{p})_0, \dots, \mathcal{F}(\mathsf{p})_n\big)$ *with elements in* $V_\mathcal{G} \cup \{\epsilon\}$ *such that the each column* $\mathcal{F}(\mathsf{p})_i$ *is defined as follows:*

$$\mathcal{F}(\mathsf{p})_i = \begin{pmatrix} \mathcal{F}(\mathsf{p})_i^{h_\mathsf{p}} \\ \vdots \\ \mathcal{F}(\mathsf{p})_i^0 \end{pmatrix} \quad \textit{where} \quad \begin{cases} \mathcal{F}(\mathsf{p})_i^{h_\mathsf{p}} = \mathsf{add}^{h_{\mathsf{p}_i}}(\mathsf{p}_i), \dots, \mathcal{F}(\mathsf{p})_i^{h_\mathsf{p} - h_{\mathsf{p}_i} + 1} = \mathsf{add}^1(\mathsf{p}_i) \\ \mathcal{F}(\mathsf{p})_i^{h_\mathsf{p} - h_{\mathsf{p}_i}} = \epsilon, \dots, \mathcal{F}(\mathsf{p})_i^1 = \epsilon \\ \mathcal{F}(\mathsf{p})_i^0 = \mathsf{p}_i \end{cases}$$

We say that p *is* well-batched *if* $|\mathsf{add}(\mathsf{s}_{2k})| = |\mathsf{add}(\mathsf{s}_{2k+1})|$ *for every* $2k \in \{0, \dots, |\mathsf{p}| - 1\}$*. Each well-batched view* p *induces an equivalence relation* $\overset{\mathcal{G}_\mathsf{p}}{\sim}$ *over* $V_\mathcal{G}$ *generated by:*

$$u \overset{\mathcal{G}_\mathsf{p}}{\sim}_1 w \quad \textit{iff} \quad u = \mathcal{F}(\mathsf{p})_{2k}^h \textit{ and } w = \mathcal{F}(\mathsf{p})_{2k+1}^h \textit{ for a } 2k < n - 1 \textit{ and a } h \le h_\mathsf{p} \tag{9}$$

A WIS $\mathcal{S}$ *is* linked *if it contains only well-batched views and if for every* $\mathsf{p} \in \mathcal{S}$ *the* $\overset{\mathcal{G}_\mathsf{p}}{\sim}$*-classes are of the shape* $\{v_1^\bullet, \ldots, v_n^\bullet, w^\circ\}$.

A CK-winning innocent strategy *(or* CK-WIS *for short) is a linked* WIS $\mathcal{S}$*.* [4]

Example 4. Consider the arenas in Fig. 6. The batched view of the (unique) maximal views in $\mathcal{S}_1$ and $\mathcal{S}_2$ are $\begin{pmatrix} \epsilon & \Box^\bullet \\ a^\circ & a^\bullet \end{pmatrix}$ and $\begin{pmatrix} \Box^\circ_{10} & \Box^\bullet_{010} & \Box^\circ_{000} & \Box^\circ_{10} \\ b^\circ & b^\bullet & a^\circ & a^\bullet \end{pmatrix}$ respectively. The first is not well-batched because a° has height 0 while $a^\bullet$ has height 1, while the second, even if well-batched, is not linked because the $\overset{\mathcal{G}_\mathsf{p}}{\sim}$-class $\{\Box^\circ_{10}, \Box^\bullet_{010}, \Box^\circ_{000}\}$ contains two $\Box^\circ$.

The definition of CK-WISs allows us to obtain a full-completeness result with respect to CK which, together with the good compositionality properties of CK-WISs shown in [5,11], provides a full-complete denotational semantics for the logic CK. That is, every given CK-WIS is the encoding of a derivation in CK, and if a derivation $\mathcal{D}$ reduces via cut-elimination to a derivation $\mathcal{D}'$, then they are encoded by the same CK-WIS.

Theorem 4 **([5]).** *The set of* CK-WIS*s is a full-complete denotational model for* CK*.*

5.3 Full Completeness for Modal Lambda Terms in Normal Form

We can prove the full completeness result using the type system CK^F and relying on Theorem 3. For this purpose, we have to extend the definition of α-equivalence from terms to type assignments in order to avoid technicality in our proofs, since in arenas we keep no track of variable names. For example, consider the α-equivalent terms $\lambda x.x$ and $\lambda y.y$ whose derivation should be considered non-equivalent due to the fact that α-equivalence does not extends to type assignments, therefore the two occurrence of the axiom rule with conclusion $x : a \vdash x : a$ and $y : a \vdash y : a$ should be considered distinct.[5]

Definition 10. *Let* $A_1, \ldots, A_n \vdash C$ *be a sequent. We define* $\Lambda(\Gamma \vdash C)$ *as the set of terms* M *such that the typing derivation* $x_1 : A_1, \ldots, x_n : A_n \vdash M : C$ *is derivable, that is,*

$$\Lambda(\Gamma \vdash C) = \left\{ M \in \Lambda \mid x_1 : A_1, \ldots, x_n : A_n \vdash M : C \text{ is derivable for some } x_1, \ldots, x_n \right\} .$$

If $M, N \in \Lambda(\Gamma \vdash C)$*, we define* $M =_\alpha^{\Gamma;C} N$ *as the smallest equivalence relation generated by the rule* $\dfrac{M\{z_1, \ldots, z_n/x_1, \ldots, x_n\} = N\{z_1, \ldots, z_n/y_1, \ldots, y_n\}}{M =_\alpha^{\Gamma;C} N}\ z_i \text{ is fresh}$.
.

[4] We here provide a simpler definition of CK-WISs w.r.t. the one in [5]. In fact, we are able here to simplify this definition because we are considering the $\Diamond$-free fragment of CK.

[5] Note that another possible way to deal with this problem is to label non-modal vertices of arenas by pairs of propositional atoms and variables instead of propositional variables only.

$$\left\{\!\!\left\{ \mathsf{ax}\frac{}{\Gamma, x : c^{\bullet} \vdash x : c^{\circ}} \right\}\!\!\right\} = \{\epsilon, c^{\circ}, c^{\circ}c^{\bullet}\} \qquad \left\{\!\!\left\{ \to_{\mathsf{R}}^{*}\frac{\overset{\mathcal{D}'\,\|}{\Gamma, x_1 : A_1, \dots, x_n : A_n \vdash M : C}}{\Gamma \vdash \lambda x_1^{A_1} \cdots x_n^{A_n}.M : (A_1, \dots A_n) \to C} \right\}\!\!\right\} = \{\!\{\mathcal{D}'\}\!\}$$

$$\left\{\!\!\left\{ \to_{\mathsf{L}}^{\mathsf{ax}}\frac{\left\{ \overset{\mathcal{D}_i\,\|}{\Gamma, (A_1, \dots, A_n) \to c \vdash N_i : A_i} \right\}_{i \in \{1,\dots,n\}}}{\Gamma, y : (A_1, \dots, A_n) \to c \vdash yN_1 \cdots N_n : c} \right\}\!\!\right\} = \{\epsilon, c^{\circ}, c^{\circ}c^{\bullet}\} \cup \{c^{\circ}c^{\bullet}\mathsf{p} \mid \epsilon \neq \mathsf{p} \in \{\!\{\mathcal{D}_i\}\!\} \text{ for a } i \in \{1, \dots, n\}\}$$

$$\left\{\!\!\left\{ \mathsf{ex}\frac{\overset{\mathcal{D}'\,\|}{\sigma(\Gamma) \vdash M : C}}{\Gamma \vdash M : C} \right\}\!\!\right\} = \{f_\sigma(\mathsf{p}) \mid \mathsf{p} \in \{\!\{\mathcal{D}'\}\!\}, f_\sigma \text{ isomorphism between } [\![\Gamma \vdash M : C]\!] \text{ and } [\![\sigma(\Gamma) \vdash M : C]\!]\}$$

where $f_\sigma(\mathsf{p})$ is the view obtained by applying f_σ to each move in p (and updating its pointer accordingly)

$$\left\{\!\!\left\{ \mathsf{K}^{\Box}\frac{\overset{\mathcal{D}'\,\|}{x_1 : A_1, \dots, x_n : A_n \vdash M : C}}{\Delta, y_1 : \Box A_1, \dots, y_n : A_n \vdash M\,[x_1, \dots, x_n / y_1, \dots, y_n]_{\blacksquare} : \Box C} \right\}\!\!\right\} = \{\!\{\mathcal{D}'\}\!\}$$

$$\left\{\!\!\left\{ \to_{\mathsf{L}}^{\mathsf{K}}\frac{\left\{ \overset{\mathcal{D}_{i,j}\,\|}{\Gamma, \Delta \vdash T_{i,j} : A_{i,j}} \right\}_{i \in \{1,\dots,n\}, j \in \{1,\dots,k_i\}} \quad \overset{\mathcal{D}_0\,\|}{\Gamma, \Delta, x_1 : \Box B_1, \dots, x_n : \Box B_n \vdash M\,[x_1, \dots, x_n, \vec{z}/y_1, \dots, y_n, \vec{w}]_{\blacksquare} : \Box C}}{\Gamma, \underbrace{f_1 : (A_{1,1}, \dots, A_{1,k_1}) \to \Box B_1, \dots, f_n : (A_{n,1}, \dots, A_{n,k_n}) \to \Box B_n}_{\Delta} \vdash M\,[N_1, \dots, N_n, \vec{z}/y_1, \dots y_n, \vec{w}]_{\blacksquare} : \Box C} \right\}\!\!\right\}$$

$$=$$

$$\{\!\{\mathcal{D}_0\}\!\} \cup \left(\bigcup_{i \in \{1,\dots,n\}} \left\{c^{\circ}b_i^{\bullet}\mathsf{p} \mid \epsilon \neq \mathsf{p} \in \left\{\!\!\left\{\mathcal{D}_{i,j}\right\}\!\!\right\} \text{ for a } j \in \{1, \dots, k_i\}\right\}\right)$$

where c° (resp. $b_i^{\bullet}$) is the unique non-$\Box$ vertex in $\vec{R}_{[\![\Box C]\!]}$ (resp. in $[\![\Box B_i]\!]$).

Fig. 7. Rules to construct a CK-WIS from a type derivation in CK^{F}. For reasons of readability, we assume there is an implicit map identifying the moves in the arenas of the type assignment in the premises with the moves in the arena of the type assignment in the conclusion. Note that c° and $c^{\bullet}$ are occurrences of the same atom c, but we have decorate them to improve readability.

From now on, we consider derivations up the α-equivalence defined above, that is, we consider derivations up to renaming of the variables occurring in a typing context.

Theorem 5. *There is a one-to-one correspondence between terms in $\widehat{\Lambda} \cap \Lambda(\Gamma \vdash C)$ and* CK-WIS *for $\Gamma \vdash C$.*

Proof. Given a CK-WIS $\mathcal{S}$ for $\Gamma \vdash C$, we can define a (unique) typing derivation $\mathcal{D}_{\mathcal{S}}$ in CK^{F} of a term $T_{\mathcal{S}} \in \widehat{\Lambda} \cap \Lambda(\Gamma \vdash C)$ by induction on the lexicographic order over the pairs $(|\mathcal{S}|, |C|)$ reasoning on the inductive definition of $\widehat{\Lambda}$.

Similarly, given a type assignment $\Gamma \vdash T : C$. for a $T \in \widehat{\Lambda}$, then, by Theorem 3, there is a (unique) derivation $\mathcal{D}_T$ in CK^{F}. We define $\mathcal{S}_T$ as the CK-WIS defined by induction on the number of rules in $\mathcal{D}_T$ using the rules in Fig. 7. We conclude since we have that $\mathcal{S}_{T_{\mathcal{S}}} = \mathcal{S}$ and $T_{\mathcal{S}_T} = T$ by definition.

6 Conclusion

In this paper we introduced a new modal λ-calculus for the $\Diamond$-free fragment of the constructive modal logic CK (without conjunction or disjunction). This lambda calculus builds on the work in [32], by adding a restricted η-reduction as well as two new reduction rules dealing with the explicit substitution constructor used to model the modality $\Box$. We proved normalization and confluence for this calculus and we provide a one-to-one correspondence between the set of terms in normal form and the set of winning strategies for the logic CK introduced in [5].

We foresee the possibility of extending the result presented in this paper to the entire disjunction-free fragment of CK, for which winning strategies are already defined in [5]. For this purpose, we should consider additional term constructors for terms whose type is a conjunction, as well as a new Let-like operator to model terms whose type is the modality $\Diamond$-formula similar to the one proposed in [10]. For this reason, in future works we plan to reformulate our lambda-calculus in the light of the novel line of research on calculi with explicit substitutions [1,2,34,35]. This approach would allow us to simplify some of the technicalities and achieve a more elegant operational semantics. Another interesting prospective is to extend our approach to operational semantics to the Fitch-style modal λ-calculus studied in [53].

At the same time, we plan to make explicit that our game semantics provides a concrete model for the *cartesian closed categories* provided with a *strong monoidal endofunctor* [10,33]. Indeed, categorical semantics of the calculus in [10] is modeled by means of *cartesian closed categories* equipped with a *strong monoidal endofunctor* taking into account the proof-theoretical behavior of the $\Box$-modality. We further conjecture that the syntactic category obtained via the quotient of modal terms modulo the relations we introduce in this paper is indeed a *free cartesian closed category* on a set of atoms with a *strong monoidal endofunctor*.

References

1. Accattoli, B.: Exponentials as substitutions and the cost of cut elimination in linear logic. In: Baier, C., Fisman, D. (eds.) LICS 2022: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, 2–5 August 2022, pp. 49:1–49:15. ACM (2022). https://doi.org/10.1145/3531130.3532445
2. Accattoli, B., Bonelli, E., Kesner, D., Lombardi, C.: A nonstandard standardization theorem. Association for Computing Machinery, New York (2014). https://doi.org/10.1145/2535838.2535886
3. Acclavio, M.: Proof diagrams for multiplicative linear logic: syntax and semantics. J. Autom. Reason. **63**(4), 911–939 (2019). https://doi.org/10.1007/s10817-018-9466-4
4. Acclavio, M., Catta, D., Olimpieri, F.: Canonicity of proofs in constructive modal logic (extended version) (2023). https://doi.org/10.48550/arXiv.2304.05465
5. Acclavio, M., Catta, D., Straßburger, L.: Game semantics for constructive modal logic. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 428–445. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_25
6. Acclavio, M., Straßburger, L.: Combinatorial proofs for constructive modal logic. In: Advances in Modal Logic 2022, Rennes, France (2022). https://hal.inria.fr/hal-03909538

7. Alechina, N., Mendler, M., de Paiva, V., Ritter, E.: Categorical and Kripke semantics for constructive S4 modal logic. In: Fribourg, L. (ed.) CSL 2001. LNCS, vol. 2142, pp. 292–307. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44802-0_21
8. Andreoli, J.M.: Focussing and proof construction. Ann. Pure Appl. Logic **107**, 131–163 (2001)
9. Barendregt, H.P., Dekkers, W., Statman, R.: Lambda Calculus with Types. Perspectives in logic. Cambridge University Press, Cambridge (2013). http://www.cambridge.org/de/academic/subjects/mathematics/logic-categories-and-sets/lambda-calculus-types
10. Bellin, G., De Paiva, V., Ritter, E.: Extended Curry-Howard correspondence for a basic constructive modal logic. In: Proceedings of Methods for Modalities (2001)
11. Catta, D.: Les preuves vues comme des jeux et réciproquement: sémantique dialogique de langages naturel ou logiques. (Proofs as games and games as proofs: dialogical semantics of logical and natural languages). Ph.D. thesis, University of Montpellier, France (2021). https://tel.archives-ouvertes.fr/tel-03588308
12. Chaudhuri, K., Marin, S., Straßburger, L.: Modular focused proof systems for intuitionistic modal logics. In: Kesner, D., Pientka, B. (eds.) 1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, Porto, Portugal, 22–26 June 2016, LIPIcs, vol. 52, pp. 16:1–16:18. Leibniz-Zentrum fuer Informatik (2016)
13. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. J. Symb. Logic **44**(1), 36–50 (1979)
14. Danos, V., Herbelin, H., Regnier, L.: Game semantics & abstract machines. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, 27–30 July 1996, pp. 394–405. IEEE Computer Society (1996). https://doi.org/10.1109/LICS.1996.561456
15. Das, A., Marin, S.: Brouwer meets kripke: constructivising modal logic. https://prooftheory.blog/2022/08/19/brouwer-meets-kripke-constructivising-modal-logic/. Accessed 19 Aug 2022
16. Das, A., Pous, D.: Non-wellfounded proof theory for (Kleene+action)(algebras+lattices). In: Computer Science Logic (CSL), Birmingham, United Kingdom (2018). https://doi.org/10.4230/LIPIcs.CSL.2018.19. https://hal.archives-ouvertes.fr/hal-01703942
17. Davies, R., Pfenning, F.: A modal analysis of staged computation. J. ACM **48**(3), 555–604 (2001). https://doi.org/10.1145/382780.382785
18. Di Cosmo, R., Kesner, D.: Combining algebraic rewriting, extensional lambda calculi, and fixpoints. Theor. Comput. Sci. **169**(2), 201–220 (1996). https://doi.org/10.1016/S0304-3975(96)00121-1
19. Došen, K.: Identity of proofs based on normalization and generality. Bull. Symb. Logic **9**, 477–503 (2003)
20. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Sci. Comput. Program. **2**(3), 241–266 (1982). https://doi.org/10.1016/0167-6423(83)90017-5
21. Fairtlough, M., Mendler, M.: Propositional lax logic. Inf. Comput. **137**(1), 1–33 (1997)
22. Fitch, F.: Intuitionistic modal logic with quantifiers. Portugaliae Mathematica **7**(2), 113–118 (1948)
23. Girard, J.Y.: Linear logic. Theor. Comput. Sci. **50**, 1–102 (1987). https://doi.org/10.1016/0304-3975(87)90045-4

24. Girard, J.Y.: Proof-nets?: the parallel syntax for proof-theory. In: Ursini, A., Agliano, P. (eds.) Logic and Algebra. Marcel Dekker, New York (1996)
25. Guglielmi, A., Gundersen, T., Parigot, M.: A proof calculus which reduces syntactic bureaucracy. In: Lynch, C. (ed.) Proceedings of the 21st International Conference on Rewriting Techniques and Applications, LIPIcs, vol. 6, pp. 135–150. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2010). https://doi.org/10.4230/LIPIcs.RTA.2010.135. http://drops.dagstuhl.de/opus/volltexte/2010/2649
26. Heijltjes, W., Hughes, D., Straßburger, L.: Intuitionistic proofs without syntax. In: LICS 2019–34th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 1–13. IEEE, Vancouver (2019). https://doi.org/10.1109/LICS.2019.8785827. https://hal.inria.fr/hal-02386878
27. Heilala, S., Pientka, B.: Bidirectional decision procedures for the intuitionistic propositional modal logic **IS4**. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 116–131. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73595-3_9
28. Hughes, D.: Proofs Without Syntax. Ann. Math. **164**(3), 1065–1076 (2006). https://doi.org/10.4007/annals.2006.164.1065
29. Hyland, J.M.E., Ong, C.L.: On full abstraction for PCF: i, ii, and III. Inf. Comput. **163**(2), 285–408 (2000). https://doi.org/10.1006/inco.2000.2917
30. Hyland, J.M.E., Ong, C.H.L.: On full abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model. Inf. Comput. **163**, 285–408 (2000)
31. Jay, C.B., Ghani, N.: The virtues of eta-expansion. J. Funct. Program. **5**(2), 135–154 (1995). https://doi.org/10.1017/S0956796800001301
32. Kakutani, Y.: Call-by-name and call-by-value in normal modal logic. In: Shao, Z. (ed.) APLAS 2007. LNCS, vol. 4807, pp. 399–414. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76637-7_27
33. Kavvos, G.A.: Dual-context calculi for modal logic. Log. Methods Comput. Sci. **16**(3) (2020). https://doi.org/10.23638/LMCS-16(3:10)2020
34. Kesner, D.: The theory of calculi with explicit substitutions revisited. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 238–252. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74915-8_20
35. Kesner, D.: A theory of explicit substitutions with safe and full composition. Log. Methods Comput. Sci. **5**(3) (2009). http://arxiv.org/abs/0905.2539
36. Kojima, K.: Semantical study of intuitionistic modal logics. Ph.D. thesis, Kyoto University (2012)
37. Kozen, D.: Results on the propositional mu-calculus. Theor. Comput. Sci. **27**, 333–354 (1983). https://doi.org/10.1016/0304-3975(82)90125-6
38. Krivine, J.: Lambda-calculus, types and models. Ellis Horwood series in computers and their applications, Masson (1993)
39. Kuznets, R., Marin, S., Straßburger, L.: Justification logic for constructive modal logic. J. Appl. Logics IfCoLog J. Logics Appl. **8**(8), 2313–2332 (2021). https://hal.inria.fr/hal-01614707
40. Mendler, M., Scheele, S.: Cut-free Gentzen calculus for multimodal CK. Inf. Comput. **209**(12), 1465–1490 (2011)
41. Meyer, J.J., Veltmanw, F.: Intelligent agents and common sense reasoning. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, Studies in Logic and Practical Reasoning, vol. 3, pp. 991–1029. Elsevier (2007). https://doi.org/10.1016/S1570-2464(07)80021-8. http://www.sciencedirect.com/science/article/pii/S1570246407800218

42. Miller, D., Volpe, M.: Focused labeled proof systems for modal logic. In: Davis, M., Fehnker, A., McIver, A., Voronkov, A. (eds.) LPAR 2015. LNCS, vol. 9450, pp. 266–280. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48899-7_19
43. Mints, G.E.: Closed categories and the theory of proofs. J. Soviet Math. (1981). https://doi.org/10.1007/BF01404107
44. Murawski, A.S., Ong, C.-H.L.: Discreet games, light affine logic and PTIME computation. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 427–441. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44622-2_29
45. Pfenning, F., Davies, R.: A judgmental reconstruction of modal logic. Math. Struct. Comput. Sci. **11**(4), 511–540 (2001). https://doi.org/10.1017/S0960129501003322
46. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October–1 November 1977, pp. 46–57. IEEE Computer Society (1977). https://doi.org/10.1109/SFCS.1977.32
47. Prawitz, D.: Natural Deduction: A Proof-Theoretical Study. Almquist and Wiksell (1965)
48. Simpson, A.: The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh (1994)
49. Sørensen, M.H., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Elsevier, Amsterdam (2006)
50. Terese: Term rewriting systems. Cambridge University Press (2003)
51. Tubella, A.A., Straßburger, L.: Introduction to Deep Inference (2019). https://hal.inria.fr/hal-02390267. lecture
52. Vakarelov, D.: Modal logics for knowledge representation systems. Theor. Comput. Sci. **90**, 433–456 (1991)
53. Valliappan, N., Ruch, F., Tom'e Corti nas, C.: Normalization for fitch-style modal calculi. Proc. ACM Program. Lang. **6**(ICFP), 772–798 (2022). https://doi.org/10.1145/3547649

Linear Logic and MV-Algebras

Proof-Theoretic Semantics for Intuitionistic Multiplicative Linear Logic

Alexander V. Gheorghiu[1](✉), Tao Gu[1](✉), and David J. Pym[1,2](✉)

[1] University College London, London WC1E 6BT, UK
{alexander.gheorghiu.19,tao.gu.18,d.pym}@ucl.ac.uk
[2] Institute of Philosophy, University of London, London WC1H 0AR, UK

Abstract. This work is the first exploration of proof-theoretic semantics for a substructural logic. It focuses on the base-extension semantics (B-eS) for intuitionistic multiplicative linear logic (IMLL). The starting point is a review of Sandqvist's B-eS for intuitionistic propositional logic (IPL), for which we propose an alternative treatment of conjunction that takes the form of the *generalized* elimination rule for the connective. The resulting semantics is shown to be sound and complete. This motivates our main contribution, a B-eS for IMLL, in which the definitions of the logical constants all take the form of their elimination rule and for which soundness and completeness are established.

Keywords: Logic · Semantics · Proof Theory · Proof-theoretic Semantics · Substructural Logic · Multiplicative Connectives

1 Introduction

In model-theoretic semantics (M-tS), logical consequence is defined in terms of models; that is, abstract mathematical structures in which propositions are interpreted and their truth is judged. As Schroeder-Heister [33] explains, in the standard reading given by Tarski [38,39], a propositional formula φ follows model-theoretically from a context Γ iff every model of Γ is a model of φ; that is,

$$\Gamma \models \varphi \quad \text{iff} \quad \text{for all models } \mathcal{M}, \text{ if } \mathcal{M} \models \psi \text{ for all } \psi \in \Gamma, \text{ then } \mathcal{M} \models \varphi$$

Therefore, consequence is understood as the transmission of truth. Importantly, on this plan, *meaning* and *validity* are characterized is terms of *truth*.

Proof-theoretic semantics (P-tS) is an alternative approach to meaning and validity in which they are characterized in terms of *proofs*—understood as objects denoting collections of acceptable inferences from accepted premisses. This is subtle. It is not that one desires a proof system that precisely characterizes the consequences of the logic of interest, but rather that one desires to express the *meaning* of the logical constants in terms of proofs and provability. Indeed, as Schroeder-Heister [33] observes, since no formal system is fixed (only notions of inference) the relationship between semantics and provability remains the same as it has always been—in particular, soundness and completeness are desirable features of formal systems. Essentially, what differs is that *proofs* serve the role of *truth* in model-theoretic semantics. The semantic paradigm supporting P-tS is *inferentialism*—the view that meaning (or validity) arises from rules of inference (see Brandom [5]).

To illustrate the paradigmatic shift from M-tS to P-tS, consider the proposition 'Tammy is a vixen'. What does it mean? Intuitively, it means, somehow, 'Tammy is female' *and* 'Tammy is a fox'. On inferentialism, its meaning is given by the rules,

$$\frac{\text{Tammy is a fox} \quad \text{Tammy is female}}{\text{Tammy is a vixen}} \qquad \frac{\text{Tammy is a vixen}}{\text{Tammy is female}} \qquad \frac{\text{Tammy is a vixen}}{\text{Tammy is a fox}}$$

These merit comparison with the laws governing $\wedge$ in IPL, which justify the sense in which the above proposition is a conjunction:

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \wedge \psi}{\varphi} \qquad \frac{\varphi \wedge \psi}{\psi}$$

There are two major branches of P-tS: proof-theoretic validity (P-tV) in the Dummett-Prawitz tradition (see, for example, Schroeder-Heister [32]) and base-extension semantics (B-eS) in the sense of, for example, Sandqvist [28–30]. The former is a semantics of arguments, and the latter is a semantics of a logic, but both are *proof-theoretic semantics*. This paper is concerned with the latter as explained below.

Tennant [40] provides a general motivation for P-tV: reading a *consequence* judgement $\Gamma \vdash \varphi$ proof-theoretically—that is, that φ follows by some reasoning from Γ—demands a notion of *valid argument* that encapsulates what the forms of valid reasoning are. That is, we require explicating the semantic conditions required for an argument that witnesses

$$\psi_1, \ldots, \psi_n; \text{ therefore, } \varphi$$

to be valid. A particular motivation comes from the following programmatic remarks by Gentzen [37]:

> The introductions represent, as it were, the 'definitions' of the symbols concerned, and the eliminations are no more, in the final analysis, than the consequences of these definitions. This fact may be expressed as follows: In eliminating a symbol, we may use the formula with whose terminal symbol we are dealing only 'in the sense afforded it by the introduction of that symbol'.

Dummett [9] developed a philosophical understanding of the normalization results of Prawitz [25], which give a kind of priority to the introduction rules, that yields a notion of valid arguments. The result is P-tV—see Schroeder-Heister [32] for a succinct explanation.

(At)	$\Vdash_{\mathscr{B}} \mathrm{p}$	iff	$\vdash_{\mathscr{B}} \mathrm{p}$
$(\to)$	$\Vdash_{\mathscr{B}} \varphi \to \psi$	iff	$\varphi \Vdash_{\mathscr{B}} \psi$
$(\wedge)$	$\Vdash_{\mathscr{B}} \varphi \wedge \psi$	iff	$\Vdash_{\mathscr{B}} \varphi$ and $\Vdash_{\mathscr{B}} \psi$
$(\vee)$	$\Vdash_{\mathscr{B}} \varphi \vee \psi$	iff	for any $\mathscr{C}$ such that $\mathscr{B} \subseteq \mathscr{C}$ and any $\mathrm{p} \in \mathbb{A}$, if $\varphi \Vdash_{\mathscr{C}} \mathrm{p}$ and $\psi \Vdash_{\mathscr{C}} \mathrm{p}$, then $\Vdash_{\mathscr{C}} \mathrm{p}$
$(\bot)$	$\Vdash_{\mathscr{B}} \bot$	iff	$\Vdash_{\mathscr{B}} \mathrm{p}$ for any $\mathrm{p} \in \mathbb{A}$
(Inf)	$\Gamma \Vdash_{\mathscr{B}} \varphi$	iff	for any $\mathscr{C}$ such that $\mathscr{B} \subseteq \mathscr{C}$, if $\Vdash_{\mathscr{C}} \psi$ for any $\psi \in \psi$, then $\Vdash_{\mathscr{C}} \varphi$

Fig. 1. Sandqvist's Support in a Base

More generally, P-tV is about defining a notion of *validity* of objects witnessing that a formula φ follows by some reasoning from a collection of formulae Γ. This is quite different from simply giving an interpretation of proofs from some formal system; for example, while the version of P-tV discussed above is closely related to the BHK interpretation of IPL, it is important to distinguish the semantic and computational aspects—see, for example, Schroeder-Heister [32].

Meanwhile, B-eS proceeds via a judgement called *support* defined inductively according to the structure of formulas with the base case (i.e., the support of atoms) given by proof in a base. A *base* is a set of inference rules over atomic propositions, thought of as defining those atoms—an example is the set of rules above that define 'Tammy is a vixen'. Though this approach is closely related to possible world semantics in the sense of Beth [2] and Kripke [17]—see, for example, Goldfarb [13] and Makinson [18]—it remains subtle. For example, there are several incompleteness results for intuitionistic logics—see, for example, Piecha et al. [20,21,23], Goldfarb [13], Sandqvist [27–30], Stafford [36]. Significantly, a sound and complete B-eS for IPL has been given by Sandqvist [29]. Gheorghiu and Pym [10] have shown that this B-eS captures the declarative content of P-tV.

Sandqvist's B-eS for IPL is the point of departure for this paper. Fix a set of atomic propositions $\mathbb{A}$. Given a base $\mathscr{B}$, we write $\vdash_{\mathscr{B}} \mathrm{p}$ to denote that $\mathrm{p} \in \mathbb{A}$ can be derived in $\mathscr{B}$. Support in a base $\mathscr{B}$—denoted $\Vdash_{\mathscr{B}}$—is defined by the clauses of Fig. 1 in which $\Gamma \neq \varnothing$. We desire to give an analogous semantics for *intuitionistic multiplicative linear logic* (IMLL). We study this logic as it is the minimal setting in which we can explore how to set-up B-eS (and P-tS in general) for substructural logics, which enables extension to, for example, (intuitionistic) Linear Logic [11] and the logic of Bunched Implications [19]. Again, the aim is not simply to give a proof-theoretic interpretation of IMLL, which already exist, but to define the logical constants in terms of proofs.

A compelling reading of IMLL is its resource interpretation, which is inherently proof-theoretic—see Girard [11]. Accordingly, looking at (Inf), we expect that φ being supported in a base $\mathscr{B}$ relative to some multiset of formulas Γ means that the 'resources' garnered by Γ suffice to produce φ. We may express this by enriching the notion of support with multisets of resources P and U combined with multiset union—denoted $\fatcomma$. Then, that the resources garnered by Γ are given to φ is captured by the following property:

$$\Gamma \Vdash^{\mathrm{P}}_{\mathscr{B}} \varphi \qquad \text{iff} \qquad \text{for any } \mathscr{X} \supseteq \mathscr{B} \text{ and any } U, \text{ if } \Vdash^{U}_{\mathscr{X}} \Gamma, \text{ then } \Vdash^{\mathrm{P}\fatcomma\mathrm{U}}_{\mathscr{X}} \varphi$$

Naively, we may define $\otimes$ as a resource-sensitive version of ($\wedge$); that is,

$$\Vdash^{\mathrm{P}}_{\mathscr{B}} \varphi \otimes \psi \quad \text{iff} \quad \text{there are } \mathrm{P}_1, \mathrm{P}_2 \text{ such that } \mathrm{P} = (\mathrm{P}_1 \fatcomma \mathrm{P}_2), \Vdash^{\mathrm{P}_1}_{\mathscr{B}} \varphi, \text{ and } \Vdash^{\mathrm{P}_2}_{\mathscr{B}} \psi$$

While the semantics is sound, proving completeness is more subtle. We aim to follow the method by Sandqvist [30], and this clause is not suitable because the following is not the case for IMLL:

$$\Gamma \vdash \varphi \otimes \psi \quad \text{iff} \quad \text{there are } \Delta_1, \Delta_2 \text{ such that } \Gamma = (\Delta_1 \fatcomma \Delta_2),\ \Delta_1 \vdash \varphi, \text{ and } \Delta_2 \vdash \psi$$

—a counter-example is the case where Γ is the (singleton) multiset consisting of $\varphi \otimes \psi$, which denies any non-trivial partition into smaller multisets. We therefore take a more complex clause, which is inspired by the treatment of disjunction in IPL, that enables us to prove completeness using the approach by Sandqvist [29].

There is an obvious difference between the B-eS for IPL and its standard possible world semantics by Kripke [17]—namely, the treatment of disjunction ($\vee$) and absurdity ($\bot$). The possible world semantics has the clause,

$$\mathfrak{M}, x \Vdash \varphi \vee \psi \qquad \text{iff} \qquad \mathfrak{M}, x \Vdash \varphi \text{ or } \mathfrak{M}, x \Vdash \psi$$

If such a clause is taken in the definition of validity in a B-eS for IPL, it leads to incompleteness —see, for example Piecha and Schroeder-Heister [20,21]. To yield completeness, Sandqvist [30] uses a more complex form that is close to the elimination rule for disjunction in natural deduction (see Gentzen [37] and Prawitz [24])—that is,

$$\begin{array}{lll} \Vdash_{\mathscr{B}} \varphi \vee \psi & \quad \text{iff} \quad & \text{for any } \mathscr{C} \text{ such that } \mathscr{B} \subseteq \mathscr{C} \text{ and any } \mathrm{p} \in \mathbb{A}, \\ & & \text{if } \varphi \Vdash_{\mathscr{C}} \mathrm{p} \text{ and } \psi \Vdash_{\mathscr{C}} \mathrm{p}, \text{ then } \Vdash_{\mathscr{C}} \mathrm{p} \end{array}$$

One justification for the clauses is the principle of *definitional reflection* (DR) (see Hallnäs [14,15] and Schroeder-Heister [31]):

> whatever follows from all the premisses of an assertion also follows from the assertion itself

Taking the perspective that the introduction rules are definitions, DR provides an answer for the way in which the elimination rules follow. Similarly, it justifies that the clauses for the logical constants take the form of their elimination rules.

Why does the clause for conjunction ($\wedge$) not take the form given by DR? What DR gives is the *generalized* elimination rule,

$$\frac{\varphi \wedge \psi \qquad \begin{array}{c}[\varphi, \psi] \\ \chi\end{array}}{\chi}$$

We may modify the B-eS for IPL by replacing ($\wedge$) with the following:

$$(\wedge^*) \quad \Vdash_{\mathscr{B}} \varphi \wedge \psi \quad \text{iff} \quad \text{for any } \mathscr{C} \supseteq \mathscr{B} \text{ and any } \mathrm{p} \in \mathbb{A}, \text{ if } \varphi, \psi \Vdash_{\mathscr{C}} \mathrm{p}, \text{ then } \Vdash_{\mathscr{C}} \mathrm{p}$$

We show in Sect. 2.3 that the result does indeed characterize IPL. Indeed, it is easy to see that the generalized elimination rule and usual elimination rule for $\wedge$ have the same expressive power.

Note, we here take the definitional view of the introduction rules for the logical constants of IPL, and not of bases themselves, thus do not contradict the distinctions made by Piecha and Schroeder-Heister [22,34].

Taking this analysis into consideration, we take the following definition of the multiplicative conjunction that corresponds to the definitional reflection of its introduction rule:

$$\Vdash^{\mathrm{P}}_{\mathscr{B}} \varphi \otimes \psi \quad \text{iff} \quad \begin{array}{l} \text{for any } \mathscr{X} \supseteq \mathscr{B}, \text{ resources U, and } \mathrm{p} \in \mathbb{A}, \\ \text{if } \varphi \mathbin{,} \psi \Vdash^{\mathrm{U}}_{\mathscr{X}} \mathrm{p}, \text{ then } \Vdash^{\mathrm{P},\mathrm{U}}_{\mathscr{X}} \mathrm{p} \end{array}$$

We show in Sect. 4 that the result does indeed characterize IMLL.

The paper is structured as follows: in Sect. 2, we review the B-eS for IPL given by Sandqvist [29]; in Sect. 3, we define IMLL and provide intuitions about its B-eS; in Sect. 4, we formally define the B-eS for IMLL and explain its soundness and completeness proofs. The paper ends in Sect. 5 with a conclusion and summary of results.

2 Base-Extension Semantics for IPL

In this section, we review the B-eS for IPL given by Sandqvist [29]. In Sect. 2.1, we give a terse but complete definition of the B-eS for IPL. In Sect. 2.2, we summarize the completeness proof. Finally, in Sect. 2.3, we discuss a modification of the treatment of conjunction. While IPL is not the focus of this paper, this review provides intuition and motivates the B-eS for IMLL in Sect. 3. Specifically, the analysis of the treatment of conjunction in IPL motivates the handling of the multiplicative conjunction in IMLL.

Throughout this section, we fix a denumerable set of atomic propositions $\mathbb{A}$, and the following conventions: $\mathrm{p}, \mathrm{q}, \ldots$ denote atoms; $\mathrm{P}, \mathrm{Q}, \ldots$ denote finite sets of atoms; $\varphi, \psi, \theta, \ldots$ denote formulas; $\Gamma, \Delta, \ldots$ denote finite sets of formulas.

We forego an introduction to IPL, which is doubless familiar—see van Dalen [7]. For clarity, note that we distinguish sequents $\Gamma \triangleright \varphi$ from judgements $\Gamma \vdash \varphi$ that say that the sequent is valid in IPL.

2.1 Support in a Base

The B-eS for IPL begins by defining *derivability in a base.* A (properly) second-level atomic rule—see Piecha and Schroeder-Heister [22,34]— is a natural deduction rule of the following form, in which $q, q_1, ..., q_n$ are atoms and Q_1,...,Q_n are (possibly empty) sets of atoms:

$$\frac{}{\mathrm{q}} \qquad \frac{\begin{matrix}[Q_1] & & [Q_n] \\ \mathrm{q}_1 & \dots & \mathrm{q}_n\end{matrix}}{\mathrm{q}}$$

Importantly, atomic rules are taken *per se* and not closed under substitution. They may be expressed inline as $(\mathrm{Q}_1 \triangleright \mathrm{q}_1, \ldots, \mathrm{Q}_n \triangleright \mathrm{q}_n) \Rightarrow \mathrm{q}$—note, the axiom case is the special case when the left-hand side is empty, $\Rightarrow \mathrm{q}$. They are read as natural deduction rules in the sense of Gentzen [37]; thus, $\Rightarrow q$ means that the atom q may be concluded whenever, while $(\mathrm{Q}_1 \triangleright \mathrm{q}_1, \ldots, \mathrm{Q}_n \triangleright \mathrm{q}_n) \Rightarrow \mathrm{q}$ means that one may derive q from a set of atoms S if one has derived q_i from S assuming Q_i for $i = 1, ..., n$.

A *base* is a set of atomic rules. We write $\mathscr{B}, \mathscr{C}, \ldots$ to denote bases, and $\varnothing$ to denote the empty base (i.e., the base with no rules). We say $\mathscr{C}$ is an *extension* of $\mathscr{B}$ if $\mathscr{C}$ is a superset of $\mathscr{B}$, denoted $\mathscr{C} \supseteq \mathscr{B}$.

Definition 1 (Derivability in a Base). Derivability in a base $\mathscr{B}$ *is the least relation* $\vdash_{\mathscr{B}}$ *satisfying the following:*

(Ref-IPL) $\mathrm{S}, \mathrm{q} \vdash_{\mathscr{B}} \mathrm{q}$.
(App-IPL) *If atomic rule* $(\mathrm{Q}_1 \triangleright \mathrm{q}_1, \ldots, \mathrm{Q}_n \triangleright \mathrm{q}_n) \Rightarrow \mathrm{q}$ *is in* $\mathscr{B}$*, and* $\mathrm{S}, \mathrm{Q}_i \vdash_{\mathscr{B}} \mathrm{q}_i$ *for all* $i = 1, \ldots, n$*, then* $\mathrm{S} \vdash_{\mathscr{B}} \mathrm{q}$.

This forms the base case of the B-eS for IPL:

Definition 2 (Sandqvist's Support in a Base). Sandqvist's support in a base $\mathscr{B}$ *is the least relation* $\Vdash_{\mathscr{B}}$ *defined by the clauses of Fig. 1. A sequent* $\Gamma \triangleright \varphi$ *is* valid*—denoted* $\Gamma \Vdash \varphi$*—iff it is supported in every base,*

$$\Gamma \Vdash \varphi \qquad \textit{iff} \qquad \Gamma \Vdash_{\mathscr{B}} \varphi \textit{ holds for any } \mathscr{B}$$

Every base is an extension of the empty base ($\varnothing$), therefore $\Gamma \Vdash \varphi$ iff $\Gamma \Vdash_{\varnothing} \varphi$. Sandqvist [29] showed that this semantics characterizes IPL:

Theorem 1 (Sandqvist [29]). $\Gamma \vdash \varphi$ *iff* $\Gamma \Vdash \varphi$

Soundness—that is, $\Gamma \vdash \varphi$ implies $\Gamma \Vdash \varphi$—follows from showing that $\Vdash$respects the rules of Gentzen's [37] NJ; for example, $\Gamma \Vdash \varphi$ and $\Delta \Vdash \psi$ implies $\Gamma, \Delta \Vdash \varphi \wedge \psi$. Completeness—that is, $\Gamma \Vdash \varphi$ implies $\Gamma \vdash \varphi$—is more subtle. We present the argument in Sect. 2.2 as it motivates the work in Sect. 4.3.

2.2 Completeness of IPL

We require to show that $\Gamma \Vdash \varphi$ implies that there is an NJ-proof witnessing $\Gamma \vdash \varphi$. To this end, we associate to each sub-formula ρ of $\Gamma \cup \{\varphi\}$ a unique atom r, and construct a base $\mathscr{N}$ such that r behaves in $\mathscr{N}$ as ρ behaves in NJ. Moreover, formulas and their atomizations are semantically equivalent in any extension of $\mathscr{N}$ so that support in $\mathscr{N}$ characterizes both validity and provability. When $\rho \in \mathbb{A}$, we take $\mathrm{r} := \rho$, but for complex ρ we choose r to be alien to Γ and φ.

$$\frac{\rho^\flat \quad \sigma^\flat}{(\rho \wedge \sigma)^\flat}\,\wedge^\flat_{\mathsf{I}} \qquad \frac{(\rho \wedge \sigma)^\flat}{\rho^\flat}\,\wedge^\flat_{\mathsf{E}} \qquad \frac{(\rho \wedge \sigma)^\flat}{\sigma^\flat}\,\wedge^\flat_{\mathsf{E}} \qquad \frac{\rho^\flat \quad (\rho \to \sigma)^\flat}{\sigma^\flat}\,\to^\flat_{\mathsf{E}}$$

$$\frac{\rho^\flat}{(\rho \vee \sigma)^\flat}\,\vee^\flat_{\mathsf{I}} \qquad \frac{\sigma^\flat}{(\rho \vee \sigma)^\flat}\,\vee^\flat_{\mathsf{I}} \qquad \frac{(\rho \vee \sigma)^\flat \quad \begin{matrix}[\rho^\flat]\\ \mathrm{p}\end{matrix} \quad \begin{matrix}[\sigma^\flat]\\ \mathrm{p}\end{matrix}}{\mathrm{p}}\,\vee^\flat_{\mathsf{E}} \qquad \frac{\begin{matrix}[\rho^\flat]\\ \sigma^\flat\end{matrix}}{(\rho \to \sigma)^\flat}\,\to^\flat_{\mathsf{I}} \qquad \frac{\bot^\flat}{\mathrm{p}}\,\mathsf{EFQ}^\flat$$

Fig. 2. Atomic System $\mathscr{N}$

Example 1. Suppose $\rho := \mathrm{p} \wedge \mathrm{q}$ is a sub-formula of $\Gamma \cup \{\varphi\}$. Associate to it a fresh atom r. Since the principal connective of ρ is $\wedge$, we require $\mathscr{N}$ to contain the following rules:

$$\frac{\mathrm{p} \quad \mathrm{q}}{\mathrm{r}} \qquad \frac{\mathrm{r}}{\mathrm{p}} \quad \frac{\mathrm{r}}{\mathrm{q}}$$

We may write $(\mathrm{p} \wedge \mathrm{q})^\flat$ for r so that these rules may be expressed as follows:

$$\frac{\mathrm{p} \quad \mathrm{q}}{(\mathrm{p} \wedge \mathrm{q})^\flat} \qquad \frac{(\mathrm{p} \wedge \mathrm{q})^\flat}{\mathrm{p}} \quad \frac{(\mathrm{p} \wedge \mathrm{q})^\flat}{\mathrm{q}}$$ ■

Formally, given a judgement $\Gamma \Vdash \varphi$, to every sub-formula ρ associate a unique atomic proposition $\rho^\flat$ as follows:

- if $\rho \notin \mathbb{A}$, then $\rho^\flat$ is an atom that does not occur in any formula in $\Gamma \cup \{\varphi\}$;
- if $\rho \in \mathbb{A}$, then $\rho^\flat = \rho$.

By *unique* we mean that $(\cdot)^\flat$ is injective—that is, if $\rho \neq \sigma$, then $\rho^\flat \neq \sigma^\flat$. The left-inverse of $(\cdot)^\flat$ is $(\cdot)^\natural$, and the domain may be extended to the entirety of $\mathbb{A}$ by identity on atoms not in the codomain of $(\cdot)^\flat$. Both functions act on sets pointwise—that is, $\Sigma^\flat := \{\varphi^\flat \mid \varphi \in \Sigma\}$ and $\mathrm{P}^\natural := \{\mathrm{p}^\natural \mid \mathrm{p} \in \mathrm{P}\}$. Relative to $(\cdot)^\flat$, let $\mathscr{N}$ be the base containing the rules of Fig. 2 for any sub-formulas ρ and σ of Γ and φ, and any $\mathrm{p} \in \mathbb{A}$.

Sandqvist [29] establishes three claims that deliver completeness:

(IPL-AtComp) Let $\mathrm{S} \subseteq \mathbb{A}$ and $\mathrm{p} \in \mathbb{A}$ and let $\mathscr{B}$ be a base: $\mathrm{S} \Vdash_{\mathscr{B}} \mathrm{p}$ iff $\mathrm{S} \vdash_{\mathscr{B}} \mathrm{p}$.
(IPL-Flat) For any sub-formula ξ of $\Gamma \cup \{\varphi\}$ and $\mathscr{N}' \supseteq \mathscr{N}$: $\Vdash_{\mathscr{N}'} \xi^\flat$ iff $\Vdash_{\mathscr{N}'} \xi$.
(IPL-Nat) Let $\mathrm{S} \subseteq \mathbb{A}$ and $\mathrm{p} \in \mathbb{A}$: if $\mathrm{S} \vdash_{\mathscr{N}} p$, then $\mathrm{S}^\natural \vdash p^\natural$.

The first claim is completeness in the atomic case. The second claim is that $\xi^\flat$ and ξ are equivalent in $\mathscr{N}$—that is, $\xi^\flat \Vdash_{\mathscr{N}} \xi$ and $\xi \Vdash_{\mathscr{N}} \xi^\flat$. Consequently,

$$\Gamma^\flat \Vdash_{\mathscr{N}'} \varphi^\flat \qquad \text{iff} \qquad \Gamma \Vdash_{\mathscr{N}'} \varphi$$

The third claim is the simulation statement which allows us to make the final move from derivability in $\mathscr{N}$ to derivability in NJ.

Proof (Theorem 1—Completeness). Assume $\Gamma \Vdash \varphi$ and let $\mathscr{N}$ be its bespoke base. By (IPL-Flat), $\Gamma^\flat \Vdash_{\mathscr{N}} \varphi^\flat$. Hence, by (IPL-AtComp), $\Gamma^\flat \vdash_{\mathscr{N}} \varphi^\flat$. Whence, by (IPL-Nat), $(\Gamma^\flat)^\natural \vdash (\varphi^\flat)^\natural$, i.e. $\Gamma \vdash \varphi$, as required. □

2.3 Base-Extension Semantics for IPL, Revisited

Goldfarb [13, 23] has also given a (complete) proof-theoretic semantics for IPL, but it mimics Kripke's [17] semantics. What is interesting about the B-eS in Sandqvist [29] is the way in which it is *not* a representation of the possible world semantics. This is most clearly seen in ($\vee$), which takes the form of the 'second-order' definition of disjunction—that is,

$$U + V = \forall X\,((U \to X) \to (U \to X) \to X)$$

—see Girard [12] and Negri [41]. This adumbrates the categorical perspective on B-eS given by Pym et al. [26]. Proof-theoretically, the clause recalls the elimination rule for the connective restricted to atomic conclusions,

$$\frac{\varphi \vee \psi \qquad \begin{matrix}[\varphi]\\ \mathrm{p}\end{matrix} \qquad \begin{matrix}[\psi]\\ \mathrm{p}\end{matrix}}{\mathrm{p}}$$

Dummett [9] has shown that such restriction in NJ is without loss of expressive power. Indeed, *all* of the clauses in Fig. 1 may be regarded as taking the form of the corresponding elimination rules.

The principle of *definitional reflection*, as described in Sect. 1 justifies this phenomenon. According to this principle, an alternative candidate clause for conjunction is as follows:

($\wedge^*$) $\Vdash^*_{\mathscr{B}} \varphi \wedge \psi$ iff for any $\mathscr{C} \supseteq \mathscr{B}$ and any $\mathrm{p} \in \mathbb{A}$, if $\varphi, \psi \Vdash^*_{\mathscr{C}} \mathrm{p}$, then $\Vdash^*_{\mathscr{C}} \mathrm{p}$

Definition 3. *The relation $\Vdash^*_{\mathscr{B}}$ is defined by the clauses of Fig. 1 with ($\wedge^*$) in place of ($\wedge$). The judgement $\Gamma \Vdash^* \varphi$ obtains iff $\Gamma \Vdash^*_{\mathscr{B}} \varphi$ for any $\mathscr{B}$.*

The resulting semantics is sound and complete for IPL:

Theorem 2. $\Gamma \Vdash^* \varphi$ *iff* $\Gamma \vdash \varphi$.

Proof. We assume the following: for arbitrary base $\mathscr{B}$, and formulas φ, ψ, χ,

(IPL*-Monotone) If $\Vdash^*_{\mathscr{B}} \varphi$, then $\Vdash^*_{\mathscr{C}} \varphi$ for any $\mathscr{C} \supseteq \mathscr{B}$.
(IPL*-AndCut) If $\Vdash^*_{\mathscr{B}} \varphi \wedge \psi$ and $\varphi, \psi \Vdash^*_{\mathscr{B}} \chi$, then $\Vdash^*_{\mathscr{B}} \chi$.

The first claim follows easily from (Inf). The second is a generalization of ($\wedge^*$); it follows by induction on the structure of χ—an analogous treatment of disjunction was given by Sandqvist [29].

By Theorem 1, it suffices to show that $\Gamma \Vdash^* \varphi$ iff $\Gamma \Vdash \varphi$. For this it suffices to show $\Vdash^*_{\mathscr{B}} \theta$ iff $\Vdash_{\mathscr{B}} \theta$ for arbitrary $\mathscr{B}$ and θ. We proceed by induction on the structure of θ. Since the two relations are defined identically except in the case when the θ is a conjunction, we restrict attention to this case.

First, we show $\Vdash_{\mathscr{B}} \theta_1 \wedge \theta_2$ implies $\Vdash^*_{\mathscr{B}} \theta_1 \wedge \theta_2$. By ($\wedge^*$), the conclusion is equivalent to the following: for any $\mathscr{C} \supseteq \mathscr{B}$ and $\mathrm{p} \in \mathbb{A}$, if $\theta_1, \theta_2 \Vdash^*_{\mathscr{C}} \mathrm{p}$, then $\Vdash^*_{\mathscr{C}} \mathrm{p}$.

Therefore, fix $\mathscr{C} \supseteq \mathscr{B}$ and $\mathrm{p} \in \mathbb{A}$ such that $\theta_1, \theta_2 \Vdash^*_{\mathscr{C}} \mathrm{p}$. By (Inf), this entails the following: if $\Vdash^*_{\mathscr{C}} \theta_1$ and $\Vdash^*_{\mathscr{C}} \theta_2$, then $\Vdash^*_{\mathscr{C}} \mathrm{p}$. By ($\wedge$) on the assumption (i.e., $\Vdash_{\mathscr{B}} \theta_1 \wedge \theta_2$), we obtain $\Vdash_{\mathscr{B}} \theta_1$ and $\Vdash_{\mathscr{B}} \theta_2$. Hence, by the induction hypothesis (IH), $\Vdash^*_{\mathscr{B}} \theta_1$ and $\Vdash^*_{\mathscr{B}} \theta_2$. Whence, by (IPL*-Monotone), $\Vdash^*_{\mathscr{C}} \theta_1$ and $\Vdash^*_{\mathscr{C}} \theta_2$. Therefore, $\Vdash^*_{\mathscr{C}} \mathrm{p}$. We have thus shown $\Vdash^*_{\mathscr{B}} \theta_1 \wedge \theta_2$, as required.

Second, we show $\Vdash^*_{\mathscr{B}} \theta_1 \wedge \theta_2$ implies $\Vdash_{\mathscr{B}} \theta_1 \wedge \theta_2$. It is easy to see that $\theta_1, \theta_2 \Vdash^*_{\mathscr{B}} \theta_i$ obtains for $i = 1, 2$. Applying (IPL*-AndCut) (setting $\varphi = \theta_1$, $\psi = \theta_2$) once with $\chi = \theta_1$ and once with $\chi = \theta_2$ yields $\Vdash^*_{\mathscr{B}} \theta_1$ and $\Vdash^*_{\mathscr{B}} \theta_2$. By the IH, $\Vdash_{\mathscr{B}} \theta_1$ and $\Vdash_{\mathscr{B}} \theta_2$. Hence, $\Vdash_{\mathscr{B}} \theta_1 \wedge \theta_2$, as required. □

A curious feature of the new semantics is that the meaning of the context-former (i.e., the comma) is not interpreted as $\wedge$; that is, defining the context-former as

$$\Vdash^*_{\mathscr{B}} \Gamma, \Delta \qquad \text{iff} \qquad \Vdash^*_{\mathscr{B}} \Gamma \text{ and } \Vdash^*_{\mathscr{B}} \Delta$$

we may express (Inf)

$$\Gamma \Vdash^*_{\mathscr{B}} \varphi \qquad \text{iff} \qquad \text{for any } \mathscr{C} \supseteq \mathscr{B}, \text{ if } \Vdash^*_{\mathscr{C}} \Gamma, \text{ then } \Vdash^*_{\mathscr{C}} \varphi$$

The clause for contexts is not the same as the clause for $\wedge$ in the new semantics. Nonetheless, as shown in the proof of Theorem 2, they are equivalent at every base—that is, $\Vdash^*_{\mathscr{B}} \varphi, \psi$ iff $\Vdash^*_{\mathscr{B}} \varphi \wedge \psi$ for any $\mathscr{B}$.

This equivalence of the two semantics yields the following:

Corollary 1. *For arbitrary base $\mathscr{B}$ and formula φ, $\Vdash_{\mathscr{B}} \varphi$ iff, for any $\mathscr{X} \supseteq \mathscr{B}$ and every atom* p*, if $\varphi \Vdash_{\mathscr{X}} \mathrm{p}$, then $\Vdash_{\mathscr{X}} \mathrm{p}$.*

The significance of this result is that we see that formulas in the B-eS are precisely characterized by their support of atoms.

3 Intuitionistic Multiplicative Linear Logic

Having reviewed the B-eS for IPL, we turn now to *intuitionistic multiplicative linear logic* (IMLL). We first define the logic and then consider the challenges of giving a B-eS for it. This motivates the technical work in Sect. 4. Henceforth, we abandon the notation of the previous section as we do not need it and may recycle symbols and conventions.

Fix a countably infinite set $\mathbb{A}$ of atoms.

Definition 4 (Formula). *The set of formulas* ($\mathsf{Form}_{\mathrm{IMLL}}$) *is defined by the following grammar:*

$$\varphi, \psi ::= \mathrm{p} \in \mathbb{A} \mid \varphi \otimes \psi \mid I \mid \varphi \multimap \psi$$

We use $\mathrm{p}, \mathrm{q}, \ldots$ for atoms and $\varphi, \psi, \chi, \ldots$ for formulas. In contrast to the work on IPL, collections of formulas in IMLL are more typically *multisets*. We use $\mathrm{P}, \mathrm{Q}, \ldots$ for *finite multisets* of atoms, and $\Gamma, \Delta, \ldots$ to denote *finite multisets* of formulas.

$$\frac{}{\varphi \triangleright \varphi}\,\mathsf{ax} \qquad \frac{\Gamma \fatsemi \varphi \triangleright \psi}{\Gamma \triangleright \varphi \multimap \psi}\,\multimap_{\mathsf{I}} \qquad \frac{\Gamma \triangleright \varphi \multimap \psi \quad \Delta \triangleright \varphi}{\Gamma \fatsemi \Delta \triangleright \psi}\,\multimap_{\mathsf{E}} \qquad \frac{}{\varnothing \triangleright I}\,I_{\mathsf{I}}$$

$$\frac{\Gamma \triangleright \varphi \quad \Delta \triangleright I}{\Gamma \fatsemi \Delta \triangleright \varphi}\,I_{\mathsf{E}} \qquad \frac{\Gamma \triangleright \varphi \quad \Delta \triangleright \psi}{\Gamma \fatsemi \Delta \triangleright \varphi \otimes \psi}\,\otimes_{\mathsf{I}} \qquad \frac{\Gamma \triangleright \varphi \otimes \psi \quad \Delta \fatsemi \varphi \fatsemi \psi \triangleright \chi}{\Gamma \fatsemi \Delta \triangleright \chi}\,\otimes_{\mathsf{E}}$$

Fig. 3. The Sequential Natural Deduction System NIMLL for IMLL

We use $[\cdot]$ to specify a multiset; for example, $[\varphi, \varphi, \psi]$ denotes the multiset consisting of two occurrence of φ and one occurrences of ψ. The empty multiset (i.e., the multiset with no members) is denoted $\varnothing$. The union of two multisets Γ and Δ is denoted $\Gamma \fatsemi \Delta$. We may identify a multiset containing one element with the element itself; thus, we may write $\psi \fatsemi \Delta$ instead of $[\psi] \fatsemi \Delta$ to denote the union of multiset Δ and the singleton multiset $[\psi]$. Thus, when no confusion arises, we may write $\varphi_1 \fatsemi \ldots \fatsemi \varphi_n$ to denote $[\varphi_1, ..., \varphi_n]$.

Definition 5 (Sequent). *A sequent is a pair $\Gamma \triangleright \varphi$ in which Γ is a multiset of formulas and φ is a formula.*

We characterize IMLL by proof in a natural deduction system. Since it is a substructural logic, we write the system in the format of a sequent calculus as this represents the context management explicitly. We assume general familiarity with sequent calculi—see, for example, Troelstra and Schwichtenberg [41].

Definition 6 (System NIMLL). *The sequential natural deduction system for* IMLL*, denoted* NIMLL*, is given by the rules in Fig. 3.*

A sequent $\Gamma \triangleright \varphi$ is a *consequence* of IMLL—denoted $\Gamma \vdash \varphi$—iff there is a NIMLL-proof of it.

One may regard IMLL as IPL without the structural rules of weakening and contraction—see Došen [8]. In other words, adding the following rules to NIMLL recovers a sequent calculus for IPL:

$$\frac{\Gamma \triangleright \varphi}{\Delta \fatsemi \Gamma \triangleright \varphi}\,\mathsf{w} \qquad \frac{\Delta \fatsemi \Delta \fatsemi \Gamma \triangleright \varphi}{\Delta \fatsemi \Gamma \triangleright \varphi}\,\mathsf{c}$$

To stay close to the work in Sect. 2, it is instructive to consider the natural deduction presentation, too. The rule figures may be the same, but their application is not; for example,

$$\frac{\varphi \quad \psi}{\varphi \otimes \psi} \quad \text{means} \quad \text{if } \Gamma \vdash \varphi \text{ and } \Delta \vdash \psi\text{, then } \Gamma \fatsemi \Delta \vdash \varphi \otimes \psi$$

(i.e., *not* 'if $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$, then $\Gamma \vdash \varphi \otimes \psi$')

Here, it is important that the context are multisets, not as sets.

The strict context management in IMLL yields the celebrated 'resource interpretations' of Linear Logic—see Girard [11]. The leading example of which is, perhaps, the number-of-uses reading in which a proof of a formula $\varphi \multimap \psi$ determines a function that *uses* its arguments exactly once. This reading is, however, entirely proof-theoretic and is not expressed in the truth-functional semantics of IMLL—see Girard [11], Allwein and Dunn [1], and Coumans et al. [6]. Though these semantics do have sense of 'resource' it is not via the number-of-uses reading, but instead denotational in the sense of the treatment of resources in the truth-functional semantics of the logic of Bunched Implications [19]. The number-of-uses reading is, however, reflected in the categorical semantics—see Seely [35] and Biermann [3,4].

How do we render support sensitive to the resource reading? The subtlety is that for $\Gamma \Vdash \varphi$ (where $\Gamma \neq \varnothing$), we must somehow transmit the resources captured by Γ to φ. From Corollary 1, we see that in B-eS the content of a formula is captured by the atoms it supports. Therefore, we enrich the support relation with an multiset of atoms P,

$$\Gamma \Vdash^{P}_{\mathscr{B}} \varphi \quad \text{iff} \quad \text{for any } \mathscr{X} \supseteq \mathscr{B} \text{ and any U, if } \Vdash^{\mathrm{U}}_{\mathscr{X}} \Gamma, \text{ then } \Vdash^{\mathrm{P}\,\mathrm{,}\,\mathrm{U}}_{\mathscr{X}} \varphi$$

where

$$\Vdash^{U}_{\mathscr{B}} \Gamma_1 \mathrm{,} \Gamma_2 \quad \text{iff} \quad \text{there are } U_1 \text{ and } U_2 \text{ such that } U = (U_1 \mathrm{,} U_2), \Vdash^{\mathrm{U}_1}_{\mathscr{X}} \Gamma_1, \text{ and } \Vdash^{\mathrm{U}_2}_{\mathscr{X}} \Gamma_2$$

This completes the background on IMLL.

4 Base-extension Semantics for IMLL

In this section, we give a B-eS for IMLL. It is structured as follows: first, we define support in a base in Sect. 4.1; second, we prove soundness in Sect. 4.2; finally, we prove completeness in Sect. 4.3.

4.1 Support in a Base

The definition of the B-eS proceeds in line with that for IPL (Sect. 2) while taking substructurality into consideration.

Definition 7 (Atomic Sequent). *An* atomic sequent *is a pair* $P \triangleright \mathrm{p}$ *in which* P *is a multiset of atoms and* q *is an atom.*

Definition 8 (Atomic Rule). *An* atomic rule *is a pair* $\mathcal{P} \Rightarrow \mathrm{p}$ *in which* $\mathcal{P}$ *is a (possibly empty) finite set of atomic sequents and* p *in an atom.*

Definition 9 (Base). *A* base $\mathscr{B}$ *is a (possibly infinite) set of atomic rules.*

Definition 10 (Derivability in a Base). *The relation* $\vdash_{\mathscr{B}}$ *of* derivability in $\mathscr{B}$ *is the least relation satisfying the following:*

(Ref) $\mathrm{p} \vdash_{\mathscr{B}} \mathrm{p}$

(At)	$\Vdash^{\mathrm{P}}_{\mathscr{B}} \mathrm{p}$	iff	$\mathrm{P} \vdash_{\mathscr{B}} \mathrm{p}$
($\otimes$)	$\Vdash^{\mathrm{P}}_{\mathscr{B}} \varphi \otimes \psi$	iff	for any $\mathscr{X} \supseteq \mathscr{B}$, multiset of atoms U, and atom p, if $\varphi , \psi \Vdash^{\mathrm{U}}_{\mathscr{X}} \mathrm{p}$, then $\Vdash^{\mathrm{P},\mathrm{U}}_{\mathscr{X}} \mathrm{p}$
(I)	$\Vdash^{\mathrm{P}}_{\mathscr{B}} I$	iff	for any $\mathscr{X} \supseteq \mathscr{B}$, multiset of atoms U, and atom p, if $\Vdash^{\mathrm{U}}_{\mathscr{X}} \mathrm{p}$, then $\Vdash^{\mathrm{P},\mathrm{U}}_{\mathscr{X}} \mathrm{p}$
($\multimap$)	$\Vdash^{\mathrm{P}}_{\mathscr{B}} \varphi \multimap \psi$	iff	$\varphi \Vdash^{\mathrm{P}}_{\mathscr{B}} \psi$
(,)	$\Vdash^{P}_{\mathscr{B}} \Gamma , \Delta$	iff	there are U and V such that $P = (U , V)$, $\Vdash^{\mathrm{U}}_{\mathscr{B}} \Gamma$, and $\Vdash^{V}_{\mathscr{B}} \Delta$
(Inf)	$\Gamma \Vdash^{P}_{\mathscr{B}} \varphi$	iff	for any $\mathscr{X} \supseteq \mathscr{B}$ and any U, if $\Vdash^{\mathrm{U}}_{\mathscr{X}} \Gamma$, then $\Vdash^{\mathrm{P},\mathrm{U}}_{\mathscr{X}} \varphi$

Fig. 4. Base-extension Semantics for IMLL

(App) *If* $\mathrm{S}_i , \mathrm{P}_i \vdash_{\mathscr{B}} \mathrm{p}_i$ *for* $i = 1, \ldots, n$ *and* $(\mathrm{P}_1 \triangleright \mathrm{p}_1, \ldots, \mathrm{P}_n \triangleright \mathrm{p}_n) \Rightarrow \mathrm{p} \in \mathscr{B}$*, then* $\mathrm{S}_1 , \ldots , \mathrm{S}_n \vdash_{\mathscr{B}} \mathrm{p}$.

Note the differences between Definition 1 and Definition 10: first, in (Ref), no redundant atoms are allowed to appear, while in (Ref-IPL) they may; second, in (App), the multisets S_1,...,S_n are collected together as a multiset, while in (App-IPL), there is one set. These differences reflect the fact in the multiplicative setting that 'resources' can neither be discharged nor shared.

Definition 11 (Support). *That a sequent* $\Gamma \triangleright \varphi$ *is* supported in the base $\mathscr{B}$ using resources S*—denoted* $\Gamma \Vdash^{\mathrm{S}}_{\mathscr{B}} \varphi$*—is defined by the clauses of Fig. 4 in which* Γ *and* Δ *are non-empty finite multisets of formulas. The sequent* $\Gamma \triangleright \varphi$ *is* supported using resources S*—denoted* $\Gamma \Vdash^{\mathrm{S}} \varphi$*—iff* $\Gamma \Vdash^{\mathrm{S}}_{\mathscr{B}} \varphi$ *for any base* $\mathscr{B}$*. The sequent* $\Gamma \triangleright \varphi$ *is* valid*—denoted* $\Gamma \Vdash \varphi$*—iff* $\Gamma \triangleright \varphi$ *is supported using the empty multiset of resources (i.e.,* $\Gamma \Vdash^{\varnothing} \varphi$*).*

It is easy to see that Fig. 4 is an inductive definition on a structure of formulas that prioritizes conjunction ($\otimes$) over implication ($\multimap$)—an analogous treatment in IPL with disjunction ($\vee$) prioritized over implication ($\rightarrow$) has been given by Sandqvist [29]. As explained in Sect. 3, the purpose of the multisets of atoms S in the support relation $\Vdash^{\mathrm{S}}_{\mathscr{B}}$ is to express the susbtructurality of the logical constants. The naive ways of using multisets of formulas rather than multisets of atoms—for example, $\Gamma \Vdash^{\Delta}_{\mathscr{B}} \varphi$ iff $\Vdash^{\Gamma,\Delta}_{\mathscr{B}} \varphi$—results in impredicative definitions of support.

We read (Inf) as saying that $\Gamma \Vdash^{\mathrm{S}}_{\mathscr{B}} \varphi$ (for $\Gamma \neq \varnothing$) means, for any extension $\mathscr{X}$ of $\mathscr{B}$, if Γ is supported in $\mathscr{X}$ with some resources U (i.e. $\Vdash^{\mathrm{U}}_{\mathscr{X}} \Gamma$), then φ is also supported by combining the resources U with the resources S (i.e., $\Vdash^{\mathrm{S},\mathrm{U}}_{\mathscr{X}} \varphi$).

The following observation on the monotonicity of the semantics with regard to base extensions follows immediately by unfolding definitions:

Proposition 1. *If* $\Gamma \Vdash^{\mathrm{S}}_{\mathscr{B}} \varphi$ *and* $\mathscr{C} \supseteq \mathscr{B}$*, then* $\Gamma \Vdash^{\mathrm{S}}_{\mathscr{C}} \varphi$.

From this proposition we see the following: $\Gamma \Vdash^{\mathrm{S}} \varphi$ iff $\Gamma \Vdash^{\mathrm{S}}_{\varnothing} \varphi$, and $\Gamma \Vdash \varphi$ iff $\Gamma \Vdash^{\varnothing}_{\varnothing} \varphi$. As expected, we do not have monotonicity on resources—that is, $\Gamma \Vdash^{\mathrm{S}} \varphi$

does not, in general, imply $\Gamma \Vdash^{\mathrm{S} \fatcomma T} \varphi$ for arbitrary T. This exposes the different parts played by bases and the resources in the semantics: bases are the setting in which a formula is supported, resources are tokens used in that setting to establish the support.

A distinguishing aspect of support is the structure of (Inf). In one direction, it is merely cut, but in the other it says something stronger. The completeness argument will go through the atomic case (analogous to the treatment of IPL in Sect. 2.2), and the following proposition suggests that the setup is correct:

Proposition 2. *The following two propositions are equivalent for arbitrary base $\mathscr{B}$, multisets of atoms* P, S*, and atom* q*, where we assume* $\mathrm{P} = [\mathrm{p}_1, \dots, \mathrm{p}_n]$*:*

1. $\mathrm{P} \fatcomma \mathrm{S} \vdash_{\mathscr{B}} \mathrm{q}$.
2. *for any* $\mathscr{X} \supseteq \mathscr{B}$ *and multisets of atoms* $\mathrm{T}_1, \dots, \mathrm{T}_\mathrm{n}$*, if* $\mathrm{T}_i \vdash_{\mathscr{X}} \mathrm{p}_i$ *holds for all* $i = 1, \dots, n$*, then* $\mathrm{T}_1 \fatcomma \dots \fatcomma \mathrm{T}_n \fatcomma \mathrm{S} \vdash_{\mathscr{X}} \mathrm{q}$.

It remains to prove soundness and completeness.

4.2 Soundness

Theorem 3 (Soundness). *If* $\Gamma \vdash \varphi$*, then* $\Gamma \Vdash \varphi$*.*

The argument follows a typical strategy of showing that the semantics respects the rules of NIMLL—that is, for any $\Gamma, \Delta, \varphi, \psi$, and χ:

(Ax)	$\varphi \Vdash \varphi$
($\multimap$I)	If $\Gamma, \varphi \Vdash \psi$, then $\Gamma \Vdash \varphi \multimap \psi$
($\multimap$E)	If $\Gamma \Vdash \varphi \multimap \psi$ and $\Delta \Vdash \varphi$, then $\Gamma \fatcomma \Delta \Vdash \psi$
($\otimes$I)	If $\Gamma \Vdash \varphi$ and $\Delta \Vdash \psi$, then $\Gamma \fatcomma \Delta \Vdash \varphi \otimes \psi$
($\otimes$E)	If $\Gamma \Vdash \varphi \otimes \psi$ and $\Delta \fatcomma \varphi \fatcomma \psi \Vdash \chi$, then $\Gamma \fatcomma \Delta \Vdash \chi$
(II)	$\Vdash I$
(IE)	If $\Gamma \Vdash \chi$ and $\Delta \Vdash I$, then $\Gamma \fatcomma \Delta \Vdash \chi$

These follow quickly from the fact that the clauses of each connective in Fig. 4 takes the form of its elimination rules. The only subtle cases are ($\otimes$E) and (IE).

To show (IE), suppose $\Gamma \Vdash \chi$ and $\Delta \Vdash I$. We require to show $\Gamma \fatcomma \Delta \Vdash \chi$. By (Inf), we fix some base $\mathscr{B}$ and multisets of atoms P and Q such that $\Vdash^{\mathrm{P}}_{\mathscr{B}} \Gamma$ and $\Vdash^{\mathrm{Q}}_{\mathscr{B}} \Delta$. It remains to verify $\Vdash^{\mathrm{P} \fatcomma \mathrm{Q}}_{\mathscr{B}} \chi$. When χ is atomic, this follows immediately from $\Vdash^{\mathrm{P}}_{\mathscr{B}} \chi$ and $\Vdash^{\mathrm{Q}}_{\mathscr{B}} I$ by (I). To handle non-atomic χ, we require the following:

Lemma 1. *For arbitrary base $\mathscr{B}$, multisets of atoms* S, T*, and formula χ, if 1.* $\Vdash^{\mathrm{S}}_{\mathscr{B}} I$*, 2.* $\Vdash^{\mathrm{T}}_{\mathscr{B}} \chi$*, then 3.* $\Vdash^{\mathrm{S} \fatcomma \mathrm{T}}_{\mathscr{B}} \chi$*.*

This lemma follows by induction on the structure of χ, with the base case given by (I). One cannot use this general form to define I as it would result in an impredicative definition of support.

Similarly, we require the following to prove ($\otimes$E):

$$\begin{array}{ll}
\multimap_{\mathsf{I}}{}^{\flat} : (\sigma^{\flat} \triangleright \tau^{\flat}) \Rightarrow (\sigma \multimap \tau)^{\flat} & \multimap_{\mathsf{E}}{}^{\flat} : \left(\triangleright(\sigma \multimap \tau)^{\flat}, \triangleright\sigma^{\flat}\right) \Rightarrow \tau^{\flat} \\
\otimes_{\mathsf{I}}{}^{\flat} : \left(\triangleright\sigma^{\flat}, \triangleright\tau^{\flat}\right) \Rightarrow (\sigma \otimes \tau)^{\flat} & \otimes_{\mathsf{E}}{}^{\flat} : \left(\triangleright(\sigma \otimes \tau)^{\flat}, \sigma^{\flat} \fatsemi \tau^{\flat} \triangleright \mathrm{p}\right) \Rightarrow \mathrm{p} \\
I_{\mathsf{I}}{}^{\flat} : \Rightarrow I^{\flat} & I_{\mathsf{E}}{}^{\flat} : \left(\triangleright I^{\flat}, \triangleright\mathrm{p}\right) \Rightarrow \mathrm{p}
\end{array}$$

Fig. 5. Atomic System $\mathscr{M}$

Lemma 2. *For arbitrary base $\mathscr{B}$, multisets of atoms S, T, and formulas φ, ψ, χ, if 1. $\Vdash_{\mathscr{B}}^{\mathrm{S}} \varphi \otimes \psi$, 2. $\varphi \fatsemi \psi \Vdash_{\mathscr{B}}^{\mathrm{T}} \chi$, then 3. $\Vdash_{\mathscr{B}}^{\mathrm{S} \fatsemi \mathrm{T}} \chi$.*

With these results, we may prove soundness:

Proof (Theorem 3 —sketch). We demonstrate ($\otimes$I) and ($\otimes$E).

($\otimes$I). Assume $\Gamma \Vdash \varphi$ and $\Delta \Vdash \psi$. We require to show $\Gamma \fatsemi \Delta \Vdash \varphi \otimes \psi$. By (Inf), the conclusion is equivalent to the following: for any base $\mathscr{B}$, for any multisets of atoms T and S , if $\Vdash_{\mathscr{B}}^{T} \Gamma$ and $\Vdash_{\mathscr{B}}^{S} \Delta$, then $\Vdash_{\mathscr{B}}^{T \fatsemi S} \varphi \otimes \psi$. So we fix some $\mathscr{B}$ and T, S such that $\Vdash_{\mathscr{B}}^{T} \Gamma$ and $\Vdash_{\mathscr{B}}^{S} \Delta$, and show that $\Vdash_{\mathscr{B}}^{\mathrm{T} \fatsemi \mathrm{S}} \varphi \otimes \psi$. By ($\otimes$), it suffices to show, for arbitrary $\mathscr{C} \supseteq \mathscr{B}$, multiset of atoms U, and atom p, if $\varphi \fatsemi \psi \Vdash_{\mathscr{C}}^{\mathrm{U}} \mathrm{p}$, then $\Vdash_{\mathscr{C}}^{\mathrm{T} \fatsemi \mathrm{S} \fatsemi \mathrm{U}} \mathrm{p}$. So we fix some $\mathscr{C} \supseteq \mathscr{B}$, multiset of atoms U, and atom p such that $\varphi \fatsemi \psi \Vdash_{\mathscr{C}}^{\mathrm{U}} \mathrm{p}$, and the goal is to show that $\Vdash_{\mathscr{C}}^{\mathrm{T} \fatsemi \mathrm{S} \fatsemi \mathrm{U}} \mathrm{p}$. From the assumptions $\Gamma \Vdash \varphi$ and $\Delta \Vdash \psi$, we see that $\Vdash_{\mathscr{B}}^{S \fatsemi \mathrm{T}} \varphi \fatsemi \psi$ obtains. Therefore, by monotonicity, $\Vdash_{\mathscr{C}}^{\mathrm{S} \fatsemi \mathrm{T}} \varphi \fatsemi \psi$ obtains. By (Inf), this suffices for $\varphi \fatsemi \psi \Vdash_{\mathscr{C}}^{\mathrm{U}} \mathrm{p}$, to yield $\Vdash_{\mathscr{C}}^{T \fatsemi S \fatsemi \mathrm{U}} \mathrm{p}$, as required.

($\otimes$E). Assume $\Gamma \Vdash \varphi \otimes \psi$ and $\Delta \fatsemi \varphi \fatsemi \psi \Vdash \chi$. We require to show $\Gamma \fatsemi \Delta \Vdash \chi$. By (Inf), it suffices to assume $\Vdash_{\mathscr{B}}^{\mathrm{S}} \Gamma$ and $\Vdash_{\mathscr{B}}^{\mathrm{T}} \Delta$ and show that $\Vdash_{\mathscr{B}}^{\mathrm{S} \fatsemi \mathrm{T}} \chi$. First, $\Gamma \Vdash \varphi \otimes \psi$ together with $\Vdash_{\mathscr{B}}^{\mathrm{S}} \Gamma$ entails that $\Vdash_{\mathscr{B}}^{\mathrm{S}} \varphi \otimes \psi$. Second, by (Inf), $\Delta \fatsemi \varphi \fatsemi \psi \Vdash \chi$ is equivalent to the following:

$$\text{for any } \mathscr{X} \text{ and } \mathrm{P}, \mathrm{Q}, \text{ if } \Vdash_{\mathscr{X}}^{\mathrm{P}} \Delta \text{ and } \Vdash_{\mathscr{X}}^{\mathrm{Q}} \varphi \fatsemi \psi, \text{ then } \Vdash_{\mathscr{X}}^{\mathrm{P} \fatsemi \mathrm{Q}} \chi$$

Since $\Vdash_{\mathscr{B}}^{\mathrm{T}} \Delta$, setting $\mathrm{P} := \mathrm{T}$ and $\mathrm{Q} := \mathrm{S}$, yields,

$$\text{for any } \mathscr{X} \supseteq \mathscr{B}, \text{ if } \Vdash_{\mathscr{X}}^{\mathrm{S}} \varphi \fatsemi \psi, \text{ then } \Vdash_{\mathscr{X}}^{\mathrm{T} \fatsemi \mathrm{S}} \chi \tag{1}$$

Now, given $\Vdash_{\mathscr{B}}^{\mathrm{S}} \varphi \otimes \psi$ and (1), we can apply Lemma 2 and conclude $\Vdash_{\mathscr{B}}^{\mathrm{S} \fatsemi \mathrm{T}} \chi$. □

4.3 Completeness

Theorem 4 (Completeness). *If $\Gamma \Vdash \varphi$, then $\Gamma \vdash \varphi$.*

The argument follows the strategy used by Sanqvist [29] for IPL—see Sect. 2.2. We explain the main steps.

Let Ξ be the set of all sub-formulas of $\Gamma \cup \{\varphi\}$. Let $(\cdot)^{\flat} : \Xi \to \mathbb{A}$ be an injection that is fixed on $\Xi \cap \mathbb{A}$—that is, $\mathrm{p}^{\flat} = \mathrm{p}$ for $\mathrm{p} \in \Xi \cap \mathbb{A}$. Let $(\cdot)^{\natural}$ be the left-inverse of $(\cdot)^{\flat}$—that is $\mathrm{p}^{\natural} = \chi$ if $\mathrm{p} = \chi^{\flat}$, and $\mathrm{p}^{\natural} = \mathrm{p}$ if p is not in the image of $(\cdot)^\flat$. Both act on multisets of formulas pointwise; that is, $\Delta^\flat := [\delta^\flat \mid \delta \in \Delta]$ and $\mathrm{P}^\natural := [\mathrm{p}^\natural \mid \mathrm{p} \in \mathrm{P}]$.

We construct a base $\mathscr{M}$ such that $\varphi^\flat$ behaves in $\mathscr{M}$ as φ behaves in NIMLL. The base $\mathscr{M}$ contains all instances of the rules of Fig. 5 when σ and τ range over Ξ, and p ranges over $\mathbb{A}$. We illustrate how $\mathscr{M}$ works with an example.

Example 2. Consider the sequent $\Gamma \rhd \varphi$ where $\Gamma = [\mathrm{p}_1 \fatsemi \mathrm{p}_2 \fatsemi \mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q}, \mathrm{p}_1]$ and $\varphi = \mathrm{q} \otimes \mathrm{p}_1$. By definition, $\Xi := \{\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q}, \mathrm{p}_1 \otimes \mathrm{p}_2, \mathrm{q}, \mathrm{q} \otimes \mathrm{p}_1\}$, and, therefore, the image of $(\cdot)^\flat$ is $\{\mathrm{p}_1, \mathrm{p}_2, \mathrm{q}, (\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q})^\flat, (\mathrm{p}_1 \otimes \mathrm{p}_2)^\flat, (\mathrm{q} \otimes \mathrm{p}_1)^\flat\}$.

That $\Gamma \vdash \varphi$ obtains is witnessed by the following NIMLL-proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{p}_1 \rhd \mathrm{p}_1}\,\mathsf{ax} \quad \dfrac{}{\mathrm{p}_2 \rhd \mathrm{p}_2}\,\mathsf{ax}}{\mathrm{p}_1 \fatsemi \mathrm{p}_2 \rhd \mathrm{p}_1 \otimes \mathrm{p}_2}\,\otimes_{\mathsf{I}} \quad \dfrac{}{\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q} \rhd \mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q}}\,\mathsf{ax}}{\mathrm{p}_1 \fatsemi \mathrm{p}_2 \fatsemi \mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q} \rhd \mathrm{q}}\,\multimap_{\mathsf{E}} \quad \dfrac{}{\mathrm{p}_1 \rhd \mathrm{p}_1}\,\mathsf{ax}}{\mathrm{p}_1 \fatsemi \mathrm{p}_2 \fatsemi \mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q} \fatsemi \mathrm{p}_1 \rhd \mathrm{q} \otimes \mathrm{p}_1}\,\otimes_{\mathsf{I}}$$

The base $\mathscr{M}$ is designed so that we may simulate the rules of NIMLL; for example, the $\otimes_{\mathsf{E}}$ is simulated by using (App) on $\otimes_{\mathsf{E}}^\flat$,

$$(\varnothing \rhd (\sigma \otimes \tau)^\flat, \sigma^\flat \fatsemi \tau^\flat \rhd \gamma^\flat) \Rightarrow \gamma^\flat \quad \text{means} \quad \text{if } \Delta^\flat \vdash_{\mathscr{M}} (\sigma \otimes \tau)^\flat \text{ and } \Sigma^\flat \fatsemi \sigma^\flat \fatsemi \tau^\flat \vdash_{\mathscr{M}} \gamma^\flat \text{ then } \Delta^\flat \fatsemi \Sigma^\flat \vdash_{\mathscr{M}} \gamma^\flat$$

In this sense, the proof above is simulated by the following steps:

(i) By (Ref), (1) $\mathrm{p}_1 \vdash_{\mathscr{M}} \mathrm{p}_1$; (2) $\mathrm{p}_2 \vdash_{\mathscr{M}} \mathrm{p}_2$; (3) $(\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q})^\flat \vdash_{\mathscr{M}} (\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q})^\flat$
(ii) By (App), using $(\otimes_{\mathsf{I}})$ on (1) and (2), we obtain (4) $\mathrm{p}_1 \fatsemi \mathrm{p}_2 \vdash_{\mathscr{M}} (\mathrm{p}_1 \otimes \mathrm{p}_2)^\flat$
(iii) By (App), using $(\multimap_{\mathsf{E}})^\flat$ on (3) and (4), we obtain (5) $(\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q})^\flat \fatsemi \mathrm{p}_1 \fatsemi \mathrm{p}_2 \vdash_{\mathscr{M}} \mathrm{q}$
(iv) By (App), using $(\otimes_{\mathsf{I}})^\flat$ on (1) and (5). we have $(\mathrm{p}_1 \otimes \mathrm{p}_2 \multimap \mathrm{q})^\flat \fatsemi \mathrm{p}_1 \fatsemi \mathrm{p}_2 \fatsemi \mathrm{p}_1 \vdash_{\mathscr{M}} (\mathrm{q} \otimes \mathrm{p}_1)^\flat$.

Significantly, steps (i)–(iv) are analogues of the steps in the proof tree above. ■

Theorem 4 (Completeness) follows from the following three observations, which are counterparts to (IPL-AtComp), (IPL-Flat), and (IPL-Nat) from Sect. 2.2:

(IMLL-AtComp) For any $\mathscr{B}$, P, S, and q, $\mathrm{P} \fatsemi \mathrm{S} \vdash_{\mathscr{B}} \mathrm{q}$ iff $\mathrm{P} \Vdash_{\mathscr{B}}^{\mathrm{S}} \mathrm{q}$.
(IMLL-Flat) For any $\xi \in \Xi$, $\mathscr{X} \supseteq \mathscr{M}$ and U, $\Vdash_{\mathscr{X}}^{\mathrm{U}} \xi^\flat$ iff $\Vdash_{\mathscr{X}}^{\mathrm{U}} \xi$.
(IMLL-Nat) For any P and q, if $\mathrm{P} \vdash_{\mathscr{M}} \mathrm{q}$ then $\mathrm{P}^\natural \vdash \mathrm{q}^\natural$.

(IMLL-AtComp) follows from Proposition 2 and is the base case of completeness. (IMLL-Flat) formalizes the idea that every formula ξ appearing in $\Gamma \rhd \varphi$ behaves the same as $\xi^\flat$ in any base extending $\mathscr{M}$. Consequently, $\Gamma^\flat \Vdash_{\mathscr{M}} \varphi^\flat$ iff $\Gamma \Vdash_{\mathscr{M}} \varphi$. (IMLL-Nat) intuitively says that $\mathscr{M}$ is a faithful atomic encoding of NIMLL, witnessed by $(\cdot)^\natural$. This together with (IMLL-Flat) guarantee that every $\xi \in \Xi$ behaves in $\mathscr{M}$ as $\xi^\flat$ in $\mathscr{M}$, thus as $\left(\xi^\flat\right)^\natural = \xi$ in NIMLL.

Proof. (Theorem 4 —Completeness). Assume $\Gamma \Vdash \varphi$ and let $\mathscr{M}$ be the bespoke base for $\Gamma \rhd \varphi$. By (IMLL-Flat), $\Gamma^\flat \Vdash^{\varnothing}_{\mathscr{M}} \varphi^\flat$. Therefore, by (IMLL-AtComp), we have $\Gamma^\flat \vdash_{\mathscr{M}} \varphi^\flat$. Finally, by (IMLL-Nat), $\left(\Gamma^\flat\right)^\natural \vdash \left(\varphi^\flat\right)^\natural$, namely $\Gamma \vdash \varphi$. $\square$

5 Conclusion

Proof-theoretic semantics (P-tS) is the paradigm of meaning in logic based on proof, as opposed to truth. A particular form of P-tS is *base-extension semantics* (B-eS) in which one defines the logical constants by means of a *support* relation indexed by a base—a system of natural deduction for atomic propositions—which grounds the meaning of atoms by proof in that base. This paper provides a sound and complete base-extension semantics for *intuitionistic multiplicative linear logic* (IMLL).

The B-eS for IPL given by Sandqvist [29] provides a strategy for the problem. The paper begins with a brief but instructive analysis of this work that reveals *definitional reflection* (DR) as an underlying principle delivering the semantics; accordingly, in Sect. 2.3, the paper modifies the B-eS for IPL to strictly adhere to DR and proves soundness and completeness of the result. Moreover, the analysis highlights that essential to B-eS is a transmission of proof-theoretic content: a formula φ is supported in a base $\mathscr{B}$ relative to a context Γ iff, for any extension $\mathscr{C}$ of $\mathscr{B}$, the formula φ is supported in $\mathscr{C}$ whenever Γ is supported in $\mathscr{C}$.

With this understanding of B-eS of IPL, the paper gives a ‘resource-sensitive’ adaptation by enriching the support relation to carry a multiset of atomic ‘resources’ that enable the transmission of proof-theoretic content. This captures the celebrated ‘resource reading’ of IMLL which is entirely proof-theoretic—see Girard [11]. The clauses of the logical constants are then delivered by DR on their introduction rules. Having set up the B-eS for IMLL in this principled way, soundness and completeness follow symmetrically to the preceding treatment of IPL.

To date, P-tS has largely been restricted to classical and intuitionistic propositional logics. This paper provides the first step toward a broader analysis. In particular, the analysis in this paper suggests a general methodology for delivering B-eS for other substructural logics such as, *inter alia*, (intuitionistic) Linear Logic [11] (LL) and the logic of Bunched Implications [19] (BI). While it is straightforward to add the additive connectives of LL, with the evident semantic clauses following IPL and with the evident additional cases in the proofs, it is less apparent how to handle the exponentials. For BI, the primary challenge is to appropriately account for the *bunched* structure of contexts, and to enable and confine weakening and contraction to the additive context-former.

Developing the P-tS for substructural logics is valuable because of their deployment in the verification and modelling of systems. Significantly, P-tS has shown the be useful in simulation modelling—see, for example, Kuorikoski and Reijula [16]. Of course, more generally, we may ask what conditions a logic must satisfy in order to provide a B-eS for it.

Acknowledgements. We are grateful to Yll Buzoku, Diana Costa, Sonia Marin, and Elaine Pimentel for many discussion on the P-tS for substructural logics, and to Jonte Deakin for his careful reading of and feedback on an earlier draft of this article. Similarly, we would like to thank the reviewers for their helpful comments and remarks.

References

1. Allwein, G., Dunn, J.M.: Kripke models for linear logic. J. Symbolic Logic **58**(2), 514–545 (1993)
2. Beth, E.W.: Semantic construction of intuitionistic logic. Indag. Math. **17**(4), 327–338 (1955)
3. Bierman, G.M.: What is a categorical model of intuitionistic linear logic? In: Dezani-Ciancaglini, M., Plotkin, G. (eds.) TLCA 1995. LNCS, vol. 902, pp. 78–93. Springer, Heilderberg (1995). https://doi.org/10.1007/bfb0014046
4. Bierman, G.M.: On Intuitionistic Linear Logic. Ph.D. thesis, University of Cambridge (1994). available as Computer Laboratory Technical report 346
5. Brandom, R.: Articulating Reasons: An Introduction to Inferentialism. Harvard University Press, Cambridge (2000)
6. Coumans, D., Gehrke, M., van Rooijen, L.: Relational semantics for full linear logic. J. Appl. Log. **12**(1), 50–66 (2014). https://doi.org/10.1016/j.jal.2013.07.005
7. van Dalen, D.: Logic and Structure, 5th edn. Universitext, Springer (2013)
8. Došen, K.: A Historical Introduction to Substructural Logics. In: Schroeder-Heister, P.J., Došen, K. (eds.) Substructural Logics. Oxford University Press (1993)
9. Dummett, M.: The Logical Basis of Metaphysics. Harvard University Press, Cambridge (1991)
10. Gheorghiu, A.V., Pym, D.J.: From Proof-theoretic Validity to Base-extension Semantics for Intuitionistic Propositional Logic. https://arxiv.org/abs/2210.05344. Accessed 08 Feb 2023
11. Girard, J.Y.: Linear Logic: its Syntax and Semantics. In: Girard, J.Y., Lafont, Y., Regnier, L. (eds.) Advances in Linear Logic, pp. 1–42. London Mathematical Society Lecture Note Series, Cambridge University Press (1995)
12. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and Types. Cambridge University Press, Cambridge (1989)
13. Goldfarb, W.: On Dummett's "proof-theoretic justifications of logical laws". In: Piecha, T., Schroeder-Heister, P. (eds.) Advances in Proof-Theoretic Semantics. TL, vol. 43, pp. 195–210. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-22686-6_13
14. Hallnäs, L.: Partial inductive definitions. Theoret. Comput. Sci. **87**(1), 115–142 (1991)
15. Hallnäs, L.: On the proof-theoretic foundation of general definition theory. Synthese **148**, 589–602 (2006)
16. Jaakko Kuorikoski, S.R.: Making It Count: An Inferentialist Account of Computer Simulation (2022) https://doi.org/10.31235/osf.io/v9bmr, https://osf.io/preprints/socarxiv/v9bmr. Accessed Jan 2023
17. Kripke, S.A.: Semantical analysis of intuitionistic logic I. In: Studies in Logic and the Foundations of Mathematics, vol. 40, pp. 92–130. Elsevier (1965)
18. Makinson, D.: On an inferential semantics for classical logic. Log. J. IGPL **22**(1), 147–154 (2014)
19. O'Hearn, P.W., Pym, D.J.: The logic of bunched implications. Bull. Symbolic Logic **5**(2), 215–244 (1999)

20. Piecha, T.: Completeness in proof-theoretic semantics. In: Piecha, T., Schroeder-Heister, P. (eds.) Advances in Proof-Theoretic Semantics. TL, vol. 43, pp. 231–251. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-22686-6_15
21. Piecha, T., de Campos Sanz, W., Schroeder-Heister, P.: Failure of completeness in proof-theoretic semantics. J. Philos. Log. **44**(3), 321–335 (2015)
22. Piecha, T., Schroeder-Heister, P.: The definitional view of atomic systems in proof-theoretic semantics. In: The Logica Yearbook 2016, pp. 185–200. College Publications London (2017)
23. Piecha, T., Schroeder-Heister, P.: Incompleteness of intuitionistic propositional logic with respect to proof-theoretic semantics. Stud. Logica. **107**(1), 233–246 (2019)
24. Prawitz, D.: Natural Deduction: A Proof-Theoretical Study. Dover Publications, New York (1965)
25. Prawitz, D.: Ideas and results in proof theory. In: Studies in Logic and the Foundations of Mathematics, vol. 63, pp. 235–307. Elsevier (1971)
26. Pym, D.J., Ritter, E., Robinson, E.: Proof-theoretic Semantics in Sheaves (Extended Abstract). In: Proceedings of the Eleventh Scandinavian Logic Symposium – SLSS 11 (2022)
27. Sandqvist, T.: An Inferentialist Interpretation of Classical Logic. Ph.D. thesis, Uppsala University (2005)
28. Sandqvist, T.: Classical logic without bivalence. Analysis **69**(2), 211–218 (2009)
29. Sandqvist, T.: Base-extension semantics for intuitionistic sentential logic. Logic J. IGPL **23**(5), 719–731 (2015)
30. Sandqvist, T.: Hypothesis-discharging rules in atomic bases. In: Wansing, H. (ed.) Dag Prawitz on Proofs and Meaning, vol. 7, pp. 313–328. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-11041-7_14
31. Schroeder-Heister, P.: Rules of definitional reflection. In: Logic in Computer Science – LICS, pp. 222–232. IEEE (1993)
32. Schroeder-Heister, P.: Validity concepts in proof-theoretic semantics. Synthese **148**(3), 525–571 (2006)
33. Schroeder-Heister, P.: Proof-theoretic versus model-theoretic consequence. In: Pelis, M. (ed.) The Logica Yearbook 2007. Filosofia (2008)
34. Piecha, T., Schroeder-Heister, P.: Atomic systems in proof-theoretic semantics: two approaches. In: Redmond, J., Pombo Martins, O., Nepomuceno Fernández, Á. (eds.) Epistemology, Knowledge and the Impact of Interaction. LEUS, vol. 38, pp. 47–62. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-26506-3_2
35. Seely, R.A.G.: Linear logic, ∗-autonomous categories and cofree coalgebras. In: Categories in Computer Science and Logic, vol. 92. American Mathematical Society (1989)
36. Stafford, W.: Proof-theoretic semantics and inquisitive logic. J. Philos. Logic **50**, 1199–1229 (2021)
37. Szabo, M.E. (ed.): The Collected Papers of Gerhard Gentzen. North-Holland Publishing Company, Amsterdam (1969)
38. Tarski, A.: O pojęciu wynikania logicznego. Przegląd Filozoficzny 39 (1936)
39. Tarski, A.: On the concept of following logically. Hist. Philos. Logic **23**(3), 155–196 (2002). https://doi.org/10.1080/0144534021000036683
40. Tennant, N.: Entailment and Proofs. Proc. Aristot. Soc. **79**, 167–189 (1978)
41. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press, Cambridge (2000)

The MaxSAT Problem in the Real-Valued MV-Algebra

Zuzana Haniková[1](✉), Felip Manyà[2], and Amanda Vidal[2]

[1] Institute of Computer Science of the Czech Academy of Sciences, Prague, Czech Republic
zuzana@cs.cas.cz

[2] Artificial Intelligence Research Institute (IIIA, CSIC), Bellaterra, Spain
{felip,amanda}@iiia.csic.es

Abstract. This work addresses the maximum satisfiability (MaxSAT) problem for a multiset of arbitrary formulas of the language of propositional Łukasiewicz logic over the MV-algebra whose universe is the real interval [0,1]. First, we reduce the MaxSAT problem to the SAT problem over the same algebra. This solution method sets a benchmark for other approaches, allowing a classification of the MaxSAT problem in terms of metric reductions introduced by Krentel. We later define an alternative analytic method with preprocessing in terms of a Tseitin transformation of the input, followed by a reduction to a system of linear constraints, in analogy to the earlier approaches of Hähnle and Olivetti. We discuss various aspects of these approaches to solving the problem.

Keywords: Maximum satisfiability · Satisfiability · Łukasiewicz logic · MV-algebra

1 Introduction

Satisfiability is a semantic problem: it relates not just to a logic (here, the infinite-valued Łukasiewicz logic), but to a semantics interpreting that logic (here, the MV-algebra on the real unit interval with natural order, called "standard MV-algebra" and denoted $[0,1]_{Ł}$).

A propositional formula $\varphi(x_1,\ldots,x_n)$ of the language of Łukasiewicz logic is *satisfiable* in an MV-algebra $\mathcal{A}$ provided there is an assignment of elements of the universe of $\mathcal{A}$ to $x_1,\ldots,x_n$ that yields the value $1^{\mathcal{A}}$ (i.e., the top element in the lattice order of $\mathcal{A}$). This definition determines, for a given MV-algebra $\mathcal{A}$, a unique set of its satisfiable formulas $\mathbf{SAT}(\mathcal{A})$. The satisfiability notion extends immediately to a *finite list* of formulas $\langle\varphi_1,\ldots,\varphi_m\rangle$, which is satisfiable in $\mathcal{A}$ if and only if so is the conjunction of the formulas on the list.[1]

[1] It is important to specify which MV-algebra is considered, since for many infinite MV-algebras $\mathcal{A}$, and even many subalgebras of $[0,1]_{Ł}$, the set $\mathbf{SAT}(\mathcal{A})$ is distinct from $\mathbf{SAT}([0,1]_{Ł})$ [16, Theorem 6.6]. Some extant works on satisfiability refer to "infinite-valued Łukasiewicz logic" while in fact working with the algebra $[0,1]_{Ł}$.

This paper works with the standard MV-algebra $[0,1]_{Ł}$ without mentioning it explicitly from now on; thus we write **SAT** for **SAT**$([0,1]_{Ł})$ and likewise for the MaxSAT problems considered in this paper. If another algebra, distinct from $[0,1]_{Ł}$, is considered, it will be indicated explicitly.

The focus of this paper is not on satisfiability, but on maximum satisfiability, an optimization problem (with a natural decision version): given a multiset (i.e., a list) of arbitrary formulas of the language of Łukasiewicz logic, find the *maximum number* among them that can be satisfied under a single assignment, over all assignments. The formulas are not required to be in a normal form. It has been recognized early on by Mundici [22] that formulas of Łukasiewicz logic are a suitable device for *counting*; his paper gives a reduction of the (decision version of) the Boolean MaxSAT problem to the problem **SAT**; see also [25].

The MaxSAT problem for a list of arbitrary formulas over the three-element MV-chain has been addressed in [19], using semantic tableaux; the approach generalizes to other finite MV-chains, but not to MV-chains with infinitely many elements. Earlier results in satisfiability go back to Mundici's proof of the **NP**-completeness of the **SAT** problem, obtained by bounding the denominators of a satisfying assignment. This line of research was continued in [1,2], see also [27].

Our main contribution consists in showing that the MaxSAT problem can be reduced to the **SAT** problem, in Sect. 3, and can then be used as a benchmark to assess the analytic method in Sect. 4; a similar analysis could then be performed with any other calculi for the maximum satisfiability problem.

This paper is structured as follows. Section 2 defines the problem and introduces technical tools. Section 3 gives a method for solving the MaxSAT problem in $[0,1]_{Ł}$ based on a Cook reduction of MaxSAT to the **SAT** problem. Section 4 outlines an analytic method with preprocessing via a Tseitin transformation, using a variant of the approach of [12,24], where each branch of a tableau tree ends with solving a system of linear constraints. The method is proved sound and complete. Eliminating the branching of the tree can also be achieved, using established tools.

2 Problem Formulation and Preliminaries

The language of propositional Łukasiewicz logic Ł, denoted $\mathcal{L}(Ł)$, has two basic connectives: $\neg$ (negation, unary) and $\oplus$ (strong disjunction, binary). Other connectives are definable: 1 is $x \oplus \neg x$; 0 is $\neg 1$; $x \odot y$ is $\neg(\neg x \oplus \neg y)$ (strong conjunction); $x \to y$ is $\neg x \oplus y$; $x \leftrightarrow y$ is $(x \to y) \odot (y \to x)$; $x \vee y$ is $(x \to y) \to y$ (weak disjunction); and $x \wedge y$ is $\neg(\neg x \vee \neg y)$ (weak conjunction).

Well-formed formulas of $\mathcal{L}(Ł)$ are built up from an infinite set of propositional variables $\mathrm{Var} = \{x_i\}_{i \in \mathbb{N}}$ using the connectives of $\mathcal{L}(Ł)$. The basic language is a point of reference for complexity considerations; other connectives are used as shortcuts. If φ is a formula of $\mathcal{L}(Ł)$ in the basic language, $|\varphi|$ denotes the *number of occurrences of* propositional variables in φ. Given that $\neg\neg\alpha \leftrightarrow \alpha$ is a theorem of Ł for any formula $\alpha \in \mathcal{L}(Ł)$, we will assume double negation does not occur in formulas. With this convention in place, the number of occurrences of connectives in φ is bounded by $2|\varphi|$. Thus $|\varphi|$ is a good notion of *length* of φ. Moreover $||\varphi||$ denotes the number of *distinct* subformulas of φ.

MV-algebras can be introduced using Mundici's Γ-functor [10,20]: any MV-algebra is isomorphic to $\Gamma(\mathcal{G}, u)$ for a lattice-ordered Abelian group $\mathcal{G}$ with a strong unit u (in particular, define $x \oplus y = u \wedge (x + y)$ and $\neg x = u - x$ for $x, y \in G$; then $\Gamma(\mathcal{G}, u) = \langle [0, u], \oplus, \neg \rangle$ is an MV-algebra). The standard MV-algebra $[0,1]_{Ł}$ is $\Gamma(\mathbb{R}, 1)$, interpreting the basic connectives in $[0,1]$ as follows: for any assignment v, $v(\neg\varphi) = 1 - v(\varphi)$ and $v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi))$. Any assignment to variables of φ in language $\mathcal{L}(Ł)$ extends to all its subformulas in the interpretation provided by $[0,1]_{Ł}$; this also determines the notion of satisfiability in $[0,1]_{Ł}$ and the set of satisfiable formulas of $[0,1]_{Ł}$, denoted **SAT**.

The interpretations of $\oplus$, $\odot$, $\wedge$ and $\vee$ are commutative and associative, so one can write $x_1 \oplus \cdots \oplus x_n$ without worrying about order and parentheses. We write x^n for $\underbrace{x \odot \cdots \odot x}_{n \text{ occurrences}}$ and nx for $\underbrace{x \oplus \cdots \oplus x}_{n \text{ occurrences}}$. Also, $\vee$ and $\wedge$ distribute over each other and $\odot$ distributes over $\vee$.

Unlike the Boolean MaxSAT problem over the two-element Boolean algebra, here we work with *arbitrary* formulas of $\mathcal{L}(Ł)$. We formulate both the optimization and the decision version of the MaxSAT problem.

MaxSAT-OPT
Instance: multiset $\langle \varphi_1, \ldots, \varphi_m \rangle$ of formulas of $\mathcal{L}(Ł)$ in variables $\{x_1, \ldots, x_n\}$.
Output: the maximum integer $k \leq m$ such that there is an assignment v to $\{x_1, \ldots, x_n\}$ that satisfies at least k formulas in the multiset $\langle \varphi_1, \ldots, \varphi_m \rangle$.

MaxSAT-DEC
Instance: multiset $\langle \varphi_1, \ldots, \varphi_m \rangle$ of formulas of $\mathcal{L}(Ł)$ in variables $\{x_1, \ldots, x_n\}$ and a positive integer $k \leq m$.
Output: (Boolean) Is **MaxSAT-OPT**$(\langle \varphi_1, \ldots, \varphi_m \rangle (x_1, \ldots, x_n))$ at least k?

Let $\mathbf{A}$ be an integer $m \times n$ matrix. Let $\mathbf{x}$ be an n-vector of variables and $\mathbf{b}$ be an integer m-vector. The **solvability of the system of inequalities $\mathbf{Ax} \leq \mathbf{b}$** in $\mathbb{R}$ can be tested in polynomial time [28].

More generally, for the system $\mathbf{Ax} \leq \mathbf{b}$, one can ask about the maximal size (number of lines) of a subsystem that is solvable in $\mathbb{R}$. This problem is known as the *maximum feasible subsystem* [4] of a system of linear constraints: the solution is a natural number k bounded by m (the total number of lines in the system). This problem is **NP**-hard. We shall refer to this problem as **Max-FS problem**. Notice that the system is not defined as a set, so the same constraint may appear multiple times.

There are many variants of the Max-FS problem, indeed many were already suggested in the paper [4]. We will use a variant that partitions the linear constraints into two groups: those that need to be satisfied by any feasible solution (often called *hard constraints*; the paper [4] refers to them as "mandatory") and those the satisfied number of which is to be maximized (often called *soft constraints*; [4] refers to them as "optional") over all feasible solutions. This variant of Max-FS problem will be called **Max-FS with hard and soft constraints** within this paper.

3 Canonical Method

First we give a polynomial-time, many-one (a.k.a. Karp) reduction of **MaxSAT-DEC** to **SAT**. Our reduction is similar to those used in [25] (which, in turn, refers to [22]) and in [15]. The differences arise from the fact that, in our case, an unsatisfied formula can take any value below 1 (but not necessarily 0), and this needs to be addressed in the definition of the set of formulas in the reduction.

Let $\langle \varphi_1, \dots, \varphi_m \rangle (x_1, \dots, x_n)$ and $k \leq m$ be an instance of **MaxSAT-DEC**. It is well known that one can implicitly define any rational value in $[0,1]_{Ł}$ with a formula of $\mathcal{L}(Ł)$: an early example of suitable formulas can be found in [30]. Let $k \geq 2$ and y be a new variable, not among $(x_1, \dots, x_n)$, and let

$$\rho_{1/k} \coloneqq \quad y \leftrightarrow \neg((k-1)y)$$

Then we have that $\rho_{1/k}$ implicitly defines the rational value $1/k$ in $[0,1]_{Ł}$ (see, e.g., [25, Lemma 2]): that is, an assignment v in $[0,1]_{Ł}$ sends $\rho_{1/k}$ to 1 if and only if it sends y to $1/k$. Moreover, the length of this formula is linear in $k \leq m$, therefore linear in the size of the instance on input.

For $1 \leq i \leq m$, consider a new variable $y_{i,k}$, let $\Phi_{\varphi_i,k}$ be the set of formulas

$$\{\ (\varphi_i \leftrightarrow k\, y_{i,k}) \vee \neg y_{i,k} \quad , \quad (y_{i,k} \leftrightarrow y) \vee \neg y_{i,k}\ \}$$

and let Φ_k be the list of formulas $\bigcup_{1 \leq i \leq m} \{\Phi_{\varphi_i,k}\}$.

Theorem 1. *The pair $\langle \varphi_1, \dots, \varphi_m \rangle (x_1, \dots, x_n)$ and k with $2 \leq k \leq m$ belongs to* **MaxSAT-DEC** *if and only if the set $\{\rho_{1/k}\} \cup \Phi_k \cup \{\bigoplus_{i=1}^m y_{i,k}\}$ belongs to* **SAT**.

Proof. For the left-to-right direction, assume v to be an assignment satisfying—without loss of generality—the first k formulas of the list. Consider then the assignment v' that coincides with v on the variables $x_1, \dots, x_n$ and puts $v'(y) = 1/k$ and

$$v'(y_{i,k}) = \begin{cases} 1/k & \text{if } i \leq k \\ 0 & \text{otherwise.} \end{cases}$$

The assignment v' clearly satisfies $\rho_{1/k}$. Next, since $v'(y_{1,k}) = \ldots = v'(y_{k,k}) = 1/k$, also $v'(\bigoplus_{i=1}^m y_{i,k}) = 1$. Lastly, the formulas in Φ_k are satisfied under v': the formulas $(y_{i,k} \leftrightarrow y) \vee \neg y_{i,k}$ are trivially satisfied, since each $y_{i,k}$ is indeed sent to either $1/k$ (and hence, $v'(y)$) or to 0. For the other formulas in Φ_k, first $v'(\varphi_j) = 1$ and $kv'(y_{j,k}) = k1/k = 1$ for each $1 \leq j \leq k$, and $v'(\neg y_{j,k}) = 1$ for $k < j \leq m$, hence they are all satisfied.

For the right-to-left direction, let v be an assignment satisfying $\{\rho_{1/k}\} \cup \Phi_k \cup \{\bigoplus_{i=1}^m y_{i,k}\}$. From Φ_k and $\rho_{1/k}$ we know $v(y_{i,k})$ is either $1/k$ or 0. Therefore, for $v(\bigoplus_{i=1}^m y_{i,k}) = 1$, necessarily at least k many y-variables are evaluated to $1/k$. Assume, again without loss of generality, that $v(y_{1,k}) = \ldots = v(y_{k,k}) = 1/k$. From Φ_k, we get that $v((\varphi_i \leftrightarrow k\, y_{i,k}) \vee \neg y_{i,k}) = 1$ for each $1 \leq i \leq m$. In particular, since $v(\neg y_{j,k}) \neq 1$ for every $1 \leq j \leq k$, necessarily $v((\varphi_j \leftrightarrow k\, y_{j,k}))$ for each such j. Together with the previously observed fact that $y_{j,k} = 1/k$ for each such j, this implies that $v(\varphi_1) = \ldots = v(\varphi_k) = 1$, concluding the proof.

For $k = 1$, it is immediate that $\langle \varphi_1, \ldots, \varphi_m \rangle$ and k is in **MaxSAT-DEC** if and only if $(\ldots(\varphi_1 \vee \varphi_2) \vee \ldots) \vee \varphi_m$ is in **SAT**. Given that for $m = k = 1$ both problems coincide, we get:

Corollary 1. *The problem* **MaxSAT-DEC** *is* **NP**-*complete.*

This reduction from **MaxSAT-DEC** to **SAT** provides a practical approach to the MaxSAT problem in $[0,1]_{\text{Ł}}$, provided that we use a competitive algorithm for solving **SAT** (i.e., the satisfiability problem in $[0,1]_{\text{Ł}}$). We could rely on either of the following two **SAT** solvers, which have been shown rather efficient. The first one is the tableau with constraints method proposed by Hähnle [12] that reduces **SAT** to Mixed Integer Programming (MIP) and can therefore use any available MIP solver. The second one is the Satisfiability Modulo Theory (SMT) methods proposed by Ansótegui et al. that reduces **SAT** to an SMT satisfiability problem and can use any available SMT solver [6,7,32]. These methods can take advantage of the latest developments and innovations in MIP and SMT solvers, avoiding the need to implement a **SAT** solver from scratch.

A polynomial-time Turing (a.k.a. Cook) reduction of **MaxSAT-OPT** to **MaxSAT-DEC** can be given, as we proceed to explain. It is this approach that prompts our referring to this method of solving **MaxSAT-OPT** as *canonical*, given its wide scope of applicability to optimization problems (see, e.g., [29]). The reduction uses an unspecified algorithm for **MaxSAT-DEC** as an *oracle*; as usual with oracle computations, any call to the oracle counts as one step in the computation and under this proviso, the oracle computation runs in time polynomial in the input size $(\Sigma_{i=1}^{m} |\varphi_i|)$. Indeed, given an instance $\langle \varphi_1, \ldots, \varphi_m \rangle$, it is easy to arrive at the optimal value for **MaxSAT-OPT** using binary search on the discrete, polynomial-size search space $\{1, \ldots, m\}$ of possible solutions, using at most $\lceil \log m \rceil$ oracle calls. Considering that **MaxSAT-DEC** is **NP**-complete by Corollary 1, we have the following:

Corollary 2. **MaxSAT-OPT** *is in* $\mathbf{FP}^{\mathbf{NP}}$.

For this conclusion, it is not important that the oracle solves **MaxSAT-DEC**; any oracle solving an **NP**-complete problem (an **NP**-oracle) would suit, and indeed one can use any algorithm for **SAT**, relying on Theorem 1. In view of the obvious reduction from **MaxSAT-DEC** to **MaxSAT-OPT**, the two problems are equivalent in the sense that if either has a polynomial-time algorithm, so does the other. This is standard, and it is why the decision version of an optimization problem is often considered *in lieu* of the problem as such.

Can one do better than $O(\log m)$ oracle calls? Below, we provide a classification of the problem in terms of Krentel's work [17] that suggests a negative answer subject to $\mathbf{P} \neq \mathbf{NP}$. Krentel ranks optimization problems in $\mathbf{FP}^{\mathbf{NP}}$ in terms of the number of calls to an **NP**-oracle. For $z : \mathbb{N} \longrightarrow \mathbb{N}$ a smooth function (i.e., z is non-decreasing and polynomial-time computable in unary representation), $\mathbf{FP}^{\mathbf{NP}}[z(n)]$ is the class of functions computable in polynomial time with an **NP** oracle with at most $z(|x|)$ oracle calls for instance x, where $|x|$ denotes the length of x. By definition, $\mathbf{FP}^{\mathbf{NP}}$ coincides with $\mathbf{FP}^{\mathbf{NP}}[n^{O(1)}]$ since a polynomial-time algorithm can make no more than a polynomial amount of oracle calls.

For Σ a finite alphabet let $f, g : \Sigma^* \longrightarrow \mathbb{N}$. A *metric reduction* [17] from f to g is a pair (h_1, h_2) of polynomial-time computable functions where $h_1 : \Sigma^* \longrightarrow \Sigma^*$ and $h_2 : \Sigma^* \times \mathbb{N} \longrightarrow \mathbb{N}$ such that $f(x) = h_2(x, g(h_1(x)))$ for all $x \in \Sigma^*$. The notion of a metric reduction is a natural generalization of polynomial-time many-one reduction to optimization problems. It follows from the definition that for each smooth function z as above, $\mathbf{FP}^{\mathbf{NP}}[z(n)]$ is closed under metric reductions.

Theorem 2. ([17], see also [29]) *Assume* $\mathbf{P} \neq \mathbf{NP}$.
Then $\mathbf{FP}^{\mathbf{NP}}[O(\log\log n)] \subsetneq \mathbf{FP}^{\mathbf{NP}}[O(\log n)] \subsetneq \mathbf{FP}^{\mathbf{NP}}[n^{O(1)}]$.

Recall that Boolean algebras form a subvariety of MV-algebras. In particular, in any Boolean algebra, the interpretations of the strong and the weak disjunction coincide, as do the interpretations of the strong conjunction and the weak conjunction. When mapping the Boolean connectives to the $\mathcal{L}(\text{Ł})$ connectives, we take $\neg$ for the Boolean negation, $\vee$ for the Boolean disjunction, and $\odot$ as the Boolean conjunction.

Moreover, in every nontrivial MV-algebra $\mathcal{A}$, the set consisting of its bottom element $0^{\mathcal{A}}$ and its top element $1^{\mathcal{A}}$ is closed under all operations of $\mathcal{A}$ and the subalgebra of $\mathcal{A}$ on the universe consisting of these two elements is isomorphic to the two-element Boolean algebra.

Now let us recall the MaxSAT problem in the two-element Boolean algebra for CNF formulas, given as multisets of clauses.

Classical-MaxSAT-OPT
Instance: multiset $\langle C_1, \ldots, C_m \rangle$ of Boolean clauses in variables $\{x_1, \ldots, x_n\}$.
Output: the maximum integer $k \leq m$ such that there is an assignment v in the two-element Boolean algebra on $\{0, 1\}$ to $\{x_1, \ldots, x_n\}$ that satisfies at least k clauses.

Krentel [17] proves the following result: **Classical-MaxSAT-OPT** is complete for $\mathbf{FP}^{\mathbf{NP}}[O(\log m)]$ under metric reductions.

We now prepare a few technical tools for eventually giving a metric reduction of **Classical-MaxSAT-OPT** to **MaxSAT-OPT**. Following [16, Def. 7.1], consider the language $\mathcal{L}(\text{Ł})$ including the definable connectives and define:

(i) a *literal* is a variable (such as x) or a negation thereof (such as $\neg x$).
(ii) A $(\odot, \vee)$-*formula* is built up from literals using arbitrary combination of $\odot$ and $\vee$.
(iii) In particular, a *clause* is built up from literals using only $\vee$.

Lemma 1. ([16, Thm. 7.4])

– *The interpretation of any* $(\odot, \vee)$*-formula with* n *variables in* $[0, 1]_{\text{Ł}}$ *is a convex function in* $[0, 1]^n$*;*
– *any* $(\odot, \vee)$*-formula (in particular, any clause) is satisfiable in* $[0, 1]_{\text{Ł}}$ *if and only if it is satisfiable in the two-element Boolean algebra* $\{0, 1\}$*.*

Lemma 2. *Let $C_1, \ldots, C_l$ be clauses in $\mathcal{L}(Ł)$ in variables $\{x_1, \ldots, x_n\}$. Assume $\bar{a} \in [0,1]^n$ is such that $C_i(\bar{a}) = 1$ for each $1 \leq i \leq l$. Then there is an element $\bar{b} \in \{0,1\}^n$ such that $C_i(\bar{b}) = 1$ for $1 \leq i \leq l$.*

Proof. We construct $\bar{b}$ from $\bar{a}$ in n independent steps. Let $\bar{b}_1 := \bar{a}$. The j-th step takes a $\bar{b}_j$, assuming the property that $C_i(\bar{b}_j) = 1$ for each $1 \leq i \leq l$, and produces $\bar{b}_{j+1}$ with the same property, replacing the real value in the j-th coordinate of $\bar{b}_j$ with a Boolean value (i.e., either a 0 or a 1). Lastly, we set $\bar{b} := \bar{b}_{n+1}$: all coordinates of $\bar{b}$ are Boolean.

We describe the j-th step. We simplify notation by writing $\bar{b}'$ for $\bar{b}_j$. We thus have $\bar{b}' = \langle b'_1, \ldots, b'_n \rangle$. Consider the j-th component of this vector: if b'_j is 0 or 1, we set $\bar{b}_{j+1} := \bar{b}_j$, whereby the step is finished. If $0 < b'_j < 1$, define $\bar{b}'_0 := \langle b'_1, \ldots, b'_{j-1}, 0, b'_{j+1}, \ldots, b'_n \rangle$ and $\bar{b}'_1 := \langle b'_1, \ldots, b'_{j-1}, 1, b'_{j+1}, \ldots, b'_n \rangle$. By assumption, we have $C_1(\bar{b}') = 1$. From Lemma 1, the interpretation of C_1 is a convex function. Now assume that either $C_1(\bar{b}'_0) \neq 1$ or $C_1(\bar{b}'_1) \neq 1$. Then there is a convex combination of $C_1(\bar{b}'_0)$ and $C_1(\bar{b}'_1)$ that is strictly below $C_1(\bar{b}')$, a contradiction with the convexity fact. We conclude that $C_1(\bar{b}'_0) = C_1(\bar{b}'_1) = 1$. An analogous argument holds for the remaining clauses $C_2, \ldots, C_l$. This means that we can set either $\bar{b}_{j+1} := \bar{b}'_0$ or $\bar{b}_{j+1} := \bar{b}'_1$ and we will indeed have $C_i(\bar{b}_{j+1}) = 1$ for each $1 \leq i \leq l$.

Theorem 3. **MaxSAT-OPT** *is complete for* $\mathbf{FP}^{\mathbf{NP}}[O(\log m)]$ *under metric reductions.*

Proof. Containment was obtained in Corollary 2 and the discussion preceding it. We prove hardness. We claim that the metric reduction of **Classical-MaxSAT-OPT** to **MaxSAT-OPT** is provided by a pair of *identity functions*. Take an arbitrary instance of **Classical-MaxSAT-OPT** problem, namely a multiset $\langle C_1, \ldots, C_m \rangle$ of Boolean clauses in variables $\{x_1, \ldots, x_n\}$, and interpret it as a multiset of clauses in $\mathcal{L}(Ł)$ (no change in notation is needed, see above). By Lemma 1, the interpretation of each C_i for $i = 1, \ldots, m$ in $[0,1]_Ł$ is a convex function. The convexity of the interpretation is not violated by rewriting each C_i in the basic connectives of $\mathcal{L}(Ł)$; this yields formulas $\langle C_1^*, \ldots, C_m^* \rangle$. Feed this m-tuple to the algorithm solving **MaxSAT-OPT**. The output is a natural number $k \leq m$ which indicates the maximal number among $\langle C_1^*, \ldots, C_m^* \rangle$ that are simultaneously satisfiable by an assignment in $[0,1]_Ł$. We assume without loss of generality that the first k formulas in the list are satisfied by some assignment; hence so are the first k among $\langle C_1, \ldots, C_m \rangle$. By Lemma 2, the same clauses (hence, the same number of clauses) are also simultaneously satisfiable by a *Boolean* assignment. This gives a lower bound on the number of simultaneously satisfiable clauses among $\langle C_1, \ldots, C_m \rangle$ in $\{0,1\}$. At the same time, the two-element Boolean algebra is a subalgebra of $[0,1]_Ł$, so any assignment in $\{0,1\}^n$ is also an assignment in $[0,1]^n$: therefore, considering that k was the answer of the algorithm solving **MaxSAT-OPT**, no more than k clauses among $\langle C_1, \ldots, C_m \rangle$ can be simultaneously satisfiable in $\{0,1\}$, because otherwise k would not be optimal for **MaxSAT-OPT**. Therefore k is the optimal value.

The binary search algorithm always makes a logarithmic number of oracle calls, no matter what the instance is. Also, the complexity analysis as given does not take into account the efficiency of the computations executed by the oracle; all that is known about the oracle is that it correctly decides a particular **NP**-complete problem. Considering the experience obtained in Boolean MaxSAT solvers based on Boolean SAT solvers, there might be alternatives to binary search that might turn out to be more efficient in practice, where one departs from the paradigm that emphasizes worst-case complexity. A typical Boolean MaxSAT solver does a *linear* search, either from unsatisfiable to satisfiable (core-guided approach), or from satisfiable to unsatisfiable (model-guided approach) [8,18]. The solvers heavily exploit the fact that the formulas in the multiset are Boolean clauses (i.e., a *normal form* is assumed) and that a SAT solver also returns a satisfying assignment or an unsatisfiable core; moreover, the calls to the SAT solver need not be its independent runs. These parallels invite an openness of mind when implementing MaxSAT solvers for Łukasiewicz logic.

4 Tableau-Like Method

4.1 Satisfiability

We give first a decision method for the **SAT** problem, combining several approaches that might be termed *analytic*. **SAT** and its complexity have been investigated in depth [1,2,6,7,9,12,14,16,21,23,26]. In particular, tableau calculi have been proposed in [12,24]. Presenting our decision method for **SAT** has several goals. It outlines our approach to a simpler problem than **MaxSAT-OPT**, to be modified in Subsect. 4.2. Our method for **SAT** can then be used as a lower bound on the complexity of the method for **MaxSAT-OPT** in Subsect. 4.2. Furthermore, the method, in its variant generating a tree with an exponential number of branches, provides a simple proof for **SAT** in **NP** and an upper bound on the runtime of a deterministic algorithm for **SAT**.

The method operates in two subsequent stages. The first one is a variant of Tseitin transformation of an arbitrary formula to a formula in *normal form* [31]; in classical logic, the target normal form is a CNF, while in our case, the target normal form is a system of equations in the language $\mathcal{L}(\text{Ł})$. The transformation preserves satisfiability, involves only a polynomial increase in size, and adds new variables. A variant of the transformation was used for testing **SAT** in [9].

Let $||\varphi||$ denote the number of pairwise distinct subformulas in φ.[2] Recall at this point the formula $\rho_{1/k}$ from Sect. 3 and its subformula $(k-1)y$. If brackets in this subformula nest to the right (or to the left), then $||(k-1)y||$ is proportional to $|(k-1)y|$. But if $(k-1)y$ is bracketed as a balanced binary tree, then $||(k-1)y||$ is proportional to $\log_2(|(k-1)y|)$.

[2] φ is viewed as a string, any subformula is a substring, and subformulas are the same if and only if the strings are. Thus $x \oplus (x \oplus x)$ is distinct from $(x \oplus x) \oplus x$. Per convention $\neg\neg\psi$ does not occur as subformula for any ψ, since $\neg\neg\psi \leftrightarrow \psi$ in Ł.

The second stage is a tableau-like procedure that utilizes the system of equations obtained in the first stage as labels for nodes in a rooted linear tree, and expands the nodes using simple rules that translate these equations of $\mathcal{L}(Ł)$ into linear equations in the reals. Subsequently, each branch is evaluated for solvability in the reals, analogously to [12,24].

The algorithm for **SAT** is given below. The presentation is informal.

Decision method TŁ_{SAT}. Let $\varphi(x_1, \ldots, x_n)$ be an input formula.

1. **List subformulas.** Let **L** be the list of all pairwise distinct subformulas occurring in φ, including φ and all its variables. Let l be the number of items in **L**. If φ does not contain any double negations, we have $l = ||\varphi||$. We assume that if α is a subformula of β, then α occurs before β in **L**.
2. **Name subformulas.** Introduce new pairwise distinct variables z_i for the i-th formula in **L** with $1 \leq i \leq l$. These will be called "z-variables". It is assumed that the z variables are also distinct from each x_j for $1 \leq j \leq n$.[3]
3. **Equations on names.** Let **S** be the list of equations in the language $\{\neg, \oplus\}$ obtained by initializing **S** as empty and taking the following step for each item in the list **L**:
   – if x is a propositional variable in φ and $1 \leq i \leq l$ and z_i is the variable for x, include in **S** the equation
   $$x = z_i;$$
   – if $\neg\alpha$ is a subformula of φ and $1 \leq i, j \leq l$ and z_i is the variable for α and z_j is the variable for $\neg\alpha$, include in **S** the equation
   $$z_j = \neg z_i;$$
   – if $\alpha \oplus \beta$ is a subformula of φ and $1 \leq i, j, k \leq l$ and z_i, z_j, z_k are the variables for α, β, $\alpha \oplus \beta$ respectively, include in **S** the equation
   $$z_i \oplus z_j = z_k.$$
   Having each item of **L** processed, **S** contains equations in the language $\mathcal{L}(Ł)$. The number of equations in **S** is l.
4. **Initialize tree.** Initialize a rooted tree **T**, linear at this stage, with l nodes. From the root down, label each node of **T** with one equations from **S**. Start with equations containing the x-variables and mark them *final*. Then process those containing $\neg$ and subsequently those containing $\oplus$ and mark each as *active*.[4]
5. **Boundary constraints.** Append before the root $2l$ new nodes labelled $0 \leq z_i \leq 1$ for each $i = 1, \ldots, l$. Mark each as *final*.
6. **Target constraint.** Append as new root of the tree a node labelled $z_l = 1$ for z_l the variable introduced for φ. (By convention taken in step 1, z_l is assigned to φ.) Mark *final*.

[3] This is a convention in favour of clarity of presentation. Avoiding introduction of new variables for atoms $x_1, \ldots, x_n$ would save n new variables.

[4] The structure of **T** will be linear up to a certain point and binary from there on. This is the case because a) the equations with the x-variables are not expanded, and b) all the equations with $\neg$ are expanded before any of the equations with $\oplus$, and the expansion rule for $\neg$ does not lead to branching. Cf. Example 1.

7. **Expand tree.** From the root of $\mathbf{T}$ towards the leaves, process each node N:
   - If the label of N is marked *final* (i.e, does not contain $\neg$ or $\oplus$), leave it intact and proceed to the next node.
   - If the label of N is marked *active* (contains $\neg$ or $\oplus$), mark it *passive*, and below each leaf of $\mathbf{T}$, append a new subtree with labelled nodes using the following **expansion rules** (one new node per each constraint), marking each new label *final*:

   $$\frac{z_i \oplus z_j = z_k}{\begin{array}{c|c} z_i + z_j \leq 1 & z_i + z_j \geq 1 \\ z_i + z_j = z_k & z_k = 1 \end{array}} \qquad \frac{z_i = \neg z_j}{z_i = 1 - z_j}$$

   An application of the rule on the left involves branching below each leaf of $\mathbf{T}$. The labels in the conclusions of these rules are linear constraints in real numbers. The mark *final* indicates the algorithm leaves them intact. Having processed all nodes of $\mathbf{T}$, each branch of $\mathbf{T}$ defines a system of linear constraints marked *final* in an unambiguous way.
8. **Solve systems.** From the leftmost branch to the right, test the system of constraints on the branch for solvability in $\mathbb{R}$[5] until a branch is found whose system of constraints is solvable. In such a case, return **'yes'** and exit.
9. **Default.** Return **'no'** and exit.

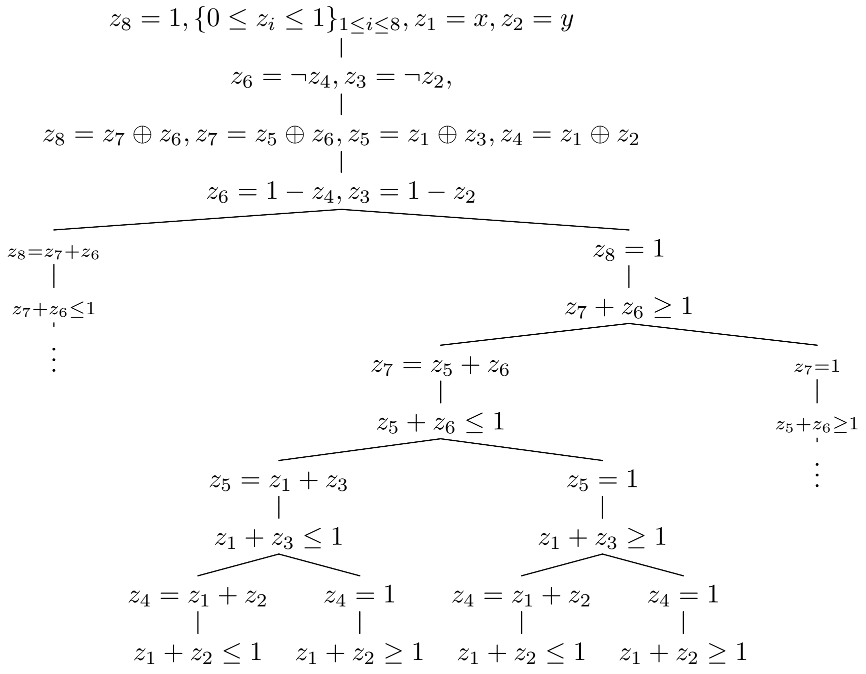
Typically in an analytic tableau method (cf. eg. Hähnle [12]), one starts with a given formula φ and decomposes it, taking one occurrence of a connective in each step and expanding the tableau using the given tableau rules. If a subformula of φ occurs multiple times in φ, it is processed multiple times and each time, new variables are introduced with it: cf. e.g. [12, section 5.1] where new variables i_1 and i_2 are introduced for each occurrence of an implication. This is a feature of the analytic method. With creating the set of subformulas first, we avoid this and have potentially less new variables. (Cf. also the introduction in [24], where our method might therefore not qualify as purely analytic.)

Example 1. A simple example will illustrate the generation of the tree and the resulting systems of constraints. Consider the formula $((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)$. A list of its subformulas is the following:

$$\langle x, y, \neg y, x \oplus y, x \oplus \neg y, \neg(x \oplus y), (x \oplus \neg y) \oplus (\neg(x \oplus y)), ((x \oplus \neg y) \oplus \neg(x \oplus y)) \oplus \neg(x \oplus y)\rangle$$

In order to present the example in a compact way, we write three initial nodes only: the first, with the boundary, target and ground equations; the second, with the equations from $\mathbf{S}$ with symbol $\neg$, and the third, with the equations from $\mathbf{S}$ with symbol $\oplus$. Below this, we expand the tree as described by the algorithm. We omit marks (active, passive, final). We use vertical dots to indicate the tree that would be included in their place is a copy of the one depicted at its side.

[5] The testing procedure is in $\mathbf{P}$. For the purpose of testing, one can render each equality $\mathbf{ax} = \mathbf{b}$ as two inequalities $\mathbf{ax} \leq \mathbf{b}$ and $-\mathbf{ax} \leq -\mathbf{b}$.



Lemma 3. *The expansion rules in step 7 of* $\mathrm{T}Ł_{\mathrm{SAT}}$ *preserve the following invariant: for any assignment v of values in $[0,1]$ to all z-variables, v satisfies the equation in the premise in the algebra $[0,1]_{Ł}$ if and only if v satisfies all constraints in at least one branch in the conclusions of the rule in the algebra $\mathbb{R}$.*

Proof. Notice that the expansion rules work as a switch between the signature of $\mathcal{L}(Ł)$ and language of real closed fields. (Where by slight abuse of language, we only differentiate between the two sets of the operation symbols, but not the relation symbols.) In both cases the statement is a straightforward consequence of the semantics of the connectives $\neg$ and $\oplus$ in $[0,1]_{Ł}$. We prove the case for $\oplus$. Top-to-bottom: let v be an assignment of values in $[0,1]$ to z-variables introduced in step 2, and consider z_i, z_j, z_k s.t. $v(z_i) \oplus v(z_j) = v(z_k)$ is true in $[0,1]_{Ł}$. Then it must be the case that either $v(z_i)+v(z_j) \le 1$ and $v(z_i)+v(z_j) = v(z_k)$ holds in $\mathbb{R}$, or $v(z_i)+v(z_j) \ge 1$ in which case we also have $v(z_k) = 1$ in $\mathbb{R}$. Bottom to top: again let v be an assignment of values in $[0,1]$ to z-variables. If $v(z_i)+v(z_j) \le 1$ and $v(z_i)+v(z_j) = v(z_k)$ both hold in $\mathbb{R}$, we have $v(z_i) \oplus v(z_j) = v(z_k)$ is true in $[0,1]_{Ł}$. If $v(z_i)+v(z_j) \ge 1$ and $v(z_k) = 1$ in $\mathbb{R}$, we have $v(z_i) \oplus v(z_j) = v(z_k)$ is true in $[0,1]_{Ł}$. This exhausts possible cases.

Theorem 4. *The method* $\mathrm{T}Ł_{\mathrm{SAT}}$ *is sound and complete for* **SAT**.

Proof. The **soundness** claim states that whenever the method answers 'yes' on input φ, then there is an assignment v to $x_1, \dots, x_n$ such that $v(\varphi) = 1$. So assume that there is a branch B of $\mathbf{T}$ such that the system of constraints given by B is solvable, under some assignment v to variables on B, and fix v. In particular, for $i = 1, \ldots, n$, the variable x_i gets value $v(x_i)$ (notice each x_i occurs on every branch). The assignment v extends to φ in a unique way and one shows by induction on the structure of φ, using Lemma 3, that for any subformula ψ of φ, we have $v(\psi) = v(z_j)$ for z_j with $j \in \{1, \ldots, l\}$ being the z-variable assigned to ψ in step 2. In particular, $v(\varphi) = 1$.

The **completeness** claim states that if $v(\varphi) = 1$ for some assignment v, then the method yields 'yes' on input φ. So fix v s.t. $v(\varphi) = 1$. We claim there is a branch of $\mathbf{T}$ with a solvable system of equations. First produce the full tree $\mathbf{T}$. Then assign values to all z-variables, starting from those that are names for $x_1, \ldots, x_n$, and then inductively on the structure of φ using again that $v(\psi) = v(z_j)$ for a z_j assigned to ψ in step 2. This is consistent with equations obtained in step 3. By abuse of language, call this assignment v. The assignment v makes it possible to travel downward from the root of $\mathbf{T}$ via labelled nodes, using Lemma 3 to show that v satisfies each label: in particular if $\mathbf{T}$ branches due to a node with label $z_i \oplus z_j = z_k$, then (assuming the label in the premise is satisfied by v), Lemma 3 guarantees that there is at least one branch on which the new (and hence, all) labels are satisfied by v. Finally a leaf L of $\mathbf{T}$ is reached: since Lemma 3 was applied at each expansion, and since the boundary and the final constraint clearly hold under v, all final constraints on the branch determined by L hold under v.

Lemma 4. *The problem* **SAT** *on instance φ can be solved deterministically by constructing the tree* $\mathbf{T}$ *and testing the solvability of systems of linear constraints in $\mathbb{R}$ on no more than $2^{||\varphi||}$ branches. Each branch has at most $4||\varphi|| + 1$ constraints and $||\varphi|| + n$ variables.*

Proof. Branching of the tree takes place at each occurrence of $\oplus$ in $\mathbf{S}$; the number of such occurrences is bounded by $||\varphi||$. Each branch has at most $2||\varphi||$ constraints for subformulas, plus $2||\varphi||$ boundary constraints, plus a target constraint. (Here we do not consider the possibility of replacing each equation with two inequalities.) Each branch of the tree uses all the variables: n input variables $x_1, \ldots, x_n$ and $||\varphi||$ z-variables.

Corollary 3. *The problem* **SAT** *is in* **NP***, in particular, a formula is satisfiable if and only if there is a polynomial-size witness consisting of a tableau branch of the method* $\mathrm{TŁ_{SAT}}$ *and matching system of constraints solvable in $\mathbb{R}$.*

Proof. Since the method $\mathrm{TŁ_{SAT}}$ is sound and complete for **SAT** by Theorem 4, any satisfiable formula has the following polynomial-size certificate of its own satisfiability in $[0, 1]_{Ł}$: the system of equations in z-variables constructed in step 3, and a branch of the tree $\mathbf{T}$, defined by a list of instructions specifying which branch to take upon each application of $\oplus$-rule, combined with a system C of constraints that matches the indicated branchings (in the sense that the equations with $\oplus$ have been expanded according to the specified branch) and such that C is solvable in $\mathbb{R}$. On the other hand, the soundness and completeness theorem also says that an unsatisfiable formula *cannot* have such a certificate.

Furthermore, any decision tree obtained from the above procedure can be linearized, using the methods of [12]. In particular, any instance of the application of the branching rule introduced in step 7 can be replaced by an instance of an application of the following lemma (observing the condition that distinct Boolean variables will be used for distinct instances):

Lemma 5. (Cf. [12, Sect. 5.1], [13, Lemma 6.2.19]) *Assume* $a_1, a_2, a_3 \in [0,1]$. *Then* $a_1 \oplus a_2 = a_3$ *holds in* $[0,1]_{Ł}$ *if and only if there is an* $y \in \{0,1\}$ *such that all of the following constraints hold in* $\mathbb{R}$*:*

(i) $a_1 + a_2 \leq 1 + y$
(ii) $y \leq a_1 + a_2$
(iii) $a_3 \leq a_1 + a_2$
(iv) $a_1 + a_2 \leq a_3 + y$
(v) $y \leq a_3$.

Proof. Assume $a_1 \oplus a_2 = a_3$ holds in $[0,1]_{Ł}$. Case 1: $a_1 + a_2 \leq 1$, then from the assumption we have $a_1 + a_2 = a_3$. We set $y := 0$. The fact that $a_1, a_2, a_3 \in [0,1]$ implies (ii) and (v); the remaining constraints in the Lemma follow from $a_1 + a_2 = a_3$. Case 2: $a_1 + a_2 > 1$. The assumption implies $a_3 = 1$; we set $y := 1$, we get (v). The fact that $a_1, a_2, a_3 \in [0,1]$ implies (i) and (iv). From $a_1 + a_2 > 1$ we get (ii) and (iii).
Now assume there is an $y \in \{0,1\}$ such that all constraints listed hold in $\mathbb{R}$. Case 1: $y = 0$. We have (i) $a_1 + a_2 \leq 1$ and (iii,iv) $a_3 \leq a_1 + a_2 \leq a_3$. Hence $a_1 \oplus a_2 = a_3$. Case 2: $y = 1$. We have (v) $1 \leq a_3$ and (ii) $1 \leq a_1 + a_2$. Hence $a_1 \oplus a_2 = a_3$.

This modification eventually yields, in step 8, a single MIP problem — one of the extant competitive ways to address the **SAT** problem. A major advantage of using a MIP solver is the advanced possibility of applying heuristics, whereas in the simple version above, the only optimization considered is aborting the computation upon finding a branch with a solvable system.[6] That is: by design, the algorithm $TŁ_{SAT}$ needs to generate and perhaps eventually test exponentially many systems of equations. However, from the viewpoint of the worst-case deterministic complexity, the MIP method does not differ substantially from testing the (possibly exponentially many) branches.

4.2 Maximum Satisfiability

In this Subsection we adapt the previous method to the **MaxSAT-OPT** problem from Sect. 2. It is easily observed that usual methods for **SAT**, the method from the previous Subsection among them (even if it easily adapts to test joint satisfiability of a list of formulas), are not applicable for **MaxSAT-OPT**; cf. [19] for a discussion. One problem is that they yield a Boolean value. Taking any satisfiable formula α and considering the m-element list $\langle \alpha, \ldots, \alpha \rangle$, for any $m > 1$,

[6] One might optimize by testing immediately on every generated branch and exiting the computation upon finding one with a solvable system. In our exposition though, we prefer to consider the size of the full decision tree.

clearly a complete method needs to produce the answer m on this input. The tableau approaches of [12, 24] uses MIP solvers on branches, also returning a Boolean value. Another feature of the method from the previous Subsection is that it considers distinct subformulas as a set; thus any repetition of the same formula in the list on input would be obliterated.

These considerations invite the approach of preserving the Tseitin-like procedure of listing equations obtained from the subformulas, but combining it with:

– updating the target constraint for a multiset of formulas on input, and
– updating the query about the system of constraints obtained on each branch.

The following algorithm updates the decision method $\mathrm{T}Ł_{\mathrm{SAT}}$ from the Subsect. 4.1. To highlight the differences, each step only gives the information that has changed compared to the previous case.

Optimization method $\mathrm{T}Ł_{\mathrm{MaxSAT}}$ for computing **MaxSAT-OPT**.
Let $\langle \varphi_1, \dots, \varphi_m \rangle$ be a list of formulas in variables $x_1, \dots, x_n$.

1. **List subformulas.** Let **L** be the list of all pairwise distinct subformulas occurring in $\varphi_1, \dots, \varphi_m$, including each formula $\varphi_1, \dots, \varphi_m$ with $1 \leq i \leq m$ and all variables $x_1, \dots, x_n$. Let l be the number of items in **L**. Conventions as in step 1 of $\mathrm{T}Ł_{\mathrm{SAT}}$.
2. **Name subformulas.** As before.
3. **Equations on names.** As before.
4. **Initialize tree.** As before.
5. **Boundary constraints.** As before.
6. **Mark hard constraints.** For each node in **T** up to this point, mark all constraints as **hard constraints**.
7. **Target constraints.** Append before the root of **T** a new chain with labels $z_{j_i} \geq 1$ for z_{j_i} the variable introduced for φ_i, with $i = 1, \dots, m$, preserving the multiplicity of φ_i in the input list. Mark these constraints as **soft constraints**.
8. **Expand tree.** As before, preserving in the expansion that a hard constraint produces hard constraints.
9. **Solve systems.** From the leftmost branch to the right, taking one branch at a time. Each branch defines, via the *final* label, a system of linear constraints in $\mathbb{R}$, with the target constraints from step 7 marked *soft* and all other constraints marked *hard*. Thus each branch defines an instance of the Max-FS problem with hard and soft constraints. Obtain the solution (i.e., a natural number, possibly 0) to the instance on each branch.[7]
10. **Maximize.** Return the maximum of satisfied soft constraints among the constraint systems over all the branches, and exit.

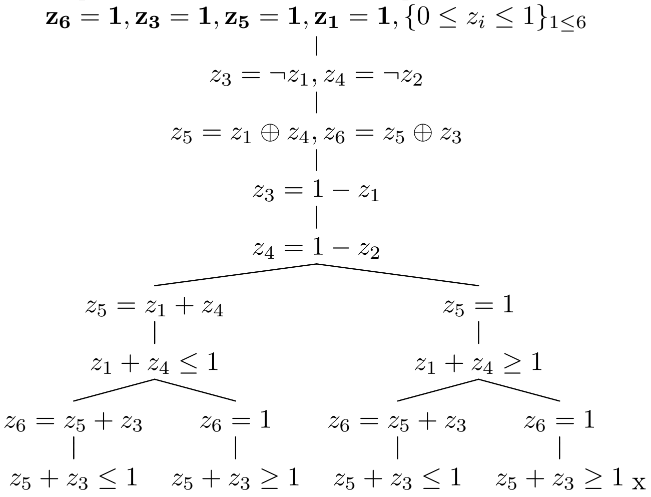
---

[7] Since all equalities are marked hard, any feasible solution to the **Max-FS** task will need to satisfy all of them. More generally, see [5, Concluding remarks] for handling soft constraints that are equalities.

Example 2. Let us consider the list of formulas $\langle (x \oplus \neg y) \oplus \neg x, \neg x, x \oplus \neg y, x\rangle$. A list of its subformulas (according to the definition in step 1) is the following:

$$\langle x, y, \neg x, \neg y, x \oplus \neg y, (x \oplus \neg y) \oplus \neg x\rangle$$

In order to depict the example in a compact way we use the same conventions as in Example 1. Furthermore, we will print in bold the soft constraints.



Theorem 5. *The method* TŁ$_{\text{MaxSAT}}$ *is sound and complete for* **MaxSAT-OPT**.

Proof. The **soundness** claim states that whenever the method returns $k \in \mathbb{N}$ on input $\langle \varphi_1, \dots, \varphi_m\rangle$, then there is an assignment v to variables $x_1, \dots, x_n$ that satisfies k formulas among $\langle \varphi_1, \dots, \varphi_m\rangle$. If TŁ$_{\text{MaxSAT}}$ returns k, that means the tree $\mathbf{T}$ was constructed with a branch B and a system of constraints given by B that yielded k upon solving the Max-FS problem with hard and soft constraints, and that this was the maximum solution among all branches. Fix such a v and notice that v defines values for $x_1, \dots, x_n$. Using Lemma 3, all hard constraints from the system, in particular, all constraints from steps 3, 5 and 8 are satisfied by v, and so are k of the target constraints. If ψ is a subformula of some φ_i with $i \in \{1, \dots, m\}$, we have $v(\psi) = v(z_j)$ whenever z_j is the z-variable assigned to ψ, by induction. In particular, from step 7 we have that there are k formulas φ_i among $\langle \varphi_1, \dots, \varphi_m\rangle$ such that $v(\varphi_i) = 1$.

The **completeness** claim states that if, for some assignment v, there are k items φ_i on the list $\langle \varphi_1, \dots, \varphi_m\rangle$ such that $v(\varphi_i) = 1$, then the method TŁ$_{\text{MaxSAT}}$ yields at least k on that instance. So assume that $v(\varphi_i) = 1$ for at least k such items and fix v. We claim there is a branch B of $\mathbf{T}$ with a system of constraints that yields at least k upon solving its instance of Max-FS problem. First construct the tree $\mathbf{T}$. From v, we get values for $x_1, \dots, x_n$, the z-variables that are their names, and using equations from step 3 for the remaining z-variables. The assignment v indicates a leaf of $\mathbf{T}$ that defines a branch B via a series of (possibly non-unique) choices on the hard constraints. If ψ is a subformula of some φ_i with $i \in \{1, \ldots, m\}$, also $v(\psi) = v(z_j)$ whenever z_j is the z-variable assigned to ψ, all the hard constraints and at least k soft constraints are satisfied on B under v. Since k formulas on input are satisfied by v, also k soft constraints are satisfied. Thus the method $\mathrm{TŁ}_{\mathrm{MaxSAT}}$, which returns a maximum over all branches, will yield a value no less than k.

To put side by side the efficiency of the method $\mathrm{TŁ}_{\mathrm{SAT}}$ from Subsect. 4.1 with the method $\mathrm{TŁ}_{\mathrm{MaxSAT}}$ above, we assume a modification of $\mathrm{TŁ}_{\mathrm{SAT}}$ that takes as input a finite list of arbitrary formulas $\langle \varphi_1, \ldots, \varphi_m \rangle$ and tests their joint satisfiability. Then we obtain comparable trees from both methods, the main difference being in the target constraints. Each branch of the tree obtained from $\mathrm{TŁ}_{\mathrm{SAT}}$ defines a set of constraints the solvability of which is in **P**. It is typically not necessary to test solvability on all the branches. On the other hand, if $\langle \varphi_1, \ldots, \varphi_m \rangle$ is an input to $\mathrm{TŁ}_{\mathrm{MaxSAT}}$, then on each branch of the generated tree, it is indeed necessary to solve the Max-FS problem with hard and soft constraints that the branch defines, because the method eventually takes a maximum over *all* the branches. Moreover, the problem on each branch is **NP**-hard [4]. In this sense, the complexity of the method $\mathrm{TŁ}_{\mathrm{SAT}}$ is a *lower bound* on the complexity of the method $\mathrm{TŁ}_{\mathrm{MaxSAT}}$ as presented above.

One can conceive optimizing the method $\mathrm{TŁ}_{\mathrm{MaxSAT}}$ by observing that, firstly, the multiset of soft constraints remains the same over all the branches, and secondly, if any subset S' of a set S of hard constraints is unsolvable, then so is S. We refrain from pursuing these considerations here, since they are addressed by the methods used in MIP solvers. The following lemma comes in useful.

Lemma 6. *The tree obtained from the* $\mathrm{TŁ}_{\mathrm{MaxSAT}}$ *method can be linearized at the cost of adding at most* $||\varphi||$ *Boolean variables. The linearization method does not affect the soft constraints.*

Proof. Any branching in step 8 of the algorithm can be replaced by expanding the tree with new nodes (without branching) using Lemma 5. The constraints obtained from the Lemma are all marked *hard*. This step therefore does not impact the set of possible solutions to the hard constraints in the system. The soft constraints are the same on all the branches, therefore the soft constraints in the linearization are well defined.

An extension of the Max-FS problem with Boolean variables among the set of hard constraints can also be rendered as a MIP problem with hard and soft constraints, with the Boolean variables not occurring in the soft constraints. Section 3 gives as benchmark for **MaxSAT-OPT** $\log m$ calls to a MIP solver for **SAT** with inputs of size $O(\Sigma_{i=1}^m |\varphi_i| + m^2)$.

5 Concluding Remarks and Future Work

Envisaged work on this material will consider finite-valued reductions of the **SAT** problem via upper bounds on denominators [1–3] to obtain a comparison with variants of TŁ_{SAT} for deterministic worst-case complexity for arbitrary formulas. Also, it remains to be seen whether upper bounds on denominators (a “small-model theorem”, cf., e.g., [11]) can be used to classify the decision version of the above Max-FS problem with Boolean variables among its hard constraints within $\mathbf{FP}^{\mathbf{NP}}$ for a conclusive comparison with the canonical approach. Another line of possible work stems from a generalized notion of satisfiability, considering, instead of the MaxSAT family of problems, their MaxSAT_r version, for a rational $r \in (0, 1]$, asking for the maximum number of formulas that are assigned a value greater than or equal to r by a single assignment.

Acknowledgements. We thank three anonymous reviewers for their useful and inspiring comments. Haniková was supported by the long-term strategic development financing of the ICS (RVO:67985807) and by mobility grant no. CSIC-20–12 of the Czech Academy of Sciences. Manyà was supported by grants PID2019-111544GB-C21, PID2022-139835NB-C21 and TED2021-129319B-I00 funded by MCIN/AEI/10.13039/501100011033. Vidal was supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 101027914.

References

1. Aguzzoli, S.: An asymptotically tight bound on countermodels for Łukasiewicz logic. Int. J. Approximate Reasoning **43**(1), 76–89 (2006)
2. Aguzzoli, S., Ciabattoni, A.: Finiteness in infinite-valued Łukasiewicz logic. J. Logic Lang. Inf. **9**(1), 5–29 (2000)
3. Aguzzoli, S., Gerla, B.: Finite-valued reductions of infinite-valued logics. Archive Math. Logic **41**(4), 361–399 (2002)
4. Amaldi, E., Kann, V.: The complexity and approximability of finding maximum feasible subsystems of linear relations. Theor. Comput. Sci. **147**, 181–210 (1995)
5. Amaldi, E., Pfetsch, M.E., Leslie, E., Trotter, J.: On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. Math. Program. Ser. A **95**, 533–554 (2003)
6. Ansótegui, C., Bofill, M., Manyà, F., Villaret, M.: Building automated theorem provers for infinitely-valued logics with satisfiability modulo theory solvers. In: Proceedings, 42nd International Symposium on Multiple-Valued Logics (ISMVL), Victoria, BC, Canada, pp. 25–30. IEEE CS Press (2012)
7. Ansótegui, C., Bofill, M., Manyà, F., Villaret, M.: Automated theorem provers for multiple-valued logics with satisfiability modulo theory solvers. Fuzzy Sets Syst. **292**, 32–48 (2016)
8. Bacchus, F., Järvisalo, M., Ruben, M.: Maximum satisfiability. In: Handbook of Satisfiability, second edition, pp. 929–991. IOS Press (2021)
9. Bofill, M., Manyà, F., Vidal, A., Villaret, M.: New complexity results for Łukasiewicz logic. Soft. Comput. **23**, 2187–2197 (2019)
10. Chang, C.C.: A new proof of the completeness of the Łukasiewicz axioms. Trans. Am. Math. Soc. **93**(1), 74–80 (1959)
11. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Inf. Comput. **87**(1–2), 78–128 (1990)
12. Hähnle, R.: Many-valued logic and mixed integer programming. Ann. Math. Artif. Intell. **12**(3–4), 231–264 (1994)

13. Hájek, P.: Metamathematics of Fuzzy Logic, Trends in Logic, vol. 4. Kluwer, Dordrecht (1998)
14. Haniková, Z.: Computational complexity of propositional fuzzy logics. In: Cintula, P., Hájek, P., Noguera, C. (eds.) Handbook of Mathematical Fuzzy Logic, vol. 2, pp. 793–851. College Publications (2011)
15. Haniková, Z.: On the complexity of validity degrees in Łukasiewicz logic. In: Anselmo, M., Della Vedova, G., Manea, F., Pauly, A. (eds.) CiE 2020. LNCS, pp. 175–188. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-51466-2_15
16. Haniková, Z., Savický, P.: Term satisfiability in FL$_{ew}$-algebras. Theor. Comput. Sci. **631**, 1–15 (2016)
17. Krentel, M.W.: The complexity of optimization problems. J. Comput. Syst. Sci. **36**, 490–509 (1988)
18. Li, C.M., Manyà, F.: MaxSAT, hard and soft constraints. In: Handbook of Satisfiability, second edition, pp. 903–927. IOS Press (2021)
19. Li, C.M., Manyà, F., Vidal, A.: Tableaux for maximum satisfiability in Łukasiewicz logic. In: IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL), pp. 243–248. IEEE Computer Society, Miyazaki (2020)
20. Mundici, D.: Mapping abelian ℓ-groups with strong unit one-one into MV-algebras. J. Algebra **98**(1), 76–81 (1986)
21. Mundici, D.: Satisfiability in many-valued sentential logic is NP-complete. Theor. Comput. Sci. **52**(1–2), 145–153 (1987)
22. Mundici, D.: Ulam game, the logic of MaxSAT, and many-valued partitions. In: Dubois, D., Klement, E.P., Prade, H. (eds.) Fuzzy Sets, Logics and Reasoning about Knowledge, pp. 121–137. Kluwer (1999)
23. Mundici, D., Olivetti, N.: Resolution and model building in the infinite-valued calculus of Łukasiewicz. Theor. Comput. Sci. **200**, 335–366 (1998)
24. Olivetti, N.: Tableaux for Łukasiewicz Infinite-valued Logic. Stud. Logica. **73**, 81–111 (2003)
25. Preto, S., Manyà, F., Finger, M.: Linking Łukasiewicz Logic and Boolean Maximum Satisfiability, ISMVL, pp. 164–169. IEEE Computer Society, Miyazaki (2023)
26. Schockaert, S., Janssen, J., Vermeir, D.: Satisfiability checking in Łukasiewicz logic as finite constraint satisfaction. J. Autom. Reasoning **49**, 493–550 (2012)
27. Schockaert, S., Janssen, J., Vermeir, D., De Cock, M.: Finite satisfiability in infinite-valued Łukasiewicz logic. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS (LNAI), vol. 5785, pp. 240–254. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04388-8_19
28. Schrijver, A.: Theory of Linear and Integral Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester (1998)
29. Stockmeyer, L.J.: Computational Complexity. In: Coffman, E.G., et al. (eds.) Handbooks in OR & MS, Vol. 3, pp. 455–517. Elsevier Science Publishers (1992)
30. Torrens, A.: Cyclic elements in MV-algebras and post algebras. Math. Logic Q. **40**(4), 431–444 (1994)
31. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Slisenko, A.O. (ed.) Studies in mathematics and mathematical logic, Part II, pp. 115–125. Steklov Mathematical Institute (1968)
32. Vidal, A.: MNiBLoS: a SMT-based solver for continuous t-norm based logics and some of their modal expansions. Inf. Sci. **372**, 709–730 (2016)

Separation Logic

The Logic of Separation Logic: Models and Proofs

Frank S. de Boer[1,2], Hans-Dieter A. Hiep[1,2](✉), and Stijn de Gouw[3]

[1] Centrum Wiskunde and Informatica (CWI), Amsterdam, The Netherlands
hdh@cwi.nl

[2] Leiden Institute of Advanced Computer Sciences (LIACS), Leiden, The Netherlands

[3] Open University (OU), Heerlen, The Netherlands

Abstract. The standard semantics of separation logic is restricted to finite heaps. This restriction already gives rise to a logic which does not satisfy compactness, hence it does not allow for an effective, sound and complete axiomatization. In this paper we therefore study both the general model theory and proof theory of the separation logic of finite and infinite heaps over arbitrary (first-order) models. We show that we can express in the resulting logic finiteness of the models and the existence of both countably infinite and uncountable models. We further show that a sound and complete sequent calculus still can be obtained by restricting the second-order quantification over heaps to first-order definable heaps.

1 Introduction

Separation logic [Rey02], in the sequel also referred to by SL, extends first-order logic with the separating connectives of conjunction and implication for reasoning about programs which feature the dynamic allocation of variables that are stored at locations of that part of the memory called the 'heap'. The *separating conjunction* allows to specify properties of a partition of the heap into two disjoint sub-heaps. The *separating implication* (also called 'the magic wand') allows to express properties of disjoint extensions of the heap. Both separating connectives involve a second-order quantification over heaps (which are represented by binary relations).

In this paper we study both the model theory and the proof theory of SL. The standard model of SL (as introduced in [Rey02]) extends the standard model of arithmetic with the so-called 'points-to' relation which provides a formalization of the heap in terms of the *graph* of a *finitely-based partial function*. This function assigns to each location of the heap its stored value, or is undefined if the location is not allocated. In the standard semantics of SL (here also called *weak* SL), the domains of heaps are finite, that is, only finitely many locations are allocated. Reasoning about finite heaps however requires an *infinitary* logic because the logic of finite heaps, and that of finite model theory in general, does not satisfy the compactness property: it is straightforward to express for each natural number that the domain of the heap contains at least that number of

R. Ramanayake and J. Urban (Eds.): TABLEAUX 2023, LNAI 14278, pp. 407–426, 2023.
https://doi.org/10.1007/978-3-031-43513-3_22

elements. It follows that every finite subset of this infinite set of sentences is satisfiable, but clearly no finite heap satisfies the entire set.

To study the general model and proof theory of *full* SL[1] we (1) extend its semantics to arbitrary first-order models and (2) generalize the notion of a heap to a partial function on the underlying domain of the given (first-order) model: no restrictions are imposed on the cardinality of the domain of heap, in contrast to weak SL which restricts to finite heaps. Our main model-theoretic results are that in this general setting we can express: (1) finiteness of models, (2) well-foundedness of the points-to relation, and (3) existence of countably infinite and uncountable models. As a consequence we have that full SL satisfies neither compactness nor the downward and upward Löwenheim-Skolem theorems (see [CK13]). Non-compactness implies that there does not exist an effective, sound and complete proof theory for SL. In fact, we will show that the well-foundedness of the points-to relation can already be expressed in full SL using only separating conjunction. Consequently, full SL without separating implication is already non-compact. For full SL without separating implication but in which separating conjunction only occurs positively, the fragment which we call separation logic light (SLL), we do have compactness, but its semantic consequence relation is not compact and therefore also does not allow for an effective, sound and complete proof theory.

The question thus arises whether there exists an *alternative* interpretation of SL that does allow for an effective, sound and complete proof theory. Clearly, the main complexity of SL stems from the (second-order) quantification over heaps (or sub-heaps, as in the case of the separating conjunction). For second-order logic a sound and complete axiomatization can be obtained by generalizing its semantics by means of so-called *general models*. Such models extend first-order models with a set of possible interpretations of the second-order variables. For example, instead of interpreting a monadic predicate over *all* possible subsets of the given first-order domain, a general model restricts its interpretation to a given set of such subsets. This generalization of the semantics of second-order logic allows for a sound and complete axiomatization by restricting to so-called Henkin models. A Henkin model is a general model for second-order logic which additionally satisfies the comprehension axiom

$$\exists R \forall x_1, \ldots, x_n (R(x_1, \ldots, x_n) \leftrightarrow \phi(x_1, \ldots, x_n))$$

for any second-order formula $\phi(x_1, \ldots, x_n)$ which does not contain the n-ary relation symbol R. In the *arithmetic* comprehension axiom $\phi(x_1, \ldots, x_n)$ is first-order.

Generalizing the semantics of SL accordingly in terms of a given set of possible heaps, which does not necessarily contain *all* heaps, we can formulate in SL the following version of the arithmetic comprehension axiom

$$\blacklozenge(\forall x, y((x \hookrightarrow y) \leftrightarrow \phi(x, y)))$$

[1] Here we adopt the terminology for second-order logic [Vää01], where the semantics of *full* second-order logic does not impose any restrictions on the *cardinality* of the interpretation of the predicates/relations, in contrast to *weak* second-order logic which restricts to *finite* interpretations (of the predicates/relations).

which expresses the existence of a heap such that its *graph*, as denoted by the points-to relation $\hookrightarrow$, satisfies the 'pure' first-order formula $\phi(x, y)$ (i.e., ϕ does not involve the separation connectives and the points-to relation). The $\blacklozenge$-modality (formally defined in Sect. 3) expresses the existence of a heap which satisfies the associated formula. Such an instance of the arithmetic comprehension axiom holds if there exists a heap which is characterized by the formula $\phi(x, y)$. We cannot generalize this axiom to arbitrary SL formulas because it is not obvious how to avoid contradictions like $\blacklozenge(\forall x, y((x \hookrightarrow y) \leftrightarrow \neg(x \hookrightarrow y)))$. Simply requiring that the points-to relation does not occur in $\phi(x, y)$ does not work because the separating connectives implicitly refer to it. Therefore, we introduce a new interpretation of SL that restricts the (second-order) quantification to *first-order definable* heaps. For this new interpretation we introduce a *sequent calculus* which is sound and complete. The completeness proof is based on the construction of a model for a *consistent* theory (a theory from which false is not derivable), following [Hen49]. From the completeness proof we further derive that this new interpretation satisfies both compactness and the downward Löwenheim-Skolem theorem. By the seminal theorem of Lindström we then infer that this new interpretation is as expressive as first-order logic.

Related Work. The model theory of SL has been focused mainly on finite heaps. For example, the computability and complexity results in [CYO01] depend on this assumption. Surprisingly, in [BDL12] the authors show that *weak* SL is as expressive as *weak* second-order logic [Man96], which is a semantics of second-order logic where quantification is restricted to finite relations. In [DD16] this result is further refined by the restriction to two variables and the separating implication (no separating conjunction) which still is as expressive as weak second-order logic. In [EIP20] the satisfiability problem for SL with k record fields has been studied for finite heaps, but over arbitrary first-order models. A tableaux method for a propositional fragment of SL has been developed in [GM10] which has been proven sound and complete. Extensions to first-order SL are discussed assuming finite heaps. In fact, the tableaux method introduced is based on a labelling mechanism for encoding finite heap structures.

In contrast, when investigating complete proof systems for SL the assumption of the finiteness of heaps has to be dropped, thus allowing for infinite heaps, because, as already observed above, finiteness leads to non-compactness. Our general model theory shows that this generalization of SL, *full* SL, is also non-compact, and therefore does not allow for a finitary sound and complete logic either. Consequently, to obtain such a logic one either has to syntactically restrict SL or further abstract or generalize its semantics. In [DLM21], for example, a sound and complete sequent calculus is described for a quantifier-free subset of SL. On the other hand, examples of further abstractions and generalizations are [HT16] and [Pym02], and both describe a finitary logic which is sound and complete. In [Pym02], models are based on very general preordered commutative monoids and there is no points-to relation. In [HT16], special commutative monoids called *separation algebras* are used to give semantics to the separating connectives. The elements of such separation algebras represent heaps as relations on the underlying (first-order) domain. This allows for a standard set-theoretic interpretation of the points-to relation. However, the semantics of separating conjunction is defined in terms of the abstract monoid, and as such is decoupled from the set-theoretic interpretation of the points-to relation. For example, a first-order specification (using plain conjunction) of an enumeration of the elements of the domain of a (finite) heap *as a set* does not in general correspond with an enumeration using separation conjunction.

A sound and complete axiomatization of the points-to relation in the general context of first-order SL *respecting its standard set-theoretic interpretation* thus remains a main challenge.

Second-order logic allows for a straightforward translation of the (weak or full) semantics of SL, and one can use second-order logic to reason about validity in SL. This approach is followed for example by the IRIS project [JKJ+18] which formalizes the semantics of weak SL in the higher-order logic of Coq [HH14]. By restricting the semantics of the separating connectives to (first-order) definable heaps, our approach instead transforms a *compositional* second-order logical description of the semantics of SL into corresponding rules of a standard first-order sequent calculus. The resulting calculus allows us to reason, in a natural manner, in first-order logic about the (hierarchical) heap structures generated by the rules for the separating connectives. As such it does not involve the additional tree structures of the so-called *bunched contexts* of the sequent calculi of [HT16] and [Pym02]. Also [Kri08] avoids the use of bunched contexts in a modal sequent calculus for propositional SL, which is proven sound. However it is incomplete because it provides limited support for equational reasoning about the modal contexts (so-called 'worlds') associated with the SL formulas.

Plan of the Paper. In the next section we introduce the syntax and semantics of full SL. In Sect. 3 we investigate the expressiveness of full SL. Section 4 introduces a restriction of the semantics to definable heaps. In Sect. 5 we introduce the sequent calculus, and discuss soundness and completeness. Finally, in the conclusion section we wrap up, and discuss some future work.

2 Separation Logic

In this section we introduce the syntax of SL and define its classical semantics with respect to arbitrary first-order models. For an intuitive introduction to separation logic, see [Rey05]. Given a first-order signature of function and predicate symbols[2] and a countably infinite set of first-order variables $x, y, z, \ldots$, the first-order terms of this signature are denoted by $t, t', \ldots$.

We have the following inductive definition of formulas of separation logic.

Definition 1 (Syntax of SL). *We define*

$$p ::= (t_1 = t_2) \mid R(t_1, \ldots, t_n) \mid (\neg p) \mid (p \wedge q) \mid \exists x(p) \mid (p * q) \mid (p \mathrel{-\!\!*} q)$$

[2] We allow for a countably infinite set of such symbols.

where R is a n-ary relation symbol. As a special case we have the binary 'points-to' relation symbol $\hookrightarrow$ (also called the weak/loose points-to).

Let $M = (D, I)$ denote a first-order model, where D denotes the non-empty domain and I provides an interpretation of the function and predicate symbols as functions and relations over D. A valuation s assigns elements of the domain D of M to the first-order variables $x, y, z, \ldots$. We omit the standard inductive definition of the value $I_s(t)$ of a term t. Given a model $M = (D, I)$, we denote by $M, h, s \models p$ that p holds in the model M, under the interpretation $h \subseteq D \times D$ of the binary relation symbol $\hookrightarrow$, where h denotes a so-called *heap*, represented as the graph of a *partial function* with *finite domain*.

Definition 2 (Semantics of SL). *We have the following main cases.*

- $M, h, s \models (t \hookrightarrow t')$ *if and only if* $\langle I_s(t), I_s(t')\rangle \in h$.
- $M, h, s \models (p * q)$ *if and only if* $M, h_1, s \models p$ *and* $M, h_2, s \models q$, *for some heaps* $h_1, h_2 \subseteq D \times D$ *such that* $h = h_1 \cup h_2$ *and* $h_1 \perp h_2$.
- $M, h, s \models (p \mathrel{-\!\!*} q)$ *if and only if* $M, h', s \models p$ *implies* $M, h \cup h', s \models q$, *for all heaps* $h' \subseteq D \times D$ *such that* $h \perp h'$.

Other cases are the Tarksi-style semantics of classical logic [Yan01, Table 5.2].

In the above definition we use the set-theoretic operation of *union* of binary relations as sets of pairs. On the other hand, by $h_1 \perp h_2$ we denote that the *domains* of the relations h_1 and h_2 are *disjoint*[3]. As such, we can introduce the strict/tight points-to relation $\mapsto$ of SL, defined by $M, h, s \models t \mapsto t'$ if and only if $h = \{\langle I_s(t), I_s(t')\rangle\}$, as a derived concept: it can be expressed by $(t \hookrightarrow t') \wedge \forall x, y((x \hookrightarrow y) \rightarrow (x = t \wedge y = t'))$. The concept **emp** of the empty relation can also be expressed by $\forall x, y(x \not\hookrightarrow y)$. *Intuitionistic* SL only allows for the weak/loose points-to relation. The strict version cannot be expressed in intuitionistic SL because of its *monotonicity* property that the truth of a formula is preserved by extensions of the domain of the heap [Rey00]. In this article we focus on classical separation logic only.

Let $(x_i \hookrightarrow -)$ abbreviate $\exists y(x_i \hookrightarrow y)$. The sentences ϕ_n defined by

$$\exists x_1, \ldots, x_n((x_1 \hookrightarrow -) * \ldots * (x_n \hookrightarrow -))$$

then state that there exist at least n allocated elements of the underlying domain of the given first-order model. Note that the semantics of the separating conjunction implies that $x_i \neq x_j$ for $i \neq j$. It is also possible to formulate the same property using propositional conjunction instead of separating conjunction by explicitly stating this fact, that the variables are not aliases. Now collect all ϕ_n in a set. Clearly, every finite subset of this set of sentences is satisfied by a finite heap, but that there does not exist a finite heap satisfying all these sentences.

[3] The domain of an arbitrary relation $\mathcal{R} \subseteq D \times D$ is the set $d \in D$ for which there exists a $d' \in D$ such that $\langle d, d'\rangle \in \mathcal{R}$. Note that for heaps $h_1 \perp h_2$ is equivalent to $h_1 \cap h_2 = \emptyset$.

This simple counterexample to compactness provides the basic motivation to study the above semantics of SL extended to unbounded heaps, i.e. heaps which potentially have an infinite domain.

Further, for technical convenience only, we generalize the semantics to arbitrary *binary relations*. For an arbitrary (binary) relation $\mathcal{R} \subseteq D \times D$ on the underlying domain D of the given first-order model, we define $M, \mathcal{R}, s \models p$ as above, where the interpretation of the separating connectives ranges over arbitrary subsets of $D \times D$. In fact, in this generalized semantics, which we call *relational* SL, we can model the restriction to heaps simply by *syntactically* restricting the separating implication to assertions of the form $(p \wedge \mathit{fun}) \mathrel{-\!\!*} q$, where fun denotes the assertion $\forall x, y, z((x \hookrightarrow y \wedge x \hookrightarrow z) \rightarrow y = z)$. Let p' denote the result of restricting syntactically all occurrences of the separating implication in p to heaps (as described above). It follows that the evaluation of $p' \wedge \mathit{fun}$ is restricted to heaps.

It is worthwhile to observe here that there exists a straightforward formalization of relational SL in second-order logic. For any formula p as defined above we define inductively the second-order formula $p(R)$, where R is a binary relation.

Definition 3 (Logical formalization of relational SL).
We have the following main cases.

- $(t \hookrightarrow t')(R) = R(t, t')$,
- $(p * q)(R) = \exists R_1, R_2(R = R_1 \uplus R_2 \wedge p(R_1) \wedge q(R_2))$,
- $(p \mathrel{-\!\!*} q)(R) = \forall R_1, R_2((R_2 = R_1 \uplus R \wedge p(R_1)) \rightarrow q(R_2))$.

Here we denote by $R = R_1 \uplus R_2$, for any binary relation symbols R, R_1, R_2, the conjunction of the formulas $\forall x, y(R(x, y) \leftrightarrow (R_1(x, y) \vee R_2(x, y)))$ and $\forall x, y, z(\neg R_1(x, y) \vee \neg R_2(x, z))$. We denote by $M, s \models \phi$ the standard truth definition of a second-order formula ϕ, where the evaluation s additionally interprets the second-order variables. Correctness of this translation, that is, $M, \mathcal{R}, s \models p$ if and only if $M, s[R := \mathcal{R}] \models p(R)$ (where $s[R := \mathcal{R}]$ denotes the update of s which assigns to the binary variable R the relation $\mathcal{R}$), can be established by a straightforward induction on p.

3 Model Theory: Compactness and Countability

To explore the general model theory of SL we introduce the modalities $\blacksquare p$ and $\Box p$ as abbreviations of $\mathbf{true} * (\mathbf{emp} \wedge (\mathbf{true} \mathrel{-\!\!*} p))$ and $\neg(\mathbf{true} * \neg p)$, respectively[4]. For $M = (D, I)$ we have $M, \mathcal{R}, s \models \blacksquare p$ if and only if $M, \mathcal{R}', s \models p$, for *every* $\mathcal{R}' \subseteq D \times D$. Further, we have $M, \mathcal{R}, s \models \Box p$ if and only if $M, \mathcal{R}', s \models p$, for *every* sub-relation $\mathcal{R}'$ of $\mathcal{R}$ (that is, $\mathcal{R}' \subseteq \mathcal{R}$). By $\blacklozenge p$ we denote the formula $\neg \blacksquare \neg p$. It follows that $M, \mathcal{R}, s \models \blacklozenge p$ if and only if $M, \mathcal{R}', s \models p$, for *some* $\mathcal{R}' \subseteq D \times D$.

Characterizing Finite Models. The above $\blacksquare$-modality allows to express that the domain D of a model $M = (D, I)$ is finite, by asserting that every injective

[4] We note that $\blacksquare$ and $\blacklozenge$ are, respectively, $\Box$ and $\Diamond$ in [HT16]. However in [HT16] they are introduced not as abbreviations but as *primitive* concepts.

function $f : D \to D$ is a surjection: Let *inj* be the conjunction of the formulas *fun* (as defined above), $\forall x, y, z((x \hookrightarrow z \land y \hookrightarrow z) \to x = y)$, and $\forall x \exists y(x \hookrightarrow y)$. We have that $M, \mathcal{R}, s \models inj$ if and only if $\mathcal{R} : D \to D$ is injective (note that the domain of $\mathcal{R}$ is D because $M, \mathcal{R}, s \models \forall x \exists y(x \hookrightarrow y)$). And so $M, \mathcal{R}, s \models \blacksquare(inj \to \forall x \exists y(y \hookrightarrow x))$ if and only if D is finite. Note that the occurrences of $\hookrightarrow$ in the scope of the $\blacksquare$-modality are universally bounded, and the interpretation of $\hookrightarrow$ thus ranges over *all* $\mathcal{R} \subseteq D \times D$.

Characterizing Countable Infinity. We next show that countability of the underlying domain of a model can be expressed, using the above two modalities. We will be working with chains related by $\hookrightarrow$, and in that sense we speak of a *predecessor* of x, being any y such that $(y \hookrightarrow x)$, and *successor* of x, being any y such that $(x \hookrightarrow y)$. Let *enum* be the conjunction of the following formulas:

- the above formula *inj*,
- the formula $\exists! x \forall y(y \not\hookrightarrow x)$[5], which states the existence of a unique *minimal* element (that is, an element that has no predecessor),
- the formula $\Box(\mathbf{emp} \lor \exists x((x \hookrightarrow -) \land \forall y((y \hookrightarrow -) \to (y \not\hookrightarrow x)))$, which expresses that the points-to relation $\hookrightarrow$ is *well-founded.*

Note that a relation $\mathcal{R}$ is well-founded iff *every* (non-empty) sub-relation of $\mathcal{R}$ has a minimal element (with respect to that sub-relation). This fact can be expressed by the use of the formula *enum*. Let $M, \mathcal{R}, s \models enum$. We show that $\mathcal{R}$ encodes an enumeration $\langle d_n \rangle_n$ of D (still we have $M = (D, I)$). We define the sequence $\langle d_n \rangle_n$ by induction on n: for d_0 we take the (unique) minimal element, and for d_{n+1} we take the unique element $d \in D$ such that $\langle d_n, d \rangle \in \mathcal{R}$. Note that *inj* implies that every element of D has a unique 'successor' and that $d_{n+1} \notin \{d_0, \ldots, d_n\}$. Well-foundedness ensures that every element of D appears in the enumeration $\langle d_n \rangle_n$. Because otherwise we can construct an infinite descending chain of elements not appearing in the enumeration $\langle d_n \rangle_n$ (since d_0 denotes the unique minimal element with respect to the functional interpretation $\mathcal{R}$ of $\hookrightarrow$, it follows that for any $d \in D$ which does not appear in the enumeration $\langle d_n \rangle_n$ there exists a $d' \in D$ which also does not appear in the enumeration $\langle d_n \rangle_n$ and $\langle d', d \rangle \in \mathcal{R}$).

We thus have that $M, \mathcal{R}, s \models enum$ implies that the domain of M is countably infinite. The formula $\blacklozenge enum$ further abstracts from the current interpretation of the points-to relation $\hookrightarrow$, so that if the domain of M is countably infinite then $M, \mathcal{R}, s \models \blacklozenge enum$, for arbitrary $\mathcal{R}$ (and s).

The class of uncountable models is characterized by $\neg(\blacklozenge enum \lor fin)$, where *fin* denotes the above formula which characterizes the class of finite models.

Summarizing, the logic of full SL is neither compact nor does it satisfy the Löwenheim-Skolem theorem because it can distinguish between countable and uncountable models. Further, we observe that the above expressiveness results do not depend on the interpretation of the points-to relation as an arbitrary relation. That is, these results also hold for the semantics restricted to (infinite) heaps.

[5] $\exists! x p$ is an abbreviation of $\exists x(p \land \forall y(p[y/x] \to y = x))$, where $p[y/x]$ denotes the substitution of x by y.

Interestingly, since we can express that the points-to relation $\hookrightarrow$ is well-founded (see above), even restricting to the separating conjunction gives rise to non-compactness: given a countably infinite set of individual constants c_n, $n \geq 0$, let Γ consist of the above formula $\Box(\mathbf{emp} \vee \exists x((x \hookrightarrow -) \wedge \forall y((y \hookrightarrow -) \rightarrow (y \not\hookrightarrow x)))$ and the formulas $c_{n+1} \hookrightarrow c_n$, $n \geq 0$. Clearly, every finite subset of Γ is satisfiable but Γ itself is not. Note that we do not need to require that all the $c_i \neq c_j$, for every $i \neq j$, because in case the formulas $c_{n+1} \hookrightarrow c_n$, $n \geq 0$, are satisfied and additionally $c_i = c_j$ holds, for some $i \neq j$, we have a loop in the interpretation of $\hookrightarrow$. Further, restricting SL to separating conjunction also does not satisfy the *upward* Löwenheim-Skolem theorem, because, as argued above, $M, \mathcal{R}, s \models \mathit{enum}$ implies (infinite) countability of the domain of M.

Separation Logic Light. What about further restricting to *positive* occurrences of the separating conjunction? Since we then can push negation inside, this restriction can be formally defined by the following syntax describing SLL ('separation logic light'):

$$p ::= (\neg)R(t_1, \ldots, t_n) \mid (p \vee q) \mid (p \wedge q) \mid \exists x(p) \mid \forall x(p) \mid (p * q)$$

Here R denotes either a n-ary relation symbol or the points-to relation $\hookrightarrow$. Thus, in this version of SL, negation can only be applied to atomic formulas. To show that the notion of satisfiability of SLL is compact, we introduce the following first-order translation $p@R$, where R is a binary predicate different from $\hookrightarrow$, $\circ$ denotes conjunction/disjunction, and Q denotes the existential/universal quantifier.

$$\begin{array}{ll}
(\neg)R(t_1, \ldots, t_n)@R' & = (\neg)R(t_1, \ldots, t_n) \\
(t \hookrightarrow t')@R & = R(t, t') \\
(p \circ q)@R & = p@R \circ q@R \\
Qx(p)@R & = Qx(p@R) \\
(p * q)@R & = R = R_1 \uplus R_2 \wedge p@R_1 \wedge q@R_2
\end{array}$$

The binary relation symbols R_1 and R_2 are 'fresh'. It follows that p is satisfiable if and only if $p@R$ is satisfiable. More precisely, $M, \mathcal{R}, s \models p$ if and only if there exists a (first-order) model M' such that $M', s \models p@R$. Consequently, compactness of first-order logic implies compactness of SLL: Let Γ be an infinite set of formulas of SLL and $\Gamma' = \{p@R \mid p \in \Gamma\}$[6], for some binary relation symbol R. If every finite subset of Γ is satisfiable, so is every finite subset of Γ'. By the compactness of first-order logic Γ' is satisfiable, and so is Γ. Along the same lines it follows that if Γ is satisfiable then there exists a model $M = (D, I)$ such that D is *countable* and $M, \mathcal{R}, s \models p$, for every $p \in \Gamma$.

Note however that compactness of the satisfiability relation does not imply that the (semantic) consequence relation is compact. In fact, non-compactness of the consequence relation for SLL follows directly from the above argument

[6] Note that Γ' may require the introduction of an infinite number of fresh (binary) relation symbols. This is however no problem because first-order logic allows for a countably infinite set of function and relation symbols.

involving well-founded relations: Let Γ denote the set formulas $c_{n+1} \hookrightarrow c_n$, $n \geq 0$. It follows that $\Gamma \models \textbf{true} * (\neg\textbf{emp} \wedge \forall x((x \hookrightarrow -) \rightarrow \exists y(y \hookrightarrow x)))$. But clearly, there does not exist a finite subset Γ_0 of Γ such that $\Gamma_0 \models \textbf{true} * (\neg\textbf{emp} \wedge \forall x((x \hookrightarrow -) \rightarrow \exists y(y \hookrightarrow x)))$.

Some Open Problems. The question remains whether restricting to separating conjunction satisfies the *downward* Löwenheim-Skolem theorem. A counterexample to the downward Löwenheim-Skolem theorem would be the expressibility of uncountable models. This seems to require the $\blacksquare p$ modality (and thus the separating implication).

Another interesting question is whether we can express finiteness of the domain of the current interpretation of the points-to relation, that is, does there exist a formula p in SL such that $M, \mathcal{R}, s \models p$ if and only if the domain of the relation $\mathcal{R}$ is finite?

A main open problem is a formalization of the relation between full SL and second-order logic. Intuitively, one of the main differences is the *local perspective* of SL, which is determined by the current heap. Remarkably, as already mentioned in the introduction, [BDL12] presents a rather intricate encoding of (dyadic) weak second-order logic into weak SL. Apparently this restriction to finite heaps allows to break the local perspective. Our conjecture however is that full SL is strictly less expressive than (dyadic) second-order logic. To illustrate how subtle this difference may be, consider the following extension of separation logic with a *binding* operator $\downarrow R(p)$ which binds the binary variable R in the evaluation of p to the current interpretation of the points-to relation. In other words, it corresponds to a bounded (second-order) quantification $\exists R((R = \hookrightarrow) \wedge p)$, where, $R = \hookrightarrow$ abbreviates the first-order formula $\forall x, y(R(x, y) \leftrightarrow (x \hookrightarrow y))$. Alternatively, we can directly define $M, \mathcal{R}, s \models \downarrow R(p)$ if and only if $M, \mathcal{R}, s[R := \mathcal{R}] \models p$. This definition thus assumes an extension of the valuation s to (binary) second-order variables. The expressive power of this binding operator lies in that it allows to 'break the spell' of the local perspective since the bound binary variable allows in the local context of the current interpretation of the points-to relation to refer to those 'outer' ones that have generated it (by the separating connectives). This extension of SL allows for a simple, compositional translation of (dyadic) second-order logic. We have the following main case which translates $\exists R(\phi)$, where ϕ a dyadic second-order formula (which is assumed not to contain occurrences of the points-to relation of SL), into the SL formula $\blacklozenge(\downarrow R(p))$.

4 Separation Logic of Definable Binary Relations

In this section we restrict the interpretation of the separating connectives to first-order definable binary relations. By ϕ we now denote a first-order formula which does not contain occurrences of the points-to relation $\hookrightarrow$ of SL. We omit the standard inductive truth definition $M, s \models \phi$ of a first-order formula ϕ.

By $\phi(x_1, \ldots, x_n)$ we denote that the free (first-order) variables of ϕ are among the distinct variables $x_1, \ldots, x_n$. A formula $\phi(x, y)$ is called a *binary* formula.

A binary formula is also simply denoted by ϕ, omitting its free variables x and y. Given a model $M = (D, I)$, and a first-order formula $\phi(x, y)$, we denote by $Rel_M(\phi)$ the relation $\{\langle s(x), s(y)\rangle \mid M, s \models \phi\} \subseteq D \times D$. Note that the evaluation of $\phi(x, y)$ only depends on the values of its free variables x and y, that is, $M, s \models \phi$ if and only if $M, s' \models \phi$, where $s(x) = s'(x)$ and $s(y) = s'(y)$. By $\phi(t, t')$ we denote the result of replacing in $\phi(x, y)$ the variables x and y by t and t', respectively (if necessary renaming bound variables to ensure that the variables of t and t' do not become bound).

Definition 4 (First-order definability). *Given a model $M = (D, I)$, a relation $\mathcal{R} \subseteq D \times D$ is* first-order definable *if $\mathcal{R} = Rel_M(\phi)$, for some binary formula $\phi(x, y)$.*

Note that, given a model $M = (D, I)$, $I(R) = Rel_M(R)$, that is, for any binary relation symbol R its interpretation $I(R)$ is trivially a first-order definable relation. We generalize the definition of $R = R_1 \uplus R_2$ to arbitrary binary formulas: we denote by $\phi = \phi_1 \uplus \phi_2$ that the binary formulas $\phi_1(x, y)$ and $\phi_2(x, y)$ represent a partition of the binary formula $\phi(x, y)$ which is expressed by the conjunction of $\forall x, y(\phi(x, y) \leftrightarrow (\phi_1(x, y) \vee \phi_2(x, y)))$ and $\forall x, y, z(\neg\phi_1(x, y) \vee \neg\phi_2(x, z))$. The latter formula, which states that the domains of the binary relations represented by $\phi_1(x, y)$ and $\phi_1(x, y)$ are disjoint, we abbreviate by $\phi_1 \perp \phi_2$.

In the sequel we denote by $M, \mathcal{R}, s \models p$ the *restriction* of the relational semantics of full SL (Definition 2 extended to binary relations) such that instead of quantifying over arbitrary binary relations, the separating connectives involve quantification over first-order definable binary relations. It is worthwhile to observe here that, as for Henkin models of second-order logic [Hen50], the implicit second-order quantification depends on the underlying signature of function and relation symbols. Extending or restricting the signature affects the semantics of formulas of the 'old' signature.

5 Sequent Calculus

To reason about the implicit quantification over definable (binary) relations, we introduce *rooted* assertions of the form $p@\phi$, where ϕ denotes a binary formula and p is a formula of SL (see Definition 1). We define $M, s \models p@\phi$ if and only if $M, \mathcal{R}, s \models p$, where $\mathcal{R} = Rel_M(\phi)$. The variables x and y of the binary formula $\phi(x, y)$ are thus implicitly bound by the @-operator, that is, $M, s \models p@\phi$ if and only if $M, s' \models p@\phi$, for any s and s' such that $s(z) = s'(z)$, for any free variable occurring in p.

Note that the separating connectives are interpreted in terms of relations which are definable by first-order formulas which do not involve the points-to relation $\hookrightarrow$. This allows for the following alternative *predicative* definition[7] of the semantics of the separating connectives in rooted assertions (used in both the soundness and completeness proofs). Here $\psi \perp \phi$, for the binary formulas $\psi(x, y)$ and $\phi(x, y)$, denotes the formula $\forall x, y, z(\neg\psi(x, y) \vee \neg\phi(x, z))$.

[7] For a foundational discussion concerning predicativity, see [Cro17].

Separating conjunction

$$\mathbf{L}_* \quad \frac{\Gamma, \phi = R_1 \uplus R_2, p@R_1, q@R_2 \Rightarrow \Delta}{\Gamma, (p * q)@\phi \Rightarrow \Delta}$$

$$\mathbf{R}_* \quad \frac{\Gamma \Rightarrow \Delta, \phi = \phi_1 \uplus \phi_2 \quad \Gamma \Rightarrow \Delta, p@\phi_1 \quad \Gamma \Rightarrow \Delta, q@\phi_2}{\Gamma \Rightarrow \Delta, (p * q)@\phi}$$

Separating implication

$$\mathbf{L}_{-\!*} \quad \frac{\Gamma \Rightarrow \Delta, \phi \perp \psi \quad \Gamma \Rightarrow \Delta, p@\psi \quad \Gamma, q@(\phi \vee \psi) \Rightarrow \Delta}{\Gamma, (p -\!* q)@\phi \Rightarrow \Delta}$$

$$\mathbf{R}_{-\!*} \quad \frac{\Gamma, R \perp \phi, p@R \Rightarrow \Delta, q@(\phi \vee R)}{\Gamma \Rightarrow \Delta, (p -\!* q)@\phi}$$

Points-to rules

$$\frac{\Gamma, p[\phi / \hookrightarrow] \Rightarrow \Delta}{\Gamma, p@\phi \Rightarrow \Delta} \qquad \frac{\Gamma \Rightarrow p[\phi / \hookrightarrow], \Delta}{\Gamma \Rightarrow p@\phi, \Delta}$$

Fig. 1. Sequent calculus. The binary relation symbols R_1, R_2 and R introduced in the rules $\mathbf{L}_*$ and $\mathbf{R}_{-\!*}$ are 'fresh'. In the points-to rules p denotes a basic formula (which does not contain occurrences of the separating connectives).

Lemma 1. *We have*

- *$M, s \models (p * q)@\phi$ if and only if there exist binary formulas ϕ_1 and ϕ_2 such that $M, s \models \phi = \phi_1 \uplus \phi_2$, $M, s \models p@\phi_1$, and $M, s \models q@\phi_2$.*
- *$M, s \models (p -\!* q)@\phi$ if and only if $M, s \models \psi \perp \phi$ and $M, s \models p@\psi$ implies $M, s \models q@(\phi \vee \psi)$, for all binary formulas ψ.*

We now develop a calculus for sequents $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$, where each A_i, $i = 1, \ldots, n$, and B_j, $j = 1, \ldots, m$, is constructed from first-order formulas and rooted assertions, which can be further composed using propositional connectives and quantification of first-order variables. This calculus is an extension of standard first-order sequent calculus (including cut), where the standard rules are applicable with respect to top-level propositional connectives and quantifiers. Figure 1 shows the left and right rules for separating conjunction and implication. These rules closely follow the translation in Definition 3 of SL into second-order logic, eliminating the explicit second-order quantification by applying the standard proof rules for second-order quantification (which themselves are straightforward generalizations of the rules for first-order quantification, instantiating the second-order variables by formulas). The binary relation symbols R_1, R_2 and R introduced in the rules $\mathbf{L}_*$ and $\mathbf{R}_{-\!*}$ are 'fresh' binary relation symbols, that is, they must not appear in the formulas of the conclusion of the rules.

We also have rules which allow classical reasoning under rooted assertions: $(p \circ q)@\phi \leftrightarrow (p@\phi) \circ (q@\phi)$, where $\circ$ denotes binary propositional connectives, e.g., conjunction, disjunction, and implication, $(\neg p)@\phi \leftrightarrow \neg(p@\phi)$, and $(\exists x p)@\phi \leftrightarrow \exists x(p@\phi)$ (and similarly $(\forall x p)@\phi \leftrightarrow \forall x(p@\phi)$). Further, we have $\forall x, y(\phi \leftrightarrow \psi) \rightarrow (p@\phi \leftrightarrow p@\psi)$. It is straightforward to validate these rules, but we omit the details of the semantics $M, s \models A$, which follows the standard Tarski-style classical semantics, given the semantics of rooted assertions which may appear in the place of atomic formulas.

In the so-called 'points-to' rules of Fig. 1 the formula p does not involve occurrences of the separating connectives. Such a formula of SL we call *basic*. Note that it differs from pure first-order formulas in that basic formulas additionally may involve the points-to relation. For such formulas we denote by $p[\phi/\hookrightarrow]$, for any binary formula $\phi(x, y)$, the result of replacing every atomic assertion $(t \hookrightarrow t')$ in p by $\phi(t, t')$, which is a pure first-order formula. It follows that $M, s \models p[\phi/\hookrightarrow]$ if and only if $M, Rel_M(\phi), s \models p$, for any basic formula p.

Example Proofs

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow q@R, R_1 \perp R_2 \quad \Gamma \Rightarrow q@R, p@R_1 \quad \Gamma, q@(R_1 \vee R_2) \Rightarrow q@R}{R = R_1 \uplus R_2, p@R_1, (p \mathrel{-\!\!*} q)@R_2 \Rightarrow q@R}\,\mathbf{L}_{-*}}{(p * (p \mathrel{-\!\!*} q))@R \Rightarrow q@R}\,\mathbf{L}_{*}}{\Rightarrow (p * (p \mathrel{-\!\!*} q))@R \rightarrow q@R}}{\Rightarrow ((p * (p \mathrel{-\!\!*} q)) \rightarrow q)@R}$$

As a first example of the use of the sequent calculus, above we have a derivation of the sequent $\Rightarrow ((p * (p \mathrel{-\!\!*} q)) \rightarrow q)@R$ which represents the validity of $(p * (p \mathrel{-\!\!*} q)) \rightarrow q$. This derivation essentially consists of an application of the rule $\mathbf{L}_*$ followed by an application of the rule $\mathbf{L}_{-*}$. In this derivation Γ denotes the formulas $R = R_1 \uplus R_2, p@R_1$ generated by the application of rule $\mathbf{L}_*$. The second premise of the application of the rule $\mathbf{L}_{-*}$ is derivable from an instance of the axiom $\Gamma, A \Rightarrow A, \Delta$. Note that ψ (in the $\mathbf{L}_{-*}$ rule) is instantiated with R_1. The first and third premise follows from the fact that $R = R_1 \uplus R_2$ reduces to $R_1 \perp R_2$ and $R = R_1 \cup R_2$ (that part of the proof is not shown above).

Next we show how to use the calculus in reasoning about the equivalence of weakest preconditions that arise in the practice of verifying the correctness of heap manipulating programs. Let p denote the weakest precondition $(u \hookrightarrow -) \wedge (z = 0 \triangleleft u = v \triangleright v \hookrightarrow z)$ of the heap update $[u] := 0$ which ensures the postcondition $v \hookrightarrow z$ after assigning the value 0 to the location denoted by the variable u (here $\phi \triangleleft b \triangleright \psi$ abbreviates $(b \wedge \phi) \vee (\neg b \wedge \psi)$) (in [dBHdG23] a dynamic logic extension of SL is introduced which generates this weakest precondition). The standard rule for backwards reasoning in [Rey02] gives the weakest precondition $(u \mapsto -) * (u \mapsto 0 \mathrel{-\!\!*} v \hookrightarrow z)$, which we denote by p'. These preconditions are equivalent because both are the weakest.

Surprisingly, a proof of the implication $p' \rightarrow p$ however exceeds the capability of all the automatic SL provers in the benchmark competition for SL [SNPR+19].

In particular, of the automatic provers, only the CVC4-SL tool [RISK16] supports the fragment of SL that includes the separating implication connective. However, from our own experiments with that tool, we found that it produces an incorrect counter-example and reported this as a bug to one of the maintainers of the project (Andrew Reynolds). In fact, the latest version, CVC5-SL, reports the same input as 'unknown', indicating that the tool is incomplete. In the case of (semi) interactive SL provers (such as Iris [JKJ+18], and VerCors [AH21,MRH22] that uses Viper [MSS16] as a back-end) we sought out expertise and collaborated in our search for a tool-supported proof of the above equivalence. Even after personally visiting the Iris team in Nijmegen (lead by Robbert Krebbers) and the VerCors team in Twente (lead by Marieke Huisman), we were unable to guide the tools to produce a proof of $p' \rightarrow p$. The problem here seems similar to that of [HT16], in that their semantics of separating connectives, which are formalized in terms of abstract monoids, are not compatible with the set-theoretic interpretation of the points-to relation.

In fact, the equivalence between the above two formulas can be expressed in quantifier-free separation logic, for which a complete axiomatization of all valid formulas has been given in [DLM21]. In the sequent calculus we can express the equivalence of p and p' in terms of the sequent $\mathit{fun}(R) \Rightarrow (p \leftrightarrow p')@R$. Here R is an arbitrary binary relation symbol used to represent the current interpretation of the points-to relation. We abbreviate $\forall x, y, z((R(x,y) \wedge R(x,z)) \rightarrow y = z)$ by $\mathit{fun}(R)$. A proof of the above sequent amounts to proving the sequents $\mathit{fun}(R), p'@R \Rightarrow p@R$ and $\mathit{fun}(R), p@R \Rightarrow p'@R$. Below we present a high-level proof of the first sequent, abstracting from some basic first-order reasoning in the calculus.

By an application of $\mathbf{L}_*$ to derive the sequent $\mathit{fun}(R), p'@R \Rightarrow p@R$ it suffices to derive

$$\mathit{fun}(R), R = R_1 \uplus R_2, (u \mapsto -)@R_1, (u \mapsto 0 \mathrel{-\!\!*} v \hookrightarrow z)@R_2 \Rightarrow p@R$$

for some fresh R_1 and R_2. Let $\psi(x,y)$ denote the binary formula $x = u \wedge y = 0$. Further, let Γ denote the set of formulas $\mathit{fun}(R), R = R_1 \uplus R_2, (u \mapsto -)@R_1$. By an application of the rule $\mathbf{L}_{-*}$ it then suffices to prove the following sequents (from $\Gamma \Rightarrow \Delta$ we can derive $\Gamma \Rightarrow A, \Delta$ by right-weakening). First we prove $\Gamma \Rightarrow R_2 \cap \psi = \emptyset$: By the points-to rules the rooted assertion $(u \mapsto -)@R_1$ (appearing in Γ) reduces to $\exists z(R_1(u,z) \wedge \forall x, y(R_1(x,y) \rightarrow x = u \wedge y = z))$ (the forall-part of the formula is due to the 'strict' points-to which states that the domain contains u as its only location). Further, $R_2 \cap \psi = \emptyset$ logically boils down to $\neg\exists x, y(R_2(x,y) \wedge (x = u \wedge y = 0))$, that is, $\neg R_2(u, 0)$, which in basic first-order logic follows from $\exists z R_1(u,z)$ and the assumptions $R = R_1 \uplus R_2$ and $\mathit{fun}(R)$.

Second, we prove $\Gamma \Rightarrow (u \mapsto 0)@\psi$: By the points-to rules $(u \mapsto 0)@\psi$ (using the expanded definition ϕ of $u \mapsto 0$ and the definition of the substitution $\phi[\psi/\hookrightarrow]$) reduces to $(u = u) \wedge (0 = 0) \wedge \forall x, y((x = u \wedge y = 0) \rightarrow (x = u \wedge y = 0))$ which is equivalent to **true**.

And, finally, we prove $\Gamma, (v \hookrightarrow z)@(R_2 \vee \psi) \Rightarrow p@R$: First note that (again, by the points-to rules)

$$((u \hookrightarrow -) \wedge (z = 0 \lhd u = v \rhd v \hookrightarrow z))@R$$

reduces to

$$(\exists z R(u, z)) \wedge (z = 0 \lhd u = v \rhd R(v, z)))$$

The assertion $\exists z R(u, z)$ clearly follows from the assumptions $R = R_1 \uplus R_2$ and $(u \mapsto -)@R_1$ in Γ. To prove $z = 0 \lhd u = v \rhd R(v, z)$, we first reduce the assumption $(v \hookrightarrow z)@(R_2 \vee \psi)$ to $R_2(v, z) \vee (v = u \wedge z = 0)$. Now, if $v = u$ then $\neg R_2(v, z)$, because of the assumptions $fun(R)$, $R = R_1 \uplus R_2$ and $(u \mapsto -)@R_1$. So we have that $z = 0$. Otherwise, we have $R_2(v, z)$, and thus $R(v, z)$, because $R = R_1 \uplus R_2$.

Soundness and Completeness. We denote by $\vdash \Gamma \Rightarrow \Delta$ that there exists a proof of the sequent $\Gamma \Rightarrow \Delta$. To define $\models \Gamma \Rightarrow \Delta$, let σ denote a substitution which assigns to every binary relation symbol R of the sequent $\Gamma \Rightarrow \Delta$ a binary formula ϕ. Such a substitution σ simply replaces occurrences of $R(t, t')$ by $\phi(t, t')$, where $\sigma(R) = \phi(x, y)$. By $\models \Gamma \Rightarrow \Delta$ we then denote that $M, s \models \bigwedge \Gamma\sigma$ (that is, $M, s \models A\sigma$, for every $A \in \Gamma$) implies $M, s \models \bigvee \Delta\sigma$ (that is, $M, s \models B\sigma$, for some $B \in \Delta$), for every M, s and every substitution σ.

In the soundness proof below we use these substitutions to instantiate the fresh binary relation symbols introduced in the rules $\mathbf{L}_*$ and $\mathbf{R}_{-*}$. Note that updating the interpretation of these symbols (as provided by M) would affect the semantics of the separating connectives if binary formulas would refer to these fresh binary relation symbols (note that they are only supposed not to appear in formulas of the conclusion of the rules $\mathbf{L}_*$ and $\mathbf{R}_{-*}$).

We generalize the above notions of derivability and validity to possibly infinite Γ: $\Gamma \vdash \Delta$ indicates that $\vdash \Gamma' \Rightarrow \Delta$, for some finite $\Gamma' \subseteq \Gamma$, and $\Gamma \models \Delta$ indicates that for every substitution σ we have that $M, s \models \Gamma\sigma$ (that is, $M, s \models A\sigma$, for every $A \in \Gamma$) implies $M, s \models B\sigma$, for some $B \in \Delta$.

Theorem 1 (Soundness). *We have that* $\vdash \Gamma \Rightarrow \Delta$ *implies* $\models \Gamma \Rightarrow \Delta$.

Proof. We prove that the rules for the separating connectives preserve validity. The points-to rules are sound because $M, Rel_M(\phi), s \models p$ if and only if $M, s \models p[\phi/\hookrightarrow]$, for any basic formula p (note that $p[\phi/\hookrightarrow]$ is a pure first-order formula which does not depend on the heap).

$\mathbf{L}_*$: Let $M, s \models \Gamma\sigma$ and $M, s \models (p\sigma * q\sigma)@\phi\sigma$. We have to show that $M, s \models \bigvee \Delta\sigma$. By Lemma 1, there exist ϕ_1 and ϕ_2 such that $M, s \models (\phi\sigma) = \phi_1 \uplus \phi_2$, $M, s \models p\sigma@\phi_1$, and $M, s \models q\sigma@\phi_2$. Let $\sigma' = \sigma[R_1, R_2 := \phi_1, \phi_2]$. Since R_1 and R_2 are fresh and as such do not appear in $\Gamma, (p * q)@\phi$, it follows that $M, s \models \Gamma'\sigma'$, where $\Gamma' = \Gamma, \phi = R_1 \uplus R_2, p@R_1, q@R_2$. By the validity of the premise we thus obtain that $M, s \models \bigvee \Delta\sigma'$. Since R_1 and R_2 also do not appear in Δ, we conclude that $M, s \models \bigvee \Delta\sigma$.

$\mathbf{R}_*$: Let $M, s \models \Gamma\sigma$ and suppose that $M, s \not\models \bigvee \Delta\sigma$. From the validity of the premises it then follows that $M, s \models \phi\sigma = (\phi_1 \uplus \phi_2)\sigma$, $M, s \models p\sigma@\phi_1\sigma$, and $M, s \models q\sigma@\phi_2\sigma$, By Lemma 1 we conclude $M, s \models (p\sigma * q\sigma)@\phi\sigma$.

$\mathbf{L}_{-\!*}$: Let $M, s \models \Gamma\sigma$ and $M, s \models (p\sigma \mathrel{-\!*} q\sigma)@\phi\sigma$, and suppose that $M, s \not\models \bigvee \Delta\sigma$. From the validity of the first two premises it then follows that $M, s \models \phi\sigma \perp \psi\sigma$ and $M, s \models p\sigma@\psi\sigma$. By Lemma 1 again, it follows that $M, s \models q\sigma@(\phi\sigma \vee \psi\sigma)$. By the validity of the third premise we thus derive that $M, s \not\models \bigvee \Delta\sigma$, which a contradicts our assumption.

$\mathbf{R}_{-\!*}$: Let $M, s \models \Gamma\sigma$ and suppose that $M, s \not\models \bigvee \Delta\sigma$. We have to show that $M, s \models (p\sigma \mathrel{-\!*} q\sigma)@\phi\sigma$. Let ψ be such that $M, s \models \psi \perp (\phi\sigma)$ and $M, s \models p\sigma@\psi$. Further, let R be a fresh variable and $\sigma' = s[R := \psi]$. It follows that $M, s \models \Gamma'\sigma'$, where $\Gamma' = \Gamma, R \perp \phi, p@R$ and $M, s \not\models \bigvee \Delta\sigma'$. And so we derive from the validity of the premise of the rule that $M, s \models q\sigma@(\phi\sigma \cup \psi)$. Since ψ was arbitrarily chosen, by Lemma 1 again we conclude that $M, s \models (p\sigma \mathrel{-\!*} q\sigma)@\phi\sigma$. $\square$

As a corollary we obtain that $\Gamma \vdash \Delta$ implies $\Gamma \models \Delta$.

Following the completeness proof of first-order logic as described in [Hen49], it suffices to show that every consistent set of formulas is satisfiable (the so-called 'model existence theorem'). A set of formulas Γ is consistent if $\Gamma \not\vdash \emptyset$. We first show that every consistent set of formulas can be extended to a maximal consistent set. To this end we assume an infinite set of 'fresh' binary relation symbols R that do not appear in Γ. We construct for any consistent set Γ a maximal consistent extension Γ^∞, assuming an enumeration of all formulas A (which also covers all first-order formulas). We define $\Gamma_0 = \Gamma$ and Γ_{n+1} satisfies the general rule: if $\Gamma_n, A_n \not\vdash \emptyset$ then $\Gamma_n \cup \{A_n\} \subseteq \Gamma_{n+1}$, otherwise $\Gamma_{n+1} = \Gamma_n$. Additionally, in case A_n is added and A_n is of the form $\exists x A$ or a rooted assertion $(p * q)@\phi$ or $\neg(p \mathrel{-\!*} q)@\phi$, we also include corresponding *witnesses* in Γ_{n+1}:

- If A_n is of the form $\exists x A$ we additionally add $A(y)$, where $A(y)$ results from replacing all free occurrences of x in A by the fresh variable y which does not appear in Γ_n.
  Note that $A(y)$ can indeed be added consistently because from $\Gamma_n, A(y) \vdash \emptyset$ we would derive $\Gamma_n, \exists x A \vdash \emptyset$, which contradicts the assumption that $\Gamma_n, \exists x A \not\vdash \emptyset$.
- If A_n is of the form $(p * q)@\phi$ we additionally add the formulas $\phi = R_1 \uplus R_2, R_1 \perp R_2, p@R_1$, and $q@R_2$, where R_1 and R_2 are fresh (e.g., not appearing in Γ_n).
  Note that these formulas can indeed be added consistently because from $\Gamma_n, \phi = R_1 \uplus R_2, R_1 \perp R_2, p@R_1, q@R_2 \vdash \emptyset$ we would derive $\Gamma_n, (p * q)@\phi \vdash \emptyset$ (by rule $\mathbf{L}_*$).
- If A_n is of the form $\neg(p \mathrel{-\!*} q)@\phi$ (which is equivalent to $\neg((p \mathrel{-\!*} q)@\phi)$) we additionally add the formulas $R \perp \phi, p@R(x, y)$, and $\neg q@(\phi \vee R)$, where R is fresh (e.g., not appearing in Γ_n).
  Note that these formulas can indeed be added consistently because from $\Gamma_n, R \perp \phi, p@R(x, y), \neg q@(\phi \vee R) \vdash \emptyset$ we would derive $\Gamma_n \vdash (p \mathrel{-\!*} q)@\phi$ (by rule $\mathbf{R}_{-\!*}$), which contradicts the assumption that $\Gamma_n, \neg(p \mathrel{-\!*} q)@\phi \not\vdash \emptyset$.

We define $\Gamma^\infty = \bigcup_n \Gamma_n$. By construction Γ^∞ is maximal consistent. Given a maximal consistent set of formulas Γ, let $M_\Gamma = (D, I)$, where D is the set of equivalences classes $[t] = \{t' \mid t = t' \in \Gamma\}$. For any function symbol f and relation symbol R (excluding the points-to relation $\hookrightarrow$) we define

- $I(f)([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$,
- $I(R)([t_1], \ldots, [t_n]) = \textbf{true}$ if and only if $R(t_1, \ldots, t_n) \in \Gamma$.

The above interpretation of the function and relational symbols is well-defined because its definition does not depend on the choice of the representatives (this follows from the equality axioms).

Given a maximal consistent set of formulas Γ and the model $M_\Gamma = (D, I)$, a corresponding valuation s assigns to every variable x an equivalence class $[t]$. However, in the sequel we will represent such a valuation by a *substitution* s which simply assigns to each variable a term. The value $I_s(x)$ of a variable x then is given by the equivalence class $[s(x)]$ of the term $s(x)$.

Given a substitution s, for any term t and formula A (of the sequent calculus) we denote by ts and As the result of replacing every free occurrence of a (first-order) variable x in t and A by $s(x)$. Note that $(p@\phi)s = ps@\phi$, because the meaning of $p@\phi$ does not depend on the free variables x and y of the binary formula $\phi(x, y)$.

Given a maximal consistent set of formulas Γ and the model $M_\Gamma = (D, I)$, it follows that $I_s(t) = [ts]$, for every term t and substitution s.

Lemma 2. *Given a maximal consistent set of formulas Γ and the model $M_\Gamma = (D, I)$, we have $M, s \models A$ if and only if $As \in \Gamma$, for every formula A and substitution s.*

Proof. The proof proceeds by induction on the following well-founded ordering $A < B$ on formulas of the sequent calculus: Let $\#A = (n, m)$, where n denotes the number of occurrences of the separating connectives and the @-binding operator of A and m denotes the number of occurrences of the (standard) first-order logical operations of A. Then $A < B$ if $\#A < \#B$, where the latter denotes the lexicographical ordering on $\mathbb{N} \times \mathbb{N}$ (w.r.t. the standard 'smaller than' ordering on the natural numbers). We treat the following main cases (for notational convenience M denotes the model M_Γ).

- Let $M, s \models A$, where A denotes the formula $(p * q)@\phi$. By Lemma 1 there exist ϕ_1 and ϕ_2 such that $M, s \models \phi = \phi_1 \uplus \phi_2$, $M, s \models p@\phi_1$ and $M, s \models q@\phi_2$. From the induction hypothesis it follows that $ps@\phi_1, qs@\phi_2, \phi = \phi_1 \uplus \phi_2 \in \Gamma$ (note that the first-order formula $\phi = \phi_1 \uplus \phi_2$ does not contain free variables, and thus is not affected by the substitution s). So we derive by rule $\mathbf{R}_*$ that $\Gamma \vdash (ps * qs)@\phi$. By maximal consistency of Γ, we then conclude that $(ps * qs)@\phi \in \Gamma$, that is, $As \in \Gamma$.
  On the other hand, let $As \in \Gamma$. That is, $(ps * qs)@\phi \in \Gamma$. By construction $\phi = R_1 \uplus R_2, ps@R_1, qs@R_2 \in \Gamma$, for some witnesses R_1 and R_2. By the induction hypothesis it then follows that $M, s \models p@R_1$ and $M, s \models p@R_2$. Further, the induction hypothesis gives $M, s \models \phi = R_1 \uplus R_2$ (again, note that the formula $\phi = R_1 \uplus R_2$ has no free variables, and thus is not affected by the substitution s). We conclude by Lemma 1 that $M, s \models (p * q)@\phi$.
- Let $M, s \models A$, where A denotes the formula $(p \mathrel{-\!\!*} q)@\phi$. Suppose $As \notin \Gamma$. By the maximal consistency of Γ, we then have $\neg(ps \mathrel{-\!\!*} qs)@\phi \in \Gamma$. By

construction $R \perp \phi, ps@R, \neg qs@(\phi \vee R) \in \Gamma$, for some witness R, which contradicts $M, s \models (p \mathrel{-\!\!*} q)@\phi$ (after application of the induction hypothesis and using Lemma 1 again).

On the other hand, let $As \in \Gamma$. To show that $M, s \models (p \mathrel{-\!\!*} q)@\phi$, let $M, s \models \phi \perp \psi$ and $M, s \models p@\psi$, for some binary formula ψ. By the induction hypothesis we have that $\phi \perp \psi, ps@\psi \in \Gamma$. Suppose that $qs@(\phi \vee \psi) \notin \Gamma$, that is $\neg qs@(\phi \vee \psi) \in \Gamma$ (Γ is maximal consistent), and thus $\Gamma, qs@(\phi \vee \psi) \vdash \emptyset$. Applying rule $\mathbf{L}_{-*}$ we then derive $\Gamma, (ps \mathrel{-\!\!*} qs)@\phi \vdash \emptyset$, which contradicts the consistency of Γ ($(ps \mathrel{-\!\!*} qs)@\phi \in \Gamma$). So we have that $qs@(\phi \vee \psi) \in \Gamma$, that is, $M, s \models q@(\phi \vee \psi)$, by the induction hypothesis. Since ψ is chosen arbitrarily, it follows by Lemma 1 that $M, s \models (p \mathrel{-\!\!*} q)@\phi$.

- Let A be a formula $p@\phi$, where p denotes a basic formula. Let $\mathcal{R} = Rel_M(\phi)$. We then have $M, s \models p@\phi$ iff (by definition)
  $M, \mathcal{R}, s \models p$ iff (straightforward induction on p)
  $M, s \models p[\phi/\hookrightarrow]$ iff (induction hypothesis for $p[\phi/\hookrightarrow]$)
  $ps[\phi/\hookrightarrow] \in \Gamma$ iff (by the points-to rules)
  $ps@\phi \in \Gamma$. Note that applying the substitution s to $p@\phi$ and $p[\phi/\hookrightarrow]$ results in $ps@\phi$ and $ps[\phi/\hookrightarrow]$. □

The downward Löwenheim-Skolem property follows. It should be noted that we cannot remove from the constructed model the binary relation symbols which are introduced as witnesses, as these determine the notion of first-order definability.

Theorem 2 (Completeness). *We have that $\Gamma \models \Delta$ implies $\Gamma \vdash \Delta$.*

Compactness follows. We thus derive (by Lindström's theorem [Vää10]) that this version of SL is as expressive as first-order logic.

6 Conclusion

We investigated the expressiveness of full SL over arbitrary first-order models. We have shown that restricting the quantification over first-order definable heaps gives rise to a semantic consequence relation that can be captured by a sound and complete extension of the standard sequent calculus for first-order logic.

The main question remains what is the exact relationship between full SL which allows for infinite heaps and second-order logic. In [KR04] a translation is given of general second-order logic in a first-order logic with *spatial conjunction*. Spatial conjunction (as defined in [KR04]) allows to split a global set of *arbitrary* relations. As such it goes beyond the *local* scope of separating conjunction which is restricted to the points-to relation. We conjecture that second-order logic is strictly more expressive than full SL.

Acknowledgements. The authors thank the anonymous referees for providing many constructive and useful suggestions for improvement.

References

[AH21] Armborst, L., Huisman, M.: Permission-based verification of red-black trees and their merging. In: 2021 IEEE/ACM 9th International Conference on Formal Methods in Software Engineering (FormaliSE), pp. 111–123. IEEE (2021)

[BDL12] Brochenin, R., Demri, S., Lozes, E.: On the almighty wand. Inf. Comput. **211**, 106–137 (2012)

[CK13] Chang, C.C., Keisler, H.J.: Model Theory: Third Edition. Dover Books on Mathematics. Dover Publications (2013)

[Cro17] Crosilla, L.: Predicativity and Feferman. In: Jäger, G., Sieg, W. (eds.) Feferman on Foundations: Logic, Mathematics, Philosophy. OCL, vol. 13, pp. 423–447. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63334-3_15

[CYO01] Calcagno, C., Yang, H., O'Hearn, P.W.: Computability and complexity results for a spatial assertion language for data structures. In: Hariharan, R., Vinay, V., Mukund, M. (eds.) FSTTCS 2001. LNCS, vol. 2245, pp. 108–119. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45294-X_10

[dBHdG23] de Boer, F., Hiep, H.-D., de Gouw, S.: Dynamic separation logic. In: Mathematical Foundations of Programming Semantics (MFPS) (2023, to appear)

[DD16] Demri, S., Deters, M.: Expressive completeness of separation logic with two variables and no separating conjunction. ACM Trans. Comput. Log. **17**(2), 12 (2016)

[DLM21] Demri, S., Lozes, É., Mansutti, A.: A complete axiomatisation for quantifier-free separation logic. Log. Methods Comput. Sci. **17**(3) (2021)

[EIP20] Echenim, M., Iosif, R., Peltier, N.: The Bernays-Schönfinkel-Ramsey class of separation logic with uninterpreted predicates. ACM Trans. Comput. Log. **21**(3), 19:1–19:46 (2020)

[GM10] Galmiche, D., Méry, D.: Tableaux and resource graphs for separation logic. J. Log. Comput. **20**(1), 189–231 (2010)

[Hen49] Henkin, L.: The completeness of the first-order functional calculus. J. Symb. Log. **14**(3), 159–166 (1949)

[Hen50] Henkin, L.: Completeness in the theory of types. J. Symb. Logic **15**(2), 81–91 (1950)

[HH14] Huet, G.P., Herbelin, H.: 30 years of research and development around Coq. In: Jagannathan, S., Sewell, P. (eds.) The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2014, San Diego, CA, USA, 20–21 January 2014, pp. 249–250. ACM (2014)

[HT16] Hóu, Z., Tiu, A.: Completeness for a first-order abstract separation logic. In: Igarashi, A. (ed.) APLAS 2016. LNCS, vol. 10017, pp. 444–463. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47958-3_23

[JKJ+18] Jung, R., Krebbers, R., Jourdan, J.-H., Bizjak, A., Birkedal, L., Dreyer, D.: Iris from the ground up: a modular foundation for higher-order concurrent separation logic. J. Funct. Program. **28** (2018)

[KR04] Kuncak, V., Rinard, M.C.: On spatial conjunction as second-order logic. CoRR, cs.LO/0410073 (2004)

[Kri08] Krishnaswami, N.R.: A modal sequent calculus for propositional separation logic (2008)

[Man96] Manzano, M.: Extensions of First-Order Logic, vol. 19. Cambridge University Press, Cambridge (1996)

[MRH22] Monti, R.E., Rubbens, R., Huisman, M.: On deductive verification of an industrial concurrent software component with VerCors. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 517–534. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19849-6_29

[MSS16] Müller, P., Schwerhoff, M., Summers, A.J.: Viper: a verification infrastructure for permission-based reasoning. In: Jobstmann, B., Leino, K.R.M. (eds.) VMCAI 2016. LNCS, vol. 9583, pp. 41–62. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49122-5_2

[Pym02] Pym, D.J.: The semantics and proof theory of the logic of bunched implications. In: Applied Logic Series (2002)

[Rey00] Reynolds, J.C.: Intuitionistic reasoning about shared mutable data structure. In: Davies, J., Roscoe, B., Woodcock, J. (eds.) Millennial Perspectives in Computer Science, Cornerstones of Computing, pp. 303–321. Macmillan Education (2000)

[Rey02] Reynolds, J.C.: Separation logic: a logic for shared mutable data structures. In: Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS 2002), Copenhagen, Denmark, 22–25 July 2002, pp. 55–74. IEEE Computer Society (2002)

[Rey05] Reynolds, J.C.: An overview of separation logic. In: Meyer, B., Woodcock, J. (eds.) VSTTE 2005. LNCS, vol. 4171, pp. 460–469. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69149-5_49

[RISK16] Reynolds, A., Iosif, R., Serban, C., King, T.: A decision procedure for separation logic in SMT. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 244–261. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_16

[SNPR+19] Sighireanu, M., et al.: SL-COMP: competition of solvers for separation logic. In: Beyer, D., Huisman, M., Kordon, F., Steffen, B. (eds.) TACAS 2019. LNCS, vol. 11429, pp. 116–132. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17502-3_8

[Vää01] Väänänen, J.: Second-order logic and foundations of mathematics. Bull. Symb. Logic **7**(4), 504–520 (2001)

[Vää10] Väänänen, J.: Lindström's theorem. Universal Logic: An Anthology, pp. 231–236 (2010)

[Yan01] Yang, H.: Local reasoning for stateful programs. Ph.D. thesis, University of Illinois at Urbana-Champaign. (Technical Report UIUCDCS-R-2001-2227) (2001)

Testing the Satisfiability of Formulas in Separation Logic with Permissions

Nicolas Peltier(✉)

Université Grenoble Alpes, LIG, CNRS, Inria, Grenoble INP,
38000 Grenoble, France
nicolas.peltier@imag.fr

Abstract. We investigate the satisfiability problem for a fragment of Separation Logic (SL) with inductively defined spatial predicates and permissions. We show that the problem is undecidable in general, but decidable under some restrictions on the rules defining the semantics of the spatial predicates. Furthermore, if the satisfiability of permission formulas can be tested in exponential time for the considered permission model then SL satisfiability is Exptime complete.

1 Introduction

Separation Logic [14,22] (SL) is a dialect of bunched logic [18] that is widely used in verification for reasoning on programs manipulating pointer-based data structures. It constitutes the theoretical basis of several industrial scale automated static program analyzers [1,2,7]. SL formulas describe *heaps*, with atoms asserting that some location (i.e., a memory address) is allocated and refers to some tuple of locations (i.e., a record), combined with a special connective $*$, called *separating conjunction*, which is used to compose heaps. Custom data structures may be described in this setting by using spatial predicates, the semantics of which is defined using *inductive rules*, similar to those used for defining recursive structures in usual programming languages. Such rules allow one to describe heaps of unbounded size with some particular structure such as lists or trees. In this setting, existing work usually focuses on the fragment of SL called *symbolic heaps* (defined as separating conjunctions of SL atoms).

Usually, SL formulas are interpreted in the *standard heap model*, where heaps are defined as partial finite functions mapping locations to tuples of locations and where the separating conjunction $*$ is interpreted as the disjoint union of heaps. Both the satisfiability and entailment problems have been extensively investigated for this heap model. It was proven that the satisfiability problem is Exptime complete [6], whereas the entailment problem is undecidable in general, and 2-Exptime complete provided the inductive rules meet some syntactic conditions [11–13,15] which are general enough to capture usual data structures used in programming. The combination of spatial reasoning with theory reasoning has also been thoroughly investigated, see for instance [16,19–21,23]).

However, richer models exist (see for instance [8]) accounting for additional features of dynamic memory. The automation of reasoning in these models received little attention. One such model that is of practical relevance is *separation logic with permissions* [3,5], where allocated locations are associated with so called *permissions* used to model the ownership of a given heap region (e.g., a process may have `read` or `write` permission over some location). The heap composition operator that is used to define the interpretation of the separating conjunction is more complex in this framework than in the above case: non disjoint heaps can be combined if they agree on all the locations on which they are both defined and if the corresponding permissions can be combined (for instance it is natural to assume that `read` permissions can be freely combined but not `write` permissions). The framework is thus parameterized by some *permission model* describing which permissions are available and how they can be combined. In [10] algorithms are provided to decide the satisfiability and entailment problems for SL formulas (symbolic heaps) with permissions in the case of lists, i.e., when all allocated locations refer to a single location (i.e., to a record of size 1) and when there is only one spatial predicate $\mathtt{lseg}_p(x, y)$ denoting a list segment from x to y, with permission p. The provided algorithms are generic w.r.t. the permission model, and it is proven that these problems are in NP and co-NP, respectively, assuming that some oracle exists for testing the satisfiability of permission formulas in the considered model.

In the present paper, we investigate the satisfiability problem for SL formulas with permission defined over arbitrary spatial predicates, with user-defined inductive rules. The goal is to allow for more genericity by tackling custom data structures (such as trees, cyclic lists, doubly linked lists etc.) with arbitrary permissions. The addition of permissions makes satisfiability testing much more difficult: we prove that the problem is undecidable in general, and we devise syntactic conditions on the inductive rules for which the problem is EXPTIME-complete. The restrictions are similar – but stronger – to those given in [13] to ensure the decidability of the entailment problem in the standard heap model. In particular, the inductive rules defining the predicate `lseg` mentioned above fulfill these restrictions[1], as well as other usual data structures such as cyclic list, trees etc. (however, doubly linked lists or trees with parent links are not captured). The considered inductive rules use a special connective $\circ$ (different from $*$) that is interpreted as a disjoint union. As we shall see, this is both more natural for defining data structures (see also [5]) and required for deciding satisfiability.

2 Definitions

Syntax. We first briefly review some basic notations. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are finite sequences, then we denote by $\boldsymbol{x}.\boldsymbol{y}$ the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$. We denote by $|\boldsymbol{x}|$ the length of $\boldsymbol{x}$ and by $\boldsymbol{x}|_i$ its i-th element (if $1 \leq i \leq |\boldsymbol{x}|$). If $E \subseteq \{1, \dots, |\boldsymbol{x}|\}$ then $\boldsymbol{x}|_E$ denotes the set $\{\boldsymbol{x}|_i \mid i \in E\}$. With a slight abuse of notations, a finite

[1] provided the considered lists are not empty.

sequence $\boldsymbol{x}$ is sometimes identified with the set $\{\boldsymbol{x}|_i \mid i = 1, \ldots, |\boldsymbol{x}|\}$, for instance, we may write $x \in (\boldsymbol{u} \cup \boldsymbol{v}) \setminus \boldsymbol{w}$ to state that x occurs in $\boldsymbol{u}$ or $\boldsymbol{v}$ but not in $\boldsymbol{w}$.

We consider a multisorted framework, with two sorts $\mathtt{l}$ (for locations) and $\mathtt{p}$ (for permissions). Let $\mathcal{V}_{\mathtt{l}}$ and $\mathcal{V}_{\mathtt{p}}$ be two countably infinite disjoint sets of *variables* with $\mathcal{V} \overset{\text{def}}{=} \mathcal{V}_{\mathtt{l}} \cup \mathcal{V}_{\mathtt{p}}$, where $\mathcal{V}_{\mathtt{l}}$ and $\mathcal{V}_{\mathtt{p}}$ denote location variables and permission variables, respectively. The set of *permission terms* $\mathcal{T}_{\mathtt{p}}$ denotes the set of terms built inductively as usual on the set of variables $\mathcal{V}_{\mathtt{p}}$ and the binary function $\oplus$ (written in infix notation). A *points-to atom* is an expression of the form $x \overset{p}{\mapsto} (y_1, \ldots, y_k)$ with $x, y_1, \ldots, y_k \in \mathcal{V}_{\mathtt{l}}$ and $p \in \mathcal{T}_{\mathtt{p}}$. An *equational atom* is an expression of the form $x \simeq y$ or $x \not\simeq y$ with either $x, y \in \mathcal{V}_{\mathtt{l}}$ or $x, y \in \mathcal{T}_{\mathtt{p}}$.

We consider two disjoint sets of predicate symbols $\mathcal{P}_{\mathtt{p}}$ and $\mathcal{P}$. The set $\mathcal{P}_{\mathtt{p}}$ denotes *permission predicates*, where each predicate $\hat{P} \in \mathcal{P}_{\mathtt{p}}$ is associated with a unique arity $\#(\hat{P})$. A *permission atom* is an expression of the form $\hat{P}(p_1, \ldots, p_n)$, $\hat{P} \in \mathcal{P}_{\mathtt{p}}$, $n = \#(\hat{P})$ and $p_1, \ldots, p_n \in \mathcal{T}_{\mathtt{p}}$. $\mathcal{P}$ is a finite set of *spatial predicate symbols*. Each symbol $P \in \mathcal{P}$ is associated with a *spatial arity* $\#_{\mathtt{l}}(P) \in \mathbb{N}$ and with an *arity* $\#(P) \in \mathbb{N}$, with $\#(P) > \#_{\mathtt{l}}(P) > 0$ ($\#_{\mathtt{l}}(P)$ and $\#(P) - \#_{\mathtt{l}}(P)$ denote the number of arguments of P that are of sort $\mathtt{l}$ and $\mathtt{p}$, respectively). A *predicate atom* is an expression of the form $P(x_1, \ldots, x_n, p_1, \ldots, p_m)$, with $n = \#_{\mathtt{l}}(P)$, $n + m = \#(P)$, $x_1, \ldots, x_n \in \mathcal{V}_{\mathtt{l}}$ and $p_1, \ldots, p_m \in \mathcal{T}_{\mathtt{p}}$. A *spatial atom* is either a points-to atom or a predicate atom.

The set of *formulas* is built inductively as usual on the logical constants $\mathtt{emp}$, and $\bot$ and on the set of spatial, equational and permission atoms, using the special connectives $*$ and $\circ$ and existential quantification on variables of sort $\mathtt{l}$ only (existential quantification over variables of type $\mathtt{p}$ is not allowed). The connective $*$ is usually called *separating conjunction*, and we call $\circ$ the *disjoint conjunction* (it is intended to capture the disjoint union of heaps[2]). Formulas are taken up to associativity and commutativity of the symbols $*$ and $\circ$, up to the commutativity of $\simeq, \not\simeq$ and up to prenex form. We denote by $|\phi|$ the size of ϕ. For technical convenience, we assume that the symbols $\circ$ and $*$ have weight of 1 and 2, respectively, and that all atoms have size 1. For conciseness, a formula $\exists x_1 \ldots \exists x_n\, \phi$ will often be written $\exists \boldsymbol{x}\, \phi$, with $\boldsymbol{x} = (x_1, \ldots, x_n)$. A *permission formula* is a formula containing no spatial atoms and no equational atom of the form $x \simeq y$ or $x \not\simeq y$ with $x, y \in \mathcal{V}_{\mathtt{l}}$ (note that $\mathtt{emp}$ is a permission formula). A formula is *spatial* if all the atoms occurring in it are spatial. A *pure formula* is a formula that contains no spatial atom (it is not necessarily a permission formula, as it may contain equations or disequations between locations) A *symbolic heap* is a formula containing no occurrence of $\circ$, and a $\circ$-*formula* is a formula containing no occurrence of $*$.

A variable x is *free* in a formula ϕ if it occurs in ϕ outside of the scope of any quantifier binding x. The set of variables (freely) occurring in a term (or formula) ϕ is denoted by $fv(\phi)$. A *substitution* is a function mapping every variable in $\mathcal{V}_{\mathtt{l}}$ to a variable in $\mathcal{V}_{\mathtt{l}}$ and every variable in $\mathcal{V}_{\mathtt{p}}$ to a term in $\mathcal{T}_{\mathtt{p}}$.

[2] The connective $\circ$ is called *strong separating conjunction* in [5] and written $*$ (whereas $*$ is written $\circledast$). Our notations are mostly consistent with those in [10].

The *domain* of a substitution σ (denoted by $dom(\sigma)$) is the set of variables x such that $\sigma(x) \neq x$. A substitution of domain $\{x_1, \ldots, x_n\}$ with $\sigma(x_i) = t_i$ is denoted by $\{x_i \leftarrow t_i \mid i = 1, \ldots, n\}$, or $\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}$, with $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{t} = (t_1, \ldots, t_n)$. For all formulas or terms ϕ, we denote by $\phi\sigma$ the formula or term obtained from ϕ by replacing every free occurrence of a variable x by $\sigma(x)$.

Semantics. Permissions are interpreted in some permission model:

Definition 1 (Adapted from [10]). *A* permission model $\mathfrak{P}$ *is a triple*

$$(\mathcal{P}_{\mathfrak{P}}, \oplus_{\mathfrak{P}}, (\hat{P}_{\mathfrak{P}})_{\hat{P} \in \mathcal{P}_{\mathtt{p}}})$$

where $\mathcal{P}_{\mathfrak{P}}$ is a non empty set, called the set of permissions, *$\oplus_{\mathfrak{P}} : \mathcal{P}_{\mathfrak{P}}^2 \to \mathcal{P}_{\mathfrak{P}}$ is a binary partial function that is commutative, associative and cancellative, and $\hat{P}_{\mathfrak{P}} \subseteq \mathcal{P}_{\mathfrak{P}}^{\#(\hat{P})}$, for all $\hat{P} \in \mathcal{P}_{\mathtt{p}}$. If $\pi, \pi' \in \mathcal{P}_{\mathfrak{P}}$, we write $\pi \leq_{\mathfrak{P}} \pi'$ if $\pi = \pi' \vee (\exists \pi'' \in \mathcal{P}_{\mathfrak{P}}\ \pi' = \pi \oplus \pi'')$.*

In what follows, $\mathfrak{P}$ always denotes a permission model. If $\pi \in \mathcal{P}_{\mathfrak{P}}$ and $n \in \mathbb{N}$, we denote by π^n the permission $\pi \oplus_{\mathfrak{P}} \ldots \oplus_{\mathfrak{P}} \pi$ (n times), note that π^n is not necessarily defined and implicitly depends on the considered permission model, which will always be clear from the context. In contrast to [10], we do not assume that a maximal "total" permission $1_{\mathfrak{P}}$ exists, we allow instead for arbitrary predicates over permissions (the total permission can be encoded as a unary predicate symbol T, with $T_{\mathfrak{P}} = \{1_{\mathfrak{P}}\}$).

Example 2. Assume that $\mathcal{P}_{\mathtt{p}} = \emptyset$. A simple example of permission model is $\mathfrak{w} = (\{\mathtt{read}, \mathtt{write}\}, \oplus_{\mathfrak{w}}, \emptyset)$, with $\mathtt{read} \oplus_{\mathfrak{w}} \mathtt{read} = \mathtt{read}$ and $\mathtt{write} \oplus_{\mathfrak{w}} \pi$ is undefined for all $\pi \in \{\mathtt{read}, \mathtt{write}\}$. Another example (from [4]) is $\mathfrak{f} = (]0,1], \oplus_{\mathfrak{f}}, \emptyset)$ where $]0,1]$ denotes the interval of rational numbers, with $\pi \oplus_{\mathfrak{f}} \pi' = \pi + \pi'$ if $\pi + \pi' \leq 1$ and $\pi \oplus_{\mathfrak{f}} \pi'$ is undefined otherwise ($\mathfrak{f}$ stands for *fractional*).

Let $\mathcal{L}$ be a countably infinite set of *locations*. A *store* (for a given permission model $\mathfrak{P}$) is a total mapping associating every variable in $\mathcal{V}_1$ to an element of $\mathcal{L}$ and every variable in $\mathcal{V}_{\mathtt{p}}$ to an element of $\mathcal{P}_{\mathfrak{P}}$. A store can be extended into a partial mapping from $\mathcal{T}_{\mathtt{p}}$ to $\mathcal{P}_{\mathfrak{P}}$ inductively defined as follows: $\mathfrak{s}(p_1 \oplus p_2) \stackrel{\text{def}}{=} \mathfrak{s}(p_1) \oplus_{\mathfrak{P}} \mathfrak{s}(p_2)$. Note that the obtained mapping is partial since $\mathfrak{s}(p_1) \oplus_{\mathfrak{P}} \mathfrak{s}(p_2)$ is not always defined. If $x_1, \ldots, x_n$ are pairwise distinct variables in $\mathcal{V}_1$ and $\ell_1, \ldots, \ell_n \in \mathcal{L}$, we denote by $\mathfrak{s}\{x_i \leftarrow \ell_i \mid i = 1, \ldots, n\}$ the store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on every variable not occurring in $\{x_1, \ldots, x_n\}$ and such that $\mathfrak{s}'(x_i) = \ell_i$ for all $i = 1, \ldots, n$.

A *heap* (for a given permission model $\mathfrak{P}$) is a partial finite function from $\mathcal{L}$ to $\mathcal{L}^* \times \mathcal{P}_{\mathfrak{P}}$. The domain of a heap $\mathfrak{h}$ is denoted by $dom(\mathfrak{h})$, and we denote by $|\mathfrak{h}|$ the finite cardinality of $dom(\mathfrak{h})$. A heap of domain $\ell_1, \ldots, \ell_n$ such that $\mathfrak{h}(\ell_i) = (\ell_1^i, \ldots, \ell_{k_i}^i, \pi_i)$ (for all $i \in \{1, \ldots, n\}$) will be denoted as a set $\{(\ell_i, \ell_1^i, \ldots, \ell_{k_i}^i, \pi_i) \mid i = 1, \ldots, n\}$. For every heap $\mathfrak{h}$ we denote by $loc(\mathfrak{h})$ the set $\{\ell_i \mid \ell_0 \in dom(\mathfrak{h}), \mathfrak{h}(\ell_0) = (\ell_1, \ldots, \ell_k, \pi), 0 \leq i \leq k\}$. A heap may be viewed as a directed (labeled) graph: the locations in $loc(\mathfrak{h})$ are the vertices of the graph and there is a edge from ℓ to ℓ' if $\mathfrak{h}(\ell) = (\ell_1, \dots, \ell_n, \pi)$ and $\ell' = \ell_i$ for some $i \in \{1, \dots, n\}$.

A *subheap* of $\mathfrak{h}$ is any heap $\mathfrak{h}'$ such that $dom(\mathfrak{h}') \subseteq dom(\mathfrak{h})$ and $\mathfrak{h}'(\ell) = \mathfrak{h}(\ell)$ for all $\ell \in dom(\mathfrak{h}')$. A $\mathtt{p}$-*weakening* of $\mathfrak{h}$ (w.r.t. some permission model $\mathfrak{P}$) is any heap $\mathfrak{h}'$ such that $dom(\mathfrak{h}') = dom(\mathfrak{h})$ and for all $\ell \in dom(\mathfrak{h})$, if $\mathfrak{h}(\ell) = (\ell_1, \dots, \ell_n, \pi)$ then $\mathfrak{h}'(\ell) = (\ell_1, \dots, \ell_n, \pi')$ with $\pi' \leq_{\mathfrak{P}} \pi$. We write $\mathfrak{h}' \leq_{\mathtt{1}} \mathfrak{h}$ (resp. $\mathfrak{h}' \leq_{\mathtt{p}} \mathfrak{h}$) if $\mathfrak{h}'$ is a subheap (resp. a $\mathtt{p}$-weakening) of $\mathfrak{h}$. The relation $\leq$ denotes the composition of $\leq_{\mathtt{1}}$ and $\leq_{\mathtt{p}}$. We write $\mathfrak{h} \sim \mathfrak{h}'$ if $\mathfrak{h}$ and $\mathfrak{h}'$ only differ by the permissions, i.e., $dom(\mathfrak{h}) = dom(\mathfrak{h}')$ and for all $\ell \in dom(\mathfrak{h})$, if $\mathfrak{h}'(\ell) = (\ell_1, \dots, \ell_n, \pi')$ then there exists π such that $\mathfrak{h}(\ell) = (\ell_1, \dots, \ell_n, \pi)$.

Example 3. Consider the permission model $\mathfrak{f}$ defined in Example 2 with $\mathcal{L} = \mathbb{N}$. Then

$$\begin{aligned} \mathfrak{h}_0 &= \{(0,0,1,0.1),(1,0,0,0.2)\}, \quad \mathfrak{h}_1 = \{(0,0,1,0.1)\}, \\ \mathfrak{h}_2 &= \{(0,0,1,0.1),(1,0,0,0.1)\} \quad \mathfrak{h}_3 = \{(1,0,0,0.1)\} \end{aligned}$$

are heaps, and we have, e.g., $\mathfrak{h}_0(0) = (0,1,0.1)$ (meaning that the location 0 is allocated and refers to $(0,1)$, with permission 0.1), $\mathfrak{h}_1 \leq_{\mathtt{1}} \mathfrak{h}_0$, $\mathfrak{h}_2 \leq_{\mathtt{p}} \mathfrak{h}_0$, $\mathfrak{h}_3 \leq_{\mathtt{1}} \mathfrak{h}_2$, and $\mathfrak{h}_3 \leq \mathfrak{h}_0$. Moreover, $\mathfrak{h}_0 \sim \mathfrak{h}_2$.

Heaps can be composed using the following partial operator. If $\mathfrak{h}_1, \mathfrak{h}_2$ are heaps, then $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ is defined iff for all $\ell \in dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2)$, we have $\mathfrak{h}_i(\ell) = (\ell_1^i, \dots, \ell_{k_i}^i, \pi_i)$ (for all $i = 1, 2$) where $k_1 = k_2$, $\ell_j^1 = \ell_j^2$ for all $j \in \{1, \dots, k_1\}$ and $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is defined. Then $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ is defined as follows: if $\ell \in dom(\mathfrak{h}_i) \setminus dom(\mathfrak{h}_j)$ with $(i,j) \in \{(1,2),(2,1)\}$ then $(\mathfrak{h}_1 \sqcup \mathfrak{h}_2)(\ell) \overset{\text{def}}{=} \mathfrak{h}_i(\ell)$, and if $\ell \in dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2)$ then $(\mathfrak{h}_1 \sqcup \mathfrak{h}_2)(\ell) \overset{\text{def}}{=} (\ell_1^1, \dots, \ell_{k_1}^1, \pi_1 \oplus_{\mathfrak{P}} \pi_2)$.

Example 4. Consider the permission model $\mathfrak{f}$ defined in Example 2, with $\mathcal{L} = \mathbb{N}$ and the following heaps:

$$\begin{aligned} \mathfrak{h}_0 &= \{(0,0,0.5),(1,0,0.6)\} \quad \mathfrak{h}_1 = \{(0,0,0.5),(1,0,0.2),(2,0.1)\} \\ \mathfrak{h}_2 &= \{(0,0,0.5),(1,0,0.6)\} \quad \mathfrak{h}_3 = \{(0,0,0.1),(1,0.1)\} \end{aligned}$$

Then $\mathfrak{h}_0 \sqcup \mathfrak{h}_1$ is defined, and we have: $\mathfrak{h}_0 \sqcup \mathfrak{h}_1 = \{(0,0,1),(1,0,0.8),(2,0.1)\}$. However, neither $\mathfrak{h}_0 \sqcup \mathfrak{h}_2$ nor $\mathfrak{h}_0 \sqcup \mathfrak{h}_3$ is defined (in the former case the permissions of location 1 cannot be combined (as $0.6 + 0.6 > 1$) and in the latter case the location 1 is associated with distinct tuples, (0) and $()$, respectively.

A *structure* (for a given permission model $\mathfrak{P}$) is a pair $(\mathfrak{s}, \mathfrak{h})$ where $\mathfrak{s}$ is a store and $\mathfrak{h}$ is a heap for $\mathfrak{P}$. It is *injective* if $\mathfrak{s}$ is injective. A location ℓ is *allocated* in a structure $(\mathfrak{s}, \mathfrak{h})$ or in a heap $\mathfrak{h}$ if $\ell \in dom(\mathfrak{h})$, and a variable x is *allocated* in $(\mathfrak{s}, \mathfrak{h})$ if $\mathfrak{s}(x) \in dom(\mathfrak{h})$.

The semantics of spatial predicate is defined by inductive rules. A *set of inductive definitions* (SID) is a set of *rules* of the form $P(x_1, \dots, x_n, y_1, \dots, y_m) \Leftarrow \phi$ where $n = \#_{\mathtt{1}}(P)$, $n + m = \#(P)$, $x_1, \dots, x_n$ are pairwise distinct variables in $\mathcal{V}_{\mathtt{1}}$, $y_1, \dots, y_m$ are pairwise distinct variables in $\mathcal{V}_{\mathtt{p}}$, and ϕ is a formula such that $fv(\phi) \subseteq \{x_1, \dots, x_n, y_1, \dots, y_m\}$. We write $P(z_1, \dots, z_n, p_1, \dots, p_m) \Leftarrow_{\mathcal{R}} \psi$ iff $\mathcal{R}$ contains a rule $P(x_1, \dots, x_n, y_1, \dots, y_m) \Leftarrow \phi$ with $\psi = \phi\{x_i \leftarrow z_i, y_j \leftarrow p_j \mid i \in \{1, \dots, n\}, j \in \{1, \dots, m\}\}$.

Definition 5. *(Semantics) For every permission model $\mathfrak{P}$ and SID $\mathcal{R}$, the satisfiability relation $\models^{\mathfrak{P}}_{\mathcal{R}}$ is the smallest relation between structures (for $\mathfrak{P}$) and formulas such that:*

1. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \mathtt{emp}$ *iff* $\mathfrak{h} = \emptyset$.
2. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{p}{\mapsto} (y_1, \dots, y_k)$ *if* $\mathfrak{s}(p)$ *is defined and* $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \dots, \mathfrak{s}(y_k), \mathfrak{s}(p))\}$. *Note that this entails that* $dom(\mathfrak{h}) = \{\mathfrak{s}(x)\}$.
3. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \simeq y$ *(resp.* $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \not\simeq y$*) if* $\mathfrak{h} = \emptyset$, $\mathfrak{s}(x)$ *and* $\mathfrak{s}(y)$ *are defined and* $\mathfrak{s}(x) = \mathfrak{s}(y)$ *(resp.* $\mathfrak{s}(x) \neq \mathfrak{s}(y)$*).*
4. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \hat{P}(p_1, \dots, p_n)$ *with* $\hat{P} \in \mathcal{P}_{\mathrm{P}}$ *if* $\mathfrak{s}(p_i)$ *is defined for all* $i \in \{1, \dots, n\}$, $(\mathfrak{s}(p_1), \dots, \mathfrak{s}(p_n)) \in \hat{P}_{\mathfrak{P}}$ *and* $\mathfrak{h} = \emptyset$.
5. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x_1, \dots, x_n, \pi_1, \dots, \pi_m)$ *with* $P \in \mathcal{P}$ *if there exists* ϕ *such that* $P(x_1, \dots, x_n, \pi_1, \dots, \pi_m) \Leftarrow_{\mathcal{R}} \phi$ *and* $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.
6. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_1 * \phi_2$ *if there exist heaps* $\mathfrak{h}_1, \mathfrak{h}_2$ *such that* $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *is defined,* $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *and* $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ *for all* $i = 1, 2$.
7. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_1 \circ \phi_2$ *if there exists heaps* $\mathfrak{h}_1, \mathfrak{h}_2$ *such that* $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$, $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *and* $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ *for all* $i = 1, 2$.
8. $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists x\, \phi$ *if* $(\mathfrak{s}\{x \leftarrow \ell\}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ *for some* $\ell \in \mathcal{L}$.

A structure $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ *is an* $(\mathcal{R}, \mathfrak{P})$-model *of* ϕ. *A formula admitting an* $(\mathcal{R}, \mathfrak{P})$*-model is* $(\mathcal{R}, \mathfrak{P})$-satisfiable. *Two formulas are* sat-equivalent *(w.r.t.* $\mathcal{R}$, $\mathfrak{P}$*) if they are both* $(\mathcal{R}, \mathfrak{P})$*-satisfiable or both* $(\mathcal{R}, \mathfrak{P})$*-unsatisfiable.*

Example 6. The formula $x \overset{u}{\mapsto} (y, z) \circ x \overset{u'}{\mapsto} (y', z')$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable, as x cannot be allocated in disjoint parts of the heap. $x \overset{u}{\mapsto} (y) * x \overset{u'}{\mapsto} (y') * y \not\simeq y'$ is also $(\mathcal{R}, \mathfrak{P})$-unsatisfiable, as x cannot refer to two distinct records, but $x \overset{u}{\mapsto} (y, z) * x \overset{u'}{\mapsto} (y', z')$ admits the model (on the permission model $\mathfrak{f}$) $(\mathfrak{s}, \mathfrak{h})$ with $\mathfrak{s}(x) = 0$, $\mathfrak{s}(y) = \mathfrak{s}(y') = 1$, $\mathfrak{s}(z) = \mathfrak{s}(z') = 2$, $\mathfrak{s}(u) = 0.5$, $\mathfrak{s}(u') = 0.2$ and $\mathfrak{h} = \{(0, 1, 2, 0.7)\}$.

Note that there is no logical constant $\top$ (true): no formula can be satisfied on all heaps. The constant `emp` is similar to $\top$ but it states that the heap is empty. For all formulas ϕ, ψ, we write $\phi \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ iff the implication $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi \implies (\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ holds for all structures $(\mathfrak{s}, \mathfrak{h})$, and $\phi \equiv^{\mathfrak{P}}_{\mathcal{R}} \psi$ iff we have both $\phi \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ and $\psi \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. If ϕ contains no predicate symbols in $\mathcal{P}$, then the truth value of ϕ in $(\mathfrak{s}, \mathfrak{h})$ does not depend on $\mathcal{R}$. We thus may write $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} \phi$ instead of $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. If, moreover, ϕ is pure, then $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} \phi$ holds only if $\mathfrak{h}$ is empty. We will write $\mathfrak{s} \models^{\mathfrak{P}} \phi$ to state that $(\mathfrak{s}, \emptyset) \models^{\mathfrak{P}} \phi$. Finally, if ϕ contains only equalities between variables then its semantics does not depend on $\mathcal{R}$ and $\mathfrak{P}$ thus we write $\mathfrak{s} \models \phi$ to state that $(\mathfrak{s}, \emptyset) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. Note that the semantics of $\phi_1 \circ \phi_2$ and $\phi_1 * \phi_2$ coincide if ϕ_1 or ϕ_2 is pure, and also coincide with that of the usual standard conjunction if *both* ϕ_1 and ϕ_2 are pure.

Shorthands. If $\boldsymbol{x} = (x_1, \dots, x_n)$ and $\boldsymbol{y} = (y_1, \dots, y_m)$ are sequences of variables in $\mathcal{V}_1$ then $\boldsymbol{x} \simeq \boldsymbol{y}$ denotes the formula $\bot$ if $n \neq m$ and $(x_1 \simeq y_1) \circ \dots \circ (x_n \simeq y_n)$ otherwise. For every permission term p, we denote by $def(p)$ the atom $p \simeq p$. By definition, $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} def(p)$ iff $\mathfrak{s}(p)$ is defined and $\mathfrak{h} = \emptyset$.

3 $\mathfrak{h}$-Regular Systems

We focus on SIDs of some particular form, defined below.

Definition 7. *A rule is* $\mathfrak{h}$-regular *if it is of the following form:*

$$P(x, \boldsymbol{y}) \Leftarrow \exists u_1, \dots, u_n \, (x \overset{p}{\mapsto} (v_1, \dots, v_k) \circ Q_1(u_1, \boldsymbol{y}_1) \dots \circ Q_n(u_n, \boldsymbol{y}_n) \circ \phi)$$

where $\{u_1, \dots, u_n\} \subseteq \{v_1, \dots, v_k\}$, $\boldsymbol{y}_i$ *is a vector of variables*[3], $Q_i \in \mathcal{P}$ *and* ϕ *is pure. We assume by* α*-renaming that* $x, \boldsymbol{y}$ *do not occur in* $\{u_1, \dots, u_n\}$. *A SID* $\mathcal{R}$ *is* $\mathfrak{h}$-regular *if all the rules in* $\mathcal{R}$ *are* $\mathfrak{h}$*-regular.*

Note that the right-hand side formula contains only the disjoint separation connective $\circ$ and not the usual separating conjunction $*$. As we will see (Theorem 33) this is crucial for the decidability of the satisfiability problem. However, as already observed in [5], this is also justified from a practical point of view. Assume for instance that we want to define the predicate `lseg` introduced in [10], denoting a list segment from x to y with some permission z. The following rules can be used[4]: $\texttt{lseg}(x, y, z) \Leftarrow x \overset{z}{\mapsto} (y)$ $\quad \texttt{lseg}(x, y, z) \Leftarrow \exists u \, (x \overset{z}{\mapsto} (u) \circ \texttt{lseg}(u, y, z))$. A structure $(\mathfrak{s}, \mathfrak{h})$ satisfies $\texttt{lseg}(x, y, z)$ if $\mathfrak{h} = \{(\ell_i, \ell_{i+1}, \mathfrak{s}(z)) \mid i = 1, \dots, n\}$ with $n > 0$, $\mathfrak{s}(x) = \ell_1$, $\mathfrak{s}(y) = \ell_{n+1}$ and $\ell_i \neq \ell_j$ if $i \neq j$ and $i, j \in \{1, \dots, n\}$. This fits in with the definition in [10] (except that $n > 0$). In contrast, if one uses instead the connective $*$: $\texttt{lseg}(x, y, z) \Leftarrow \exists u \, (x \overset{z}{\mapsto} (u) * \texttt{lseg}(u, y, z))$, then one could obtain models where the list "loops" on itself an arbitrary number of times, such as, for instance $(\mathfrak{s}, \{(\mathfrak{s}(x), \mathfrak{s}(x), p))\})$, with $\mathfrak{s}(y) = \mathfrak{s}(x)$ and $p = \mathfrak{s}(z)^n$, for any $n > 0$ such that $\mathfrak{s}(z)^n$ is defined. In the former definition, $\mathfrak{s}(y)$ possibly occurs in $\{\ell_1, \dots, \ell_n\}$, but each location can only be allocated once.

Intuitively, $\mathfrak{h}$-regular sets of inductive rules generate heaps with a regular structure (in the sense that it may be represented by a tree automaton [9]), enriched with some additional edges (referring to the nodes corresponding to the variables passed as parameters to the spatial predicates at some recursive calls). These additional edges may refer to locations corresponding to free variables (e.g. the root of the structure) but also to existential variables (for instance they may refer to the parent node in the tree). $\mathfrak{h}$-Regular SID are related to the PCE systems introduced in [13] (for **p**rogressing, **c**onnected and **e**stablished), extended to formulas with permissions, but our conditions are slightly stronger, because we require that every existential variable be allocated at the next recursive call. Note that structures with mixed permissions are allowed, for instance

[3] i.e., compound permission terms are not allowed in predicate atoms.

[4] As $\mathfrak{h}$-regular rules allocate exactly one location, we assume that the segment is non empty, the case of an empty segment must be considered apart.

the rules $P(x, z_1, z_2) \Leftarrow x \overset{z_1}{\mapsto} ()$ and $P(x, z_1, z_2) \Leftarrow \exists u\, (x \overset{z_1}{\mapsto} (u) \circ P(u, z_2, z_1))$ defines a list with permissions alternating between z_1 and z_2. Rules with compound permission terms in points-to or permission atoms are allowed (such as $P(x, y_1, y_2) \Leftarrow x \overset{y_1 \oplus y_2}{\mapsto} () \circ \mathit{def}(y_1 \oplus y_1))$, but not those with compound permission terms in spatial predicate atoms[5] (e.g., $P(x, y_1, y_2) \Leftarrow x \overset{y_1}{\mapsto} () \circ Q(x, y_1 \oplus y_2)$ is *not* $\mathfrak{h}$-regular).

For every quantifier-free formula ϕ, we denote by $\mathit{roots}(\phi)$ the set of variables x (called the *roots of* ϕ) inductively defined as follows: $\mathit{roots}(x \overset{p}{\mapsto} (y_1, \dots, y_k)) \overset{\text{def}}{=} \{x\}$, $\mathit{roots}(P(x, y_1, \dots, y_k)) \overset{\text{def}}{=} \{x\}$, $\mathit{roots}(\exists y\, \phi) = \mathit{roots}(\phi) \setminus \{y\}$, $\mathit{roots}(\phi) = \emptyset$ if ϕ is pure and $\mathit{roots}(\phi_1 * \phi_2) = \mathit{roots}(\phi_1 \circ \phi_2) = \mathit{roots}(\phi_1) \cup \mathit{roots}(\phi_2)$. By Definition 7, roots are always allocated:

Proposition 8. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. If $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} \phi$ and $x \in \mathit{roots}(\phi)$ then $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$. Consequently, every formula of the form $\phi_1 \circ \phi_2$ with $\mathit{roots}(\phi_1) \cap \mathit{roots}(\phi_2) \neq \emptyset$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable.*

The conditions in Definition 7 are actually not sufficient to ensure that the satisfiability problem is decidable:

Theorem 9. *If there exist (not necessary distinct) permissions $\pi_1, \pi_2 \in \mathcal{P}_{\mathfrak{P}}$ such that $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is defined, then the $(\mathcal{R}, \mathfrak{P})$-satisfiability problem is undecidable for $\mathfrak{h}$-regular SID $\mathcal{R}$.*

To ensure decidability, we need to further restrict the way existential variables are passed as parameters during recursive calls. This is the goal of the next definition.

Definition 10. *Assume that $\mathcal{R}$ is $\mathfrak{h}$-regular. Given two spatial predicates P and Q, of arities n and m respectively, we write $P \bowtie_{\mathcal{R}} Q$ if $P(x, x_1, \dots, x_{n-1}) * Q(x, y_1, \dots, y_{m-1})$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable[6] (where $x_1, \dots, x_{n-1}, y_1, \dots, y_{m-1}$ denote pairwise distinct variables of the appropriate sorts). We denote by $\gamma_{\mathcal{R}}$ the function associating every predicate symbol P of spatial arity n to a subset of $\{2, \dots, n\}$ inductively defined as follows: for every rule $P(x_1, \dots, x_n, \boldsymbol{u}) \Leftarrow \exists y_1, \dots, y_m\, \phi$ in $\mathcal{R}$, for every predicate atom $Q(z_1, \dots, z_k, \boldsymbol{u}_k)$ in ϕ with $\#_1(Q) = k$ and for all $i \in \{2, \dots, k\}$:*

1. *$z_i \in \{y_1, \dots, y_m\} \Rightarrow i \in \gamma_{\mathcal{R}}(Q)$.*
2. *$z_i \in \{x_j \mid j \in \gamma_{\mathcal{R}}(P)\} \implies i \in \gamma_{\mathcal{R}}(Q)$.*

[5] Otherwise the unfolding of spatial predicates could yield terms of arbitrary depth.

[6] In practice, as this condition is hard to test, some stronger syntactic condition can be tested instead, for instance one can check that all the formulas ϕ and ϕ' such that $P(x, x_1, \dots, x_{n-1}) \Leftarrow_{\mathcal{R}} \phi$ and $Q(x, y_1, \dots, y_{m-1}) \Leftarrow_{\mathcal{R}} \phi'$ are of the form $\phi = (x \mapsto (\boldsymbol{u}) \circ \psi)$ and $\phi' = (x \mapsto (\boldsymbol{u}') \circ \psi')$ with $|\boldsymbol{u}| \neq |\boldsymbol{u}'|$ (this condition is used in Theorem 33 and for the Exptime-hardness proof in Theorem 32.). More generally, it is sufficient to test that the "shape" of the structures generated by P and Q, up to a certain fixed unfolding depth, are incompatible.

Let $\mathcal{P}^\star$ be a subset of $\mathcal{P}$, such that: (3) $P \in \mathcal{P}^\star \implies \gamma_{\mathcal{R}}(P) = \emptyset$; and (4) $P \in \mathcal{P}^\star \wedge Q \in \mathcal{P} \setminus \mathcal{P}^\star \implies P \bowtie_{\mathcal{R}} Q$. A $\mathfrak{h}$-regular rule is ∃-restricted *(w.r.t. $\mathcal{R}$ and $\mathcal{P}^\star$) if it satisfies the following condition (using the notations of Definition 7):*

5. $\forall i \in \{1, \dots, n\} \forall j \in \{1, \dots, n\} \, (u_i \in \boldsymbol{y}_j \implies Q_i \in \mathcal{P}^\star)$.

A SID $\mathcal{R}$ is ∃-restricted *if all the rules in $\mathcal{R}$ are ∃-restricted.*

Conditions 1 and 2 in Definition 10 are meant to ensure that $\gamma_{\mathcal{R}}(P)$ denotes the indices of the parameters of P that may (but do not have to) be instantiated by some existential variable introduced during the unfolding of the inductive rules in $\mathcal{R}$ (the other parameters may only be instantiated by variables occurring in the initial formula). Condition 1 corresponds to a base case, where an existential variable is passed as a parameter to a predicate symbol, and Condition 2 handles the inductive case, when the variable is carried through recursive calls[7]. Then, Condition 5 ensures that an existential variable may only be passed as a parameter to a predicate symbol if it is the root of a structure defined by an atom $Q_i(\boldsymbol{y}_i)$ containing no variables introduced by unfolding (by Condition 3).

Example 11. The rules of the predicate `lseg` are ∃-restricted (with $\mathcal{P}^\star = \emptyset$). Indeed, they contain only one existential variable u, which occurs only as the first argument of a predicate. Hence Condition 5 in Definition 10 trivially holds. If $\mathcal{R}$ contains no other rule then $\gamma_{\mathcal{R}}(\texttt{lseg}) = \emptyset$. Note that $\gamma_{\mathcal{R}}(\texttt{lseg})$ depends on the entire set $\mathcal{R}$. For instance, if $\mathcal{R}$ contains a rule $P(x, y) \Leftarrow \exists u \, (x \overset{y}{\mapsto} (u) \circ \texttt{lseg}(u, u, y))$ then the second argument of `lseg` may be instantiated by an existential variable hence $\gamma_{\mathcal{R}}(\texttt{lseg}) = \{2\}$, and the latter rule is not ∃-restricted. On the other hand, if $\mathcal{P}^\star = \{Q\}$, then the rules $Q(x, y) \Leftarrow x \overset{y}{\mapsto} ()$, $R(x, y) \Leftarrow \exists u, v \, (x \overset{y}{\mapsto} (u, v) \circ \texttt{lseg}(u, v, y) \circ Q(v, y))$ are ∃-restricted, with $\mathcal{P}^\star = \{Q\}$. Indeed, the variable u occurs only at the root of a predicate, and the variable v is the root of $Q(v, y)$. Note that $\texttt{lseg}(x, y, z) * Q(x, u)$ and $R(x, y) * Q(x, u)$ are $(\mathcal{R}, \mathfrak{P})$-unsatisfiable, thus $\texttt{lseg} \bowtie_{\mathcal{R}} Q$ and $R \bowtie_{\mathcal{R}} Q$.

Intuitively, the structures generated by ∃-restricted rules are regular tree-shaped structures, enriched with two kinds of additional edges: (i) a *bounded* number of *arbitrary* edges (corresponding to free variables, which may be freely passed as arguments to any predicate, thus may be referred to in an arbitrary way); (ii) an *unbounded* number of other edges (corresponding to existential variables) which are only allowed to point to structures that contain no edge of type (ii). Condition 4 ensures that the structures containing only edges of type (i) do not overlap with those containing both kinds of edges. Note that the conditions of Definition 10 always hold if the existential variables occur only

[7] For generality, one could assume that all the equalities occurring in the rules are propagated before $\gamma_{\mathcal{R}}$ is computed (so that existential variables are eliminated if they are equal to a free variable), but this is not essential for our purposes hence the corresponding formal definitions are omitted.

as roots (with $\mathcal{P}^\star = \mathcal{P}$ or $\mathcal{P}^\star = \emptyset$). In this case there is no edge of type (ii), i.e., the obtained structures are regular sets of trees with a bounded number of additional edges (for instance trees with pointers to the root, or cyclic lists). Note that doubly linked lists cannot be captured (as they contain an unbounded number of additional edges from every node to the previous one). In the following we devise an algorithm to test the $(\mathcal{R}, \mathfrak{P})$-satisfiability of symbolic heaps when $\mathcal{R}$ is $\exists$-restricted.

4 A Decision Procedure for Testing Satisfiability

Before entering into technical details we start with a general overview of the procedure for testing satisfiability (assuming the considered SID is $\exists$-restricted).

1. Starting with a formula of the form $\delta_1 * \cdots * \delta_n$ where the δ_i's are atoms, we first reduce every spatial atom δ_i into an equivalent disjunction of $\circ$-conjunctions $\delta_1^i \circ \ldots \circ \delta_{m_i}^i$ such that the *only* free variables allocated by an atom δ_j^i are its roots $\mathit{roots}(\delta_j^i)$ (as δ_j^i is an atom, $\mathit{card}(\mathit{roots}(\delta_j^i)) \leq 1$). Due to the particular properties of the $\mathfrak{h}$-regular rules (more precisely, due to the fact that the rules satisfy the "establishment" property of [13], i.e., every existential variable is allocated), this entails that, for all structures $(\mathfrak{s}, \mathfrak{h}_{i,j})$ satisfying δ_j^i, the domains of $\mathfrak{h}_{i,j}$ and $\mathfrak{h}_{i',j'}$ are either equal (if δ_j^i and $\delta_{j'}^{i'}$ have the same roots) or disjoint (otherwise). Indeed, the establishment property ensures that the considered heaps have no "pending edges" (i.e., no location that is referred to but not allocated), other than those denoted by free variables. This step can be considered as the key part of the procedure. It requires to (automatically) enrich the language with additional predicates and rules, and the termination of the transformation crucially depends on the conditions on $\exists$-restricted rules. For instance, an atom $\mathtt{lseg}(x, x)$ occurring in a formula with free variables x, y could be written $(x \simeq y \circ \mathtt{lseg}(x, x)) \vee \mathtt{lseg}'(x, x, y) \vee (\mathtt{lseg}'(x, y, y) \circ \mathtt{lseg}'(y, x, x))$ where $\mathtt{lseg}'(u, v, w)$ denotes a list segment from u to v not allocating w. The previous decomposition depends on whether y is equal to x and whether y occurs in the list segment from x to x.
2. By distributivity, we get at this point $*$-conjunctions of $\circ$-conjunctions of atoms. Taking advantage of the previous property, we then reduce these formulas into $\circ$-conjunctions of $*$-conjunctions of atoms, by regrouping the atoms with the same roots, e.g., $(P(x, y) \circ Q(y, x)) * (P'(x, y) \circ Q'(y, x))$ may be written $(P(x, y) * P'(x, y)) \circ (Q(y, x) * Q'(y, x))$.
3. Next, we show that a $*$-conjunction of atoms sharing the same root (such as $P(x, y) * P'(x, y)$ or $Q(y, x) * Q'(y, x)$) can be denoted by a single atom, the rules of which are obtained by "merging" the rules of the initial atoms.
4. At this point we get a $\circ$-conjunction of atoms. To ensure that the formula is satisfiable it suffices to test that all these atoms have a model and that all these models are compatible, w.r.t. the equality constraints, allocated locations and permission constraints. To this aim, we construct finite abstractions of the models of the considered atoms using a bottom-up fixpoint algorithm.

In the next subsections, each of these steps is explained in details.

4.1 Normalization

We first show that every formula can be transformed into an equivalent formula (that we call *normalized*) in which every allocated variable occurs as a root:

Definition 12. *A formula ϕ is* normalized *if it is of the form $\exists \boldsymbol{x}\, \psi$ where ψ is quantifier-free and for all spatial atoms δ in ψ, for all $(\mathcal{R}, \mathfrak{P})$-models $(\mathfrak{s}, \mathfrak{h})$ of δ and for all variables $y \in fv(\psi)$: $\mathfrak{s}(y) \in dom(\mathfrak{h}) \iff y \in roots(\psi)$.*

For instance, $\mathtt{lseg}(x, y)$ is not normalized, because y may be allocated (e.g., if $\mathfrak{s}(x) = \mathfrak{s}(y)$) and does not occur in $roots(\mathtt{lseg}(x, y)) = \{x\}$. To enforce this condition, we introduce new predicate symbols (called *derived predicates*), the rules of which can be automatically computed from those of the predicates already occurring in this formula. We first define predicate symbols that ensure that some given variable is not allocated.

Definition 13. *For all predicate atoms $P(\boldsymbol{x}, \boldsymbol{p})$ (where $\boldsymbol{x}$ and $\boldsymbol{p}$ are vectors of location variables and permission terms, respectively) and for all location variables v, we denote by $P(\boldsymbol{x}, \boldsymbol{p})[v]^-$ any atom of the form $Q(\boldsymbol{x}, v, \boldsymbol{p})$, where Q is a fresh predicate symbol, associated with the rules:*

$$Q(\boldsymbol{y}, w, \boldsymbol{z}) \Leftarrow \exists \boldsymbol{u}\, (Q_1(\boldsymbol{y}_1, \boldsymbol{p}_1)[w]^- \circ \ldots \circ Q_m(\boldsymbol{y}_m, \boldsymbol{p}_m)[w]^- \circ \phi \circ \boldsymbol{y}|_1 \not\simeq w)$$

for all rules $P(\boldsymbol{y}, \boldsymbol{z}) \Leftarrow \exists \boldsymbol{u}\, (Q_1(\boldsymbol{y}_1, \boldsymbol{p}_1) \circ \ldots \circ Q_m(\boldsymbol{y}_m, \boldsymbol{p}_m) \circ \phi)$ in $\mathcal{R}$ (up to AC), where $\boldsymbol{y}, \boldsymbol{y}_i$ are vectors of location variables, $\boldsymbol{z}$, $\boldsymbol{p}_i$ are vectors of permission variables, and ϕ contains no predicate atom.

For instance $\mathtt{lseg}(x, y, z)[u]^-$ is a predicate atom $Q(x, y, u, z)$ defined by the following rules: $\{Q(x, y, u, z) \Leftarrow \exists x'(x \overset{z}{\mapsto} (x') \circ Q(x', y, u, z) \circ x \not\simeq u), Q(x, y, u, z) \Leftarrow x \overset{z}{\mapsto} (y) \circ x \not\simeq u\}$. It denotes a list segment from x to y not allocating u. The following result is straightforward to prove:

Proposition 14. *For every $\exists$-restricted SID $\mathcal{R}$, the set $\mathcal{R}$ enriched with the rules associated with the predicate Q corresponding to $P(\boldsymbol{x}, p)[v]^-$ in Definition 13 is $\exists$-restricted, with $\gamma_{\mathcal{R}}(Q) = \gamma_{\mathcal{R}}(P)$ and $Q \in \mathcal{P}^\star \iff P \in \mathcal{P}^\star$.*

Intuitively the structures that satisfy $P(\boldsymbol{x}, \boldsymbol{p})[v]^-$ are exactly those that satisfy $P(\boldsymbol{x}, \boldsymbol{p})$ and do not allocate v:

Lemma 15. *For all $\mathfrak{h}$-regular SID $\mathcal{R}$, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} P(\boldsymbol{x}, p)[v]^-$ iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} P(\boldsymbol{x}, p)$ and $\mathfrak{s}(v) \notin dom(\mathfrak{h})$.*

The operator $\delta \mapsto \delta[x]^-$ can be applied recursively, e.g., one can consider atoms of the form $\delta[x]^-[y]^-$, etc. For all predicate atoms δ, we denote by $unalloc(\delta)$ the set of variables inductively defined as follows: $unalloc(\delta[x]^-) \overset{\text{def}}{=} \{x\} \cup unalloc(\delta)$, and $unalloc(\delta) \overset{\text{def}}{=} \emptyset$ if δ is not of the form $\delta'[x]^-$. The following proposition is an immediate consequence of Lemma 15:

Proposition 16. *If $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} \delta$ then $\mathfrak{s}(x) \notin dom(\mathfrak{h})$, for all $x \in unalloc(\delta)$.*

Next, we define predicate symbols allowing one to remove some part of a structure. Intuitively, the expression ($\phi \multimap\!\bullet\ \psi$) will hold exactly in the structures that satisfy ψ when a disjoint structure satisfying ϕ is added. For instance given the rules $\mathtt{tree}(x,y) \Leftarrow \exists x_1, x_2\, x \overset{y}{\mapsto} (x_1,x_2) \circ \mathtt{tree}(x_1,y) \circ \mathtt{tree}(x_2,y)$ and $\mathtt{tree}(x,y) \Leftarrow x \overset{y}{\mapsto} ()$, $\mathtt{tree}(z,y)$ and $\mathtt{tree}(x,y)$ denote binary trees with roots z and x, respectively, and $\mathtt{tree}(z,y) \multimap\!\bullet\ \mathtt{tree}(x,y)$ denotes a tree of root x with a "hole" at z (the structures satisfying $\mathtt{tree}(z,y) \multimap\!\bullet\ \mathtt{tree}(x,y)$ are obtained from models of $\mathtt{tree}(x,y)$ by removing the part of the heap that corresponds to $\mathtt{tree}(z,y)$). The formula $\phi \multimap\!\bullet\ \psi$ is similar to the *strong magic wand* introduced in [17] and to the *context predicates* in [12] and also close in spirit to the separating implication of SL although the semantics are slightly different.

Definition 17. *For all finite sequences of predicate atoms $P_i(\boldsymbol{x}_i,\boldsymbol{p}_i)$ (with $i = 0,\dots,n$), where $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are vectors of location variables and permission terms, respectively, we denote by $(P_1(\boldsymbol{x}_1,\boldsymbol{p}_1) \circ \dots \circ \hat{P}_n(\boldsymbol{x}_n,\boldsymbol{p}_n)) \multimap\!\bullet\ P_0(\boldsymbol{x}_0,\boldsymbol{p}_0)$ any atom $P(\boldsymbol{x},\boldsymbol{p})$ with $\boldsymbol{x} = \boldsymbol{x}_0.\dots.\boldsymbol{x}_n$, $\boldsymbol{p} = \boldsymbol{p}_0.\dots.\boldsymbol{p}_n$, and such that $P = P_0$ if $n = 0$ and otherwise P is a fresh symbol associated with rules of the form*

$$P(\boldsymbol{y},\boldsymbol{z}) \Leftarrow \exists \boldsymbol{w}\, (\psi_1 \circ \dots \circ \psi_m \circ \phi)$$

for all rules

$$P_0(\boldsymbol{y}_0,\boldsymbol{z}_0) \Leftarrow \exists \boldsymbol{w}\, (Q_1(\boldsymbol{u}_1,\boldsymbol{q}_1) \circ \dots \circ Q_m(\boldsymbol{u}_m,\boldsymbol{q}_m) \circ \phi)$$

in $\mathcal{R}$ and for all decompositions $\alpha_1 \circ \dots \circ \alpha_m = P_1(\boldsymbol{y}_1,\boldsymbol{z}_1) \circ \dots \circ P_n(\boldsymbol{y}_n,\boldsymbol{z}_n)$ (up to AC, where the α_i's may be empty), where:

- *$\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ are sequences of pairwise distinct location and permission variables, respectively, with $|\boldsymbol{y}_i| = |\boldsymbol{x}_i|$ and $|\boldsymbol{z}_i| = |\boldsymbol{p}_i|$;*
- *$\boldsymbol{y} = \boldsymbol{y}_0.\dots.\boldsymbol{y}_n$, $\boldsymbol{z} = \boldsymbol{z}_1.\dots.\boldsymbol{z}_n$;*
- *ψ_i is of one of the following forms:*
  - *either $\alpha_i \multimap\!\bullet\ Q_i(\boldsymbol{u}_i,\boldsymbol{q}_i)$;*
  - *or $\boldsymbol{y}_j \simeq \boldsymbol{u}_i \circ \boldsymbol{z}_j \simeq \boldsymbol{q}_i$, if $\alpha_i = P_j(\boldsymbol{y}_j,\boldsymbol{z}_j)$ and $P_j = Q_i$.*

For instance $\mathtt{tree}(z,y) \multimap\!\bullet\ \mathtt{tree}(x,y)$ denotes an atom $P(x,z,y,y)$ with the rules:

$$P(x,z,y_1,y_2) \Leftarrow \exists x_1,x_2\ (x \overset{y_1}{\mapsto} (x_1,x_2) \circ P(x_1,z,y_1,y_2) \circ \mathtt{tree}(x_2,z,y_1))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1,x_2\ (x \overset{y_1}{\mapsto} (x_1,x_2) \circ \mathtt{tree}(x_1,z,y_1) \circ P(x_2,z,y_1,y_2))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1,x_2\ (x \overset{y_1}{\mapsto} (x_1,x_2) \circ x_1 \simeq z \circ y_1 \simeq y_2 \circ \mathtt{tree}(x_2,z,y_1))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1,x_2\ (x \overset{y_1}{\mapsto} (x_1,x_2) \circ \mathtt{tree}(x_1,z,y_1) \circ x_2 \simeq z \circ y_1 \simeq y_2)$$

For readability, all the expressions of the form $\mathtt{emp} \multimap\!\bullet\ \mathtt{tree}(x_2,z,y_1)$ have been replaced by $\mathtt{tree}(x_2,z,y_1)$. Note that the rules are not $\mathfrak{h}$-regular, as x_1 and x_2 do not occur as roots in every rule, but they can easily be transformed into $\mathfrak{h}$-regular rules by replacing x_1 and x_2 by z in the third and fourth rule, respectively (using the equations $x_1 \simeq z$ and $x_2 \simeq z$). The definition can be applied recursively (i.e., $P_0,\dots,P_n$ may be derived predicates). The next proposition is an immediate consequence of Definition 17:

Proposition 18. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. The rules associated with any predicate P corresponding to an expression $\alpha \multimapdotbothB \delta$ (Definition 17) are $\mathfrak{h}$-regular, up to the following equivalence: $\exists x \, (x \simeq y \circ \phi) \equiv^{\mathfrak{P}}_{\mathcal{R}} \phi\{x \leftarrow y\}$. Moreover, the rules are also $\exists$-restricted, with $\gamma_{\mathcal{R}}(P) = \gamma_{\mathcal{R}}(P_0)$ and $P \in \mathcal{P}^\star \iff P_0 \in \mathcal{P}^\star$. Finally if $\alpha = \mathtt{emp}$ then $(\alpha \multimapdotbothB \delta) = \delta$.*

Note that, however, the implication $P \in \mathcal{P}^\star \wedge Q \in \mathcal{P} \setminus \mathcal{P}^\star \implies P \bowtie_{\mathcal{R}} Q$ (Condition 4 in Definition 10) does *not* necessarily hold for derived predicates P, Q. The following lemma states a form of modus ponens, relating the connective $\circ$ with $\multimapdotbothB$:

Lemma 19. *If $\mathcal{R}$ is $\mathfrak{h}$-regular then $P(\boldsymbol{x}, \boldsymbol{p}) \circ ((P(\boldsymbol{x}, \boldsymbol{p}) \circ \alpha) \multimapdotbothB Q(\boldsymbol{y}, \boldsymbol{q})) \models^{\mathfrak{P}}_{\mathcal{R}} \alpha \multimapdotbothB Q(\boldsymbol{y}, \boldsymbol{q})$.*

The next lemma states that every predicate atom allocating x can be written as a $\circ$-formula in which x occurs as a root.

Lemma 20. *Assume that $\mathcal{R}$ is $\exists$-restricted. Let $\boldsymbol{y}, \boldsymbol{p}$ be vectors of location variables and permission terms, respectively. If $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} Q(\boldsymbol{y}, \boldsymbol{p})$, $\mathfrak{s}(x) \neq \mathfrak{s}(\boldsymbol{y}|_1)$ and $\mathfrak{s}(x) \in dom(\mathfrak{h})$, then there exist atoms of the form $P(x, \boldsymbol{z}, \boldsymbol{q})$, $P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i)$ (with $i \in \{1, \dots, n\}$), where $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \{x_1, \dots, x_n\}$, $\boldsymbol{y}_i \subseteq \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\}$, $\boldsymbol{q} \subseteq \boldsymbol{p}$ and $\boldsymbol{q}_i \subseteq \boldsymbol{p}$, such that: $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists x_1, \dots, x_n \; (\beta \circ (\beta \multimapdotbothB Q(\boldsymbol{y}, \boldsymbol{p})))$, with $\beta = P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=1}^m P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i)$. Moreover, $P_i \in \mathcal{P}^\star$, $\{x_1, \dots, x_n\} \subseteq (x, \boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$ and $y \in \boldsymbol{y} \cap \boldsymbol{z} \wedge y \notin \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\} \implies y \in (x, \boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$.*

Intuitively, since x is allocated and the rules are $\mathfrak{h}$-regular, then necessarily some predicate atom of the form $P(x, \boldsymbol{z}, \boldsymbol{q})$ must be called at some point during the unfolding of the rules. Using $\multimapdotbothB$, this predicate can be removed from the call tree of $Q(\boldsymbol{y}, \boldsymbol{p})$ and lifted at the root level in the formula. The atom $P(x, \boldsymbol{z}, \boldsymbol{q})$ may contain variables not occurring in $Q(\boldsymbol{y}, \boldsymbol{p})$ corresponding to existential variables introduced by unfolding. As the rules are $\exists$-restricted, all such variables x_i must themselves appear as the root of some predicate atom $P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i)$ which contains (beside x_i) only variables occurring in $Q(\boldsymbol{y}, \boldsymbol{p})$ (since $\gamma_{\mathcal{R}}(P_i) = \emptyset$, due to Condition 5 in Definition 10). Again, these atoms can be moved at the root level.

Definition 21. *For all atoms $Q(\boldsymbol{y}, \boldsymbol{p})$ we denote by $\delta[x]^+$ the set of formulas of the form $\exists x_1, \dots, x_n \; (\beta \circ (\beta \multimapdotbothB Q(\boldsymbol{y}, \boldsymbol{p})))$ as defined in Lemma 20. We also denote by $\delta[x]^=$ the formula: $\delta \circ (x \simeq \boldsymbol{y}|_1)$.*

For every model of δ, $\delta[x]^-$ holds if x is not allocated in δ, $\delta[x]^=$ holds if x is equal to the root of δ and $\delta[x]^+$ holds if x is allocated but is not the root of δ. The following result follows immediately from Lemmata 19 and 20:

Lemma 22. *Assume that $\mathcal{R}$ is $\exists$-restricted. Let $x \in \mathcal{V}_1$. For every predicate atom δ such that $x \notin roots(\delta)$, and for all structures $(\mathfrak{s}, \mathfrak{h})$: $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \delta$ iff there exists $\psi \in \{\delta[x]^-, \delta[x]^=\} \cup \delta[x]^+$ such that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$.*

For instance the atom $\mathtt{lseg}(x, y, z)$ holds iff one of the formulas $\mathtt{lseg}(x, y, z) \circ x \simeq y$, $\mathtt{lseg}(x, y, z)[y]^-$ or $\mathtt{lseg}(y, y, z) \circ (\mathtt{lseg}(y, y, z) \multimapdot \mathtt{lseg}(x, y, z))$ holds. The second formula corresponds to the case where y is not allocated, and the first and third ones correspond to the case where there is a loop on y. By applying repeatedly Lemma 22 on every variable x and atom δ we eventually obtain a disjunction of normalized formulas:

Lemma 23. *Let $\mathcal{R}$ be a $\exists$-restricted SID. There exists an algorithm transforming any symbolic heap ϕ containing no points-to atom into a set of normalized formulas Ψ such that for all structures $(\mathfrak{s}, \mathfrak{h})$: $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} \phi$ iff there exists $\psi \in \Psi$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} \psi$. Furthermore, every formula in Ψ is a (quantified) separating conjunction of $\circ$-formulas.*

4.2 Commuting Separating and Disjoint Connections

The next step consists in showing that – under some particular conditions enforced by the previous transformation – the operator $*$ can be pushed innermost in the formula (below the operator $\circ$). To this aim, we exploit an essential property of $\mathfrak{h}$-regular SIDs, namely that all the locations that occur in the heap of some model of a formula ϕ but are not allocated correspond to a variable in $fv(\phi)$. We shall denote by $cut(L, L', \mathfrak{h})$ the set of locations reachable from L in $\mathfrak{h}$, from a path not crossing L':

Definition 24. *Let $\mathfrak{h}$ be a heap, let $L, L' \subseteq \mathcal{L}$. We denote by $cut(L, L', \mathfrak{h})$ the set of locations inductively defined as follows: $L \subseteq cut(L, L', \mathfrak{h})$, and if $\ell' \in cut(L, L', \mathfrak{h})$, $\mathfrak{h}(\ell') = (\ell_1, \ldots, \ell_k, \pi)$, $i \in \{1, \ldots, k\}$ and $\ell_i \notin L'$ then $\ell_i \in cut(L, L', \mathfrak{h})$.*

The following lemma characterizes the domain of the part of the heap satisfying some formula ϕ:

Lemma 25. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID and let ϕ be a $\circ$-formula containing no quantifier. Let $\mathfrak{s}$ be a store and let $\mathfrak{h}, \mathfrak{h}'$ be heaps, with $\mathfrak{h}' \leq \mathfrak{h}$. Let V be a set of variables, with $fv(\phi) \subseteq V \cup roots(\phi)$ and $\mathfrak{s}(V) \cap dom(\mathfrak{h}') = \emptyset$. If $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}}^{\mathfrak{P}} \phi$ then $dom(\mathfrak{h}') = cut(\mathfrak{s}(roots(\phi)), \mathfrak{s}(V), \mathfrak{h})$.*

The commutation property, pushing $*$ below $\circ$, is given by Lemma 26:

Lemma 26. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. Let $V \subseteq \mathcal{V}_1$ and let ϕ be a normalized formula, of the form $\phi = \phi' \circ (\mathop{\Large *}_{i=1}^{n} (\phi_i \circ \psi_i) * \psi')$, where, for all $i \in \{1, \ldots, n\}$, $roots(\phi_i) = V$ and $(roots(\psi_i) \cup roots(\psi')) \cap V = \emptyset$. Then ϕ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff $(\phi' \circ \mathop{\Large *}_{i=1}^{n} \phi_i) \circ ((\mathop{\Large *}_{i=1}^{n} \psi_i) * \psi')$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable.*

Roughly speaking, as $roots(\phi_i) = V$ and ϕ_i is normalized, it is possible to prove, using the characterization given in Lemma 25, that the parts of the heap that correspond to the formulas ϕ_i have all the same domain. This entails that the heaps corresponding to the formulas ψ_i and $\phi_{i'}$ are disjoint, which permits to prove that $\mathop{\Large *}_{i=1}^{n} (\phi_i \circ \psi_i)$ can be written $(\mathop{\Large *}_{i=1}^{n} \phi_i) \circ (\mathop{\Large *}_{i=1}^{n} \psi_i)$, yielding the result.

4.3 Merging of Spatial Predicates

We show that, under some particular conditions, it is possible to replace the separating conjunction of two spatial atoms having the same root by a single spatial atom. The rules defining this atom are obtained by combining the rules of the two initial atoms. More precisely, consider any $\mathfrak{h}$-regular SID $\mathcal{R}$ and two spatial atoms $P(x, \boldsymbol{y}, \boldsymbol{p})$ and $P'(x, \boldsymbol{y}', \boldsymbol{p}')$ sharing the same root x, where $\boldsymbol{y}, \boldsymbol{y}'$ are vectors of location variables and $\boldsymbol{p}$ and $\boldsymbol{p}'$ are vectors of permission terms. We denote by $P(x, \boldsymbol{y}, \boldsymbol{p}) \nabla P'(x, \boldsymbol{y}', \boldsymbol{p}')$ any atom $Q(x, \boldsymbol{y}, \boldsymbol{y}', \boldsymbol{p}, \boldsymbol{p}')$ where Q is associated with rules of the form:

$$Q(v, \boldsymbol{w}, \boldsymbol{w}', \boldsymbol{z}, \boldsymbol{z}') \Leftarrow \exists u_1, \dots, u_n \quad v \overset{q}{\mapsto} (v_1, \dots, v_k) \\ \circ \bigcirc_{i=1}^{n} (Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \nabla Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q_i}')) \circ \phi \circ \phi' \circ \psi$$

with $q \overset{\text{def}}{=} p \oplus p'$, for all pairs of rules of the following forms in $\mathcal{R}$ (with the same numbers k and n, and up to α-renaming, so that the rules share the same existential variables):

$$\begin{aligned} P(v, \boldsymbol{w}, \boldsymbol{z}) \quad &\Leftarrow \exists u_1, \dots, u_n \quad v \overset{p}{\mapsto} (v_1, \dots, v_k) \circ \bigcirc_{i=1}^{n} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \phi \\ P'(v, \boldsymbol{w}', \boldsymbol{z}') &\Leftarrow \exists u_1, \dots, u_n \quad v \overset{p'}{\mapsto} (v'_1, \dots, v'_k) \circ \bigcirc_{i=1}^{n} Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q}'_i) \circ \phi' \end{aligned}$$

where $\psi = \bigcirc_{i=1}^{k} (v_i \simeq v'_i)$. Note that all the produced rules are $\mathfrak{h}$-regular[8].

Lemma 27. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. Let $x \in \mathcal{V}_1$ and let $(\mathfrak{s}, \mathfrak{h})$ be a structure such that $\mathfrak{s}(y) \notin dom(\mathfrak{h})$ holds for all variables y such that $\mathfrak{s}(x) \neq \mathfrak{s}(y)$. Then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} P(x, \boldsymbol{y}, \boldsymbol{p}) \nabla P'(x, \boldsymbol{y}', \boldsymbol{p}') \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathfrak{P}} P(x, \boldsymbol{y}, \boldsymbol{p}) * P'(x, \boldsymbol{y}', \boldsymbol{p}')$.*

The result crucially depends on the fact that the parts of the heap that correspond to $P(x, \boldsymbol{y}, \boldsymbol{p})$ and $P'(x, \boldsymbol{y}', \boldsymbol{p}')$ respectively must share the same domain, since otherwise, as $\mathcal{R}$ is $\mathfrak{h}$-regular, a free variable would be allocated, contradicting the hypothesis. This ensures that the heap can be generated by the above rules.

4.4 Heap Abstractions and Main Result

As we shall see later, the previous transformations can be used to transform any symbolic heap into a $\circ$-formula (while preserving satisfiability). The final step is to devise an algorithm to test the satisfiability of $\circ$-formulas. As it is done in [6] for standard heap models, the algorithm works by constructing relevant abstractions of the models of the predicate atoms. It suffices to keep track of the truth value of the equational atoms, of the allocated variables and of the permission atoms satisfied by the structure.

[8] However $\exists$-restrictedness is not necessarily preserved.

Definition 28. *A* heap abstraction *is a tuple* $\mathfrak{a} = (V_\mathfrak{a}, \sim_\mathfrak{a}, A_\mathfrak{a}, \rho_\mathfrak{a})$ *where* $V_\mathfrak{a}$ *is a finite set of variables,* $\sim_\mathfrak{a}$ *is an equivalence relation on the variables of sort* $\mathtt{l}$ *occurring in* $V_\mathfrak{a}$*,* $A_\mathfrak{a}$ *is a subset of* $V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*, closed under* $\sim_\mathfrak{a}$ *(i.e., for all* $x, y \in \mathcal{V}_\mathtt{l}$*:* $x \in A_\mathfrak{a} \wedge x \sim_\mathfrak{a} y \implies y \in A_\mathfrak{a}$*), and* $\rho_\mathfrak{a}$ *is a permission formula (with variables in* $V_\mathfrak{a}$*).*

Definition 29. *Let* $(\mathfrak{s}, \mathfrak{h})$ *be a structure and let* $\mathfrak{a} = (V_\mathfrak{a}, \sim_\mathfrak{a}, A_\mathfrak{a}, \rho_\mathfrak{a})$ *be a heap abstraction. We write* $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} \mathfrak{a}$ *if all the following conditions are satisfied: (i) For all variables* $x, y \in V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*:* $x \sim_\mathfrak{a} y \iff \mathfrak{s}(x) = \mathfrak{s}(y)$*; (ii) for all* $x \in V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*,* $x \in A_\mathfrak{a} \iff \mathfrak{s}(x) \in dom(\mathfrak{h})$*; and (iii)* $\mathfrak{s} \models^{\mathfrak{P}} \rho_\mathfrak{a}$*. A heap abstraction is* $\mathfrak{P}$-satisfiable *if there exists a structure* $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} \mathfrak{a}$*.*

Proposition 30. *A heap abstraction* $\mathfrak{a}$ *is* $\mathfrak{P}$*-satisfiable iff* $\rho_\mathfrak{a}$ *is* $\mathfrak{P}$*-satisfiable.*

For all $\circ$-formulas ϕ, we define a set of heap abstractions $\mathfrak{A}(\phi)$ by mutual induction as follows. The sets $\mathfrak{A}(\phi)$ are the least sets of heap abstractions satisfying the following properties, for all finite sets of variables[9] $V \supseteq fv(\phi)$ and for all equivalence relations $\sim$ on $V \cap \mathcal{V}_\mathtt{l}$: (i) if $\phi = x \overset{p}{\mapsto} (y_1, \dots, y_n)$ then $(V, \sim, \{y \mid y \mid y \sim x\}, def(p)) \in \mathfrak{A}(\phi)$. (ii) if $\phi = x \simeq y$ (resp. $x \not\simeq y$) with $x, y \in \mathcal{V}_\mathtt{l}$ and $x \sim y$ (resp. $x \not\sim y$) then $(V, \sim, \emptyset, \mathtt{emp}) \in \mathfrak{A}(\phi)$; (iii) if ϕ is a permission formula then $(V, \sim, \emptyset, \phi) \in \mathfrak{A}(\phi)$; (iv) if $\phi = \exists x\, \psi$, $(V, \sim, A, \rho) \in \mathfrak{A}(\psi)$ then $(V \setminus \{x\}, \sim', A \setminus \{x\}, \rho) \in \mathfrak{A}(\phi)$, where $\sim'$ denotes the restriction of $\sim$ to the variables distinct from x, i.e., $\sim' \overset{\text{def}}{=} \{(u, v) \mid u \sim v \wedge u, v \neq x\}$ (note that x cannot occur in ρ, since quantification over permission variables is not allowed); (v) if $\phi = \phi_1 \circ \phi_2$, $(V, \sim, A_i, \rho_i) \in \mathfrak{A}(\phi_i)$ (for all $i = 1, 2$) with $A_1 \cap A_2 = \emptyset$, then $(V, \sim, A_1 \cup A_2, \rho_1 \circ \rho_2) \in \mathfrak{A}(\phi)$; (vi) if $\phi = P(\boldsymbol{x}, \boldsymbol{p})$ and $\phi \Leftarrow_\mathcal{R} \xi$ then $\mathfrak{A}(\xi) \subseteq \mathfrak{A}(\phi)$.

Lemma 31. *A* $\circ$*-formula* ϕ *is* $(\mathcal{R}, \mathfrak{P})$*-satisfiable iff at least one of the abstractions in* $\mathfrak{A}(\phi)$ *is* $\mathfrak{P}$*-satisfiable.*

Putting things together we get the following result:

Theorem 32. *If* $\mathfrak{P}$*-satisfiability is decidable for permission formulas, then there exists an algorithm that, for every* $\exists$*-restricted SID, decides whether a given formula* ϕ *is* $(\mathcal{R}, \mathfrak{P})$*-satisfiable. If, moreover,* $\mathfrak{P}$*-satisfiability is in* Exptime*, then* $(\mathcal{R}, \mathfrak{P})$*-satisfiability is also in* Exptime *(for* $\exists$*-restricted SID). Finally, for every permission model* $\mathfrak{P}$*,* $(\mathcal{R}, \mathfrak{P})$*-satisfiability is* Exptime*-hard (for* $\exists$*-restricted SID).*

5 Using Separating Conjunctions Inside Rules

To end the paper, we wish to point out that the satisfiability problem is undecidable from $\exists$-restricted SID if the disjoint separation $\circ$ is replaced by the standard

[9] For technical convenience we do not impose any bound on the cardinality of V, hence the set $\mathfrak{A}(\phi)$ is infinite. This simplifies the theoretical definition of the abstraction for disjoint conjunctions. In practice only variables occurring in the initial formula or in the rules need to be considered.

separating connective $*$ in the inductive definitions (see Definition 7). We think that the result is of some theoretical interest, although, as explained above, rules using $\circ$ are actually more convenient for describing data structures. The notions of $*$-$\mathfrak{h}$-*regular* and $*$-$\exists$-*restricted* SID are defined exactly as $\mathfrak{h}$-regular SID and $\exists$-restricted SID (Definitions 7 and 10) except that the symbol $\circ$ is replaced by $*$ everywhere (for conciseness the formal definitions are omitted).

Theorem 33. *Let $\mathfrak{P}$ be any permission model and assume that for every $n \in \mathbb{N}$, there exists $\pi \in \mathcal{P}_{\mathfrak{P}}$ such that π^n is defined. The $(\mathcal{R}, \mathfrak{P})$-satisfiability problem is undecidable for $*$-$\exists$-restricted SID.*

6 Conclusion and Future Work

An algorithm was devised to test the satisfiability of symbolic heaps in Separation Logic with inductively defined predicates and permissions, under some (syntactic) conditions on the inductive rules giving the semantics of the spatial predicates. The algorithm runs in exponential time, provided the satisfiability of permission formulas is in Exptime. In addition, we showed that some natural relaxings of these conditions make the problem undecidable (under some minimal assumptions on the permission model). The next step is to investigate the entailment problem for the considered fragment. The techniques devised in the present paper for transforming symbolic heaps into disjoint conjunctions of atoms should serve as a basis for this purpose, but the extension is not straightforward. Another (much easier) extension that could be of practical relevance is to consider formulas with labels (in the sense of [5]) which allow one to express additional equality conditions on some parts of the structures. In our context, labels would simply yield additional conditions on the decomposition generated during the normalization step: two formulas sharing the same label should be decomposed into formulas with the same set of roots. It could also be interesting to relax some of the conditions on the rules, for instance to allow for existential variables not occurring as roots in the rules. This is required to encode data structures with forward pointers, such as skip lists. It is also unclear whether Condition 4 in Definition 10 is required for decidability. Finally, the decision algorithm could probably be extended to handle arbitrary combinations of disjoint and separating conjunctions.

Acknowledgments. This work has been partially funded by the French National Research Agency (`ANR-21-CE48-0011`)

References

1. Berdine, J., Calcagno, C., O'Hearn, P.W.: Smallfoot: modular automatic assertion checking with separation logic. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 115–137. Springer, Heidelberg (2006). https://doi.org/10.1007/11804192_6

2. Berdine, J., Cook, B., Ishtiaq, S.: SLAyer: memory safety for systems-level code. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 178–183. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_15
3. Bornat, R., Calcagno, C., O'Hearn, P.W., Parkinson, M.J.: Permission accounting in separation logic. In: Palsberg, J., Abadi, M., (eds.) Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, 12–14 January 2005, pp. 259–270. ACM (2005)
4. Boyland, J.: Fractional permissions. In: Clarke, D., Noble, J., Wrigstad, T. (eds.) Aliasing in Object-Oriented Programming. Types, Analysis and Verification. LNCS, vol. 7850, pp. 270–288. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36946-9_10
5. Brotherston, J., Costa, D., Hobor, A., Wickerson, J.: Reasoning over permissions regions in concurrent separation logic. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12225, pp. 203–224. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_13
6. Brotherston, J., Fuhs, C., Pérez, J.A.N., Gorogiannis, N.: A decision procedure for satisfiability in separation logic with inductive predicates. In: Henzinger, T.A., Miller, D. (eds.), Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS 2014, Vienna, Austria, 14–18 July 2014, pp. 25:1–25:10. ACM (2014)
7. Calcagno, C., Distefano, D.: Infer: an automatic program verifier for memory safety of C programs. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 459–465. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_33
8. Calcagno, C., O'Hearn, P.W., Yang, H.: Local action and abstract separation logic. In 22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10–12 July 2007, Wroclaw, Poland, Proceedings, pp. 366–378. IEEE Computer Society (2007)
9. Comon, H., et al.: Tree automata techniques and applications (1997). http://www.grappa.univ-lille3.fr/tata
10. Demri, S., Lozes, É., Lugiez, D.: On symbolic heaps modulo permission theories. In: Lokam, S.V., Ramanujam, R., (eds.), 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, 11–15 December 2017, Kanpur, India, vol. 93 of LIPIcs, pp. 25:1–25:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
11. Echenim, M., Iosif, R., Peltier, N.: Entailment checking in separation logic with inductive definitions is 2-exptime hard. In: Albert, E., Kovács, L., (eds.) LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, 22–27 May 2020, vol. 73 of EPiC Series in Computing, pp. 191–211. EasyChair (2020)
12. Echenim, M., Iosif, R., Peltier, N.: Decidable entailments in separation logic with inductive definitions: beyond establishment. In: CSL 2021: 29th International Conference on Computer Science Logic, EPiC Series in Computing. EasyChair (2021)
13. Iosif, R., Rogalewicz, A., Simacek, J.: The tree width of separation logic with recursive definitions. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 21–38. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_2
14. Ishtiaq, S.S., O'Hearn, P.W.: BI as an assertion language for mutable data structures. In: ACM SIGPLAN Notices, vol. 36, pp. 14–26 (2001)

15. Katelaan, J., Zuleger, F.: Beyond symbolic heaps: deciding separation logic with inductive definitions. In: Albert, E., Kovács, L., (eds.), LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, 22–27 May 2020. vol. 73 of EPiC Series in Computing, pp. 390–408. EasyChair (2020)
16. Le, Q.L.: Compositional satisfiability solving in separation logic. In: Henglein, F., Shoham, S., Vizel, Y. (eds.) VMCAI 2021. LNCS, vol. 12597, pp. 578–602. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67067-2_26
17. Nakazawa, K., Tatsuta, M., Kimura, D., Yamamura, M.: Cyclic theorem prover for separation logic by magic wand. In: ADSL 18 (First Workshop on Automated Deduction for Separation Logics). Oxford, United Kingdom (2018)
18. O'Hearn, P.W., Pym, D.J.: The logic of bunched implications. Bull. Symb. Log. **5**(2), 215–244 (1999)
19. Navarro Pérez, J.A., Rybalchenko, A.: Separation logic modulo theories. In: Shan, C. (ed.) APLAS 2013. LNCS, vol. 8301, pp. 90–106. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03542-0_7
20. Piskac, R., Wies, T., Zufferey, D.: Automating separation logic using SMT. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 773–789. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_54
21. Qiu, X., Garg, P., Stefanescu, A., Madhusudan, P.: Natural proofs for structure, data, and separation. In: Boehm, H., Flanagan, C., (eds.) ACM SIGPLAN PLDI 2013, pp. 231–242. ACM (2013)
22. Reynolds, J.: Separation logic: a logic for shared mutable data structures. In: Proceedings of the LICS 2002 (2002)
23. Xu, Z., Chen, T., Wu, Z.: Satisfiability of compositional separation logic with tree predicates and data constraints. In: de Moura, L. (ed.) CADE 2017. LNCS (LNAI), vol. 10395, pp. 509–527. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63046-5_31

First-Order Logics

Nested Sequents for Quantified Modal Logics

Tim S. Lyon[1](✉) and Eugenio Orlandelli[2]

[1] Institute of Artificial Intelligence, TU Dresden, Dresden, Germany
timothy_stephen.lyon@tu-dresden.de

[2] Department of the Arts, University of Bologna, Bologna, Italy
eugenio.orlandelli@unibo.it

Abstract. This paper studies nested sequents for quantified modal logics. In particular, it considers extensions of the propositional modal logics definable by the axioms **D**, **T**, **B**, **4**, and **5** with varying, increasing, decreasing, and constant domains. Each calculus is proved to have good structural properties: weakening and contraction are height-preserving admissible and cut is (syntactically) admissible. Each calculus is shown to be equivalent to the corresponding axiomatic system and, thus, to be sound and complete. Finally, it is argued that the calculi are internal—i.e., each sequent has a formula interpretation—whenever the existence predicate is expressible in the language.

Keywords: Cut elimination · Nested sequent · Quantified modal logic

1 Introduction

Generalisations of Gentzen-style sequent calculi have proven useful for developing cut-free and analytic proof systems for many propositional non-classical logics, including modal and intermediate ones. Among these generalisations are *display calculi* [2], *hypersequents* [1], *labelled calculi* [23,25], and *nested sequents* [5,12]. They often allow one to give constructive proofs of important meta-theoretical properties such as decidability [3], interpolation [9], and automatic countermodel extraction [16]. These systems generalise the structural level of Gentzen-style calculi in different ways in order to express wider classes of logics. In the case of propositional modal logics they can express the structure of various relational models. In particular, nested sequents encode tree-like relational models and labelled calculi encode graph-like models. In contrast to other formalisms (e.g. labelled sequents) nested sequents have the advantage of being internal calculi: each nested sequent has a formula interpretation, and thus, such expressions are not a major departure from the modal language.

Things become more difficult when we add the quantifiers. As is well known [7,10], in quantified modal logics (QMLs) we have *interaction formulas* such as

$$\mathbf{CBF} := \Box \forall x A \supset \forall x \Box A \qquad \text{and} \qquad \mathbf{BF} := \forall x \Box A \supset \Box \forall x A$$

Tim S. Lyon was supported by the European Research Council (ERC) Consolidator Grant 771779 (DeciGUT).

whose validity depends on the interrelations between the domains of quantification ($\mathcal{D}_w$) of the different worlds (w) of the model: **CBF** is valid only if domains are *increasing*—$w\mathcal{R}v$ implies $\mathcal{D}_w \subseteq \mathcal{D}_v$—and **BF** is valid only if domains are *decreasing*—$w\mathcal{R}v$ implies $\mathcal{D}_w \supseteq \mathcal{D}_v$. Axiomatically, **CBF** is derivable from the interaction of the axioms/rules for modalities and those for the classical quantifiers, and **BF** is independent from them. However, the situation is radically different for sequent calculi than for axiomatic calculi. The problem is that **BF** becomes derivable when we add standard sequent rules for the quantifiers to a calculus having separated left and right rules for the modalities—i.e., it is derivable in all generalisations of Gentzen-style calculi mentioned above.

To overcome this issue for nested sequents, we employ a formulation technique motivated by labelled sequent calculi. One way of making **CBF** and **BF** independent of the rules for quantifiers within labelled sequent calculi is to extend the language with *domain atoms* of shape $y \in D(w)$ whose intended meaning is that 'y belong to the quantificational domain of the label w' [20,25]. In this way, one can restrict the rules for the quantifiers to the terms belonging to the domain of the label under consideration:

$$\frac{w : A(y/x), y \in D(w), w : \forall x A, \Gamma \Rightarrow \Delta}{y \in D(w), w : \forall x A, \Gamma \Rightarrow \Delta} \qquad \frac{z \in D(w), \Gamma \Rightarrow \Delta, w : A(z/x)}{\Gamma \Rightarrow \Delta, w : \forall x A}\ z \text{ fresh}$$

As a consequence, **CBF** and **BF** are derivable only if we extend the basic calculus with rules relating the domains of the distinct labels.

In this paper, we study nested sequent calculi for QMLs with varying, increasing, decreasing, and constant domains. Similar to the use of domain atoms in labelled sequents, we will formulate our nested calculi by extending the syntax of sequents with *signatures*—i.e., multisets of terms that restrict the applicability of the rules for the quantifiers at that node of the nested sequent—as was done in [24] to define hypersequents for Gödel-Dummett logic with non-constant domains. In particular, we will use the following rules for the universal quantifier:

$$\frac{\mathcal{S}\{X, y; A(y/x), \forall x A, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, y; \forall x A, \Gamma \Rightarrow \Delta\}}\ L\forall \qquad \frac{\mathcal{S}\{X, z; \Gamma \Rightarrow \Delta, A(z/x)\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \forall x A\}}\ R\forall,\ z \text{ fresh}$$

and will add signature structural rules for increasing, decreasing, and constant domains (Table 3).

As a consequence, we will be able to define nested calculi that are equivalent to the labelled calculi considered in [25, Ch. 6] and [20, Ch. 12.1]. We will show that our nested calculi have good structural properties—all rules are height-preserving invertible, weakening and contraction are height-preserving admissible, and cut is syntactically admissible—and that they characterise the quantified extensions of the propositional modal logics in the cube of normal modalities. One advantage of the present approach is that nested sequents with signatures have a formula interpretation given that the language can express the *existence predicate* $\mathcal{E}$. In this paper, we will consider a language with identity so that $\mathcal{E}x$ can be expressed as $\exists y(y = x)$ and it need not be taken as an additional primitive symbol; cf. [7]. Thus, our calculi utilise (nested) sequents as expressive as the modal language, showing that our calculi are syntactically economical.

The rest of the paper is organised as follows: Sect. 2 sketches the QMLs considered in the paper, and Sect. 3 introduces the nested calculi for these logics. Then, Sect. 4 shows that these calculi have good structural properties distinctive of G3-style calculi, including syntactic cut-elimination, and Sect. 5 shows that each calculus is sound and compete with respect to its intended semantics. Finally, Sect. 6 presents some future lines of research.

2 Quantified Modal Logics

-Syntax. Let *Rel* be a set containing, for each $n \in \mathbb{N}$, an at most countable set of n-ary predicates R_1^n, $R_2^n, \dots$, and let *Var* be a denumerable set of individual variables. The language $\mathcal{L}$ is defined by the following grammar:

$$A ::= R_i^n(x_1, \dots, x_n) \mid x_1 = x_2 \mid \bot \mid A \supset A \mid \forall x A \mid \Box A \qquad (\mathcal{L})$$

where $x, x_1, \dots, x_n \in Var$ and $R_i^n \in Rel$. An *atomic formula* is a formula of the shape $R_i^n(x_1, \dots, x_n)$ or $x_1 = x_2$. We use the following metavariables: x, y, z for variables; P, Q, R for atomic formulas; and A, B, C for formulas. An occurrence of a variable x in a formula is *free* if it is not in the scope of $\forall x$; otherwise, it is *bound*. A *sentence* is a formula without free occurrences of variables. The formulas $\neg A$, $A \wedge B$, $A \vee B$, $\exists x A$, and $\Diamond A$ are defined as expected. We follow the usual conventions for parentheses. The *weight* of a formula $|A|$ is defined accordingly: $|R_i^n(x_1, \dots, x_n)| = |x = y| = |\bot| = 0$, $|A \supset B| = |A| + |B| + 1$, and $|\forall x A| = |\Box A| = |A| + 1$. We use $A(y/x)$ to denote the formula obtained from A by replacing each free occurrence of x with an occurrence of y, possibly renaming bound variables to avoid capture: if $y \not\equiv x$, then $(\forall y A)(y/x) \equiv \forall z((A(z/y))(y/x))$, where z is fresh.

-Semantics. A *frame* is a triple $\mathcal{F} = \langle \mathcal{W}, \mathcal{R}, \mathcal{D} \rangle$, where:

- $\mathcal{W}$ is a non-empty set of *worlds*;
- $\mathcal{R}$ is a binary *accessibility relation* defined over $\mathcal{W}$;
- $\mathcal{D}$ is a function mapping each $w \in \mathcal{W}$ to a possibly empty set of objects $\mathcal{D}_w$ (the *domain* of w); we impose that $\mathcal{D}$ is such that $\mathcal{D}_v \neq \varnothing$ for some $v \in \mathcal{W}$.

We say that $\mathcal{F}$ has:

1. *increasing domains* if for all $w, v \in \mathcal{W}$, $w\mathcal{R}v$ implies $D_w \subseteq D_v$;
2. *decreasing domains* if for all $w, v \in \mathcal{W}$, $w\mathcal{R}v$ implies $D_w \supseteq D_v$;
3. *constant domains* if for all $w, v \in \mathcal{W}$, $D_w = D_v$;
4. *varying domains* if none of the above conditions hold.

A *model* $\mathcal{M}$ is a frame together with a valuation function $\mathcal{V}$ such that for each $w \in W$ and each R^n in *Rel*, $\mathcal{V}(w, R_n) \subseteq (D_{\mathcal{W}})^n$, where $D_{\mathcal{W}} = \bigcup_{v \in \mathcal{W}} D_v$. An assignment σ is a function mapping each variable to an object in $\mathcal{D}_{\mathcal{W}}$. We let $\sigma^{x \rhd o}$ be the assignment mapping x to $o \in \mathcal{D}_{\mathcal{W}}$, which behaves like σ for all

Table 1. Axioms and corresponding properties

Name	Axiom	Property ($w, v, u \in \mathcal{W}$)	Name	Axiom	Property ($w, v, u \in \mathcal{W}$)
D	$\Box A \supset \Diamond A$	$\forall w \exists u \in \mathcal{W}(w\mathcal{R}u)$	**5**	$\Diamond A \supset \Box\Diamond A$	$\forall w, v, u(w\mathcal{R}v \wedge w\mathcal{R}u \supset v\mathcal{R}u)$
T	$\Box A \supset A$	$\forall w(w\mathcal{R}w)$	**CBF**	$\Box\forall xA \supset \forall x\Box A$	$\forall w, v(w\mathcal{R}v \supset \mathcal{D}_w \subseteq \mathcal{D}_v)$
B	$A \supset \Box\Diamond A$	$\forall w, v(w\mathcal{R}v \supset v\mathcal{R}w)$	**BF**	$\forall x\Box A \supset \Box\forall xA$	$\forall w, v(w\mathcal{R}v \supset \mathcal{D}_w \supseteq \mathcal{D}_v)$
4	$\Box A \supset \Box\Box A$	$\forall w, v, u(w\mathcal{R}v \wedge v\mathcal{R}u \supset w\mathcal{R}u)$	**UI**	$\forall xA \supset A[y/x]$	$\forall w, v(\mathcal{D}_w = \mathcal{D}_v)$

other variables. Observe that variables are *rigid designators* in that their value does not change from one world to another.

The notion of *satisfaction* of a formula A at a world w of a model $\mathcal{M}$ under an assignment σ—to be denoted by $\sigma \Vdash_w^{\mathcal{M}} A$, possibly omitting $\mathcal{M}$—is defined as follows:

$$
\begin{array}{ll}
\sigma \Vdash_w^{\mathcal{M}} R^n(x_1, \ldots, x_n) & \text{iff } \langle \sigma(x_1), \ldots, \sigma(x_n) \rangle \in \mathcal{V}(w, R^n) \\
\sigma \Vdash_w^{\mathcal{M}} x = y & \text{iff } \sigma(x) = \sigma(y) \\
\sigma \not\Vdash_w^{\mathcal{M}} \bot & \\
\sigma \Vdash_w^{\mathcal{M}} A \supset B & \text{iff } \sigma \not\Vdash_w^{\mathcal{M}} A \text{ or } \sigma \Vdash_w^{\mathcal{M}} B \\
\sigma \Vdash_w^{\mathcal{M}} \forall xA & \text{iff for each } o \in \mathcal{D}_w,\ \sigma^{x \triangleright o} \Vdash_w^{\mathcal{M}} A \\
\sigma \Vdash_w^{\mathcal{M}} \Box A & \text{iff for each } v \in \mathcal{W},\ w\mathcal{R}v \text{ implies } \sigma \Vdash_v^{\mathcal{M}} A
\end{array}
$$

The notions of *truth at a world* w ($\Vdash_w^{\mathcal{M}} A$), *truth in a model* $\mathcal{M}$ ($\Vdash^{\mathcal{M}} A$), *validity in a frame* $\mathcal{F}$ ($\mathcal{F} \Vdash A$), and validity in class $\mathcal{C}$ of frames ($\mathcal{C} \Vdash A$) are defined as usual. It is well-known that the formula:

CBF:$= \Box\forall xA \supset \forall x\Box A$ is valid over frames with increasing domains;
BF:$= \forall x\Box A \supset \Box\forall xA$ is valid over frames with decreasing domains;
UI:$= \forall xA \supset A(y/x)$ is valid over frames with constant domains.

Over frames with non-constant domains the valid theory of quantification is that of positive free logic instead of that of classical logic. This means that the axiom **UI** is replaced by the weaker axiom $\mathbf{UI}^\circ := \forall y(\forall xA \supset A(y/x))$. If we extend the language with an *existence predicate* $\mathcal{E}$—whose satisfaction clause is $\sigma \models_w^{\mathcal{M}} \mathcal{E}x$ iff $\sigma(x) \in \mathcal{D}_w$—then we have the following weaker form of UI that is valid $\mathbf{UI}^{\mathcal{E}} := \forall xA \wedge \mathcal{E}y \supset A(y/x)$. Over the language $\mathcal{L}$ the formula $\mathcal{E}x$ can be defined as $\exists y(y = x)$, but over an identity-free language the existence predicate has to be taken as an additional primitive symbol. This distinction has an impact on the calculi introduced in the next section: nested sequents have a formula interpretation when $\mathcal{E}$ is expressible in the language.

-Logics. A *QML* is defined to be the set of all formulas that are valid in some given class of frames. In this paper, we consider logics that are defined by imposing combinations of the properties in Table 1. We use Q.L for a generic logic and we say that a formula is Q.L-*valid* if it belong to the logic Q.L. The formulas that

Table 2. Axiomatisation of $\mathsf{Q.K}$.

TAUT. Propositional tautologies	**REF.** $x = x$
K. $\Box(A \supset B) \supset (\Box A \supset B)$	**REPL.** $x = y \wedge A(x/z) \supset A(y/z)$
$\mathbf{UI}^{\circ}$. $\forall y(\forall x A \supset A(y/x))$	**ND.** $x \neq y \supset \Box(x \neq y)$
$\forall$ **-COMM.** $\forall x \forall y A \supset \forall y \forall x A$	
$\forall$**-DIST.** $\forall x(A \supset B) \supset (\forall x A \supset \forall x B)$	**MP.** If A and $A \supset B$ are theorem so is B
$\forall$**-VAQ.** $A \supset \forall x A$, if x is not free in A	**N.** If A is a theorem so is $\Box A$
	UG. If A is a theorem so is $\forall x A$

are valid over the class of all frames is called $\mathsf{Q.K}$ and it is axiomatised by the axioms and rules given in Table 2. We notice that $\mathbf{UI}^{\mathcal{E}}$ is a theorem of $\mathsf{Q.K}$, see [7, Lem. 2.1(iii)]. The additional axioms for the logics extending $\mathsf{Q.K}$ are given in Table 1. We follow the usual conventions for naming logics—e.g., $\mathsf{Q.S4} \oplus \mathsf{CBF}$ is the set of formulas that are valid over all reflexive and transitive frames with increasing domains and it is axiomatised by adding axioms **T**, **4**, and **CBF** to $\mathsf{Q.K}$. We will not distinguish between a logic and its axiomatisation. This is justified by the following theorem.

Theorem 1 ([7]). *A formula is a theorem of* $\mathsf{Q.L}$ *if and only if it is* $\mathsf{Q.L}$*-valid.*

3 Nested Calculi for QML

A *sequent* is an expression $X; \Gamma \Rightarrow \Delta$ where X is a multiset of variables, called a *signature*, and Γ, Δ are multisets of formulas of the language $\mathcal{L}$. The signature of a sequent is a syntactic counterpart of the existence atoms used in calculi where **UI** is replaced by $\mathbf{UI}^{\circ}$ or $\mathbf{UI}^{\mathcal{E}}$, see [19]. *Nested sequents* are defined as follows:

$$\mathcal{S} ::= \ X; \Gamma \Rightarrow \Delta \mid \mathcal{S}, [\mathcal{S}], \ldots, [\mathcal{S}]$$

A nested sequent $\mathcal{S}$ codifies the tree of sequents $\mathtt{tr}(\mathcal{S})$, as shown in Fig. 1.

$$\begin{tikzcd}
\mathtt{tr}(\mathcal{S}_1) & \ldots & \mathtt{tr}(\mathcal{S}_n) \\
 & \ldots & \\
 & X; \Gamma \Rightarrow \Delta \arrow[uul] \arrow[uur] &
\end{tikzcd}$$

Fig. 1. The tree of the sequent $X; \Gamma \Rightarrow \Delta, [\mathcal{S}_1], \ldots, [\mathcal{S}_n]$.

Substitution of free variables are extended to (nested) sequents and to multisets of formulas by applying them component-wise. The formula interpretation of a sequent is defined as follows:

$$\mathtt{fm}(X; \Gamma \Rightarrow \Delta) \equiv \bigwedge_{x \in X} \mathcal{E}x \wedge \bigwedge \Gamma \supset \bigvee \Delta$$

where $\mathcal{E}x$ is short for the formula $\exists y(y = x)$ and an empty conjunction (disjunction) is $\top$ ($\bot$, resp.). To provide a formula reading of nested sequents over the identity-free language we could add $\mathcal{E}$ to the language or interpret formulas via their universal closure. In the latter case, for example, the formula interpretation of a sequent would be $\mathtt{fm}(X; \Gamma \Rightarrow \Delta) \equiv \forall x \in X(\bigwedge \Gamma \supset \bigvee \Delta)$, and it seems our nested calculi would capture the QMLs in [13].[1] Nonetheless, we believe there are independent reasons for studying QMLs over a language containing identity; cf. [7,10]. The formula interpretation of a nested sequent is defined recursively as:

$$\mathtt{fm}(X; \Gamma \Rightarrow \Delta, [\mathcal{S}_1], \dots, [\mathcal{S}_n]) \equiv (\bigwedge_{x \in X} \mathcal{E}x \wedge \bigwedge \Gamma \supset \bigvee \Delta) \vee \bigvee_{k=1}^{n} \Box \mathtt{fm}(\mathcal{S}_k)$$

Rules are based on the notion of a *hole* $\{\cdot\}$, which is a placeholder for a subtree of (the tree of) a nested sequent and, thus, allows one to apply a rule at an arbitrary node in the tree of a nested sequent. A *context* is defined as follows:

$$\mathcal{C} ::= X; \Gamma \Rightarrow \Delta, \{\cdot\}, \dots, \{\cdot\} \mid \mathcal{C}, [\mathcal{C}], \dots, [\mathcal{C}]$$

In other words, a context $\mathcal{C}$ is a nested sequent with $n \geq 0$ hole occurrences, which do not occur inside formulas and must occur within consequent position. We hitherto write contexts as $\mathcal{S}\{\cdot\} \cdots \{\cdot\}$ indicating each of the holes occurring within the context. The *depth* of a hole in a context is defined as the height of the branch from that hole to the root (cf. [3]), and we write $\mathit{Depth}(\mathcal{S}\{\cdot\}) \geq n$ for $n \in \mathbb{N}$ to mean that the depth of the hole in $\mathtt{tr}(\mathcal{S}\{\cdot\})$ is n or greater.

We define *substitutions* of nested sequents into contexts recursively on the number and depth of holes in a given context: suppose first that our context is of the form $\mathcal{S}\{\cdot\} \equiv X; \Gamma \Rightarrow \Delta, \{\cdot\}, [\mathcal{S}_1], \dots, [\mathcal{S}_n]$ with a single hole at a depth of 0 and let $\mathcal{S}' \equiv Y, \Pi \Rightarrow \Sigma, [\mathcal{S}'_1], \dots, [\mathcal{S}'_k]$ be a nested sequent. Then,

$$\mathcal{S}\{\mathcal{S}'\} \equiv X, Y; \Pi, \Gamma \Rightarrow \Delta, \Sigma, [\mathcal{S}_1], \dots, [\mathcal{S}_n], [\mathcal{S}'_1], \dots, [\mathcal{S}'_k]$$

If our context is of the form $\mathcal{S}\{\cdot\} \equiv X; \Gamma \Rightarrow \Delta, [\mathcal{S}_1\{\cdot\}], \dots, [\mathcal{S}_n]$ with a single hole at a depth greater then 0, then we recursively define $\mathcal{S}\{\mathcal{S}'\}$ to be the nested sequent $X; \Gamma \Rightarrow \Delta, [\mathcal{S}_1\{\mathcal{S}'\}], \dots, [\mathcal{S}_n]$. This definition extends to a context $\mathcal{S}\{\cdot\} \cdots \{\cdot\}$ with n holes in the expected way, and for nested sequents $\mathcal{S}_1, \dots, \mathcal{S}_n$, we let $\mathcal{S}\{\mathcal{S}_1\} \cdots \{\mathcal{S}_n\}$ denote the nested sequent obtained by replacing, for each $i \in \{1, \dots, n\}$, the i-th hole $\{\cdot\}$ in $\mathcal{S}\{\cdot\} \cdots \{\cdot\}$ with $\mathcal{S}_i$. We may also write $\mathcal{S}\{\mathcal{S}_1\}\{\mathcal{S}_i\}_{i=2}^{n}$ to indicate $\mathcal{S}\{\mathcal{S}_1\} \cdots \{\mathcal{S}_n\}$ more succinctly. Plugging $\emptyset$ into a hole suggests the removal of the hole; for instance, if $\mathcal{S}\{\cdot\}\{\cdot\} \equiv x; A \Rightarrow B, \{\cdot\}, [x, y, B, C \Rightarrow D, \{\cdot\}]$, then $\mathcal{S}\{\cdot\}\{\emptyset\} \equiv x; A \Rightarrow B, \{\cdot\}, [x, y; B, C \Rightarrow D]$.

The rules of the nested calculi for QMLs are given in Table 3. The minimal calculus $\mathsf{NQ.K}$ contains initial sequents, the logical rules, and the rules for identity (rule *Rig* is needed—and is sound—because variables are rigid designators). If $\mathsf{Q.L}$ is an extension of $\mathsf{Q.K}$ as discussed in Sect. 2, then $\mathsf{NQ.L}$ denotes the nested

[1] We thank the anonymous reviewer who suggested this latter possibility.

calculus extending $\mathsf{NQ.K}$ with the rules for the axioms of those logics. Observe that to capture axioms **D**, **CBF**, **BF**, and **UI** we have added structural rules instead of logical ones since the former have a better behaviour.

In [3], Brünnler only considers nested calculi (for propositional modal logics) defined relative to *45-complete sets* of axioms. This restriction is required to ensure that the nested calculi contain all rules required for their completeness. Similarly, in the first-order setting, we only consider nested calculi defined relative to *properly closed* sets of axioms, which is a generalisation of 45-completeness and takes care of the interaction of **B** with **CBF** and **BF** (for example), ensuring the completeness of our nested calculi.

Definition 1 (Properly Closed). *Let* $\mathsf{L} \subseteq \{\mathbf{D}, \mathbf{T}, \mathbf{B}, \mathbf{4}, \mathbf{5}, \mathbf{CBF}, \mathbf{BF}, \mathbf{UI}\}$. *We define* L *to be* properly closed *iff if all* $\mathsf{Q.L}$*-frames satisfy* $X \in \{\mathbf{4}, \mathbf{5}, \mathbf{CBF}, \mathbf{BF}\}$*, then* $X \in \mathsf{L}$*. We define a nested calculus* $\mathsf{NQ.L}$ *to be* properly closed *iff (1)* L *is properly closed, and (2)* $R_{5dom} \in \mathsf{NQ.L}$ *iff* $\mathbf{5} \in \mathsf{L}$ *and* $\{\mathbf{CBF}, \mathbf{BF}\} \cap \mathsf{L} \neq \emptyset$*.*

Remark 1. All nested calculi hitherto considered will be assumed properly closed.

Given a calculus $\mathsf{NQ.L}$, an $\mathsf{NQ.L}$-*derivation* of a nested sequent $\mathcal{S}$ is a tree of nested sequents, whose leaves are initial sequents, whose root is $\mathcal{S}$, and which grows according to the rules of $\mathsf{NQ.L}$. We consider only derivations of *pure sequents*, meaning no variable has both free and bound occurrences and each *eigenvariable* (i.e., a fresh variable participating in an $R\forall$ inference) is distinct. The *height* of an $\mathsf{NQ.L}$-derivation is the number of nodes of one of its longest branches. We say that $\mathcal{S}$ is $\mathsf{NQ.L}$-derivable if there is an $\mathsf{NQ.L}$-derivation of $\mathcal{S}$ or of an alphabetical variant of $\mathcal{S}$. We let $\mathsf{NQ.L} \vdash \mathcal{S}$ denote that $\mathcal{S}$ is $\mathsf{NQ.L}$-derivable. A rule is said to be *(height-preserving) admissible* in $\mathsf{NQ.L}$, if, whenever its premisses are $\mathsf{NQ.L}$-derivable (with height at most n), also its conclusion is $\mathsf{NQ.L}$-derivable (with height at most n). A rule is said to be *(height-preserving) invertible* in $\mathsf{NQ.L}$, if, whenever its conclusion is $\mathsf{NQ.L}$-derivable (with height at most n), each premiss is $\mathsf{NQ.L}$-derivable (with height at most n). For each rule displayed in Table 3, the formulas explicitly displayed in the conclusion are called *principal*, those explicitly displayed in the premisses are called *auxiliary*, and everything else constitutes the *context*.

4 Properties and Cut-Elimination

We now show that our nested calculi satisfy fundamental admissibility and invertibility properties. Ultimately, we will apply these properties in our proof of syntactic cut-elimination.

Lemma 1 (Generalised Initial Sequents). $\mathsf{NQ.L} \vdash \mathcal{S}\{X; A, \Gamma \Rightarrow \Delta, A\}$*, for any arbitrary* $\mathcal{L}$*-formula* A*.*

Proof. By a standard induction on the weight of A. □

Lemma 2. *The sequents* $\mathcal{S}\{\Rightarrow x = x\}$ *and* $\mathcal{S}\{x = y, A(x/z) \Rightarrow A(y/z)\}$ *are* $\mathsf{NQ.L}$*-derivable.* □

Table 3. Nested rules for QML

Initial Sequents: $\mathcal{S}\{X; P, \Gamma \Rightarrow \Delta, P\}$ with P atomic

Logical Rules:

$$\frac{\mathcal{S}\{X;\Gamma \Rightarrow \Delta, A\} \quad \mathcal{S}\{X; B, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; A \supset B, \Gamma \Rightarrow \Delta\}}\ L{\supset} \qquad \frac{\mathcal{S}\{X; A, \Gamma \Rightarrow \Delta, B\}}{\mathcal{S}\{X;\Gamma \Rightarrow \Delta, A \supset B\}}\ R{\supset} \qquad \frac{}{\mathcal{S}\{X; \bot, \Gamma \Rightarrow \Delta\}}\ L\bot$$

$$\frac{\mathcal{S}\{X, z; A(z/x), \forall x A, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, z; \forall x A, \Gamma \Rightarrow \Delta\}}\ L\forall \qquad \frac{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, A(y/x)\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \forall x A\}}\ R\forall,\ y \text{ fresh}$$

$$\frac{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta, [Y; A, \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}\ L\Box \qquad \frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [\emptyset; \Rightarrow A]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \Box A\}}\ R\Box$$

Identity Rules:

$$\frac{\mathcal{S}\{X; x = x, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}\ Ref \qquad \frac{\mathcal{S}\{X; P(y/z), x = y, P(x/z), \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; x = y, P(x/z), \Gamma \Rightarrow \Delta\}}\ Repl$$

$$\frac{\mathcal{S}\{X, x, y; x = y, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, x; x = y, \Gamma \Rightarrow \Delta\}}\ Repl_X \qquad \frac{\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}\{Y; x = y, \Pi \Rightarrow \Sigma\}}{\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}\{Y; \Pi \Rightarrow \Sigma\}}\ Rig$$

Rules for Propositional Axioms:

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [\emptyset; \Rightarrow]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}\ R_D \qquad \frac{\mathcal{S}\{X; A, \Gamma \Rightarrow \Delta, [Y; \Box A, \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Box A, \Pi \Rightarrow \Sigma]\}}\ R_B \qquad \frac{\mathcal{S}\{X; A, \Box A, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta\}}\ R_T$$

$$\frac{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta, [Y; \Box A, \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}\ R_4 \qquad \frac{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta\}\{Y; \Box A, \Pi \Rightarrow \Sigma\}}{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta\}\{Y; \Pi \Rightarrow \Sigma\}}\ R_5,\ Depth(\mathcal{S}\{\cdot\}\{\emptyset\}) \geq 1$$

Rules for Domains:

$$\frac{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}\ R_{cbf} \qquad \frac{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y, x; \Pi \Rightarrow \Sigma]\}}\ R_{bf} \qquad \frac{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}\ R_{ui}$$

$$\frac{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta\}\{Y, x; \Pi \Rightarrow \Sigma\}}{\mathcal{S}\{X, x; \Gamma \Rightarrow \Delta\}\{Y; \Pi \Rightarrow \Sigma\}}\ R_{5dom},\ Depth(\mathcal{S}\{\emptyset\}\{\cdot\}) \geq 1 \text{ and } Depth(\mathcal{S}\{\cdot\}\{\emptyset\}) \geq 1$$

Proof. $\mathcal{S}\{\ \Rightarrow x = x\}$ is derivable by applying an instance of rule *Ref* to the initial sequent $\mathcal{S}\{\ x = x \Rightarrow x = x\}$. The case of $\mathcal{S}\{x = y, A(x/z) \Rightarrow A(y/z)\}$ is handled by induction on $|A(x/z)|$. We consider only the case where $A(x/z) = \Box B(x/z)$.

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\mathcal{S}\{x = y, \Box B(x/z) \Rightarrow\ , [x = y, B(x/z) \Rightarrow B(y/z)]\}}\ IH}{\mathcal{S}\{x = y, \Box B(x/z) \Rightarrow\ , [B(x/z) \Rightarrow B(y/z)\}]}\ Rig}{\mathcal{S}\{x = y, \Box B(x/z) \Rightarrow\ , [\Rightarrow B(y/z)]\}}\ L\Box}{\mathcal{S}\{x = y, \Box B(x/z) \Rightarrow \Box B(y/z)\}}\ R\Box$$

□

Lemma 3. *The following $R\bot$ rule is height-preserving admissible in* NQ.L*:*

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \bot\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}\ R\bot$$

Proof. By a straightforward induction on the height of the derivation $\mathcal{D}$ of the premiss. The proof is almost trivial as any application of $R\bot$ to an initial sequent of an instance of $L\bot$ gives another initial sequent or instance of $L\bot$, respectively, and $R\bot$ permutes above every other rule of $\mathsf{NQ.L}$. □

Lemma 4 (Substitution). *The following rule of substitution of free variables is height-preserving admissible in* $\mathsf{NQ.L}$*:*

$$\frac{\mathcal{S}\{X;\Gamma \Rightarrow \Delta\}}{\mathcal{S}(y/x)\{X(y/x);\Gamma(y/x) \Rightarrow \Delta(y/x)\}}\ (y/x)$$

Proof. By induction on the height of the derivation $\mathcal{D}$ of the premiss. The only interesting case is when the last step of $\mathcal{D}$ is an instance of $R\forall$:

$$\frac{\mathcal{S}\{X, z_2;\Gamma \Rightarrow \Delta, A(z_2/z_1)\}}{\mathcal{S}\{X;\Gamma \Rightarrow \Delta, \forall z_1 A\}}\ R\forall,\ z_2 \text{ fresh}$$

We transform the derivation of the premiss by applying the inductive hypothesis twice to ensure the freshness condition is preserved: the first time to replace z_2 with a fresh variable z_3 and then to replace x with y. We conclude by applying $R\forall$ with z_3 as the *eigenvariable.* □

Typically, admissible structural rules operate on either formulas (e.g., see the internal weakening rule IW below) or nesting structure (e.g., see the Merge rule below) in nested calculi. An interesting observation in the first-order setting is that admissible structural rules also act on the signatures occurring in nested sequents. This gives rise to forms of weakening and contraction for terms, which are reminiscent of analogous rules formulated in the context of hypersequents with signatures [24].

Lemma 5 (Signature Structural Rules). *The following rules of signature weakening and signature contraction are height-preserving admissible in* $\mathsf{NQ.L}$*:*

$$\frac{\mathcal{S}\{X;\Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, x;\Gamma \Rightarrow \Delta\}}\ SW \qquad \frac{\mathcal{S}\{X, x, x;\Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, x;\Gamma \Rightarrow \Delta\}}\ SC$$

Proof. By a standard induction on the height of the derivation $\mathcal{D}$ of the premiss. Proving height-preserving admissibility of SC is trivial as the rule permutes above all rules of $\mathsf{NQ.L}$. Proving the height-preserving admissibility of SW is also straightforward with the only interesting case arising when $\mathcal{D}$ ends with an instance of $R\forall$ with x as the *eigenvariable.* However, this case is easily managed by applying the height-preserving admissible substitution (y/x) to ensure the freshness condition for $R\forall$ is satisfied, followed by the inductive hypothesis, and an application of $R\forall$. □

As in the setting of first-order intuitionistic logics with increasing and constant domains (see [14]), we find that our structural rules for domains give rise to admissible logical rules generalising the $L\forall$ rule. Such rules (presented in the proposition below) combine the functionality of the associated domain structural rules with the $L\forall$ rule. The $L\forall_{bf}$ and $L\forall_{cbf}$ rules are instances of *reachability rules* [16,17], which bottom-up operate by searching for terms along edges in a nested sequent used to instantiate universal formulas.

Proposition 1. *The following logical rules for 'domain-axioms' and for axiom* **D** *are admissible in the nested calculi including the appropriate structural rules for domains or R_D:*

$$\frac{\mathcal{S}\{X; A(y/x), \forall x A, \Gamma \Rightarrow \Delta, [Y, y; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \forall x A, \Gamma \Rightarrow \Delta, [Y, y; \Pi \Rightarrow \Sigma]\}}\ L\forall_{bf} \qquad \frac{\mathcal{S}\{X; A(y/x), \forall x A, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \forall x A, \Gamma \Rightarrow \Delta\}}\ L\forall_{ui}$$

$$\frac{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y; A(y/x), \forall x A, \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y; \forall x A, \Pi \Rightarrow \Sigma]\}}\ L\forall_{cbf} \qquad \frac{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta, [\emptyset; A \Rightarrow\ ]\}}{\mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta\}}\ L_D$$

Proof. The admissibility of $L\forall_{cbf}$ from R_{cbf} and SW is proven as follows:

$$\cfrac{\cfrac{\cfrac{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y; A(y/x), \forall x A, \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y, y; A(y/x), \forall x A, \Pi \Rightarrow \Sigma]\}}\ SW}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y, y; \forall x A, \Pi \Rightarrow \Sigma]\}}\ L\forall}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, [Y; \forall x A, \Pi \Rightarrow \Sigma]\}}\ R_{cbf}$$

The cases of $L\forall_{bf}$ and $L\forall_{ui}$ are similar, and the case of L_D follows immediately from R_D. □

Lemma 6 (Weakenings). *The following rules of internal and external weakening are height-preserving admissible in* NQ.L*:*

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \Pi, \Gamma \Rightarrow \Delta, \Sigma\}}\ IW \qquad \frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}\ EW$$

Proof. By induction on the height of the derivation $\mathcal{D}$ of the premiss. If $\mathcal{D}$ ends with an instance of rule $R\forall$ with y the *eigenvariable*, we apply the (height-preserving admissible) substitution rule to replace y with a fresh variable z occurring neither in $\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}$, nor in Π, Σ (in the IW case) or in Y, Π, Σ (in the EW case). Then, we apply the inductive hypothesis and an instance of $R\forall$ to conclude $\mathcal{S}\{X; \Pi, \Gamma \Rightarrow \Delta, \Sigma\}$ in the IW case and $\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}$ in the EW case. □

Lemma 7 (Necessitation and Merge). *The following rules are height-preserving admissible in* N.QL*:*

$$\frac{\mathcal{S}}{\Rightarrow, [\mathcal{S}]}\ Nec \qquad \frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi_1 \Rightarrow \Delta_1], [Z; \Pi_2 \Rightarrow \Delta_2]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y, Z; \Pi_1, \Pi_2 \Rightarrow \Delta_1, \Delta_2]\}}\ Merge$$

Proof. By a simple induction on the height of the derivation of the premiss. □

Lemma 8 (Invertibility). *Each rule of* NQ.L *is height-preserving invertible.*

Proof. The proof is by induction on the height of the derivation. The height-preserving invertibility of all rules but $L\supset$, $R\supset$, $R\forall$ and $R\Box$ follows from Lemmas 5 and 6, and the proof of the remaining cases is standard. □

Lemma 9 (Contraction). *The following rules of left and right contraction are height-preserving admissible in* $\mathsf{NQ.L}$*:*

$$\frac{\mathcal{S}\{X; \Gamma, A, A \Rightarrow \Delta\}}{\mathcal{S}\{X; \Gamma, A \Rightarrow \Delta\}}\ CL \qquad \frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, A, A\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, A\}}\ CR$$

Proof. By simultaneous induction on the height of the derivation of the premisses of CL and CR. We consider only the non-trivial $R\forall$ case for CR as the remaining cases are similar or simpler. Assume that the last step of $\mathcal{D}$ is:

$$\frac{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, A(y/x), \forall x A\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \forall x A, \forall x A\}}\ R\forall$$

To resolve the case, we apply the height-preserving invertibility of $R\forall$, the height-preserving admissibility of (y/z) and SC, followed by the inductive hypothesis. Finally, an application of $R\forall$ gives the desired conclusion.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, A(y/x), \forall x A\}}{\mathcal{S}\{X, y, z; \Gamma \Rightarrow \Delta, A(y/x), A(z/x)\}}\ \text{Lemma 8}}{\mathcal{S}\{X, y, y; \Gamma \Rightarrow \Delta, A(y/x), A(y/x)\}}\ (y/z)}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, A(y/x), A(y/x)\}}\ SC}{\mathcal{S}\{X, y; \Gamma \Rightarrow \Delta, A(y/x)\}}\ IH}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \forall x A\}}\ R\forall$$

□

Due to the presence of R_4 and R_5 in specific nested calculi, our cut elimination theorem (Theorem 2 below) requires us to simultaneously eliminate a second form of cut that acts on modal formulas. We refer to this rule as L-Cut and note that it is essentially Brünnler's Y-cut rule [3]. Since the principal and auxiliary formulas of R_4 and R_5 are of the same weight (i.e. both are $\Box A$), L-Cut is needed to permute the cut upward in these special cases as cuts cannot be reduced to formulas of a smaller weight.

Definition 2 (L-Cut and L-Str). *Let* $\mathsf{NQ.L}$ *be properly closed. We define* L-*Cut to be the following rule:*

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \Box A\}\{Y_i; \Pi_i \Rightarrow \Sigma_i\}_{i=1}^{n} \qquad \mathcal{S}\{X; \Box A, \Gamma \Rightarrow \Delta\}\{Y_i; \Box A, \Pi_i \Rightarrow \Sigma_i\}_{i=1}^{n}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}\{Y_i; \Pi_i \Rightarrow \Sigma_i\}_{i=1}^{n}}\ \mathsf{L}\text{-}Cut$$

which is subject to the following side conditions:

- *if* $\mathbf{4}, \mathbf{5} \notin \mathsf{L}$*, then* $n = 0$*;*
- *if* $\mathbf{4} \in \mathsf{L}$ *and* $\mathbf{5} \notin \mathsf{L}$*, then* $\mathcal{S}\{\cdot\}\{\cdot\}$ *is of the form* $\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \{\cdot\}, \{\mathcal{S}_1\{\cdot\}^n\}\}$*;*
- *if* $\mathbf{5} \in \mathsf{L}$ *and* $\mathbf{4} \notin \mathsf{L}$*, then* $Depth(\mathcal{S}\{\cdot\}\{\emptyset\}^n) \geq 1$*;*
- *otherwise, if* $\mathbf{4}, \mathbf{5} \in \mathsf{L}$*, then no restriction on the shape of the rule is enforced.*

Table 4. Structural rules for propositional axioms

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X, Y; \Pi, \Gamma \Rightarrow \Delta, \Sigma\}}\, S_T \qquad \frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [\emptyset; \Rightarrow, [Y; \Pi \Rightarrow \Sigma]]\}}\, S_4$$

$$\frac{\mathcal{S}\{Y_1; \Pi_1 \Rightarrow \Sigma_1, [X; \Gamma \Rightarrow \Delta]\}\{Y_2; \Pi_2 \Rightarrow \Sigma_2\}}{\mathcal{S}\{Y_1; \Pi_1 \Rightarrow \Sigma_1\}\{Y_2; \Pi_2 \Rightarrow \Sigma_2, [X; \Gamma \Rightarrow \Delta]\}}\, S_5,\, Depth(\mathcal{S}\{\cdot\}\{\emptyset\}) \geq 1$$

$$\frac{\mathcal{S}\{X; \Gamma \Rightarrow \Delta, [Y; \Pi_2 \Rightarrow \Sigma_2, [Z; \Pi_1 \Rightarrow \Sigma_1]]\}}{\mathcal{S}\{X, Z; \Pi_1, \Gamma \Rightarrow \Delta, \Sigma_1, [Y; \Pi_2 \Rightarrow \Sigma_2]\}}\, S_B$$

We define L-*Str to be the following rule:*

$$\frac{\mathcal{S}\{Y_1; \Pi_1 \Rightarrow \Sigma_1, [X; \Gamma \Rightarrow \Delta]\}\{Y_2; \Pi_2 \Rightarrow \Sigma_2\}}{\mathcal{S}\{Y_1; \Pi_1 \Rightarrow \Sigma_1\}\{Y_2; \Pi_2 \Rightarrow \Sigma_2, [X; \Gamma \Rightarrow \Delta]\}}\, \mathsf{L}\text{-}Str$$

which is subject to the following side conditions:

- *if* $\mathbf{4}, \mathbf{5} \notin \mathsf{L}$*, then* $\mathcal{S}\{\cdot\}\{\cdot\}$ *is of the form* $\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \{\cdot\}, \{\cdot\}\}$*;*
- *if* $\mathbf{4} \in \mathsf{L}$ *and* $\mathbf{5} \notin \mathsf{L}$*, then* $\mathcal{S}\{\cdot\}\{\cdot\}$ *is of the form* $\mathcal{S}\{X; \Gamma \Rightarrow \Delta, \{\cdot\}, \{\mathcal{S}_1\{\cdot\}\}\}$*;*
- *if* $\mathbf{5} \in \mathsf{L}$ *and* $\mathbf{4} \notin \mathsf{L}$*, then* $Depth(\mathcal{S}\{\cdot\}\{\emptyset\}) \geq 1$*;*
- *otherwise, if* $\mathbf{4}, \mathbf{5} \in \mathsf{L}$*, then no restriction on the shape of the rule is enforced.*

Lemma 10 (Special Structural Rules). *If* $\mathsf{NQ.L}$ *contains the rule* R_X *for the propositional axiom* X*, then the corresponding structural rule from Table 4 is admissible in* $\mathsf{NQ.L}$*. Moreover,* L-*Str is admissible in* $\mathsf{NQ.L}$*.*

Proof. We argue the S_B case by induction on the height of the given derivation; the remaining cases are considered in the appended version of this paper [18]. We only consider the R_{bf} and R_{5dom} cases of the inductive step as the remaining cases are simple or similar.

$$\dfrac{\dfrac{\mathcal{S}\{Z; \Pi_1 \Rightarrow \Sigma_1, [X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi_2 \Rightarrow \Sigma_2]]\}}{\mathcal{S}\{Z; \Pi_1 \Rightarrow \Sigma_1, [X; \Gamma \Rightarrow \Delta, [Y, x; \Pi_2 \Rightarrow \Sigma_2]]\}}\, R_{bf}}{\mathcal{S}\{Z, Y, x; \Pi_1, \Pi_2 \Rightarrow \Sigma_1, \Sigma_2, [X; \Gamma \Rightarrow \Delta]\}}\, S_B$$

As our nested calculi are assumed to be properly closed, we know that if $\mathsf{NQ.L}$ contains R_B and R_{bf}, then it must contain R_{cbf}, showing that we can apply IH first and then R_{cbf} as shown below.

$$\dfrac{\dfrac{\mathcal{S}\{Z; \Pi_1 \Rightarrow \Sigma_1, [X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi_2 \Rightarrow \Sigma_2]]\}}{\mathcal{S}\{Z, Y, x; \Pi_1, \Pi_2 \Rightarrow \Sigma_1, \Sigma_2, [X, x; \Gamma \Rightarrow \Delta]\}}\, IH}{\mathcal{S}\{Z, Y, x; \Pi_1, \Pi_2 \Rightarrow \Sigma_1, \Sigma_2, [X; \Gamma \Rightarrow \Delta]\}}\, R_{cbf}$$

Last, we consider an interesting R_{5dom} case:

$$\dfrac{\dfrac{Z; \varPi_1 \Rightarrow \varSigma_1, [X_1; \varGamma_1 \Rightarrow \varDelta_1, [X_2, x; \varGamma_2 \Rightarrow \varDelta_2]], [\mathcal{S}\{Y, x; \varPi_2 \Rightarrow \varSigma_2\}]}{Z; \varPi_1 \Rightarrow \varSigma_1, [X_1; \varGamma_1 \Rightarrow \varDelta_1, [X_2, x; \varGamma_2 \Rightarrow \varDelta_2]], [\mathcal{S}\{Y; \varPi_2 \Rightarrow \varSigma_2\}]}\, R_{5dom}}{Z, X_2, x; \varPi_1, \varGamma_2 \Rightarrow \varSigma_1, \varDelta_2, [X_1, \varGamma_1 \Rightarrow \varDelta_1], [\mathcal{S}\{Y; \varPi_2 \Rightarrow \varSigma_2\}]}\, S_B$$

To resolve the case, we apply the inductive hypothesis, followed by the height-preserving admissible rule SW. We apply the SW rule $n-1$ times adding the variable x along the path from the root to $Y, x; \varPi_2 \Rightarrow \varSigma_2$, and then the R_{cbf} rule n times to delete the $n-1$ copies of x up to the root. We may apply R_{cbf} as our nested calculi are properly closed, that is, $\mathbf{B}, \mathbf{BF} \in \mathsf{L}$ only if $\mathbf{CBF} \in \mathsf{L}$.

$$\dfrac{\dfrac{\dfrac{Z; \varPi_1 \Rightarrow \varSigma_1, [X; \varGamma_1 \Rightarrow \varDelta_1, [X_2, x; \varGamma_2 \Rightarrow \varDelta_2]], [\mathcal{S}\{Y, x; \varPi_2 \Rightarrow \varSigma_2\}]}{Z, X_2, x; \varPi_1, \varGamma_2 \Rightarrow \varSigma_1, \varDelta_2, [X, \varGamma_1 \Rightarrow \varDelta_1], [\mathcal{S}\{Y, x; \varPi_2 \Rightarrow \varSigma_2\}]}\, IH}{Z, X_2, x; \varPi_1, \varGamma_2 \Rightarrow \varSigma_1, \varDelta_2, [X, \varGamma_1 \Rightarrow \varDelta_1], [\mathcal{S}\{Y, x; \varPi_2 \Rightarrow \varSigma_2\}]}\, SW\ (n-1 \text{ times})}{Z, X_2, x; \varPi_1, \varGamma_2 \Rightarrow \varSigma_1, \varDelta_2, [X, \varGamma_1 \Rightarrow \varDelta_1], [\mathcal{S}\{Y; \varPi_2 \Rightarrow \varSigma_2\}]}\, R_{cbf}\ (n \text{ times})$$

$\square$

In our cut-elimination theorem below, we provide a procedure to eliminate an additive (i.e. context-sharing) version of cut as in the work on nested sequents for propositional modal logics by Brünnler [3]. We note that we could have considered an equivalent, multiplicative (i.e. context-independent) version—like the cut rule shown eliminable in the tree-hypersequent systems of Poggiolesi [22]—however, we find the additive version of the rule to be simpler as we can forgo considerations of how to fuse nested sequents of a different form.[2]

Theorem 2 (Cut). L-*Cut and the following rule of Cut are admissible in* NQ.L*:*

$$\dfrac{\mathcal{S}\{X; \varGamma \Rightarrow \varDelta, A\} \quad \mathcal{S}\{X; A, \varGamma \Rightarrow \varDelta\}}{\mathcal{S}\{X; \varGamma \Rightarrow \varDelta\}}\, Cut$$

Proof. We consider an uppermost instance of L-Cut or *Cut* with $A \equiv \Box B$ and A the cut formula of each rule, respectively. We argue by simultaneous induction on the lexicographic ordering of pairs $(|A|, h_1 + h_2)$, where $|A|$ is the weight of A and h_1 (h_2) is the height of the derivation $\mathcal{D}_1$ ($\mathcal{D}_2$) of the left (right) premiss of the instance of L-Cut or *Cut* under consideration.

Let us first consider the case where the weight of A is zero, i.e. A is a formula of the form $R^n_i(x_1, \ldots, x_n)$, $\bot$, or $x = y$. The first two cases are standard, so we consider the case when A is of the form $x = y$. We suppose first that $x = y$ is not principal in the left premiss of *Cut*. If the left premiss is an initial sequent or an instance of $L\bot$, then the conclusion will be as well, so we may assume that the left premiss was derived by means of another rule. We suppose w.l.o.g. that the left premiss was derived by means of a unary rule as the binary case for $L \supset$ is similar, meaning our Cut is of the following form:

$$\dfrac{\dfrac{\mathcal{S}_1\{X_1; \varGamma_1 \Rightarrow \varDelta_1, x = y\}}{\mathcal{S}\{X; \varGamma \Rightarrow \varDelta, x = y\}}\, R1 \qquad \dfrac{\mathcal{S}_2\{X_2; x = y, \varGamma_2 \Rightarrow \varDelta_2\}}{\mathcal{S}\{X; x = y, \varGamma \Rightarrow \varDelta\}}\, R2}{\mathcal{S}\{X; \varGamma \Rightarrow \varDelta\}}\, Cut$$

[2] Nested sequents and tree-hypersequents are equivalent formalisms; cf. [3,22].

As shown below, we can resolve the case by applying the height-preserving invertibility of *R1* to the right premiss of *Cut*, applying *Cut* with the premiss of *R1*, and then applying *R1* after (note that *R1* is applicable after the Cut since $x = y$ is neither auxiliary nor principal in *R1* by the shape of the rules in $\mathsf{NQ.L}$).

$$\dfrac{\dfrac{\mathcal{S}_1\{X_1; \Gamma_1 \Rightarrow \Delta_1, x = y\} \qquad \dfrac{\dfrac{\mathcal{S}_2\{X_2; x = y, \Gamma_2 \Rightarrow \Delta_2\}}{\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}}\,{\scriptstyle R2}}{\mathcal{S}_1\{X_1; x = y, \Gamma_1 \Rightarrow \Delta_1\}}\,{\scriptstyle Lemma\ 8}}{\mathcal{S}_1\{X_1; \Gamma_1 \Rightarrow \Delta_1\}}\,{\scriptstyle Cut}}{\mathcal{S}\{X; \Gamma \Rightarrow \Delta\}}\,{\scriptstyle R1}$$

If we suppose now that $x = y$ is principal in the left premiss of *Cut*, then the left premiss must be an initial sequent of the form $\mathcal{S}\{X, x = y, \Gamma \Rightarrow \Delta, x = y\}$. We have cases according to whether $x = y$ is principal or not in the right premiss. If it is principal then the right premiss is either (i) an initial sequent or (ii) the conclusion of an instance of a rule in $\{Repl, Repl_X, Rig\}$. In case (i) the conclusion of *Cut* is an initial sequent and in case (ii) the conclusion of *Cut* is identical to the conclusion of its right premiss, which is cut-free derivable. Else, the Cut is of the form shown below, where two copies of $x = y$ must occur in the right premiss since the contexts must match in Cut.

$$\dfrac{\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta, x = y\} \qquad \dfrac{\mathcal{S}'\{X'; x = y, x = y, \Gamma' \Rightarrow \Delta'\}}{\mathcal{S}\{X; x = y, x = y, \Gamma \Rightarrow \Delta\}}\,{\scriptstyle R2}}{\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}}\,{\scriptstyle Cut}$$

Applying the height-preserving admissible rule CL to the right premiss of *Cut* gives the desired conclusion.

Let us suppose now that the weight of the cut formula is greater than zero. We also assume that the cut formula is principal in both premisses of *Cut* and consider the interesting cases when $A \equiv \forall x B$ and $A \equiv \Box B$ as all other cases are standard, see [3, Thm. 5]. If the cut formula $A \equiv \forall x B$ is principal in both premisses of *Cut*, then our Cut is of the following form:

$$\dfrac{\dfrac{\mathcal{S}\{X, y, z; \Gamma \Rightarrow \Delta, B(y/x)\}}{\mathcal{S}\{X, z; \Gamma \Rightarrow \Delta, \forall x B\}}\,{\scriptstyle R\forall} \qquad \dfrac{\mathcal{S}\{X, z; B(z/x), \forall x B, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, z; \forall x B, \Gamma \Rightarrow \Delta\}}\,{\scriptstyle L\forall}}{\mathcal{S}\{X, z; \Gamma \Rightarrow \Delta\}}\,{\scriptstyle Cut}$$

We first shift the Cut upward by applying the height-preserving admissibility of IW to the left premiss of *Cut*, and then apply *Cut* with the premiss of $L\forall$ as shown below, thus reducing $h_1 + h_2$.

$$\dfrac{\dfrac{\dfrac{\mathcal{S}\{X, y, z; \Gamma \Rightarrow \Delta, B(y/x)\}}{\mathcal{S}\{X, z; \Gamma \Rightarrow \Delta, \forall x B\}}\,{\scriptstyle R\forall}}{\mathcal{S}\{X, z; B(z/x), \Gamma \Rightarrow \Delta, \forall x B\}}\,{\scriptstyle IW} \qquad \mathcal{S}\{X, z; B(z/x), \forall x B, \Gamma \Rightarrow \Delta\}}{\mathcal{S}\{X, z; B(z/x), \Gamma \Rightarrow \Delta\}}\,{\scriptstyle Cut}$$

Let us refer to the above proof as $\mathcal{D}$. We now reduce the weight of the cut formula by applying Cut as shown below, giving the desired conclusion.

$$\dfrac{\dfrac{\dfrac{\mathcal{S}\{X,y,z;\Gamma\Rightarrow\Delta,B(y/x)\}}{\mathcal{S}\{X,z,z;\Gamma\Rightarrow\Delta,B(z/x)\}}\,{\scriptstyle (z/y)}}{\mathcal{S}\{X,z;\Gamma\Rightarrow\Delta,B(z/x)\}}\,{\scriptstyle SC} \qquad \mathcal{D}}{\mathcal{S}\{X,z;\Gamma\Rightarrow\Delta\}}\,{\scriptstyle Cut}$$

We now assume that the cut formula $A\equiv\Box B$ is principal in both premisses and we may assume w.l.o.g. that the cut is an instance of L-Cut. We consider the case where the right premiss of L-Cut is an instance of R_T and the left premiss of L-Cut is an instance of $R\Box$. The remaining cases are proven in a similar fashion. The trick is to use the height-preserving admissibility of the special structural rules (see Lemma 10), namely, the S_T rule. Our L-Cut is of the following form:

$$\dfrac{\dfrac{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,[\emptyset;\Rightarrow B]\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,\Box B\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle R\Box} \qquad \dfrac{\mathcal{S}\{X;\Box B,B,\Gamma\Rightarrow\Delta\}\{Y_i;\Box B,\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}{\mathcal{S}\{X;\Box B,\Gamma\Rightarrow\Delta\}\{Y_i;\Box B,\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle R_T}}{\mathcal{S}\{X;\Gamma\Rightarrow\Delta\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle \mathsf{L}\text{-}Cut}$$

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ denote the derivation of the left and right premiss of L-Cut, respectively. To resolve the case, we first apply the height-preserving admissible rule IW to the conclusion of $\mathcal{D}_1$, yielding the derivation $\mathcal{D}_3$ shown below top. We then apply L-Cut to the conclusion of $\mathcal{D}_3$ and the premiss of $\mathcal{D}_2$ (where h_1+h_2 is strictly smaller), giving the second derivation shown below, which we refer to as $\mathcal{D}_4$. Finally, as shown in the third derivation below, we can apply Cut to B (which has a strictly smaller weight than $\Box B$), and derive the desired conclusion after applying a single application of the admissible rule S_T to the left premiss.

$$\mathcal{D}_3\left\{\quad \dfrac{\dfrac{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,[\emptyset;\Rightarrow B]\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,\Box B\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle R\Box}}{\mathcal{S}\{X;B,\Gamma\Rightarrow\Delta,\Box B\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle IW}\right.$$

$$\mathcal{D}_4\left\{\quad \dfrac{\mathcal{D}_3 \qquad \mathcal{S}\{X;\Box B,B,\Gamma\Rightarrow\Delta\}\{Y_i;\Box B,\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}{\mathcal{S}\{X;B,\Gamma\Rightarrow\Delta\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle \mathsf{L}\text{-}Cut}\right.$$

$$\dfrac{\dfrac{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,[\emptyset;\Rightarrow B]\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}{\mathcal{S}\{X;\Gamma\Rightarrow\Delta,B\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle S_T} \qquad \mathcal{D}_4}{\mathcal{S}\{X;\Gamma\Rightarrow\Delta\}\{Y_i;\Pi_i\Rightarrow\Sigma_i\}_{i=1}^n}\,{\scriptstyle Cut}$$

□

5 Soundness and Completeness

Theorem 3 (Soundness). *If* $\mathsf{NQ.L}\vdash\mathcal{S}$ *then* $\mathtt{fm}(\mathcal{S})$ *is* $\mathsf{Q.L}$*-valid.*

Proof. We first note that nested application of rules is sound: for each context $\mathcal{S}\{\cdot\}$, if $A\supset B$ is $\mathsf{Q.L}$-valid then $\mathtt{fm}(\mathcal{S}\{A\})\supset\mathtt{fm}(\mathcal{S}\{B\})$ is $\mathsf{Q.L}$-valid. This can be shown by induction on the depth of the context $\mathcal{S}\{\cdot\}$; see [3, Lem. 3] for details.

The $\mathsf{Q.L}$-soundness of the rules of $\mathsf{NQ.L}$ is proved by induction on the height of the derivation. The cases of initial sequents and of propositional rules of

NQ.L are given in [3, Thm. 1]. We present the cases of $L\forall$, R_{cbf}, Rig, and R_{5dom}, all other cases being similar. If $\mathtt{fm}(X, z; A(z/x), \forall xA, \Gamma \Rightarrow \Delta)$ is Q.L-valid, then the Q.L-validity of $\mathtt{fm}(X, z; \forall xA, \Gamma \Rightarrow \Delta)$ follows by the soundness of the axiom $\mathbf{UI}^{\mathcal{E}}$. If $\mathtt{fm}(X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi \Rightarrow \Sigma])$ is Q.L.CBF-valid, then the formula $\mathtt{fm}(X, x; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma])$ is as well because frames for Q.L.CBF have increasing domains. The Q.L-validity of $\mathtt{fm}(\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}\{Y; \Pi \Rightarrow \Sigma\})$ follows from that of $\mathtt{fm}(\mathcal{S}\{X; x = y, \Gamma \Rightarrow \Delta\}\{Y; x = y, \Pi \Rightarrow \Sigma\})$ since variables are rigid designators—i.e., the validity of $\mathbf{NI} := x = y \supset \Box(x = y)$ and that of **ND** allow identities to be duplicated up and down the accessibility relation, respectively. Finally, we argue that R_{5dom} preserves Q.L-validity when either $\mathbf{5}, \mathbf{CBF} \in$ L or $\mathbf{5}, \mathbf{BF} \in$ L. We show this holds for the following one-context rules from which R_{5dom} is NQ.L-derivable (if x is in the signature of a non-root node, these rules bottom-up copy x into the signature of another non-root node):

$$\frac{\mathcal{S}\{[X, x; \Gamma \Rightarrow \Delta], [Y, x; \Pi \Rightarrow \Sigma]\}}{\mathcal{S}\{[X, x; \Gamma \Rightarrow \Delta], [Y; \Pi \Rightarrow \Sigma]\}}\ R_{5dom_1} \qquad \frac{\mathcal{S}\{[X, x; \Gamma \Rightarrow \Delta, [Y, x; \Pi \Rightarrow \Sigma]]\}}{\mathcal{S}\{[X, x; \Gamma \Rightarrow \Delta, [Y; \Pi \Rightarrow \Sigma]]\}}\ R_{5dom_2}$$

$$\frac{\mathcal{S}\{[Y, x; \Pi \Rightarrow \Sigma, [X, x; \Gamma \Rightarrow \Delta]]\}}{\mathcal{S}\{[Y; \Pi \Rightarrow \Sigma, [X, x; \Gamma \Rightarrow \Delta]]\}}\ R_{5dom_3}$$

If the premiss of one of these rules is Q.L-valid, then so is the respective conclusion since for **5**-frames with increasing or decreasing domains the points satisfying $X, x; \Gamma \Rightarrow \Delta$ and $Y; \Pi \Rightarrow \Sigma$ are mutually accessible and have the same domain. □

Theorem 4 (Completeness). *If $\mathtt{fm}(\mathcal{S})$ is* Q.L*-valid, then* NQ.L $\vdash \mathcal{S}$.

Proof. We show that Q.L $\vdash \mathtt{fm}(\mathcal{S})$ implies NQ.L $\vdash \mathcal{S}$; the theorem follows by the completeness of Q.L (Theorem 1). We proceed by induction on the height of the derivation of $\mathtt{fm}(\mathcal{S})$ in Q.L. The NQ.L-admissibility of rule **MP**/**UG**/**N** is a corollary of Theorem 2/Lemma 6/Lemma 7. We consider only axioms $\mathbf{UI}^{\circ}$ (assuming $y \notin A$ for simplicity), **ND**, and **CBF**. The cases of axioms **REF** and **REPL** follows from Lemma 2 and the other cases are similar.

$$\cfrac{\cfrac{\cfrac{\cfrac{}{y; A(y/x), \forall xA \Rightarrow A(y/x)}\ L.\ 1}{y; \forall xA \Rightarrow A(y/x)}\ L\forall}{y; \Rightarrow \forall xA \supset A(y/x)}\ R{\supset}}{\Rightarrow \forall y(\forall xA \supset A(y/x))}\ R\forall \qquad \cfrac{\cfrac{\cfrac{\cfrac{}{x = y \Rightarrow x = y, [x = y \Rightarrow]}\ L.\ 1}{\Rightarrow x = y, [x = y \Rightarrow]}\ Rig}{x \neq y \Rightarrow [\Rightarrow x \neq y]}\ L\neg{+}R\neg}{x \neq y \Rightarrow \Box(x \neq y)}\ R\Box$$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{y; \Box\forall xA \Rightarrow, [y; A(y/x), \forall xA \Rightarrow A(y/x)]}\ L.\ 1}{y; \Box\forall xA \Rightarrow [y; \forall xA \Rightarrow A(y/x)]}\ L\forall}{y; \Box\forall xA \Rightarrow [\forall xA \Rightarrow A(y/x)]}\ R_{cbf}}{y; \Box\forall xA \Rightarrow [\Rightarrow A(y/x)]}\ L\Box}{y; \Box\forall xA \Rightarrow \Box A(y/x)}\ R\Box}{\Box\forall xA \Rightarrow \forall x\Box A}\ R\forall$$

□

6 Conclusion and Future Work

We provided a uniform nested sequent presentation of quantified modal logics characterised by combinations of fundamental properties. Due to the inclusion of equality in the language of the QMLs considered, our nested calculi permit a formula translation by means of the (definable) existence predicate. As a consequence, our systems possess both a good degree of modularity *and* utilise a language as expressive as that of each logic, yielding more economical systems in contrast to the labelled calculi given for the same QMLs, which employ a more expressive language [20,25]. Beyond formula interpretability, our nested calculi satisfy fundamental properties such as the admissibility of important structural rules, invertibility of all rules, and syntactic cut-elimination.

In future work, we aim to investigate constructive proofs of interpolation properties with our nested calculi (cf. [9,15]), to use (variations of) our nested calculi to identify decidable QML fragments, as well as extend the present approach to QMLs with non-rigid designators and, possibly, definite descriptions based on λ-abstraction (see [10]) as was done in [21] for labelled sequent calculi. Another open problem is to give nested sequents with a formula interpretation for QMLs where the existence predicate is not expressible; we conjecture that this might be achieved by using the 'universally closed nesting' defined by Brünner for free logics [4].

We also aim to generalise our approach by employing a wider selection of propagation rules [6,8] and reachability rules [16,17] in our systems. As shown in various works [11,16], diverse classes of logics characterised by Horn properties can be supplied cut-free nested calculi by utilising logical rules that propagate or consume data along paths within nested sequents specified by formal grammars. Applying this technique, we plan to see if we can capture a much wider class of QMLs in a uniform and modular fashion, and plan to investigate admissibility and invertibility properties as well as cut-elimination in this more general setting. It would also be worthwhile to examine the relationship between our nested calculi and other calculi for QMLs; e.g., we could study the computational relationship between our nested calculi and the labelled calculi for QMLs, showing how proofs can be translated and determining complexity bounds for the relative sizes of proofs.

References

1. Avron, A.: The method of hypersequents in the proof theory of propositional non-classical logics. In: From Foundations to Applications: European Logic Colloquium, USA, pp. 1–32. Clarendon Press (2010)
2. Belnap, N.D.: Display logic. J. Philos. Logic **11**(4), 375–417 (1982). https://doi.org/10.1007/BF00284976
3. Brünnler, K.: Deep sequent systems for modal logic. Arch. Math. Logic **48**, 551–577 (2009). https://doi.org/10.1007/s00153-009-0137-3
4. Brünnler, K.: How to universally close the existential rule. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR 2010. LNCS, vol. 6397, pp. 172–186. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16242-8_13
5. Bull, R.: Cut elimination for propositional dynamic logic without *. Math. Log. Q. **38**(1), 85–100 (1992). https://doi.org/10.1002/malq.19920380107

6. Castilho, M., del Cerro, L., Gasquet, O., Herzig, A.: Modal tableaux with propagation rules and structural rules. Fundamenta Informaticae **32**(3,4), 281–297 (1997). https://doi.org/10.3233/FI-1997-323404
7. Corsi, G.: A unified completeness theorem for quantified modal logics. J. Symb. Logic **67**(2), 1483–1510 (2002). https://doi.org/10.2178/jsl/1190150295
8. Fitting, M.: Tableau methods of proof for modal logics. Notre Dame J. Formal Logic **13**(2), 237–247 (1972). https://doi.org/10.1305/ndjfl/1093894722
9. Fitting, M., Kuznets, R.: Modal interpolation via nested sequents. Ann. Pure Appl. Logic **166**(3), 274–305 (2015). https://doi.org/10.1016/j.apal.2014.11.002
10. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Springer, Dordrecht (1998)
11. Goré, R., Postniece, L., Tiu, A.: On the correspondence between display postulates and deep inference in nested sequent calculi for tense logics. Logical Methods Comput. Sci. **7**(2), 1–38 (2011). https://doi.org/10.2168/LMCS-7(2:8)2011
12. Kashima, R.: Cut-free sequent calculi for some tense logics. Stud. Logica. **53**(1), 119–135 (1994). https://doi.org/10.1007/BF01053026
13. Kripke, S.: Semantical considerations on modal logic. Acta Philosophica Fennica **16**, 83–94 (1963)
14. Lyon, T.: On the correspondence between nested calculi and semantic systems for intuitionistic logics. J. Log. Comput. **31**(1), 213–265 (2020). https://doi.org/10.1093/logcom/exaa078
15. Lyon, T.: Syntactic interpolation for tense logics and bi-intuitionistic logic via nested sequents. In: Fernández, M., Muscholl, A. (eds.) Annual Conference on Computer Science Logic CSL 2020, vol. 152, pp. 28:1–28:16. Leibniz International Proceedings in Informatics (2020). https://doi.org/10.4230/LIPIcs.CSL.2020.28
16. Lyon, T.: Refining labelled systems for modal and constructive logics with applications. Dissertation, Technische Universität Wien (2021). https://doi.org/10.34726/hss.2021.97064
17. Lyon, T.: Nested Sequents for First-Order Modal Logics via Reachability Rules. arXiv (2022, unpublished). https://doi.org/10.48550/arXiv.2210.00789
18. Lyon, T., Orlandelli, E.: Nested Sequents for Quantified Modal Logics. arXiv (2023). https://doi.org/10.48550/arXiv.2210.00789
19. Maffezioli, P., Orlandelli, E.: Full cut elimination and interpolation for intuitionistic logic with existence predicate. Bull. Section Logic **48**(2), 137–158 (2019). https://doi.org/10.18778/0138-0680.48.2.04
20. Negri, S., von Plato, J.: Proof Analysis: A Contribution to Hilbert's Last Problem. Cambridge University Press, Cambridge (2011)
21. Orlandelli, E.: Labelled calculi for quantified modal logics with definite descriptions. J. Log. Comput. **31**(3), 923–946 (2021). https://doi.org/10.1093/logcom/exab018
22. 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
23. Simpson, A.: The proof theory and semantics of intuitionistic modal logic. Dissertation, University of Edinburgh (1994). https://www.cs.cmu.edu/~fp/courses/15816-s10/papers/Simpson94.pdf

24. Tiu, A.: A hypersequent system for Gödel-Dummett logic with non-constant domains. In: Brünnler, K., Metcalfe, G. (eds.) TABLEAUX 2011. LNCS (LNAI), vol. 6793, pp. 248–262. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22119-4_20
25. Viganò, L.: Labelled Non-classical Logics. Springer, Heidelberg (2000)

A Naive Prover for First-Order Logic: A Minimal Example of Analytic Completeness

Asta Halkjær From and Jørgen Villadsen(✉)

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

Abstract. The analytic technique for proving completeness gives a very operational perspective: build a countermodel to the unproved formula from a failed proof attempt in your calculus. We have to be careful, however, that the proof attempt did not fail because our strategy in finding it was flawed. Overcoming this concern requires designing a prover. We design and formalize in Isabelle/HOL a sequent calculus prover for first-order logic with functions. We formalize soundness and completeness theorems using an existing framework and extract executable code to Haskell. The crucial idea is to move complexity from the prover itself to a stream of instructions that it follows. The result serves as a minimal example of the analytic technique, a naive prover for first-order logic, and a case study in formal verification.

Keywords: First-Order Logic · Prover · Completeness · Isabelle/HOL

1 Introduction

We present a sound and complete (naive) prover for classical first-order logic with functions. There are several ways to prove that a proof system for first-order logic is complete. Gödel's approach [14], later refined by Henkin [15] is now known as the *synthetic* way. This technique abstractly builds *maximal consistent (and saturated) sets* of formulas as a bridge between the proof system and the semantics. This is a useful technique and has been used in formalizations of the completeness of axiomatic systems for first-order logic [9] and epistemic logic [8], a tableau system for hybrid logic [7] and more. Unfortunately, as pointed out by Blanchette et al. [5] in the context of formalization in Isabelle/HOL, there is no useful connection between this technique and the execution of an actual prover.

The technique by Beth and Hintikka [17] offers a more operational perspective. Here, we consider unsuccessful proof attempts in the given calculus and build countermodels from these. Such a countermodel refutes the validity of the formula that we tried to prove. To build such a countermodel, however, we must ensure that the proof attempt was sufficiently sophisticated and, essentially, that it would have found a proof if one existed. In proving this property of the proof strategy, we are effectively designing a prover based on the calculus. This means that, in practice, we can extract a prover from our completeness proof.

Blanchette et al. [5] have made this very concrete by developing a framework in Isabelle/HOL for analytic completeness proofs. Their paper includes a first-order logic example, but their entry in the Archive of Formal Proofs [3] only includes a propositional example. In this paper, we describe a *naive prover* based on the framework, designed to be as simple as possible. This augments the framework with a concrete first-order logic example showcasing the analytic technique. Moreover it serves as an introduction to automated reasoning by making explicit the requirements for completeness of a prover for first-order logic. It also serves as a small case study for formal verification in a proof assistant.

Then the question remains of how to design this proof strategy. We want it to be sufficiently intricate to be both sound and complete, but we also want it to be simple enough that we can reasonably demonstrate these properties (in a proof assistant). We might follow something like Ben-Ari's tableau algorithm [1] (essentially sequent calculus), but we discover that it is surprisingly complex. There are nodes with labels, branches with markings, and concerns about which kinds of formulas to process first, later or even together. Instead, we will design a prover with minimal structure that tries to apply sequent calculus proof rules over and over, in the belief that we will eventually apply the right ones.

The problem changes from working out which rule to apply in a given situation, to designing a stream of instructions that will cover whatever we encounter and embedding enough structure into these instructions to keep the prover itself elementary. This perspective shift greatly simplifies the prover: the rules are indexed by formulas and specify exactly what the prover should do in each case. Moreover, the nodes in the proof tree are simply sequents, no additional state is needed. The rules apply straightforwardly to these sequents to form the next nodes of the tree. This simplifies the completeness proof and makes it a non-issue to handle first-order logic with functions, which can otherwise require extra consideration.

The formalization of the (naive) prover is available in the Archive of Formal Proofs [11]. It consists of less than 900 lines of Isabelle/HOL listings, the majority of which are proofs that are not included when exporting Haskell code for the prover. A short, manually written `Main.hs` file augments the exported code with a command line interface and pretty-printed output. The Isabelle theory *Export.thy* includes instructions on how to export and compile the Haskell code (which closely resembles the programs listed here). The code in this paper is exported to LaTeX by Isabelle from the formalization, but differs slightly in names and layout for presentation reasons. Likewise, to focus on essentials, we often omit the technical commands needed in the formalization.

2 Related Work

Blanchette [2] gives an overview of a number of verification efforts including the metatheory of SAT and SMT solvers, the resolution and superposition calculi, and a series of proof systems for propositional logic [18]. The aim is to develop a methodology for formalizing modern research in automated reasoning and the present work points in this direction with a minimal example of a formally verified prover for classical first-order logic based on the sequent calculus.

The prover is based on the abstract completeness framework by Blanchette, Popescu and Traytel [4,5]. Their formalization contains a simple example prover for propositional logic, while their paper contains the ideas for a (naive) prover for first-order logic. Our prover realizes these ideas by formalizing them in Isabelle/HOL. Instead of a prover, Blanchette et al. [5] used the framework to formalize soundness and completeness of a *calculus* for first-order logic with equality in negation normal form. From and Jacobsen [10,12] used the framework to formalize a much less naive prover for first-order logic based on the SeCaV proof system [13]. Instead of indexed rules, they employ "multi-rules" that apply to every applicable formula in a sequent at once and they store more than just the sequent at each node in the proof tree. Their prover performs better, but the formalization does not enjoy the simplicity of the naive prover, with close to 3000 lines of Isabelle/HOL against 900 lines.

The indexed rules of the naive prover automatically yield readable proofs. In the same vein, THINKER by Pelletier [21] is a natural deduction proof system and attached automated theorem prover, designed for "direct proofs", as opposed to proofs based on reduction to a resolution system. MUSCADET by Pastre [20] is another automated theorem prover based on natural deduction. Neither of these has been formally verified. Schulz and Pease [24] focused on readable code rather than proofs. They have developed a saturation-based theorem prover in Python for first-order logic to teach automated theorem proving by example. They have not formally verified soundness and completeness, but our projects are similar.

In the world of formalization, Schlichtkrull et al. [23] formalized an ordered resolution prover for *clausal* first-order logic in Isabelle/HOL. Jensen et al. [16] formalized the soundness, but not the completeness, of a prover for first-order logic with equality in Isabelle/HOL. Villadsen et al. [25] verified a simple prover for first-order logic in Isabelle/HOL aiming for students to understand both the prover and the formalization. That work simplified a formalization by Ridge and Margetson [22]. Neither of the last two provers support functions.

3 Isabelle/HOL Overview

We give a quick overview of the Isabelle/HOL features used in the present paper. Nipkow and Klein [19, Part 1] give a more complete introduction.

The **datatype** command defines a new inductive type from a series of constructors, where each can be given custom syntax. The natural numbers are built from the nullary constructor *0* and unary *Suc*. The constructors *True* and *False* belong to the built-in type *bool*. The usual connectives and quantifiers from first-order logic ($\longrightarrow$, $\forall$, etc.) are available for *bool*, as well as *if-then-else* expressions. The parametric *'a list* is the type of lists with elements of type *'a*. The type variable *'a* stands in the place of another type. Lists are built from [], the empty list, and #, an infix constructor that adjoins an element to an

```
datatype tm                          datatype fm
  = Var nat (#)                        = Falsity (⊥)
  | Fun nat (tm list) (†)              | Pre nat (tm list) (‡)
                                       | Imp fm fm (infixr ⟶ 55)
                                       | Uni fm (∀)
```

Fig. 1. The first-order logic syntax in Isabelle/HOL.

existing list. The notation $[a,\ b,\ c]$ is shorthand for these primitive operations. The function *set* turns a list into a set of its elements, *map* applies a given function to every element of a list, @ appends two lists, *concat* flattens a list of lists and *upt j k* creates the list $[j, j+1, \ldots, k-1]$. We use $[\in]$ for list membership and $[\div]$ to remove all occurrences of a given element from a list. The two types *'a set* and *'a fset* form sets and finite sets respectively. The usual operations are available on sets. On finite sets they are typically prefixed by *f* as in *fimage*. Two additional types are important: sum types with the two unary constructors *Inl* and *Inr*, and *option* types constructed by the unary *Some* or nullary *None*. Constructors can be examined using *case* expressions.

The **codatatype** command defines a new coinductive type from a series of constructors. The canonical example is the type *'a stream* of "lists with no base case", i.e. infinite sequences. The functions *shd* and *stl* return the head and tail of a stream, respectively, while *flat* transforms a stream of lists into a stream of all the elements in the constituent lists, *sset* returns a set of its elements, *smap* applies a function to every element, !! returns the element at a given index and *sdrop-while* removes a prefix of a stream that satisfies a given predicate. The stream *nats* contains all natural numbers.

The type $A \Rightarrow B$ denotes a function from A to B. Type signatures are specified after "::". Types can be shortened using type synonyms. The term *UNIV* stands for the set of all values of a given type. In this paper, both $=$ and $\equiv$ are used to form new definitions. Function application resembles functional programming languages: $f(x, y)$ is written as $f\ x\ y$ and partial application is allowed. Anonymous functions are built using λ-expressions, e.g. $\lambda n.\ n + n$ for $f(n) = n + n$.

A **locale** in Isabelle/HOL **fixes** a number of terms, then **assumes** a number of properties about those terms. The meta-logical implication $\Longrightarrow$ separates premises from conclusions in each assumption. The keyword **and** acts as a separator. A locale for a group, for instance, *fixes* a set and a binary operation and *assumes* the group axioms.

4 First-Order Logic in Isabelle/HOL

Figure 1 contains a formalization of the syntax of first-order logic as a datatype in Isabelle/HOL. The syntax is *deeply embedded* as an object in the meta-logic so we can manipulate it. We use de Bruijn indices [6] to represent binding: each variable n is bound by the quantifier that is n quantifiers away, moving outwards.

$$\begin{array}{l}
\textbf{type-synonym}\ {'a}\ \mathit{var\text{-}denot} = \mathit{nat} \Rightarrow {'a} \\
\textbf{type-synonym}\ {'a}\ \mathit{fun\text{-}denot} = \mathit{nat} \Rightarrow {'a}\ \mathit{list} \Rightarrow {'a} \\
\textbf{type-synonym}\ {'a}\ \mathit{pre\text{-}denot} = \mathit{nat} \Rightarrow {'a}\ \mathit{list} \Rightarrow \mathit{bool}
\end{array}$$

$$\begin{array}{l}
\fatsemi :: {'a} \Rightarrow (\mathit{nat} \Rightarrow {'a}) \Rightarrow \mathit{nat} \Rightarrow {'a} \\
(t \fatsemi s)\ 0 = t \\
(t \fatsemi s)\ (\mathit{Suc}\ n) = s\ n
\end{array}$$

$$\begin{array}{l}
(\!| \text{-}, \text{-} |\!) :: {'a}\ \mathit{var\text{-}denot} \Rightarrow {'a}\ \mathit{fun\text{-}denot} \Rightarrow \mathit{tm} \Rightarrow {'a} \\
(\!| E, F |\!)\ (\#n) = E\ n \\
(\!| E, F |\!)\ (\dagger f\ \mathit{ts}) = F\ f\ (\mathit{map}\ (\!| E, F |\!)\ \mathit{ts})
\end{array}$$

$$\begin{array}{l}
[\![\text{-}, \text{-}, \text{-}]\!] :: {'a}\ \mathit{var\text{-}denot} \Rightarrow {'a}\ \mathit{fun\text{-}denot} \Rightarrow {'a}\ \mathit{pre\text{-}denot} \Rightarrow \mathit{fm} \Rightarrow \mathit{bool} \\
[\![\text{-}, \text{-}, \text{-}]\!]\ \bot = \mathit{False} \\
[\![E, F, G]\!]\ (\ddagger P\ \mathit{ts}) = G\ P\ (\mathit{map}\ (\!| E, F |\!)\ \mathit{ts}) \\
[\![E, F, G]\!]\ (p \longrightarrow q) = ([\![E, F, G]\!]\ p \longrightarrow [\![E, F, G]\!]\ q) \\
[\![E, F, G]\!]\ (\forall\, p) = (\forall\, x.\ [\![x \fatsemi E, F, G]\!]\ p)
\end{array}$$

Fig. 2. The semantics of first-order logic in Isabelle/HOL.

A term t, type *tm*, is then either a variable $\#n$ for some de Bruijn index n (a natural number) or a function application $\dagger f$ [...] for some natural number f representing the function name and list of argument terms. [...]. A formula p, type *fm*, is the constant for falsity, $\bot$, a predicate $\ddagger P$ [...] for some natural number P representing the predicate name and list of argument terms [...], an implication $p_1 \longrightarrow p_2$ between two formulas p_1, p_2 or a universally quantified formula $\forall p$.

Figure 2 contains a formalization of the semantics in Isabelle/HOL. A model consists of three denotations: one each for variables (E), function symbols (F) and predicate symbols (G). Terms evaluate to a member of the domain, here represented as a type variable, while formulas evaluate to truth values in the higher-order logic. We can use the connectives and quantifiers of Isabelle/HOL to interpret the first-order logic syntax. For the universal quantifier, we modify the environment such that we evaluate the quantified variable 0 as every element of the domain.

Figure 3 lists the rules for instantiating a quantifier with a term without capturing any free variables in the process. The operation *lift-tm* increments every variable in the term t by one. The operation *sub-tm* s t applies the substitution s to every variable in term t. The operation *sub-fm* s p applies the substitution s to the formula p, taking account of binders. In the case for $\forall p$, the substitution is augmented using $\fatsemi$ to preserve the bound variable $\#0$ in p and to *lift* the variables in the output of the substitution s to point past the binder. We write the instantiation of a quantified formula $\forall p$ with a concrete term t as $\langle t \rangle p$. The notation $\langle t \rangle$ represents the simultaneous substitution that maps variable 0 to t and every other variable $n + 1$ to n to account for the removed binder. Figure 4 lists the operations for generating a variable *fresh* to a list of formulas, i.e. one that does not appear in any formula in the list.

$$\begin{aligned}
&\textit{lift-tm} :: \textit{tm} \Rightarrow \textit{tm}\\
&\textit{lift-tm}\ (\#n) = \#(n{+}\mathit{1})\\
&\textit{lift-tm}\ (\dagger f\ \textit{ts}) = \dagger f\ (\textit{map lift-tm ts})\\[1ex]
&\textit{sub-tm} :: (\textit{nat} \Rightarrow \textit{tm}) \Rightarrow \textit{tm} \Rightarrow \textit{tm}\\
&\textit{sub-tm}\ s\ (\#n) = s\ n\\
&\textit{sub-tm}\ s\ (\dagger f\ \textit{ts}) = \dagger f\ (\textit{map}\ (\textit{sub-tm}\ s)\ \textit{ts})\\[1ex]
&\textit{sub-fm} :: (\textit{nat} \Rightarrow \textit{tm}) \Rightarrow \textit{fm} \Rightarrow \textit{fm}\\
&\textit{sub-fm}\ \text{-}\ \bot = \bot\\
&\textit{sub-fm}\ s\ (\ddagger P\ \textit{ts}) = \ddagger P\ (\textit{map}\ (\textit{sub-tm}\ s)\ \textit{ts})\\
&\textit{sub-fm}\ s\ (p \longrightarrow q) = \textit{sub-fm}\ s\ p \longrightarrow \textit{sub-fm}\ s\ q\\
&\textit{sub-fm}\ s\ (\forall\, p) = \forall\, (\textit{sub-fm}\ (\#\mathit{0} \mathbin{⨾} \lambda n.\ \textit{lift-tm}\ (s\ n))\ p)\\[1ex]
&\langle\text{-}\rangle :: \textit{tm} \Rightarrow \textit{fm} \Rightarrow \textit{fm}\\
&\langle t\rangle \equiv \textit{sub-fm}\ (t \mathbin{⨾} \#)
\end{aligned}$$

Fig. 3. The simultaneous substitution and quantifier instantiation in Isabelle/HOL.

$$\begin{aligned}
&\textit{vars-tm} :: \textit{tm} \Rightarrow \textit{nat list}\\
&\textit{vars-tm}\ (\#n) = [n]\\
&\textit{vars-tm}\ (\dagger\text{-}\ \textit{ts}) = \textit{concat}\ (\textit{map vars-tm ts})\\[1ex]
&\textit{vars-fm} :: \textit{fm} \Rightarrow \textit{nat list}\\
&\textit{vars-fm}\ \bot = []\\
&\textit{vars-fm}\ (\ddagger\text{-}\ \textit{ts}) = \textit{concat}\ (\textit{map vars-tm ts})\\
&\textit{vars-fm}\ (p \longrightarrow q) = \textit{vars-fm}\ p\ @\ \textit{vars-fm}\ q\\
&\textit{vars-fm}\ (\forall\, p) = \textit{vars-fm}\ p\\[1ex]
&\textit{vars-fms} :: \textit{fm list} \Rightarrow \textit{nat list}\\
&\textit{vars-fms}\ A \equiv \textit{concat}\ (\textit{map vars-fm}\ A)\\[1ex]
&\textit{max-list} :: \textit{nat list} \Rightarrow \textit{nat}\\
&\textit{max-list}\ [] = \mathit{0}\\
&\textit{max-list}\ (x\ \#\ \textit{xs}) = \textit{max}\ x\ (\textit{max-list xs})\\[1ex]
&\textit{fresh} :: \textit{fm list} \Rightarrow \textit{nat}\\
&\textit{fresh}\ A \equiv \textit{Suc}\ (\textit{max-list}\ (\textit{vars-fms}\ A))
\end{aligned}$$

Fig. 4. The rules for generating a fresh variable in Isabelle/HOL.

type-synonym *sequent* = *fm list* × *fm list*

$$\begin{aligned}
&\textit{sc} :: ({'a}\ \textit{var-denot} \times {'a}\ \textit{fun-denot} \times {'a}\ \textit{pre-denot}) \Rightarrow \textit{sequent} \Rightarrow \textit{bool}\\
&\textit{sc}\ (E,\ F,\ G)\ (A,\ B) = ((\forall\, p\ [\in]\ A.\ [\![E,\ F,\ G]\!]\ p) \longrightarrow (\exists\, q\ [\in]\ B.\ [\![E,\ F,\ G]\!]\ q))
\end{aligned}$$

Fig. 5. The syntax and semantics of sequents in Isabelle/HOL.

$$\textsc{Idle}\ \frac{A \vdash B}{A \vdash B} \qquad \textsc{Axiom}\ P\ ts\ \frac{}{A \vdash B}\ \textsc{if}\ \ddagger P\ ts\ [\in]\ A\ \textsc{and}\ \ddagger P\ ts\ [\in]\ B$$

$$\textsc{FlsL}\ \frac{}{A \vdash B}\ \textsc{if}\ \bot\ [\in]\ A \qquad \textsc{FlsR}\ \frac{A \vdash B\ [\div]\ \bot}{A \vdash B}\ \textsc{if}\ \bot\ [\in]\ B$$

$$\textsc{ImpL}\ p\ q\ \frac{A\ [\div]\ (p \longrightarrow q) \vdash p\ \#\ B \qquad q\ \#\ A\ [\div]\ (p \longrightarrow q) \vdash B}{A \vdash B}\ \textsc{if}\ (p \longrightarrow q)\ [\in]\ A$$

$$\textsc{ImpR}\ p\ q\ \frac{p\ \#\ A \vdash q\ \#\ B\ [\div]\ (p \longrightarrow q)}{A \vdash B}\ \textsc{if}\ (p \longrightarrow q)\ [\in]\ B$$

$$\textsc{UniL}\ t\ p\ \frac{\langle t \rangle p\ \#\ A \vdash B}{A \vdash B}\ \textsc{if}\ \forall p\ [\in]\ A$$

$$\textsc{UniR}\ p\ \frac{A \vdash \langle \#\mathit{fresh}(A@B) \rangle p\ \#\ B\ [\div]\ \forall p}{A \vdash B}\ \textsc{if}\ \forall p\ [\in]\ B$$

Fig. 6. The rules of the sequent calculus presented visually.

The calculus works on two-sided sequents, of type *sequent*, which are represented as pairs of lists of formulas (cf. Fig. 5). We can think of the left-hand side as assumptions and the right-hand side as conclusions. Moreover, the left-hand side is conjunctive, so we can assume all of the formulas there to be true, while the right-hand side is disjunctive, so we only need to prove one.

Sequent calculus has the benefit of the *subformula property*: to prove a formula we only need to look at its subformulas. Contrast this with axiomatic systems using modus ponens (from $p \longrightarrow q$ and p infer q), where we need to guess a suitable "lemma" formula. However, a sequent calculus may still leave too much freedom for comfort. In particular, we want to remove the need for structural rules, since these are too applicable.

Figure 6 lists the underlying rules of the prover in a somewhat idiosyncratic manner. The reason will become apparent later. Each rule has a name to the left of the horizontal line. Below the horizontal line is the conclusion and above are the premises, if any. Any side conditions are given to the right of the line. Note that each rule is indexed by the exact (sub)formulas it works on: the rule Axiom 0 [] is distinct from the rule Axiom 1 [] etc. This rigidity means that we do not need any structural rules. It also means that there is no pattern matching in any of the rules and that the three primary operations are membership checking ($[\in]$), removal of concrete formulas ($[\div]$) and adding new formulas to a list ($\#$).

The Idle rule appears for technical reasons (there should always be an enabled rule). The Axiom rule is indexed by a predicate symbol P and argument list *ts* and checks whether such a predicate appears on both sides of the sequent: if so, the rule applies and there are no child sequents. The FlsL rule checks if $\bot$ occurs among the assumptions, in which case the sequent is proved. The FlsR rule, when it applies, drops all occurrences of $\bot$ from the conclusions, since we can never prove any of them. The ImpL and ImpR rules decompose implications on either side of the sequent in the standard way. The UniL rule is indexed by a term t and a formula p. If $\forall p$ occurs on the left, then the rule instantiates it with t, adding $\langle t \rangle p$ to the left-hand side of the child sequent. The UniR rule is only indexed by a formula p. When $\forall p$ occurs on the right, it is instantiated with a fresh variable and removed.

In order to obtain a prover based on the rules of the sequent calculus we use the abstract completeness framework for Isabelle/HOL developed by Blanchette, Popescu and Traytel [3,5]. This framework formalizes the mechanics of sequent calculus and semantic tableaux provers in an abstract way that we can instantiate with concrete rules. There are two possible perspectives on the framework: (i) the proof perspective, where we use the framework to obtain theorems about proof trees built from our rules and (ii) the code generation perspective, where we use the framework to generate an executable prover. In this paper, both perspectives come into play but the two perspectives can be used on their own.

The framework needs: a stream of rules, a function describing their effect, a proof that some rule is always enabled and a guarantee that rules are persistent. We formalize the calculus in Isabelle/HOL as a datatype of rules, *rule*, with constructors *Idle*, *Axiom*, *FlsL*, *FlsR*, *ImpL*, *ImpR*, *UniL* and *UniR*, and an effect function, *eff*, that encodes the relationship between premises and conclusions in the manner expected by the framework.

5 Soundness and Completeness

Soundness requires that we do not prove a sequent without having proper reasons to do so. It is a local property of our calculus that we can easily check. Completeness, on the other hand, requires that we have sufficient rules available to prove every valid formula. Thus, proving completeness requires a more involved strategy.

Lemma 1 (Local soundness). *If all premises of a rule are valid, then its conclusion is valid. In Isabelle, if* $\mathit{eff}\ r\ (A,\ B) = \mathit{Some}\ ss$ *and* $\forall\, A\ B.\ (A,\ B)\ |{\in}|\ ss \longrightarrow (\forall (E :: \text{-} \Rightarrow {}'a).\ sc\ (E,\ F,\ G)\ (A,\ B))$*, then* $sc\ (E,\ F,\ G)\ (A,\ B)$*.*

Proof. By induction on the call structure of *eff*. The induction hypothesis then applies to the sequents produced by *eff*. All cases except UniR are trivial. For UniR, by the induction hypothesis, the premise holds under all variable denotations: no matter the assignment to the fresh variable. This justifies forming the universal quantifier and since the fresh variable does not appear elsewhere in the sequent, the semantics there are unaffected.

Theorem 1 (Prover soundness). *If a proof tree (attempt) is well formed and finite, then the root sequent is valid. In Isabelle, if* $\mathit{tfinite}\ t$ *and* $\mathit{wf}\ t$*, then* $sc\ (E,\ F,\ G)\ (\mathit{fst}\ (\mathit{root}\ t))$*.*

Proof. By induction on the *finite* proof tree using Lemma 1.

locale *Hintikka* =
  fixes $A\ B :: \mathit{fm\ set}$
  assumes
    Basic: $\ddagger P\ ts \in A \Longrightarrow \ddagger P\ ts \in B \Longrightarrow \mathit{False}$ **and**
    FlsA: $\bot \notin A$ **and**
    ImpA: $p \longrightarrow q \in A \Longrightarrow p \in B \vee q \in A$ **and**
    ImpB: $p \longrightarrow q \in B \Longrightarrow p \in A \wedge q \in B$ **and**
    UniA: $\forall\, p \in A \Longrightarrow \forall\, t.\ \langle t \rangle p \in A$ **and**
    UniB: $\forall\, p \in B \Longrightarrow \exists\, t.\ \langle t \rangle p \in B$

$M\ A \equiv [\![\#,\ \dagger,\ \lambda P\ ts.\ \ddagger P\ ts \in A]\!]$

Fig. 7. Formalizations of Hintikka sets and the countermodel $M\ A$.

For completeness we must now show that, for every valid sequent, the prover finds a proof. We do so contrapositively: if the prover does not find a proof, we produce a countermodel to the sequent. To do so, we characterize saturated escape paths syntactically using Hintikka sets and show that such sets induce countermodels. Figure 7 characterizes Hintikka sets in our setting. There are two perspectives on these: one, that they characterize saturated escape paths and two, that they characterize the semantics of the countermodel.

To understand the first perspective, read the set A as consisting of all formulas that appear as assumptions on the saturated escape path (on the left-hand side of sequents) and the set B as consisting of all formulas that appear as conclusions (on the right-hand side of sequents). The Isabelle/HOL functions *treeA* and *treeB* collect these sets, respectively.

Lemma 2 (Hintikka sets characterize saturated escape paths). *Let A and B be sets of assumption and conclusion formulas on a saturated escape path. Then they fulfill all Hintikka requirements. In Isabelle, if epath steps and Saturated steps, then Hintikka (treeA steps) (treeB steps).*

Proof. We check each condition separately.

Basic states that a predicate cannot appear as both assumption and conclusion on the epath. Otherwise the AXIOM rule would have terminated the (infinite) epath.

FlsA states that $\bot$ does not appear among the assumptions. Similar to the above, the FLSL rule would have terminated the epath if so.

ImpA and *ImpB* break down implications in accordance with the IMPL and IMPR rules. For a given p, q, if $p \longrightarrow q$ appears in A (respectively B), then at some point in the proof tree attempt, the rule IMPL $p\ q$ (respectively IMPR p q) becomes enabled. Since the epath is saturated, any enabled rule is eventually taken and the effect matches the thesis.

UniA states that any universally quantified formula $\forall p$ on the left is instantiated with all possible terms. Fix an arbitrary term t. Since $\forall p$ occurs as an assumption, the specific rule UNIL $p\ t$ is eventually enabled, taken, and has the desired effect.

UniB is similar, except the witnessing term is the fresh variable.

Remark 1. We see the usefulness of indexed rules in the above proof. If we simply had an ImpR rule, rather than an ImpR $p\ q$ rule for each formula p and q, we would have to further argue that this rule eventually applies to exactly the implication $p \longrightarrow q$ we need it to. Perhaps we need to argue first that $p \longrightarrow q$ eventually reaches the front of the sequent or similar delicate reasoning. This is where fairness concerns would show up. We have sidestepped the issue by using very specific rules.

Consider now the second perspective. The countermodel in Fig. 7 uses the term universe (also called Herbrand universe) where every variable and function symbol evaluates to itself. Thus, the universal quantifier, which ranges over a given domain, ranges over terms. Now, read the sets A and B as formulas we wish to satisfy and falsify, respectively.

Lemma 3 (A Hintikka set induces a countermodel). *Let A and B be sets of formulas fulfilling the Hintikka requirements. Then $M\ A$ satisfies formulas in A and falsifies formulas in B. In Isabelle, if Hintikka $A\ B$ then $(p \in A \longrightarrow M\ A\ p) \wedge (p \in B \longrightarrow \neg\ M\ A\ p)$.*

Proof. By well founded induction on the size of the formula, such that the induction hypothesis applies to subformulas and instances of universally quantified formulas.

For $\bot \in A$, this contradicts *FlsA* so the thesis holds vacuously. For $\bot \in B$, the thesis holds trivially since $\bot$ is falsified by every model.

For $\dagger P\ ts \in A$, the thesis holds by the definition of *M*. For $\dagger P\ ts \in B$, we cannot have $\dagger P\ ts \in A$ due to *Basic* and so the thesis holds by the definition of *M*.

For $p \longrightarrow q \in A$ and $p \longrightarrow q \in B$ the theses hold by the induction hypotheses at p and q and the conditions *ImpA* and *ImpB*, respectively.

For $\forall p \in A$ and $\forall p \in B$ the theses hold by the induction hypotheses at $\langle t \rangle p$ for all t and by the conditions *UniA* and *UniB*, respectively.

Any saturated escape path induces a countermodel, contradicting validity.

Theorem 2 (Prover completeness). *For any valid sequent, the prover terminates.*

Proof. If the prover does not find a proof, then by the framework, the proof attempt contains a saturated escape path. By Lemma 2, this epath fulfills the Hintikka requirements. By Lemma 3, we can build a model that satisfies every assumption formula and falsifies every conclusion formula. This model contradicts the validity of the sequent.

We join the soundness and completeness theorems in a corollary on formulas.

Corollary 1. *The prover terminates if, and only if, the given formula is valid. In Isabelle, fix p :: fm and let $t \equiv$ prover $([], [p])$, then tfinite $t \wedge$ wf $t \longleftrightarrow (\forall\,(E :: - \Rightarrow tm)\ F\ G.\ [\![E,\ F,\ G]\!]\ p)$.*

References

1. Ben-Ari, M.: Mathematical Logic for Computer Science. Springer, Cham (2012). https://doi.org/10.1007/978-1-4471-4129-7
2. Blanchette, J.C.: Formalizing the metatheory of logical calculi and automatic provers in Isabelle/HOL (invited talk). In: Mahboubi, A., Myreen, M.O. (eds.) Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, pp. 1–13. ACM (2019). https://doi.org/10.1145/3293880.3294087
3. Blanchette, J.C., Popescu, A., Traytel, D.: Abstract completeness. Archive of Formal Proofs (2014). https://isa-afp.org/entries/Abstract_Completeness.html. Formal proof development
4. Blanchette, J.C., Popescu, A., Traytel, D.: Unified classical logic completeness. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS (LNAI), vol. 8562, pp. 46–60. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08587-6_4
5. Blanchette, J.C., Popescu, A., Traytel, D.: Soundness and completeness proofs by coinductive methods. J. Autom. Reason. **58**(1), 149–179 (2016). https://doi.org/10.1007/s10817-016-9391-3
6. de Bruijn, N.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. In: Nederpelt, R., Geuvers, J., de Vrijer, R. (eds.) Selected Papers on Automath, Studies in Logic and the Foundations of Mathematics, vol. 133, pp. 375–388. Elsevier (1994). https://doi.org/10.1016/S0049-237X(08)70216-7, reprinted from: Indagationes Math, 34, 5, pp. 381–392, by courtesy of the Koninklijke Nederlandse Akademie van Wetenschappen, Amsterdam
7. From, A.H.: Synthetic completeness for a terminating Seligman-style tableau system. In: de'Liguoro, U., Berardi, S., Altenkirch, T. (eds.) 26th International Conference on Types for Proofs and Programs, TYPES 2020, University of Turin, Italy, 2–5 March 2020. LIPIcs, vol. 188, pp. 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.TYPES.2020.5
8. From, A.H.: Formalized soundness and completeness of epistemic logic. In: Silva, A., Wassermann, R., de Queiroz, R.J.G.B. (eds.) WoLLIC 2021. LNCS, vol. 13038, pp. 1–15. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88853-4_1
9. From, A.H.: A succinct formalization of the completeness of first-order logic. In: Basold, H., Cockx, J., Ghilezan, S. (eds.) 27th International Conference on Types for Proofs and Programs, TYPES 2021, Leiden, The Netherlands, 14–18 June 2021 (Virtual Conference). LIPIcs, vol. 239, pp. 8:1–8:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.TYPES.2021.8
10. From, A.H., Jacobsen, F.K.: Verifying a sequent calculus prover for first-order logic with functions in Isabelle/HOL. In: Andronick, J., de Moura, L. (eds.) 13th International Conference on Interactive Theorem Proving, ITP 2022, Haifa, Israel, 7–10 August 2022. LIPIcs, vol. 237, pp. 13:1–13:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ITP.2022.13
11. From, A.H.: A Naive prover for first-order logic. Archive of Formal Proofs (2022). https://isa-afp.org/entries/FOL_Seq_Calc3.html, Formal proof development
12. From, A.H., Jacobsen, F.K.: A sequent calculus prover for first-order logic with functions. Archive of Formal Proofs (2022). https://isa-afp.org/entries/FOL_Seq_Calc2.html, Formal proof development
13. From, A.H., Jensen, A.B., Schlichtkrull, A., Villadsen, J.: Teaching a formalized logical calculus. Electron. Proc. Theor. Comput. Sci. **313**, 73–92 (2020). https://doi.org/10.4204/EPTCS.313.5

14. Gödel, K.: Die Vollständigkeit der Axiome des logischen Funktionenkalküls. Monatshefte für Mathematik und Physik **37**(1), 349–360 (1930). https://doi.org/10.1007/BF01696781
15. Henkin, L.: The discovery of my completeness proofs. Bull. Symb. Log. **2**(2), 127–158 (1996). https://doi.org/10.2307/421107
16. Jensen, A.B., Larsen, J.B., Schlichtkrull, A., Villadsen, J.: Programming and verifying a declarative first-order prover in Isabelle/HOL. AI Commun. Eur. J. Artif. Intell. **31**(3), 281–299 (2018). https://doi.org/10.3233/AIC-180764
17. Kleene, S.C.: Mathematical Logic. Courier Corporation (2002)
18. Michaelis, J., Nipkow, T.: Formalized proof systems for propositional logic. In: Abel, A., Forsberg, F.N., Kaposi, A. (eds.) 23rd International Conference on Types for Proofs and Programs (TYPES 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 104, pp. 5:1–5:16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.TYPES.2017.5
19. Nipkow, T., Klein, G.: Concrete Semantics - With Isabelle/HOL. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10542-0
20. Pastre, D.: Muscadet 2.3: a knowledge-based theorem prover based on natural deduction. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS, vol. 2083, pp. 685–689. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45744-5_56
21. Pelletier, F.J.: Automated natural deduction in THINKER. Stud. Logica. **60**(1), 3–43 (1998). https://doi.org/10.1023/A:1005035316026
22. Ridge, T., Margetson, J.: A mechanically verified, sound and complete theorem prover for first order logic. In: Hurd, J., Melham, T. (eds.) TPHOLs 2005. LNCS, vol. 3603, pp. 294–309. Springer, Heidelberg (2005). https://doi.org/10.1007/11541868_19
23. Schlichtkrull, A., Blanchette, J.C., Traytel, D.: A verified prover based on ordered resolution. In: Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, pp. 152–165. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3293880.3294100
24. Schulz, S., Pease, A.: Teaching automated theorem proving by example: PyRes 1.2. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 158–166. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_9
25. Villadsen, J., Schlichtkrull, A., From, A.H.: A verified simple prover for first-order logic. In: Konev, B., Urban, J., Rümmer, P. (eds.) Proceedings of the 6th Workshop on Practical Aspects of Automated Reasoning. CEUR Workshop Proceedings, vol. 2162, pp. 88–104. CEUR-WS.org (2018). https://ceur-ws.org/Vol-2162/paper-08.pdf

Author Index