Christoph Benzmüller
Jens Otten
Revantha Ramanayake (Eds.)

Automated Reasoning in Quantified Non-Classical Logics

5th International Workshop, ARQNL 2024,
Nancy, France, July 1st, 2024

Proceedings

Originally published online by
CEUR Workshop Proceedings (CEUR-WS.org)

Preface

This volume contains the proceedings of the Fifth International Workshop on *Automated Reasoning in Quantified Non-Classical Logics* (ARQNL 2024), held July 1st, 2024, in Nancy, France. The workshop was affiliated and co-located with the *International Joint Conference on Automated Reasoning* (IJCAR 2024). The aim of the ARQNL 2024 Workshop has been to foster the development of proof calculi, automated theorem proving (ATP) systems and model finders for all sorts of quantified non-classical logics. The ARQNL workshop series provides a forum for researchers to present and discuss recent developments in this area.

Non-classical logics — such as modal logics, conditional logics, intuitionistic logic, description logics, temporal logics, linear logic, multivalued logic, dynamic logic, deontic logic, fuzzy logic, paraconsistent logic, relevance logic, free logic, natural logic — have many applications in Artificial Intelligence, Computer Science, Philosophy, Linguistics, and Mathematics. Hence, the automation of proof search in these logics is a crucial task. It is in particular the aim of the ARQNL workshop series to initiate and foster practical implementations and evaluations of such ATP systems for non-classical logics.

The ARQNL 2024 Workshop received eight paper submissions. Each paper was reviewed by at least three referees, and following an online discussion, all eight research papers were selected to be included in the proceedings. The ARQNL 2024 Workshop included invited talks by Didier Galmiche (“Separation Logics: Semantics and Proofs”) and Amir Akbar Tabatabai (“On the Computational Content of Intuitionistic Modal Proofs”).

We would like to sincerely thank the invited speakers, all authors for their contributions and all active participants of the ARQNL 2024 workshop. We would also like to thank the members of the Program Committee of ARQNL 2024 for their professional work in the reviewing process. Finally, we would like to acknowledge the support of the EasyChair conference management system.

Bamberg, Oslo and Groningen
July 2024

Christoph Benzmüller
Jens Otten
Revantha Ramanayake

Organization

Program Committee

Christoph Benzmüller	University of Bamberg, Germany – co-chair
Ana de Almeida Borges	University of Barcelona, Spain
Camillo Fiorentini	University of Milano, Italy
Marianna Girlando	University of Amsterdam, Netherlands
Andrzej Indrzejczak	University of Lodz, Poland
Annika Kanckos	University of Helsinki, Finland
Dominik Kirst	Ben-Gurion University, Israel
Timo Lang	University College London, UK
Tomer Libal	University of Luxembourg and Enidia AI
Larry Moss	Indiana University Bloomington, USA
Nicola Olivetti	Aix-Marseille University, France
Jens Otten	University of Oslo, Norway – co-chair
Xavier Parent	Vienna University of Technology, Austria
Revantha Ramanayake	University of Groningen, Netherlands – co-chair
Ramaswamy Ramanujam	Institute of Mathematical Sciences, Chennai, India

Editors & Workshop Chairs

Christoph Benzmüller
University of Bamberg (and Freie Universität Berlin)
E-mail: c.benzmueller@gmail.com
Web: http://christoph-benzmueller.de

Jens Otten
University of Oslo (and Potassco Solutions)
E-mail: jeotten@jens-otten.de
Web: https://jens-otten.de

Revantha Ramanayake
University of Groningen
E-mail: d.r.s.ramanayake@rug.nl
Web: https://rug.nl/staff/d.r.s.ramanayake/

Contents

Separation Logics: Semantics and Proofs (Extended Abstract)

Didier Galmiche

Université de Lorraine, CNRS, LORIA Vandoeuvre-lès-Nancy, F-54506, France

Abstract

In this talk we give an overview of works and results about so-called BI-based Separation Logics, with a main focus on semantics and proofs. We present some key ideas and works mainly developed in our research team in LORIA laboratory (Nancy, France) since more than twenty years. After a reminder about resource models and resource logics we start to present the BI logic (with intuitionistic additives), Boolean BI (BBI) logic (with classical additives), and also BI's Pointer logic, called now Separation Logic, that deals with memory cells. We summarize the main results about semantics and proofs in these logics with an emphasis on the notions of constraints and resource graphs on which the design of labelled proof systems is based. The next part is devoted to the presentation of various modal and epistemic BBI-based (or separation) logics that manage different kinds of modalities, again with a focus on semantics, expressiveness, and proofs. We complete this overview by mentioning recent works on proof translations between calculi in BI and their possible consequences on some completeness results. Then we conclude with some perspectives about separation logics from current studies of non-aggregative models of resource composition.

Keywords

Logics with Separation, Resources, Semantics, Labelled Calculi, Modal logics

1. Extended Abstract

In this talk we give an overview of researchs and results about so-called BI-based Separation Logics, with a main focus on semantics and proofs. We present some key ideas, works and results mainly developed in our research team in LORIA laboratory (Nancy, France) since more than twenty years. After a reminder about resource models and resource logics we start by giving the BI logic (with intuitionistic additives) [1], its bunched calculus (LBI) and its resource semantics that is complete only for BI without $\bot$ [2]. We also remind that BI logic, that focuses on resource separation and sharing, is different from Linear Logic [3], that focuses instead on resource comsumption. We also consider some variants of BI logic like Boolean BI (BBI) (with classical additives) [4] and BI's Pointer logic, also called Separation Logic (SL), that is based on BBI and deals with memory cells [5]. Separation Logic has provided key developments in formal reasoning about programs with the frame rule that allows a proof to be localized to the resources that a program component accesses andalso with the key notion of local reasoning [6, 7]. We do not consider here the impressive developments from SL in the last twenty years but we can mention the Concurrent Separation Logic [8], that allows modular reasoning about threads that share storage and other resources, the Incorrectness Separation Logic [9], with the goal of proving that compositional bug catchers find actual bugs and its concurrent extension to account for bug catching in concurrent programs [10]. In the rest of the talk Separation logics denote the (B)BI-based logics with separation and their extensions.

After this first part we introduce BI logic, its semantics and mainly on so-called resource tableaux, that are labelled tableaux with resource constraints of two kinds (assertions and requirements), and also define the key notion of resource graph [2]. The tableau calculus designed for BI logic is proved sound and complete w.r.t. the Grothendick topological semantics (GR models). To solve the question to have a semantics of BI based on partial monoids we propose different new semantics for BI logic, namely a relational semantics (RM models), a Kripke resource semantics (KR models) and a partially defined

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

Didier.Galmiche@loria.fr (D. Galmiche)

monoid semantics (PDM models) and show that the tableaux calculus is sound and complete w.r.t. these models. Moreover BI logic is sound and complete w.r.t these models except for KR models, that is still an open question [11].

From our notion of resource graph we also show that one can define a connection-based characterization of BI's validity with constraints without using prefixes like in other non-classical logics [12]. Moreover we study and define resource graphs for other logics and then provide a tableaux calculus for BI's Pointer logic (or SL) [13, 14], a new connection-based characterization of validity for Non Commutative Logic [15] and also such a connection-based characterization for Bi-intuitionistic logic (Bi-Int) with both implication and co-implication [16]. These works illustrate the interest to study semantics and then to define and use constraints and resource graphs for designing calculi for different resource logics.

We also mention works on semantics for Boolean BI (BBI) with the proposal of a Kripke relational semantics for BBI (a non-deterministic monoidal semantics) with faithful embeddings of S4 and of IL into BBI. It provides also a logical characterization of the observational power of BBI through an adequate definition of bisimulation [17]. From this study of BBI semantics one can propose a labelled tableau for BBI that is sound and also a sound and faithful embedding of BI into BBI [18]. In addition we propose a complete phase semantics for BBI and an embedding between phase semantics for ILL and Kripke semantics of BBI. By defining a fragment of ILL undecidable and complete for phase semantics one can prove the undecidability of BBI [19, 20]. Concerning the labelled tableaux for partial monoidal Boolean BI, it is important to note that the schema of its proof of strong completeness is original and it has been completely formalized in Coq [21].

In the next part we present some modal extensions of (B)BI-based logics with different kinds of modalities. A first one, called BI-Loc, considers a spatial modality for locations and resource trees and proposes a new logic for resource distribution [22]. A second one, called DBI, extends BBI logic with two modalities for expressing properties on states of process or on interacting systems. As the related semantics introduces states in addition to resources, one also introduces state constraints in addition to the resource constraints and then define a labelled tableaux calculus that is sound and complete w.r.t. the semantics [23]. A third one, called DMBI (Dynamic Modal BI), is an extension of DBI for introducing dynamics (resource transformations) with modalities à la Hennessy-Milner. Then new kinds of constraints (resources, actions, states) are considered in the tableaux calculus that is proved sound and complete w.r.t. the semantics [24]. A fourth one, called LSM, considers modalities, generalyzing S4 modalities. They are defined with two-dimensional worlds, one for S4 accessibility and one for resource parametrization. and allow us to express properties of models of distributed computing. A sound and complete labelled calculus is provided [25].
For these new modal separation logics we have mentioned modelling examples in order to illustrate their expressivity and also the related labelled calculi with their properties of soundness and completeness. We emphasize that the completeness proofs are based on the proof schema given in [21] for BBI logic, that is extended and adapted, in a non-trivial way, for these logics.

In this context we also consider works on separation logics with knowledge and then propose some epistemic extensions of BBI. One first, called ESL, considers epistemic modalities with a semantics for which the possible (epistemic) worlds are resources that can be composed and decomposed. The tableaux calculus for ESL is defined with resource constraints but also with agent constraints and it is proved sound and complete w.r.t. the semantics [26]. The addition of public announcements modalities to ESL has been also studied and results in a public announcement separation logic (PASL) [27].
Another epistemic extension of BBI, called ERL, deals with epistemic modalities that are parametrized on agents' local resources and allow us the modelling of some acces control problems. Let us note that it is a conservative extension of BBI and Epistemic Logic. Again the study of the semantics and the expressiveness has been completed by the design of a sound and complete labelled tableaux calculus [28].

We then continue this overview of BI and BBI modal and/or epistemic extensions, with a strong focus on semantics and proof theory, by mentioning two works that benefit from some of the previous results. A first work studies proof translations in BI logic between labelled and label-free calculi. The results and their proofs emphasize the difficulty to design a translation from a proof in a labelled calculus into a proof in a label-free calculus in BI logic, like in other logics. We expect that having a general schema for such a translation would help us to prove a still open question for BI logic, that is the completeness of the bunched calculus LBI w.r.t. KRM semantics [29].
A second work is about Separation Logic (SL) and inductive predicates. Proof systems for this logic are often dedicated to some fragments of SL (symbolic heaps) that cannot express some properties about pointers. They mainly allow either the full set of connectives, or the definition of arbitrary inductive predicates, but not both. Then we comment a cyclic labelled system for SL that allows both [30]. This work about cyclic proofs opens perspectives for some extensions of BI that will be developed in the future. .

We conclude with a current work on a temporal extension of BI, called LTBI (Linear Time Bunched Implication Logic) that is dedicated to resource evolution over time by combining BI separation connectives and LTL temporal connectives [31]. A new semantics is given and a labelled calculus is defined for LTBI and is proved sound but the completeness, proved for bounded timelines, is not trivial in the general case of unbounded timelines. We expect that the definition of a cyclic proof system for this logic would lead to the completeness result in this general setting. Finally we briefly present a current study of new non-aggregative models of resource composition, namely compositions that do not obey the principle that the whole is the sum of its parts. Our first objectives are to find algebraic properties characterizing such compositions and then to design resource logics for such non-aggregative compositions, if possible in the spirit of our previous works for BI, BBI and their extensions.

References

[1] P. W. O'Hearn, D. Pym, The Logic of Bunched Implications, Bulletin of Symbolic Logic 5 (1999) 215–244.

[2] D. Galmiche, D. Méry, Semantic Labelled Tableaux for propositional BI (without bottom), Journal of Logic and Computation 13 (2003) 707–753.

[3] J. Girard, Linear logic, Theoretical Computer Science 50 (1987) 1–102.

[4] D. J. Pym, The Semantics and Proof Theory of the Logic of Bunched Implications, volume 26, Applied Logic Series. Kluwer Academic Publishers, 2002.

[5] S. S. Ishtiaq, P. W. O'Hearn, BI as an Assertion Language for Mutable Data Structures, in: Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2001, pp. 14—26.

[6] J. C. Reynolds, Separation Logic: A Logic for Shared Mutable Data Structures., 17th Annual IEEE Symposium on Logic in Computer Science (LICS'02) (2002) 55–74.

[7] P. W. O'Hearn, Separation Logic, Communications of ACM 62 (2019) 86–95.

[8] S. Brookes, P. W. O'Hearn, Concurrent Separation Logic, ACM SILOG News 3 (2016) 47–65.

[9] P. W. O'Hearn, Incorrectness logic, in: Proc. ACM on Programming Languages 4, POPL, Article 10, 2019, pp. 1–32.

[10] A. Raad, J. Berdine, D. Dreyer, P. W. O'Hearn, Concurrent Incorrectness Separation Logic, in: Proc. ACM on Programming Languages 6, POPL, Article 34, 2022, pp. 1–29.

[11] D. Galmiche, D. Méry, D. Pym, The semantics of BI and Resource Tableaux, Mathematical Structures in Computer Science 15 (2005) 1033–1088.

[12] D. Galmiche, D. Méry, Connection-based proof search in propositional BI logic, in: 18th Int. Conference on Automated Deduction, CADE-18, LNAI 2392, 2002, pp. 111–128. Copenhagen, Danemark.

[13] D. Galmiche, D. Méry, Characterizing provability in BI's pointer logic through resource graphs, in: Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2005, LNAI 3835, Montego Bay, Jamaica, 2005, pp. 459–473.
[14] D. Galmiche, D. Méry, Tableaux and Resource Graphs for Separation Logic, Journal of Logic and Computation 20 (2010) 189–231.
[15] D. Galmiche, J. Notin, Connection-based Proof Construction in Non-commutative Logic, in: 10th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'03, LNCS 2850, 2003, pp. 422–436. Almaty, Kazakhstan.
[16] D. Galmiche, D. Mery, A Connection-based Characterization of Bi-intuitionistic Validity, Journal of Automated Reasoning 51 (2013) 3–26.
[17] D. Galmiche, D. Larchey-Wendling, Expressivity properties of Boolean BI through Relational Models, in: 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2006, LNCS 4337, 2006, pp. 358–369. Kolkata, India.
[18] D. Larchey-Wendling, D. Galmiche, Exploring the Relation between Intuitionistic BI and Boolean BI: An unexpected Embedding, Mathematical Structures in Computer Science 19 (2009) 435–500.
[19] D. Larchey-Wendling, D. Galmiche, The Undecidability of Boolean BI through Phase Semantics, in: 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, Edinburgh, UK, 2010, pp. 147–156.
[20] D. Larchey-Wendling, D. Galmiche, Nondeterministic Phase Semantics and the Undecidability of Boolean BI, ACM Transactions on Computational Logic 14 (2013) 6.
[21] D. Larchey-Wendling, The Formal Strong Completeness of Partial Monoidal Boolean BI, Journal of Logic and Computation 26 (2014) 605–640.
[22] N. Biri, D. Galmiche, Models and Separation Logics for Resource Trees, Journal of Logic and Computation 17 (2007) 687–726.
[23] J. Courtault, D. Galmiche, A Modal BI Logic for Dynamic Resource Properties, in: Logical Foundations of Computer Science, LFCS 2013, LNCS 7734, 2013, pp. 134–148. San Diego, CA.
[24] J.-R. Courtault, D. Galmiche, A Modal Separation Logic for Resource Dynamics., Journal of Logic and Computation 28 (2018) 733–778.
[25] J.-R. Courtault, D. Galmiche, D. Pym, A Logic of Separating Modalities, Theoretical Computer Science 637 (2016) 30–58.
[26] J.-R. Courtault, H. van Ditmarsch, D. Galmiche, An Epistemic Separation Logic, in: 22nd Int. Workshop on Logic, Language, Information, and Computation, WoLLIC 2015, LNCS 9160, Bloomington, IN, United States, 2015, pp. 156–173.
[27] J.-R. Courtault, H. van Ditmarsch, D. Galmiche, A Public Announcement Separation Logic, Mathematical Structures in Computer Science 29 (2019) 828–871.
[28] D. Galmiche, P. Kimmel, D. Pym, A Substructural Epistemic Resource Logic: Theory and Modelling Applications, Journal of Logic and Computation 29 (2019) 1251–1287.
[29] D. Galmiche, M. Marti, D. Méry, Relating Labelled and Label-Free Bunched Calculi in BI Logic, in: 28th Int. Conference on Automated Reasoning with Analytic tableaux and Related Methods, Tableaux 2019, LNAI 11714, 2019, pp. 130–146. London, UK.
[30] D. Galmiche, D. Mery, Labelled Cyclic Proofs for Separation Logic, Journal of Logic and Computation 31 (2021) 892–922.
[31] D. Galmiche, D. Méry, Labelled Tableaux for Linear Time Bunched Implication Logic, in: 8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, LIPIcs, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagsthul, Roma, Italy, 2023, p. 27:1–27:17.

On the Computational Content of Intuitionistic Modal Proofs (Abstract)

Amirhossein Akbar Tabatabai

Institute of Mathematics, Czech Academy of Sciences

Abstract

The term computational content typically refers to the computational information hidden in a proof of a first-order sentence. For instance, for a statement of the form $\forall x \exists y \phi(x, y)$, the computational content is a method to compute y for a given x such that $\phi(x, y)$ holds. However, the concept of extracting computational data from logical proofs is not limited to first-order theories. For example, in intuitionistic propositional or modal systems, the computational content of a disjunction $\phi \vee \psi$ is a way to read a proof of the disjunction and compute which disjunct is provable and then provide a proof for that disjunct [1]. When this computation can be carried out in polynomial time, the system is said to exhibit the *feasible disjunction property*.

In this talk, we will present our recent work [2] on the feasible disjunction property in intuitionistic modal logics. We begin by introducing a syntactically defined family of formulas, referred to as *constructive formulas*, to formalize the notion of *constructively acceptable axioms*. This class is chosen to be tight: it includes all commonly accepted axioms, yet any deviation from its syntactical form results in systems that lack the disjunction property and are therefore constructively unacceptable. Next, we demonstrate that any intuitionistic modal system axiomatized by constructive axioms and satisfying a mild technical condition possesses the feasible disjunction property. On the positive side, this result establishes the feasible disjunction property for several intuitionistic modal systems, including CK, IK, their extensions with the modal axioms *T*, *B*, 4, 5, axioms for bounded width and depth, and their fragments such as $\mathsf{CK}_{\Box}$, propositional lax logic, and IPC. On the negative side, we show that if a sufficiently strong intuitionistic modal logic (meeting a mild technical condition) lacks the disjunction property, it cannot be axiomatized using constructive axioms. Furthermore, by generalizing our main theorem, we prove that IPC is the only intermediate logic that admits a constructive axiomatization.

Keywords

admissible rules, feasible disjunction property, intuitionistic modal logics

References

[1] S. R. Buss, P. Pudlák, On the computational content of intuitionistic propositional proofs, Annals of Pure and Applied Logic 109 (2001) 49–64.

[2] A. Akbar Tabatabai, R. Jalali, Universal proof theory: Feasible admissibility in intuitionistic modal logics, Annals of Pure and Applied Logic 176 (2025) 103526.

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

amir.akbar@gmail.com (A. A. Tabatabai)

https://sites.google.com/view/amirtabatabai (A. A. Tabatabai)

A Fresh Look at Relevant Number Theory

John Slaney[1]

[1]Australian National University, Canberra, ACT 2601, Australia

Abstract

Underlying numerical reasoning is the formal theory of arithmetic, and if the reasoning is to be carried out in a nonclassical logic then this will be a correspondingly nonclassical arithmetic. The relevant arithmetic $\mathbf{R}^\sharp$ was proposed around 50 years ago and is one of the few theories based on substructural logic to have been investigated in much detail. This paper surveys some old results concerning $\mathbf{R}^\sharp$ and recent attempts to extend it to deal with the rational numbers as well as the naturals. While the formal results here are not new, it is worthwhile to put them together and to present the topic as one of interest for contemporary research into nonclassical reasoning.

In almost every domain of genuine importance, reasoning needs to encompass not only pure logic but also numerical inferences. From the number of timesteps in a computation to the cost of an action or the state of a stockpile, quantitative as well as qualitative reasoning is everywhere in practice. At the root of such reasoning is elementary arithmetic—primitively, as a theory about natural numbers, extending to integers and to rational number theory, and eventually to analysis.

Arithmetic has of course been studied intensively as part of mathematical logic in the classical and constructivist traditions. In the history of more radically nonclassical logics, however, the literature on arithmetic is relatively sparse. This is unfortunate, as the question of which numerical inferences are available, and with what kind of logical guarantee, is sensitive to the choice of logic and is therefore of importance in the nonclassical setting. The purpose of the present paper is to note some results, ancient and modern, concerning theories of arithmetic in substructural logics.

One of the few such theories to have been seriously investigated is the arithmetic $\mathbf{R}^\sharp$, put forward by R.K. Meyer in the 1970s. $\mathbf{R}^\sharp$ has the proper axioms of Peano arithmetic, but the underlying logic is the relevant logic **R** rather than classical logic.

1. The logic RQ and $\mathbf{R}^\sharp$

The propositional logic **R** and its first order extension **RQ** are usually specified by means of a Hilbert system. The language has the unary connective $\neg$, binary connectives $\wedge$ and $\rightarrow$, and quantifiers $\forall$x. It is usual to define

$$\begin{aligned} A \vee B &= \neg(\neg A \wedge \neg B) \\ A \leftrightarrow B &= (A \rightarrow B) \wedge (B \rightarrow A) \\ A \circ B &= \neg(A \rightarrow \neg B) \\ \exists x A &= \neg\forall x \neg A \end{aligned}$$

The pure implication ($\rightarrow$) fragment of **R** extends that of linear logic by the addition of contraction:

Axioms:

r1	$A \rightarrow A$
r2	$A \rightarrow ((A \rightarrow B) \rightarrow B)$
r3	$(A \rightarrow B) \rightarrow ((C \rightarrow A) \rightarrow (C \rightarrow B))$
r4	$(A \rightarrow (A \rightarrow B)) \rightarrow (A \rightarrow B)$

Rule:

$A \rightarrow B,\ A \implies B$

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

john.slaney@anu.edu.au (J. Slaney)

Negation is rather classical, like the "strong negation" in logics of constructible falsity:

Axioms:

r5	$(A \to \neg B) \to (B \to \neg A)$
r6	$\neg\neg A \to A$

Conjunction and disjunction are additive (extensional) connectives satisfying the postulates:

Axioms:

r7	$(A \wedge B) \to A$
r8	$(A \wedge B) \to B$
r9	$((A \to B) \wedge (A \to C)) \to (A \to (B \wedge C))$
r10	$(A \wedge (B \vee C)) \to ((A \wedge B) \vee C)$

Rule:

$$A,\ B \implies A \wedge B$$

Quantifiers are added by means of very standard axioms, of which r14 is the only postulate of the positive logic which is not intuitionistically valid:

Axioms:

r11	$\forall x A \to A_{[x \leftarrow t]}$	(t free for x in A)
r12	$\forall x(A \to B) \to (\forall x A \to \forall x B)$	
r13	$(\forall x A \wedge \forall x B) \to \forall x(A \wedge B)$	
r14	$\forall x(A \vee B) \to (A \vee \forall x B)$	(x not free in A)
r15	$A \to \forall x A$	(x not free in A)
r16	$\forall x A$	(A an axiom)

The constructive form of quantifier confinement

$$\forall x(A \to B) \to (A \to \forall x B) \qquad (x \text{ not free in } A)$$

is derivable using r12 and r15. We note this now, as it will be needed later. For accounts of **R**, including its semantics and proof theory, see the original presentation by Anderson and Belnap [1], and Mares [2] for instance.

$\mathbf{R}^\sharp$ is a theory in an arithmetical vocabulary with one constant, 0 (zero), the unary operator $'$ (successor) and binary operators $+$ and $\cdot$ (addition and multiplication). Its axioms are the universal closures of:

a1	$x = x$
a2	$x = y \to (x = z \to y = z)$
a3	$x = y \to x' = y'$
a4	$x' = y' \to x = y$
a5	$0 \neq x'$
a6	$x + 0 = x$
a7	$x + y' = (x + y)'$
a8	$x \cdot 0 = 0$
a9	$x \cdot y' = (x \cdot y) + x$
a10	$(A_{[x \leftarrow 0]} \wedge \forall x(A \to A_{[x \leftarrow x']})) \to A$

The theorems of $\mathbf{R}^\sharp$are the **RQ** consequences of the axioms—note that a10 is an axiom *scheme* with infinitely many instances. Briefly, the arithmetic has most of the properties one would expect of a version of Peano arithmetic, including closure under universal generalisation and a fully classical equality relation satisfying all instances of the scheme

$\rightarrow$	T	t	F
T	T	F	F
t	T	t	F
F	T	T	T

$=$	0	1	2	3	4
0	t	F	F	F	F
1	F	t	F	F	F
2	F	F	t	F	F
3	F	F	F	t	F
4	F	F	F	F	t

Figure 1: Inconsistent model in the integers modulo 5

$$a = b \rightarrow (A_{[x \leftarrow a]} \rightarrow A_{[x \leftarrow b]})$$

Meyer's early work on $\mathbf{R}^\sharp$, unpublished in his lifetime, appeared in 2021 [3] prompting a renewal of interest in the topic. While establishing that $\mathbf{R}^\sharp$ has many classically familiar features, he observed that it has some decidedly unclassical ones too. Most notably, it has *finite* models! What, for example, if 0 is equal to 5? Classically, the answer is simple: 0 is not 5 and there is no more to say. Relevantly, the answer is more nuanced: if $0 = 5$ then numerical equality is indistinguishable from congruence *modulo* 5. That supposition is inconsistent, of course, but **R** is a paraconsistent logic and allows models of such inconsistent thoughts to exist.

Even the logic **RM3**, a very strong 3-valued extension of **R** which is almost classical, allows this (Figure 1). There are only 5 numbers in the domain. Addition and multiplication are interpreted *modulo* 5, and the successor of 4 is 0 (despite axiom a5 which says it isn't). There are three possible truth values for propositions: T and F are what you expect, while t is a weak kind of truth which is a fixed point for negation. An equation $a = b$ has the value t if a and b are the same object and the value F if they are different. All theorems of $\mathbf{R}^\sharp$ all get values T or t on this interpretation—never the value F. Clearly, the construction can be repeated for any modulus, providing a purely finitary proof (Gödel notwithstanding) that $\mathbf{R}^\sharp$ is *reliable* in that no false equations are provable in it.

$\mathbf{R}^\sharp$ permits easy proofs, by induction on z, of:

$$x = y \leftrightarrow x + z = y + z$$
$$x = y \rightarrow xz = yz$$

Note that while the first of these is an equivalence, the second holds in one direction only. In fact, the above theorems together with the models in the integers *modulo* n suffice for the observation that for any numerals a, b, c and d, the equation $a = b$ relevantly implies $c = d$ according to $\mathbf{R}^\sharp$ iff $|a - b|$ divides $|c - d|$. As special cases, $0 = 1$ implies all equations, while every equation implies $0 = 0$, because 1 divides everything while everything divides 0. Since $0 = 0$ provably implies all and only the theorems of $\mathbf{R}^\sharp$, we may abbreviate it to 't' and its negation $0 \neq 0$ to 'f'.

The monotonicity of multiplication, as recorded in the theorem noted above, depends crucially on the contraction axiom r4. If that axiom is dropped from the logic, giving a system with the same intensional fragment as linear logic, many of the implications between equations are lost. Slaney, Meyer and Restall [4] showed that in arithmetic based on relevant logics without contraction, $a = b$ implies $c = d$ if and only if $|a - b| = |c - d|$. In particular, in such arithmetics, no false equation ever implies a true one.

2. Difficulties

$\mathbf{R}^\sharp$ indeed provides an interesting view of arithmetical reasoning, but as a theory in the relevant logical tradition it faces some formidable difficulties. Two of these in particular stand out. Firstly, the theory is sadly incomplete, not just because of Gödel's theorems, which apply to every arithmetic, and not just because it was always supposed to be agnostic concerning some intensional formulae (involving the '$\rightarrow$' connective), but because it misses some of the purely extensional arithmetical facts from the corresponding classical Peano arithmetic. Secondly, $\mathbf{R}^\sharp$ is a theory of natural numbers only; unlike its

classical counterpart, it cannot easily be embedded in a wider mathematical account of number theory. We consider each of these issues in turn.

2.1. Problem: the classical sub-theory

The theorems of classical Peano arithmetic **PA** are, by definition, the consequences of the axioms a1 – a10 and suitable axioms for Boolean first order logic by closure under the rule of material detachment. By playing a little with De Morgan's laws and double negation, we may take material detachment in the form of the "disjunctive syllogism" or the rule γ:

$$A \vee B,\ \neg A \implies B$$

This is famously not a *derivable* rule of **R**, but for the logic **R** and also for **RQ** it is *admissible* in that there is no counter-example to it: no case in which its premises are theorems while its conclusion is not. This accords well with the world view associated with **R** on which truth-functional logic is right about truth-functional matters, but stands in need of a better theory of implication and a more sophisticated account of inference. The classical rule γ preserves truth in the intended models of **R**, but does not preserve satisfacton at arbitrary worlds in those models. Now the axioms of classical logic and of Peano arithmetic are all theorems of $\mathbf{R}^\sharp$, so if $\mathbf{R}^\sharp$ is closed under γ then it exactly agrees with classical **PA** in its classical (arrow-free) vocabulary. While γ is not a derivable rule of $\mathbf{R}^\sharp$, the closely related rule γ_f

$$A \vee B \vee f,\ \neg A \vee f \implies B \vee f$$

is easily seen to be derivable, so clearly any formula A in the $\rightarrow$-free vocabulary is a theorem of **PA** iff $A \vee f$ is a theorem of $\mathbf{R}^\sharp$. This embedding of classical arithmetic into $\mathbf{R}^\sharp$ is enough to secure many results, such as Gödel's theorems, but it falls short of what the **R** enthusiast would really want. At the time of writing his exposition [3], Meyer expressed the hope that γ would prove to be admissible for $\mathbf{R}^\sharp$ just as it is for **RQ** and for $\mathbf{R}^{\sharp\sharp}$, the exension of $\mathbf{R}^\sharp$ resulting by closing under the ω rule (if $\vdash A_{[x \leftarrow n]}$ for every numeral n then $\vdash \forall x A$).

That hope, however, was vain. Friedman and Meyer [5] showed that there are theorems of **PA** which contain only the positive connectives $\wedge$ and $\vee$ and quantifiers, but which cannot be proved without using axiom a5. Such formulae likewise have no purely positive (negation-free) proofs in $\mathbf{R}^\sharp$; but $\mathbf{R}^\sharp$ is a conservative extension of its positive fragment, so there are theorems of **PA** which are not theorems of $\mathbf{R}^\sharp$, and consequently γ fails.

2.2. Problem: rational extension

The failure of γ was disappointing for the early proponents of relevant arithmetic, but we could perhaps learn to live with that. There is, however, a more serious issue which is potentially devastating for the entire program. A theory of arithmetic cannot merely be an account of natural numbers (or integers). If it is to be proposed seriously for use in mathematics, it must extend at least to the rational numbers. Here we consider the non-negative rationals, as the extension to deal with negative ones goes along with the extension from natural numbers to integers, which complicates the theory slightly but not in a fundamental way.

Any extension of a natural number theory to a theory of rational arithmetic is subject to at least three obvious desiderata. A theory of the numbers should be:

a) mathematically reasonable—for instance, it should contain all true ground equations, and allow ordinary reasoning steps such as paramodulation (replacement of equals), appeals to the transitivity of equality and the like;
b) related to natural number theory at least in that for any natural numbers a, b, c, d ($b, d > 0$) it should be provable that $\frac{a}{b} = \frac{c}{d}$ is equivalent to $ad = bc$;
c) a conservative extension of the theory of naturals.

Unfortunately, every extension of $\mathbf{R}^\sharp$ from natural to rational number theory violates at least one of these desiderata.

The proof of this is very simple: by condition (a) $\frac{4}{6}$ is provably equal to $\frac{2}{3}$, and so by either transitivity or replacement it is a theorem that $\frac{4}{6} = \frac{1}{1}$ implies $\frac{2}{3} = \frac{1}{1}$; but by condion (b) this means that $4 = 6$ implies $2 = 3$, which is not a theorem of $\mathbf{R}^\sharp$ because of the models in the integers modulo 2. Any finite model with greater modulus will give rise in this way to similar counter-examples to intuitively well-motivated principles, so in the presence of (a) and (b), desideratum (c) cannot be met.

3. Possible solutions within $\mathbf{R}^\sharp$

Leaving aside for the moment the failure of γ, we may note some possible solutions to the problem of extension to rational number theory. One bold solution to the trilemma is to hold onto $\mathbf{R}^\sharp$ just as it is, to introduce rational number theory by means of contextual definitions, so that the relationship between it and natural number theory is as close as it could be, and simply to let the theorems fall where they will. That is, we regard a term like $\frac{a}{b} + \frac{c}{d}$ as nothing more than syntactic sugar for the expression $\frac{ad+bc}{bd}$ in which the addition function is applied only to naturals. Similarly, when we write $\frac{a}{b} \times \frac{c}{d}$ we really mean $\frac{ac}{bd}$ and an equation of the form $\frac{a}{b} = \frac{c}{d}$ is nothing but another way of saying $ac = bd$, which is a formula in the primitive language of $\mathbf{R}^\sharp$ and makes no reference to anything beyond natural numbers. There is a *little* more to be done, to avoid terms like $\frac{a}{0}$, but this can be managed.

On this account, rational number theory is by definition part of natural number theory, so desiderata (b) and (c) are met. Desideratum (a) however is comprehensively violated. Any model of $\mathbf{R}^\sharp$ is a model of rational number theory on this account. That includes the finite models, in which rational equality as just defined looks nothing like an identity relation. It is not even transitive, and does not support the most basic paramodulation inferences. Hence, although $\mathbf{R}^\sharp$ as a rational arithmetic is an *interesting* theory, it is hardly convincing as a basis for numerical reasoning.

The alternative to giving up desideratum (a) is, of course, to give up desideratum (b). This might be done in many ways, as there is no unique **RQ** theory lacking a particular equivalence. The most promising line seems to be to take as axioms the analogues of a1 and a2 for rational equations, together with the postulate $\frac{ac'}{b'c'} = \frac{a}{b'}$ (avoiding division by zero by requiring b and c to be successors) and the monotonicity postulate $a = b \rightarrow \frac{a}{c'} = \frac{b}{c'}$ but not its converse. The effect is that we secure half of desideratum (b), allowing the relevant inference from $ad' = b'c$ to the rational equation $\frac{a}{b'} = \frac{c}{d'}$, but not the converse except as an admissible rule. This asymmetry is in harmony with the overall style of $\mathbf{R}^\sharp$, whereby multiplication is monotonic but not cancellative. This version of relevant arithmetic is another theory worthy of investigation, as it promises a workable account in keeping with the view of numbers embodied in $\mathbf{R}^\sharp$.

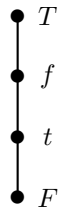
The finite models in the integers modulo n are still there, of course, and still give us non-trivial information about the natural numbers, but they say nothing about rationals because in those models all rational numbers collapse to a single point. This is easy to see: if $0 = n$ then $\frac{n}{n} = \frac{0}{n}$ which is to say the rational 1 is the same as the rational 0. But where q is any rational, $q \cdot 1 = q$ while $q \cdot 0 = 0$, so all rationals are equal to the rational zero and so equal to each other.

4. Extending $\mathbf{R}^\sharp$

The final option is to abandon the goal of staying within $\mathbf{R}^\sharp$, and instead to strengthen natural number theory to the point that the three listed desiderata for rational arithmetic can all be satisfied together. The arithmetic $\mathbf{R}^\natural$ [6] adds to $\mathbf{R}^\sharp$ an axiom

a11 $\quad 0 = x' \rightarrow 0 = 1$

Since the equation $0 = 1$ implies all other equations in $\mathbf{R}^\sharp$, this is equivalent to the principle that every incorrect equation implies that all numbers are equal. It is also equivalent to adding cancellation in the form



$\rightarrow$	T	f	t	F
T	T	F	F	F
f	T	t	F	F
t	T	f	t	F
F	T	T	T	T

$T: \quad f \circ f$

$f: \quad 0 \neq 0$

$t: \quad 0 = 0$

$F: \quad f \rightarrow t$

Figure 2: Arithmetic modulo 1

$$ax' = bx' \rightarrow a = b$$

and in the presence of desideratum (b) above, to transitivity in the form

$$\tfrac{a}{b'} = \tfrac{c}{d'} \rightarrow (\tfrac{c}{d'} = \tfrac{e}{f'} \rightarrow \tfrac{a}{b'} = \tfrac{e}{f'})$$

Hence $\mathbf{R}^\natural$ is the minimum supertheory of $\mathbf{R}^\sharp$ capable of extension to rational arithmetic without violating desiderata (a) and (b).

Of course, the finite models are no longer available, with the sole exception of the most extreme, in which there is only one number. In this model, zero is a fixed point for all arithmetical functions, and the propositional structure is 4-valued (Figure 2). All equations take the value t, which counts as true, so this structure—the only finite model of $\mathbf{R}^\natural$—cannot be used to show reliability in the way this could be done for $\mathbf{R}^\sharp$. However, it does show some formulae, such as $0 \neq 0 \rightarrow 2 + 2 = 4$ to be non-theorems, so there is still a finitary proof of non-triviality.

5. Classical arithmetic regained

The move from $\mathbf{R}^\sharp$ to $\mathbf{R}^\natural$ does more than create a workable theory of rational arithmetic. It also restores the desired relationship between the relevant and classical theories of the naturals. Recall that since classical Peano arithmetic is obtainable from the fragment of $\mathbf{R}^\sharp$ in the classical (arrow-free) vocabulary by closing under the rule γ or material detachment. The classical fragment of $\mathbf{R}^\natural$ therefore coincides with classical arithmetic if it is γ-closed. For the whole of $\mathbf{R}^\natural$, the admissibility of γ is an open question, but the special case for formulae in the classical vocabulary does hold, and this is enough to secure all of the classical theorems.

The first lemma towards this result is due to Dunn, Meyer and Leblanc [7] and is one of the earliest important results on quantified relevant logic. By an **RQ** theory, we mean a set of formulae closed under adjunction and **RQ**-provable implication. A theory is *prime* if it never contains a disjunction unless it contains one of the disjuncts, *regular* if it contains all theorems of **RQ**, and *rich* if every universal $\forall x A$ is in the theory if every ground instance $A_{[x \leftarrow t]}$ is.

Lemma 1. Let θ be a regular **RQ** theory and $B \notin \theta$. Then there is a prime, rich theory θ' such that $\theta \subseteq \theta'$ and $B \notin \theta'$.

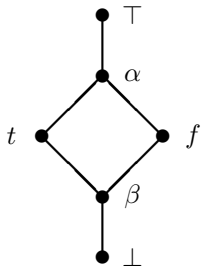
For proof see the original paper [7]. As a consequence:

Lemma 2. Let θ' be as above and let A be a ground formula. Then the principal θ'-theory of A (i.e. $\{C : A \rightarrow C \in \theta'\}$) is rich.

Lemma 2 is easily proved using lemma 1 and the confinement law

$$\forall x(A \rightarrow C) \rightarrow (A \rightarrow \forall x C)$$

Remember that x is not free in A.



$\top$: $0 \neq 1$

α : $0 = 0 \vee 0 \neq 0$

t : $0 = 0$

f : $0 \neq 0$

β : $0 = 0 \wedge 0 \neq 0$

$\bot$: $0 = 1$

Figure 3: The De Morgan lattice DM6

Lemma 3. Every quantifier-free ground formula in the extensional vocabulary is equivalent in $\mathbf{R}^\natural$ to one of the following 6:

$$0 = 0 \qquad 0 = 1 \qquad 0 = 0 \wedge 0 = 1$$
$$0 \neq 0 \qquad 0 \neq 1 \qquad 0 = 0 \vee 0 = 1$$

Proof: as in classical Peano arithmetic, every ground equation is provably equivalent either to $0 = 0$ or to $0 = 1$, and clearly the set of 6 is closed (up to provable equivalence) under the extensional connectives $\wedge$, $\vee$ and $\neg$.

Lemma 4. Let θ be a regular, prime, rich **RQ** theory. Then every ground formula in the extensional vocabulary is θ-equivalent to one of the above 6 formulae.

Proof is by induction on the structure of formulae. All cases are trivial except for those of $\forall x A$ and $\exists x A$ where A is an extensional formula with one free variable x. For the case $\forall x A$, note that every ground instance of A is equivalent to one of the Extensional Six, so choose one ground instance from each equivalence class and let C be their conjunction. Obviously $\forall x A$ implies C, and C θ-implies every ground instance of A. By lemma 2, therefore, C θ-implies $\forall x A$, so $\forall x A$ and C are equivalent according to θ. The case $\exists x A$ is immediate from this and the negation case, by quantifier duality.

Theorem. The rule γ is admissible in $\mathbf{R}^\natural$ for formulae in the extensional vocabulary.

Proof: Suppose for contradiction that A and B are extensional formulae such that $A \vee B$ and $\neg A$ are theorems of $\mathbf{R}^\natural$ but B is not. By lemma 1, there is a prime, rich supertheory θ of $\mathbf{R}^\natural$ which also excludes B. Inside θ is its extensional fragment—the set of extensional formulae in θ—and by lemma 3, this is easily seen to have a model in a homomorphic image of DM6 (figure 3), so B also has a counter-model in DM6. DM6 may be embedded in a prime, consistent algebraic model (a De Morgan monoid [8]) modelling the whole of $\mathbf{R}^\natural$. This model of $\mathbf{R}^\natural$ is prime and satisfies $A \vee B$, so either it satisfies both A and $\neg A$ or else it satisfies B, contrary to the fact that it is consistent and the supposition that B fails in it.

6. Summary

Such is the state of research in relevant Peano arithmetic. When $\mathbf{R}^\sharp$ was first proposed in 1974, it was discovered almost immediately that the theory has unintended models including finite ones. At the time, these were certainly startling, but were they a blessing or a curse? The main blessing flowing from them is the finitary proof of reliability. While there can be no finitary proof of freedom from contradiction, it can be shown, by methods representable inside the system, that no derivations exist proving a term to have two different values. Note also that since every relevant proof is also a classical proof, the classical arithmetician can have this same guarantee as long as no irrelevant moves were made during a proof.

To pursue this last point a little further, if γ were to hold for $\mathbf{R}^\sharp$, this would show that the whole of classical Peano arithmetic could be derived from its axioms by means incapable of proving an incorrect

result for any calculation. Since this would immediately show classical arithmetic to be consistent, it follows that there is no finitary proof of admissibility for γ. In fact, as Meyer and Friedman [5] showed, there is no proof of γ at all, so the point is moot, but it is known [9] that $\mathbf{R}^{\sharp\sharp}$, the result of extending $\mathbf{R}^{\sharp}$ with the ω rule, is closed under γ and still has the finite models. The ω rule is unusable in general, but at least any classical proof using only inferences valid in $\mathbf{R}^{\sharp\sharp}$ enjoys the finitary proof of reliability.

Peano arithmetic does not stand alone, but is essentially a part of number theory, which includes reasoning about rational as well as natural numbers. On stepping up from $\mathbf{R}^{\sharp}$ to a theory encompassing rational arithmetic, we must decide how to treat the finite models. There is no unique way to do this—no one approach representing *the* relevant logical account of rational numbers. One idea is to keep $\mathbf{R}^{\sharp}$ itself as the whole theory, defining the operations on fractions and the equations between them as mere abbreviations for the equivalent expressions concerning natural numbers. On such an account, the finite models remain as they always were. Rational number theory is not expected to make sense on its own over such structures, and indeed it does not, with its non-transitive equality relation and failures of substitutivity, but it is still what it is as part of natural number arithmetic and it remains coherent in those terms when the definitions are unpacked. A different approach is to loosen the ties between natural and rational equations, axiomatising the latter so as to ensure transitivity and the like. Now the integers modulo n still provide inconsistent but non-trivial models, but in them there is no interesting rational arithmetic because there is only one rational number (though n different natural ones). The third option is to keep the equivalence between natural and rational equations, and to secure mathematical respectability for the whole theory by strengthening the underlying Peano postulates, taking us from $\mathbf{R}^{\sharp}$ to $\mathbf{R}^{\natural}$. On this account, the finite models are indeed a curse: they show that $\mathbf{R}^{\sharp}$ is too weak, so they are banished. The axiom saying that zero is not a successor does not succeed in a paraconsistent logic like $\mathbf{R}$, because nothing prevents zero from being a successor anyway. $\mathbf{R}^{\natural}$ imposes a penalty for violations of the axiom, and this restores mathematical respectability, removes the unwanted models and, as noted, captures the whole of classical number theory as intended.

Important open questions and future work include:

- Is the rule γ admissible in $\mathbf{R}^{\natural}$?
- How, if at all, can we make sense of rational number theory as a defined fragment of $\mathbf{R}^{\sharp}$?
- What is the best way to axiomatise rational arithmetic as a conservative extension of $\mathbf{R}^{\sharp}$ with a transitivity postulate for rational equality?
- Still weaker logics bring their own perspectives to quantitative inference. Without contraction, for instance, there are models in which numerical equality not even a congruence on the rational field. What else is there to discover by weakening the logical base still further?

References

[1] A. R. Anderson, N. D. Belnap, Entailmant: The Logic of Relevance and Necessity, Vol. 1, Princeton University Press, Princeton, 1975.

[2] E. Mares, Relevant Logic: A Philosophical Interpretation, Cambridge University Press, Cambridge, 2004.

[3] R. K. Meyer, Arithmetic formulated relevantly, Australasian Journal of Logic 18 (2021) 154–288.

[4] J. Slaney, R. K. Meyer, G. Restall, Linear arithmetic desecsed, Logique et Analyse 39 (1998) 379–388.

[5] H. Friedman, R. K. Meyer, Whither relevant arithmetic, Journal of Symbolic Logic 57 (1992) 824–831.

[6] J. Slaney, Relevant number theory with cancellation, Journal of the IGPL (forthcoming).

[7] R. K. Meyer, J. M. Dunn, H. Leblanc, Completeness of relevant quantification theories, Notre Dame Journal of Formal Logic 15 (1974) 97–121.

[8] T. Moraschini, J. Raftery, J. Wannenburg, Varieties of De Morgan monoids: Minimality and reducible algebras, Journal of Pure and Applied Algebra 223 (2019) 2780–2803.

[9] R. K. Meyer, $\supset$E is admissible in "true" Relevant arithmetic, Journal of Philosophical Logic 27 (1998) 327–351.

Implementing Intermediate Logics

Bastiaan Haaksema[1,2], Jens Otten[3,4] and Revantha Ramanayake[1,5]

[1]Bernoulli Institute, University of Groningen, The Netherlands
[2]Department of Information and Computing Sciences, Utrecht University, The Netherlands
[3]Department of Informatics, University of Oslo, Norway
[4]Potassco Solutions, Germany
[5]CogniGron, University of Groningen, The Netherlands

Abstract
We present automated theorem provers implementing systems for intermediate logics, in the propositional and first-order setting. They use an axiomatic embedding into intuitionistic logic based on cut-restricted sequent calculi. All provers are evaluated on a large benchmark set of propositional and first-order formulas.

Keywords
non-classical logics, intermediate logics, automated theorem provers, cut-restriction

1. Introduction

Intermediate logics lie between intuitionistic logic and classical logic in terms of subset inclusion, in the propositional or first-order setting. Intermediate logics offer a more nuanced approach to the usual 2-valued classical logic and this motivates their study in logic, computer science, and artificial intelligence. While many theorem provers exist for classical logic and several for intuitionistic logic, only very few are available for intermediate logics. Our aim is to address this gap by presenting provers for intermediate logic—propositional and first-order—in a systematic manner.

On the proof-theoretic side, it is well known that most propositional intermediate logics lack a cut-free sequent calculus. This is a formidable obstacle for automated theorem proving, and meta-theoretic investigations, since cut-freeness is the typical route towards a proof calculus with the subformula property, and the latter property is crucial for pruning in backward proof search. Indeed, recall the situation that arises with a Hilbert proof calculus where it is unclear when and on what formula the rule of *modus ponens* needs to be applied backwards. Ciabattoni et al. [1] present a general solution via cut-free hypersequent calculi for extensions of intuitionistic propositional logic (**IPL**) with axioms up to $\mathcal{P}_3'$ in the substructural hierarchy. Although the hypersequent calculus is a natural generalisation of the sequent calculus (use a multiset of sequents instead of a single sequent), from the perspective of automated theorem proving it is much more complex to implement, and the hypersequent calculus formalism is much less well-known outside the structural proof theory community.

A new solution is proposed by Ciabattoni et al. [2, 3]: sound and complete *sequent calculi* for propositional intermediate (and substructural) logics by permitting restricted cuts (as mentioned above, without a restriction on the cuts, the backward proof search space is simply too large). Specifically, the cut-formulas are restricted to instantiations of the axioms with conjunctions of subformulas of the end sequent. In the case of intermediate logics, the restricted cuts can be traded for a cut-free proof in the intuitionistic calculus with axiom instances added to the antecedent, making use of the deduction theorem in the latter. Consequently, the intermediate logics embed into intuitionistic logic.

Our focus is on the implementations of the theory described above, and also for first-order intermediate logics that are obtained through the addition of quantification rules. After describing the theoretical foundation in Section 2, we present our implementations in Section 3 and evaluate them in Section 4. We conclude with a summary, outlook and related research in Section 5.

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France
[†]The third author acknowledges the financial support of CogniGron, the Ubbo Emmius Funds, and FWF project P33548.
b.haaksema@student.rug.nl (B. Haaksema); jeotten@jens-otten.de (J. Otten); d.r.s.ramanayake@rug.nl (R. Ramanayake)
https://jens-otten.de/ (J. Otten); https://www.rug.nl/staff/d.r.s.ramanayake/ (R. Ramanayake)
0009-0001-0311-1553 (B. Haaksema); 0000-0002-4331-8698 (J. Otten); 0000-0002-7940-9065 (R. Ramanayake)

2. Intermediate Logics

2.1. Preliminaries

Propositional formulas are defined inductively from propositional atoms $p, q, \ldots$ and constants $\bot$ and $\top$ using the connectives $\wedge$, $\vee$, $\to$, and $\neg$. In the first-order setting, the propositional atoms are replaced by predicates on terms built from variables and function symbols, and the language is extended with first-order quantifiers. A term is a variable or $f(t_1, \ldots, t_n)$ for a n-ary function symbol f and terms $t_1, \ldots, t_n$. First-order formulas are defined inductively as the constants $\bot$ and $\top$, $P(t_1, \ldots, t_n)$ for an n-ary predicate symbol P and terms $t_1, \ldots, t_n$, and $A \wedge B$, $A \vee B$, $A \to B$, $\neg A$, $\forall x.A$ and $\exists x.A$ for formulas A and B. Free variables in a formula are those not in the scope of a quantifier. Also, $A(t/x)$ denotes the formula obtained by uniformly substituting the term t for all occurrences of the variable x that are free.

Throughout, we identify a logic with the set of its theorems. An axiomatic extension $L + \mathcal{A}$ is obtained by extending the base logic L with every instantiation of the atomic formulas of the axiom schema $\mathcal{A}$ by arbitrary formulas, and closing under the axioms and rules of the proof calculus.

The Hilbert proof calculus consists of axioms and rules that directly manipulate the logical formulas. There are many equivalent variants for **IPL** e.g., [4, Section 6]. A Hilbert calculus for first-order intuitionistic logic (**IL**) is obtained by adding the following to **IPL**. Here x is not a free variable in C.

$$\frac{C \to A(y/x)}{C \to \forall x.A} \qquad \frac{A(y/x) \to C}{\exists x.A \to C} \qquad \forall x.A \to A(t/x) \qquad A(t/x) \to \exists x.A$$

For economy of notation, we use **IPL** to denote the set of theorems of intuitionistic propositional logic and also its Hilbert proof calculus, and similarly for intuitionistic first-order logic **IL**.

A sequent calculus is a type of proof calculus that manipulates sequents of the form $A_1, \ldots, A_m \Rightarrow A_{m+1} \ldots, A_{m+n}$ where $A_1, \ldots, A_{m+n}$ are formulas. The intended interpretation of a sequent is the formula $A_1 \wedge \ldots \wedge A_m \to A_{m+1} \vee \ldots \vee A_{m+n}$. When required, $\top$ and $\bot$ serve as the identity element for conjunction and disjunction respectively. A sequent (calculus) is single-succedent if at most a single formula is permitted on the right-hand side of the sequent, i.e. $n \in \{0, 1\}$, else it is multi-succedent. Roughly speaking, the meta-level implication and conjunction/disjunction provided by $\Rightarrow$ and comma permit reasoning *inside* the formula (fixed depth deep inference), and this is what enables the definition of inference rules with nice properties that aid automated theorem proving and meta-theoretic investigations.

The theoretical foundations can be described using any reasonable cut-free sequent calculus for intuitionistic logic. For the sake of concreteness, we use the well-known sequent calculus **LJ** presented by Gentzen [5]. In contrast, the implementations employ variants of this system that are specifically optimized for proof searching.

2.2. Cut-restricted Sequent Calculi

In this work, we consider the propositional axiomatic extensions of **IPL** listed further below.

The soundness and completeness with respect to sequent calculi with restricted cuts was established in [3]. As the reader may be unfamiliar with cut-restriction, let us sketch briefly how the completeness result was obtained there: any theorem of the logic under consideration has a cut-free hypersequent proof [1] with proper hypersequent structural rules e.g., the communication rule (com) in Gödel logic which rearranges the contents of two components in the hypersequent. A hypersequent proof without proper structural rules is obtained by repeatedly eliminating bottom-most hypersequent structural rules. The latter is achieved by accepting an additional formula (slightly more than a subformula) in each active component. These additional formulas are eliminated at the bottom of the proof via a cut on an instance of a proper axiom of the logic. The cut-restricted sequent proof can now be read off the hypersequent proof since the latter contains no proper hypersequent structural rules.

1. Jankov logic: **IPL** $+ \neg A \vee \neg\neg A$. The sequent calculus $\mathbf{LJ}^J$ extends **LJ** with the cut-rule restricted to instances of the axiom schema $\neg A \vee \neg\neg A$. Specifically, in a proof of $\Rightarrow F$, the atomic formula A in the axiom schema can be replaced by any conjunction of subformulas of F.
2. Gödel logic: **IPL** $+ (A \to B) \vee (B \to A)$. The sequent calculus $\mathbf{LJ}^G$ extends **LJ** with the cut-rule restricted to instances of the axiom schema $(A \to B) \vee (B \to A)$ so in a proof of $\Rightarrow F$, the atomic formulas A and B are replaced by any conjunction of subformulas of F.
   While the above axiom schema for Gödel logic is the standard one, we will actually use the equivalent axiomatisation **IPL** $+ (A \to B) \vee ((A \to B) \to A) + \neg A \vee \neg\neg A$. As observed in [3], this allows us to restrict cut-formulas to a much smaller set, namely instances of the axiom schema where the atomic formulas are replaced by propositional atoms from F.

First-order axiom schemas. In the first-order setting, the above axiom schemas are written as *universal sentences*. For example, $\forall \bar{x}(\neg A \vee \neg\neg A)$ and $\forall \bar{x}((A \to B) \vee (B \to A))$.

Let $\mathbf{FLJ}^J$ and $\mathbf{FLJ}^G$ denote first-order the sequent calculi obtained from $\mathbf{LJ}^J$ and $\mathbf{LJ}^G$ by adding the usual Gentzen first-order quantifiers. Here, the *eigenvariable* y must not occur in the conclusion of the (R$\forall$) and (L$\exists$) rules.

$$\frac{A(t/x), \Gamma \Rightarrow C}{\forall x A, \Gamma \Rightarrow C}\,(\mathrm{L}\forall) \qquad \frac{\Gamma \Rightarrow A(y/x)}{\Gamma \Rightarrow \forall x A}\,(\mathrm{R}\forall) \qquad \frac{A(y/x), \Gamma \Rightarrow C}{\exists x A, \Gamma \Rightarrow C}\,(\mathrm{L}\exists) \qquad \frac{\Gamma \Rightarrow A(t/x)}{\Gamma \Rightarrow \exists x A}\,(\mathrm{R}\exists)$$

We observe that $\mathbf{FLJ}^J$ and $\mathbf{FLJ}^G$ are sound and complete for the corresponding cut-free hypersequent calculi $\mathbf{HLJ}^J$ and $\mathbf{HLJ}^G$ by a straightforward extension of the argument in [3]. The latter hypersequent calculi consist of proper hypersequent structural rules added to the base calculus **HLJ** for **IL**. Soundness is straightforward. For completeness, we extend the transformation for propositional logics in [3] that was sketched above. In the case of $\mathbf{HLJ}^G$, an instance of (com) in the hypersequent proof is replaced with a formula of the form $\wedge\Gamma \to \wedge\Gamma'$ that is added to the antecedent of the of the active component. Here $\wedge\Gamma$ is the conjunction of all formulas in Γ. The remaining rules in the cut-free hypersequent proof are faithfully simulated in the sequent proof that is ultimately obtained. It remains to simulate the quantifier rules in $\mathbf{HLJ}^G$ with the quantifier rules in $\mathbf{LJ}^G$ and verify by inspection that the eigenvariable condition for the former implies it for the latter even in the presence of the added formulas.

It still remains to clarify the relationship between the first-order hypersequent calculus and the Hilbert calculus[1]. Consider the following Hilbert calculus rule with the condition that x is not free in C.

$$\frac{\forall x(C \vee A(y/x))}{C \vee \forall x A}\,(\mathrm{QSR})$$

Armed with this rule, for each rule in the hypersequent calculus, there is a derivation in the Hilbert calculus of the conclusion from the premises under the standard formula translation. It follows that $\mathbf{FLJ}^J$ is sound for **IL** $+ \forall \bar{x}(\neg A \vee \neg\neg A) +$ (QSR). Completeness is immediate since $\mathbf{HLJ}^J$ has cut-elimination. Similarly, $\mathbf{FLJ}^G$ is sound and complete for **IL** $+ \forall \bar{x}(A \to B \vee B \to A) +$ (QSR).

In certain cases, the (QSR) rule may be replaced by an axiom schema. For example, it is easy to see that **IL**$+\forall \bar{x}(A \to B \vee B \to A) +$(QSR) is equivalent to **IL**$+\forall \bar{x}(A \to B \vee B \to A) + \forall \bar{z}\forall x(C \vee A(y/x)) \to \forall \bar{z}(C \vee \forall \bar{x} A)$ where x is not free in C in the latter axiom.

2.3. Cut-restricted Sequent Calculi and Embeddings

We have already noted that the sequent calculi obtained in [3] restrict the cut formulas to certain axiom instances that depend on the formula F that is being proved. Specifically, the set of cut formulas that are required in a proof of $\Rightarrow F$ is defined by the function below, with the set $\mathcal{A}$ of axiom schemas as parameter, and with the help of an auxiliary function ψ.

$$F \;\mapsto\; \{A \mid A \in \psi(\mathcal{A}, F)\}$$

[1]The third author thanks Timo Lang for a helpful discussion on this topic.

Ciabattoni et al. [3] identify several candidates for ψ.

- The set-bounding function $\psi_s(\mathcal{A}, F)$ contains all instances of formulas in $\mathcal{A}$ whose atomic formulas are substituted by non-repeating conjunctions of subformulas of F.
- The formula-bounding function $\psi_f(\mathcal{A}, F)$ contains all instances of formulas in $\mathcal{A}$ whose atomic formulas have been substituted by subformulas of F.
- The variable-bounding function $\psi_v(\mathcal{A}, F)$ contains all instances of formulas in $\mathcal{A}$ whose atomic formulas have been substituted by atoms in F.

We have omitted the multiset-bounding function, as it is not relevant for intermediate logics. Observe that the set-bounding function has as image a set whose size is exponential in the size of F.

Notice that $\psi_v(\mathcal{A}, F) \subset \psi_f(\mathcal{A}, F) \subset \psi_s(\mathcal{A}, F)$ for any F containing two distinct subformulas. In certain cases including Jankov logic and Gödel logic, it is possible to identify an axiomatisation that supports the preferred variable-bounding function. As Ciabattoni et al. [3] observe: for axiomatisations that satisfy the $\{\wedge\}$-propagation property, the formula-bounding function can be used. Also, for axiomatisations that satisfy the $\{\wedge, \vee, \rightarrow\}$-propagation property, the variable-bounding function can be used.

The set of axiom instances (and hence cut formulas) that are required in a proof of F in Jankov logic and Gödel logic are explicitly described below. It is precisely this variable-bounding function that we implement in the embedding-preprocessing step of the provers.

Jankov logic	$\{\neg A \vee \neg\neg A \mid A \mapsto \text{atoms in } F\}$
Gödel logic	$\{(A \rightarrow B) \vee ((A \rightarrow B) \rightarrow A) \mid A, B \mapsto \text{atoms in } F\} \cup$ $\{\neg A \vee \neg\neg A \mid A \mapsto \text{atoms in } F\}$

From Restricted Cuts to Embedding into Intuitionistic Logic

Observe that a proof of $\Rightarrow F$ in the intuitionistic calculus with cuts from a finite set Ω (in the present setting, this represents the image of the bounding function) can be transformed to a proof of $\wedge\Omega \Rightarrow F$; simply replace each left premise $\Gamma \Rightarrow A$ of a cut instance ($A \in \Omega$) with the trivial proof in **LJ**, where (L$\wedge$)* denotes multiple applications of the left conjunction rule.

$$\dfrac{\dfrac{}{A, \Gamma \Rightarrow A}}{\wedge\Omega, \Gamma \Rightarrow A}\,(\mathrm{L}\wedge)^*$$

Now proceed downwards, applying the obvious weakenings and contractions as required; the end sequent is then transformed to $\wedge\Omega \Rightarrow F$. By cut-elimination in the intuitionistic calculus, we obtain a cut-free proof of the latter sequent that witnesses an embedding of the intermediate logic into intuitionistic logic. What this means is that the prover can now conduct backward proof search in the setting of a cut-free intuitionistic proof of $\wedge\Omega \Rightarrow F$. This is the embedding perspective [3] that our provers adopt.

We remark that the number of subformulas of F is bounded by the size $|F|$ of F, and the number of atomic formulas in each of the Gödel axioms listed above is ≤ 2, and the Jankov axiom has just a single atomic formula. Hence the set-bounding function yields an exponential embedding of Gödel logic into intuitionistic logic, and the variable-bounding function that we actually implement in the prover yields a linear and quadratic embedding of Jankov and Gödel logic, respectively.

3. Implementations

3.1. SuperJ Prover

SuperJ is an automated theorem prover written in Haskell that supports many intermediate propositional logics via an embedding-preprocessing step followed by intuitionistic proof search.

$$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow \Delta, A \to B}\ (\mathsf{R}\to) \qquad \frac{A, B, \Gamma \Rightarrow \Delta}{A, A \to B, \Gamma \Rightarrow \Delta}\ (\mathsf{mp}) \qquad \frac{A \to (B \to C), \Gamma \Rightarrow \Delta}{(A \wedge B) \to C, \Gamma \Rightarrow \Delta}\ (\mathsf{L}{\to}{\wedge})$$

$$\frac{A \to p, B \to p, p \to C, \Gamma \Rightarrow \Delta}{(A \vee B) \to C, \Gamma \Rightarrow \Delta}\ (\mathsf{L}{\to}{\vee}) \quad \text{fresh atomic } p$$

$$\frac{A, p \to C, B \to p, \Gamma \Rightarrow p \qquad C, \Gamma \Rightarrow \Delta}{(A \to B) \to C, \Gamma \Rightarrow \Delta}\ (\mathsf{L}{\to}{\to}) \quad \text{fresh atomic } p$$

Figure 1: Implication rules of the intuitionistic sequent calculus.

SuperJ automatically selects one of the three bounding functions mentioned earlier based on the propagation properties in the embedding step. Besides supporting arbitrary axiomatisations, the prover explicitly supports Classical Propositional Logic (**CPL**), **IPL**, Jankov logic and Gödel logic. For **IPL** the embedding step is omitted, and for **CPL** the prover immediately switches to classical proof search. The two intermediate logics are supported through the variable-bounded axiomatisations as noted in the previous section.

The intuitionistic proof search procedure of SuperJ is based on that of Avellone et al. [6]. It uses a contraction-free multi-succedent intuitionistic calculus equivalent to **LJ** and is decidable in $O(n \log n)$-Space [7]. This calculus can be seen as a refinement of Dyckhoff's **LJT*** [8], it includes rules for negation and uses fresh propositional variables in two of the left implication rules (as shown in Figure 1) to restrict the size of proofs. As in the original procedure by Avellone et al., SuperJ classifies formulas in the sequent into six groups according to their behavior with respect to branching and backtracking, which can be considered a naive form of focusing [9].

The most important optimization follows from the addition of boolean simplification and replacement rules [6, 10, 11]. The atomic replacement rules below are adjusted to sequent calculus notation, where $[A/B]$ denotes the uniform substitution of A for all occurrences of B. This implemented optimization works well for our intermediate logics, since even though the embedding step may introduce new formulas into the sequent, these are all composed of subformulas of the end sequent.

$$\frac{(\Gamma \Rightarrow \Delta)[\top/p]}{p, \Gamma \Rightarrow \Delta}\ (\text{L rep}) \qquad \frac{(\Gamma \Rightarrow \Delta)[\bot/p]}{\neg p, \Gamma \Rightarrow \Delta}\ (\text{L}\neg \text{ rep})$$

When possible, SuperJ reverts to classical proof search through the sequent calculi **LK*** [12] for classical logic. This happens whenever the succedent of the sequent is empty. Furthermore, the classical provability of the sequent is checked before attempting to apply a left non-invertible left rule, which allows pruning of the search space when false. Finally, for non-invertible right rules, backtracking can be avoided in the case of a singleton succedent.

3.2. ileanSeP-im and ileanTAP-im

The theorem prover ileanSeP-im is an axiomatic extension of the intuitionistic sequent prover ileanSeP for Jankov and Gödel first-order logics. ileanSeP is a compact Prolog implementation of the single-succedent intuitionistic sequent calculus. Similar to tableau calculi, it uses a bottom-up proof search, free variables and a dynamic skolemization to deal with quantifiers. Together with the occurs check of term unification, this ensures that the eigenvariable condition is respected.

The theorem prover ileanTAP-im is an axiomatic extension of the intuitionistic tableau prover ileanTAP for Jankov and Gödel first-order logics. ileanTAP [13] is a compact Prolog implementation of a prefixed tableau calculus, similar to Fitting [14]. It extends the classical calculus [15] by adding prefixes to capture the Kripke semantics of intuitionistic logic, uses free prefix variables [16] and extends skolemization to prefix constants. First, ileanTAP-im performs a classical proof search collecting prefixes of literals that close branches. If this search succeeds, a prefix unification is used to unify these prefixes.

Table 1
The (prefixed) non-clausal matrix for intuitionistic logic.

type	$F^{pol} : p$	$M(F^{pol} : p)$
atomic	$A^0 : p$	$\{\{A^0 : p\underline{a^*}\}\}$
α	$(G \wedge H)^1 : p$	$\{\{M(G^1 : p)\}\}, \{\{M(H^1 : p)\}\}$
	$(G \vee H)^0 : p$	$\{\{M(G^0 : p)\}\}, \{\{M(H^0 : p)\}\}$
	$(G \to H)^0 : p$	$\{\{M(G^1 : p\underline{a^*})\}\}, \{\{M(H^0 : p\underline{a^*})\}\}$
β	$(G \wedge H)^0 : p$	$\{\{M(G^0 : p), M(H^0 : p)\}\}$
	$(G \vee H)^1 : p$	$\{\{M(G^1 : p), M(H^1 : p)\}\}$
	$(G \to H)^1 : p$	$\{\{M(G^0 : p\underline{V^*}), M(H^1 : p\underline{V^*})\}\}$

type	$F^{pol} : p$	$M(F^{pol} : p)$
atomic	$A^1 : p$	$\{\{A^1 : p\underline{V^*}\}\}$
α	$(\neg G)^0 : p$	$M(G^1 : p\underline{a^*})$
	$(\neg G)^1 : p$	$M(G^0 : p\underline{V^*})$
γ	$(\forall x G)^1 : p$	$M(G[x\backslash x^*]^1 : p\underline{V^*})$
	$(\exists x G)^0 : p$	$M(G[x\backslash x^*]^0 : p)$
δ	$(\forall x G)^0 : p$	$M(G[x\backslash t^*]^0 : p\underline{a^*})$
	$(\exists x G)^1 : p$	$M(G[x\backslash t^*]^1 : p)$

3.3. nanoCoP-im

The automated theorem prover nanoCoP-im is an axiomatic extension of the intuitionisitic non-clausal connection prover nanoCoP-i for the Jankov and Gödel first-order logics. nanoCoP-i is a compact Prolog implementation of the non-clausal connection calculus for first-order *intuitionistic* logic (with equality) [17, 18] and an extension of the classical prover nanoCoP [19, 20]. It is based on a prefixed non-clausal connection calculus [17]. In contrast to sequent and tableau calculi, which are *connective-driven*, connection calculi use a *connection*-driven search strategy. A *connection* is a set $\{A_1{}^0, A_2{}^1\}$ of literals with the same predicate symbol but different polarities.

The *non-clausal connection calculus* works on *non-clausal* matrices, where a matrix M is a set of clauses and a clause C is a set of literals L and (sub)matrices. It represents a formula in negation normal form. A *prefix* is a string consisting of variables (V) and constants (a) and assigned to each literal.

For a formula F, polarity $pol \in \{0, 1\}$ and prefix p, the *intuitionistic non-clausal matrix* $M(F^{pol})$ of F^{pol} is defined inductively according to Table 1. x^* is a new term variable, t^* is the Skolem term $f^*(x_1, \ldots, x_n)$, V^* is a new prefix variable, a^* is the prefix constant of the form $f^*(x_1, \ldots, x_n)$, f^* is a new function symbol and $x_1, \ldots, x_n$ are all free term and prefix variables in the corresponding formula $F^{pol} : p$. The *intuitionistic non-clausal matrix* $M^i(F)$ of F is the matrix $M(F^0 : \varepsilon)$. A *term substitution* σ_T assigns terms to variables, a *prefix substitution* σ_P assigns strings to prefix variables (and is calculated by a *prefix unification*). For intuitionistic logic, a connection $\{A_1^0 : p_1, A_2^1 : p_2\}$ is *σ-complementary* iff $\sigma_T(A_1) = \sigma_T(A_2)$ and $\sigma_P(p_1) = \sigma_P(p_2)$ for a combined substitution $\sigma = (\sigma_T, \sigma_P)$.

The *non-clausal connection calculus* for intuitionistic logic [17] is given in Figure 2. An *intuitionistic connection proof* for F is a derivation of $\varepsilon, M^i(F), \varepsilon$. Compared to the formal *clausal* connection calculus [21, 22], the extension rule is generalized and a *decomposition rule* is added [23, 19].

First, nanoCoP-im performs a classical proof search, in which the prefixes of each connection are collected. If the search succeeds the prefixes of the literals in each connection are unified. Additional optimization techniques are regularity, lemmata, restricted backtracking and strategy scheduling [24, 18].

Axiom (A) $\dfrac{}{\{\}, M, \mathit{Path}}$ *Start (S)* $\dfrac{C_2, M, \{\}}{\varepsilon,\ M,\ \varepsilon}$ and C_2 is copy of $C_1 \in M$

Reduction (R) $\dfrac{C, M, \mathit{Path} \cup \{L_2 : \underline{p_2}\}}{C \cup \{L_1 : \underline{p_1}\}, M, \mathit{Path} \cup \{L_2 : \underline{p_2}\}}$ and $\{L_1 : \underline{p_1}, L_2 : \underline{p_2}\}$ is σ-complementary

Extension (E) $\dfrac{C_3, M[C_1 \backslash C_2], \mathit{Path} \cup \{L_1 : \underline{p_1}\} \quad C, M, \mathit{Path}}{C \cup \{L_1 : \underline{p_1}\}, M, \mathit{Path}}$ $C_3 := \beta\text{-}clause_{L_2}(C_2)$, C_2 is copy of C_1, C_1 is e-clause of M wrt. $\mathit{Path} \cup \{L_1 : \underline{p_1}\}$, C_2 contains $L_2 : \underline{p_2}$, $\{L_1 : \underline{p_1}, L_2 : \underline{p_2}\}$ is σ-complementary

Decomposition (D) $\dfrac{C \cup C_1, M, \mathit{Path}}{C \cup \{M_1\}, M, \mathit{Path}}$ and $C_1 \in M_1$

Figure 2: The non-clausal connection calculus for intuitionistic logic.

4. Experimental Evaluation

4.1. Benchmark Problems

At present, sets of formulas for testing automated theorem provers for intermediate logic are not available. As the syntax of intermediate logic is the same as classical and intuitionistic logic, we can use existing benchmark formulas of these logics. As intermediate logics are an extension of intuitionistic logic, we decided to use the ILTP problem library for intuitionistic logic [25].

Version 1.1.2 of the ILTP problem library contains 274 propositional and 2550 first-order formulas with status and difficulty rating information. The problems are in TPTP syntax and divided into 24 categories. While the propositional formulas belong mainly to the intuitionistic syntactic category (SYJ), the first-order formulas are taken from a wide range of domains, from general algebra (ALG), computing (COM), set (SET) and number (NUM) theory to software creation (SWC) and verification (SWV).

4.2. Propositional Logic

The SuperJ prover described in Section 3 was evaluated on all 274 propositional problems of the ILTP library v1.1.2. The test were conducted on a 3.6 GHz Ryzen system with 32 GB of RAM running Ubuntu 23.10 with kernel version 5.15. Running time was restricted to 60 seconds of CPU time per individual problem. Table 2 shows the number of problems solved within the time limit, and the required time in seconds, best results for each logic are marked bold.

Performance results of intuitRIL have also been collected for comparison. This intermediate propositional logic prover, due to Fiorentini and Ferrari [26], is the only automated theorem prover known to us that supports the same selection of intermediate propositional logics. It was obtained through modification of an existing SAT-based theorem prover for intuitionistic propositional logic. Their methods have similar theoretical foundations, though more akin to using restricted cuts, as opposed to our embedding approach.

intuitRIL outperforms (or matches) SuperJ in intuitionistic propositional logic, particularly so for domains SYJ206 and SYJ209. Except for SYJ208, SuperJ is only ever faster with a minimal constant factor, this might be due to there being a slightly smaller preprocessing overhead compared to intuitRIL that uses a clausification procedure.

For Jankov logic, intuitRIL solved the same number of problems within almost the same amount of time as for **IPL**. There is no real slowdown by the addition of the axiom instantiations for this prover. The SuperJ prover is more sensitive to the addition of the axiom instantiations, e.g., it finds a proof earlier for problems in SYJ209 but is slower in SYJ201-SYJ205, SYJ211 and SYJ212.

Table 2
Results on the propositional problems of the ILTP library

		SuperJ (**IPL**)		intuitRIL (**IPL**)		SuperJ (Jan)		intuitRIL (Jan)		SuperJ (Göd)		intuitRIL (Göd)	
Domain	Total	Solved	Time	Solved	Time	Solved	Time	Solved	Time	Solved	Time	Solved	Time
LCL	2	**2**	**0.023**	2	0.024	**2**	**0.023**	2	0.024	**2**	**0.023**	2	0.024
SYJ1	12	**12**	**0.138**	12	0.145	**12**	**0.138**	12	0.144	8	0.092	**12**	**0.144**
SYJ201	20	20	6.533	**20**	**2.765**	14	53.583	**20**	**2.754**	1	0.022	**20**	**2.754**
SYJ202	20	9	28.834	**10**	**14.282**	4	15.417	**10**	**14.322**	1	0.012	**10**	**14.352**
SYJ203	20	**20**	**0.232**	20	0.242	20	5.162	**20**	**0.243**	3	1.724	**20**	**0.244**
SYJ204	20	**20**	**0.234**	20	0.241	20	15.192	**20**	**0.244**	3	0.326	**20**	**0.242**
SYJ205	20	**20**	**0.232**	20	0.242	12	70.909	**20**	**0.248**	0	—	**20**	**0.242**
SYJ206	20	11	25.887	**20**	**0.240**	11	29.026	**20**	**0.239**	4	3.946	**20**	**0.243**
SYJ207	20	20	0.674	**20**	**0.613**	**20**	**0.231**	20	0.612	**20**	**0.496**	20	1.054
SYJ208	20	**20**	**0.856**	20	2.296	**20**	**0.771**	20	2.306	10	1.446	**17**	**165.925**
SYJ209	20	8	6.862	**20**	**0.243**	9	56.864	**20**	**0.240**	4	23.207	**20**	**0.264**
SYJ210	20	**20**	**0.230**	20	0.241	20	16.641	**20**	**0.240**	4	4.636	**20**	**0.313**
SYJ211	20	20	97.895	**20**	**0.241**	12	45.389	**20**	**0.241**	1	3.132	**20**	**97.985**
SYJ212	20	20	0.720	**20**	**0.242**	12	48.558	**20**	**0.241**	5	19.868	**20**	**0.302**
SYN	20	20	11.423	**20**	**0.240**	19	0.217	**20**	**0.239**	19	0.218	**20**	**0.252**

Finally, for Gödel logic, intuitRIL again managed to solve most of the problems within the time limit. However, compared to itself for other logics, it is slower for domains SYJ208 and SYJ211. The performance of SuperJ on the same logic is below that of intuitRIL, except for domains LCL and SYJ207. Perhaps this is not so surprising, considering the axiomatisation of Gödel logic contains (nested) implications, a known weak point for sequent based provers such as SuperJ.

4.3. First-Order Logic

The automated theorem provers for first-order intermediate logic described in Section 3 were evaluated on the problems of the ILTP library. All test were conducted on a 2.0 GHz Xeon server with 64 GB of RAM running Linux Mint with kernel version 3.10 and ECLiPSe Prolog 5.10. Table 3 shows the results of the evaluation on all 2550 first-order problems of the ILTP library v1.1.2 [25] for a CPU time limit of 10 seconds. Included are the provers ileanSeP-im 1.0, ileanTAP-im 1.17 and nanoCoP-im 2.0. Each of these implementations were tested for intuitionistic logic ("**IL**"), Jankov logic ("Jan") and Gödel logic ("Göd"). Furthermore, the prover leanCoP 2.2 for classical first-order logic was included, which provides an upper limit for the number of problems that can be proved by one of the provers for intermediate logic.

Table 3
Results on the first-order problems of the ILTP library

	— ileanSeP-im —			— ileanTAP-im —			— nanoCoP-im —			leanCoP 2.2
Logic	**IL**	Jan	Göd	**IL**	Jan	Göd	**IL**	Jan	Göd	Classical
proved	298	238	222	310	196	163	788	750	604	1064
0 to 1sec.	266	214	184	305	190	161	695	602	496	936
1 to 10sec.	32	24	38	5	6	2	93	148	108	128
refuted	4	0	0	4	0	0	88	60	33	28
error	35	53	7	54	323	234	3	3	3	0

nanoCoP-im proves the most problems for each of the non-classical logics. ileanSeP-im proves slightly less problems than ileanTAP-im for **IL**, but more problems than ileanTAP-im for Jankov and Gödel logic. nanoCoP-im also refutes the largest number of problems for all three logics.

All three provers prove less problems for Jankov logic than for **IL** and less problems for Gödel logic than for Jankov logic. This is explained by the overhead in the search space caused by the additional intermediate axioms that are added to the formulas. In general, the problems proved in Gödel logic are a subset of the problems proved in Jankov logic, which are again a subset of the problems proved in **IL**. Table 4 shows the few exceptions where a problem was proved in Jankov (or Gödel) logic but not in **IL**, or proved in Gödel logic but not in Jankov logic. The entries show the CPU time in seconds necessary to prove a problem or (in parentheses) to refute it. These problems are from the domains Set Theory (SET), Software Verification (SWV) and Syntactic (SYN). Given a larger CPU time limit, the problems in the SET and SWV domains can be proved in **IL** by nanoCoP-i or Slakje [18], i.e., they are valid in **IL**.

Table 4
Detailed results for selected problems of the ILTP library

	— ileanSeP-im —			— ileanTAP-im —			— nanoCoP-im —			leanCoP 2.2
Logic	**IL**	Jan	Göd	**IL**	Jan	Göd	**IL**	Jan	Göd	Classical
SET095+4	–	–	–	–	–	–	–	7.5	–	0.6
SET638+3	–	–	–	–	–	–	–	7.9	–	0.1
SWV181+1	–	–	–	–	–	–	–	9.7	–	0.3
SWV188+1	–	–	–	–	–	–	–	8.9	–	–
SWV189+1	–	–	–	–	–	–	–	9.3	–	2.1
SYN081+1	–	–	–	–	–	–	–	0.2	0.2	0.1
SYN416+1	(0.1)	–	0.1	(0.1)	–	0.1	(0.1)	(0.1)	0.1	0.1

5. Conclusion

We implemented four provers for Jankov and Gödel logic via proof search for intuitionistic logic and bounding functions: SuperJ for the propositional Jankov and Gödel logics and ileanSeP-im, ileanTAP-im and nanoCoP-im for the first-order Jankov and Gödel logics. The SuperJ implementation is available at https://github.com/bhaaksema/superintuition, the ileanSeP-im, ileanTAP-im and nanoCoP-im implementations are available at https://leancop.de/imed/. While bounding functions could be added to any existing intuitionistic theorem prover, SuperJ is a new prover aimed at facilitating quick experimentation with heuristics. To the best of our knowledge, ileanSeP-im, ileanTAP-im and nanoCoP-im are the first provers for *first-order* Jankov and *first-order* Gödel logic.

Our tests have shown that very few problems in the ILTP library which are valid in Jankov or Gödel logic are not valid in intuitionistic logic. Therefore, it seems advisable to build a collection of such problems for future benchmarking, possibly generated by (forward) proofs in the sequent calculus.

The methodology applies to other intermediate logics, e.g., Scott's and Kreisel-Putnam logic axiomatized over **IPL** respectively by $((\neg\neg A \to A) \to (A \vee \neg A)) \to (\neg\neg A \vee \neg A)$ and $(\neg A \to (B \vee C)) \to ((\neg A \to B) \vee (\neg A \to C))$. It would also be interesting to extend the methodology to substructural logics.

Fiorentini and Ferrari [26] give a propositional intermediate logic prover intuitRIL that modularly extends a SAT-based prover for **IPL**. Fiorino [27, 28] present duplication-free tableau calculi for Gödel logic and three other propositional intermediate logics, including Jankov logic. Kuznets and Lellmann [29] give semantically inspired constructions of nested sequent calculi for propositional intermediate logics including Gödel logic, and a prototype proof search implementation was also presented.

For classical logic, many state-of-the-art theorem provers use heuristics that select a subset of appropriate axioms in a preprocessing step before the actual proof search [30]. As our approach adds many axioms to the formula to be proven, integrating such heuristics will likely improve the performance of our provers. This will be part of future work, along with further optimization and evaluations of the provers.

References

[1] A. Ciabattoni, N. Galatos, K. Terui, From axioms to analytic rules in nonclassical logics, in: 2008 23rd Annual IEEE Symposium on Logic in Computer Science, 2008, pp. 229–240. doi:10.1109/LICS.2008.39.

[2] A. Ciabattoni, T. Lang, R. Ramanayake, Bounded sequent calculi for non-classical logics via hypersequents, in: TABLEAUX 2019, Springer, 2019, pp. 94–110. doi:10.1007/978-3-030-29026-9_6.

[3] A. Ciabattoni, T. Lang, R. Ramanayake, Bounded-analytic sequent calculi and embeddings for hypersequent logics, The Journal of Symbolic Logic 86 (2021) 635–668. doi:10.1017/jsl.2021.42.

[4] A. Chagrov, M. Zakharyaschev, Modal logic, volume 35 of *Oxford Logic Guides*, The Clarendon Press Oxford University Press, New York, 1997. Oxford Science Publications.

[5] G. Gentzen, Untersuchungen über das Logische Schließen, Mathematische Zeitschrift 39 (1935) 176–210, 405–431.

[6] A. Avellone, G. Fiorino, U. Moscato, Optimization techniques for propositional intuitionistic logic and their implementation, Theoretical Computer Science 409 (2008) 41–58. doi:10.1016/j.tcs.2008.08.013.

[7] J. Hudelmaier, An O(n log n)-Space Decision Procedure for Intuitionistic Propositional Logic, Journal of Logic and Computation 3 (1993) 63–75. doi:10.1093/logcom/3.1.63.

[8] R. Dyckhoff, Contraction-free sequent calculi for intuitionistic logic, The Journal of Symbolic Logic 57 (1992) 795–807. doi:10.2307/2275431.

[9] R. Dyckhoff, Intuitionistic decision procedures since Gentzen, in: Advances in Proof Theory, Springer International Publishing, Cham, 2016, pp. 245–267.

[10] M. Ferrari, C. Fiorentini, G. Fiorino, fcube: An efficient prover for intuitionistic propositional logic, in: Logic for Programming, Artificial Intelligence, and Reasoning, Springer, Berlin, Heidelberg, 2010, pp. 294–301.
[11] M. Ferrari, C. Fiorentini, G. Fiorino, Simplification rules for intuitionistic propositional tableaux, ACM Trans. Comput. Logic 13 (2012). doi:10.1145/2159531.2159536.
[12] H. Ono, Proof Theory and Algebra in Logic, Short Textbooks in Logic, 1st ed., Springer, Singapore, 2019. doi:10.1007/978-981-13-7997-0.
[13] J. Otten, ileanTAP: An intuitionistic theorem prover, in: D. Galmiche (Ed.), TABLEAUX 1997, volume 1227 of *LNAI*, Springer, Heidelberg, 1997, pp. 307–312. doi:10.1007/BFb0027422.
[14] M. Fitting, Proof Methods for Modal and Intuitionistic Logics, D. Reidel, Dordrecht, 1983.
[15] R. M. Smullyan, First-Order Logic, Springer, Berlin, Heidelberg, New York, 1968.
[16] L. A. Wallen, Automated Deduction in Nonclassical Logics, MIT Press, Cambridge, 1990.
[17] J. Otten, Non-clausal connection calculi for non-classical logics, in: R. Schmidt, C. Nalon (Eds.), TABLEAUX 2017, volume 10501 of *LNAI*, Springer, 2017, pp. 209–227. doi:10.1007/978-3-319-66902-1_13.
[18] J. Otten, The nanoCoP 2.0 connection provers for classical, intuitionistic and modal logics, in: A. Das, S. Negri (Eds.), TABLEAUX 2021, volume 12842 of *LNAI*, Springer, 2021, pp. 236–249. doi:10.1007/978-3-030-86059-2_14.
[19] J. Otten, nanoCoP: A non-clausal connection prover, in: N. Olivetti, A. Tiwari (Eds.), IJCAR 2016, volume 9706 of *LNAI*, Springer, 2016, pp. 302–312. doi:10.1007/978-3-319-40229-1_21.
[20] J. Otten, nanoCoP: Natural non-clausal theorem proving, in: C. Sierra (Ed.), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, Sister Conference Best Paper Track, IJCAI, 2017, pp. 4924–4928. doi:10.24963/ijcai.2017/695.
[21] J. Otten, W. Bibel, leanCoP: lean connection-based theorem proving, Journal of Symbolic Computation 36 (2003) 139–161. doi:10.1016/S0747-7171(03)00037-3.
[22] J. Otten, W. Bibel, Advances in connection-based automated theorem proving, in: M. Hinchey, J. P. Bowen, E.-R. Olderog (Eds.), Provably Correct Systems, NASA Monographs in Systems and Software Engineering, Springer, Cham, 2017, pp. 211–241. doi:10.1007/978-3-319-48628-4_9.
[23] J. Otten, A non-clausal connection calculus, in: K. Brünnler, G. Metcalfe (Eds.), TABLEAUX 2011, volume 6793 of *LNAI*, Springer, 2011, pp. 226–241. doi:10.1007/978-3-642-22119-4_18.
[24] J. Otten, Restricting backtracking in connection calculi, AI Commun. 23 (2010) 159–182. doi:10.3233/AIC-2010-0464.
[25] T. Raths, J. Otten, C. Kreitz, The ILTP problem library for intuitionistic logic, Journal of Automated Reasoning 38 (2007) 261–271. doi:10.1007/s10817-006-9060-z.
[26] C. Fiorentini, M. Ferrari, Sat-based proof search in intermediate propositional logics, in: Automated Reasoning, Springer International Publishing, Cham, 2022, pp. 57–74. doi:10.1007/978-3-031-10769-6_5.
[27] G. Fiorino, An O(n log n)-space decision procedure for the propositional dummett logic., Journal of Automated Reasoning 27 (2001) 297–311. doi:10.1023/a:1017515831550.
[28] G. Fiorino, Space-efficient Decision Procedures for Three Interpolable Propositional Intermediate Logics, Journal of Logic and Computation 12 (2002) 955–992. doi:10.1093/logcom/12.6.955.
[29] R. Kuznets, B. Lellmann, Interpolation for intermediate logics via injective nested sequents, Journal of Logic and Computation 31 (2021) 797–831. doi:10.1093/logcom/exab015.
[30] K. Hoder, A. Voronkov, Sine qua non for large theory reasoning, in: N. Bjørner, V. Sofronie-Stokkermans (Eds.), CADE-23, volume 6803 of *LNCS*, Springer, 2011, pp. 299–314. doi:10.1007/978-3-642-22438-6_23.

Automated Proof Search in Intuitionistic Sentential Logic.

Didier Galmiche[1], Brandon Hornbeck[1] and Daniel Méry[1]

[1]Université de Lorraine, CNRS, LORIA Vandoeuvre-lès-Nancy, F-54506, France

Abstract

In this paper we describe an automated theorem prover for the intuitionistic non-Fregean sentential calculus with Suszko's identity ISCI. We first review the basic concepts of the logic, recall the recently proposed Topological Beth semantics for ISCI and a corresponding sound and complete labelled calculus. From this calculus we investigate automated proof search for ISCI and present the theorem prover AutoPSI through its architecture, its proof search strategies and optimizations. We complete with tests and benchmarks that illustrate the impact of strategies.

Keywords

Automated Proof Search, Labelled Calculi, Non Fregean Logic, Intuitionistic Logic with Identity

1. Introduction

In this paper we consider the intuitionistic sentential calculus with identity (ISCI) which extends intuitionistic logic with Suszko's identity operator $\approx$ introduced in [1] for non-Fregean logics. In the non-Fregean approach identity and logical equivalence have distinct meanings: two sentences with the same truth value can have different denotations. For example, two logically equivalent formulas might have distinct sets of proofs. Suszko's identity has been studied as an extension of classical logic in [2]. The resulting logic is called SCI. The intuitionistic variant ISCI has been studied in [3] and we have recently proposed a new semantics, called Topological Beth semantics and a new sequent-style labelled calculus $\mathsf{L}_{\mathsf{ISCI}}$ for ISCI in [4]. From these results we study automated proof search in this logic and present the theorem prover AutoPSI through its architecture and its proof search strategies and optimizations. Tests and benchmarks complete this study.

2. Intuitionistic Sentential Calculus with Identity

In this section, we recall the basic notions of ISCI [2, 1]. ISCI extends propositional intuitionistic logic (IL) with axioms that formalize the non-truth functional nature of the identity connective $\approx$.

Definition 1. *Let* $\mathbf{P} = \{ \mathrm{p}, \mathrm{q}, \dots \}$ *be a countable set of propositional letters. The formulas of* ISCI*, the set of which is denoted* $\mathbf{F}$*, are given by the grammar:*

$$\mathrm{A} ::= \mathbf{P} \mid \bot \mid \mathrm{A} \wedge \mathrm{A} \mid \mathrm{A} \vee \mathrm{A} \mid \mathrm{A} \supset \mathrm{A} \mid \mathrm{A} \approx \mathrm{A}$$

Formulas of the form $\mathrm{A} \approx \mathrm{B}$ are called *equations*. We write $\mathbf{F}_{\approx}$ for the restriction of $\mathbf{F}$ to equations. Negation $\neg\mathrm{A}$ and truth $\top$ are respectively defined as $\mathrm{A} \supset \bot$ and $\bot \supset \bot$.

ISCI can be axiomatized by adding the four identity axioms described in Fig. 1 to any axiom schemata for IL [2]. We call "$\mathsf{H}_{\mathsf{ISCI}}$" the Hilbert proof system consisting of the four axioms for identity, the ten axioms for IL and the rule of modus ponens. We write $\mathrm{S} \vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{B}$ to mean that a formula B is derivable in $\mathsf{H}_{\mathsf{ISCI}}$ from a finite set $\mathrm{S} = \{ \mathrm{A}_1, \dots, \mathrm{A}_\mathrm{n} \}$ of assumptions. Whenever S is empty, B is called a *thesis* or a *theorem* of $\mathsf{H}_{\mathsf{ISCI}}$. Let us note that the deduction theorem holds for $\mathsf{H}_{\mathsf{ISCI}}$, i.e. $\mathrm{A}_1, \dots, \mathrm{A}_\mathrm{n} \vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{B}$ iff $\vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{A}_1 \wedge \dots \wedge \mathrm{A}_\mathrm{n} \supset \mathrm{B}$.

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

*Corresponding author.

†These authors contributed equally.

didier.galmiche@loria.fr (D. Galmiche); brandon.hornbeck@loria.fr (B. Hornbeck); daniel.mery@loria.fr (D. Méry)

$(\approx_1)$ $\mathrm{A} \approx \mathrm{A}$

$(\approx_2)$ $(\mathrm{A} \approx \mathrm{B}) \supset (\neg\mathrm{A} \approx \neg\mathrm{B})$

$(\approx_3)$ $(\mathrm{A} \approx \mathrm{B}) \supset (\mathrm{B} \supset \mathrm{A})$

$(\approx_4)$ $(\mathrm{A} \approx \mathrm{B}) \wedge (\mathrm{C} \approx \mathrm{D}) \supset (\mathrm{A} \otimes \mathrm{C}) \approx (\mathrm{B} \otimes \mathrm{D})$ where $\otimes \in \{\wedge, \vee, \supset, \approx\}$

(IL_1)	$\mathrm{A} \supset (\mathrm{B} \supset \mathrm{A})$	(IL_2)	$(\mathrm{A} \supset \mathrm{B}) \supset ((\mathrm{A} \supset (\mathrm{B} \supset \mathrm{C})) \supset (\mathrm{A} \supset \mathrm{C}))$
(IL_3)	$\mathrm{A} \supset (\mathrm{B} \supset (\mathrm{A} \wedge \mathrm{B}))$	(IL_4)	$(\mathrm{A} \wedge \mathrm{B}) \supset \mathrm{A}$
(IL_5)	$(\mathrm{A} \wedge \mathrm{B}) \supset \mathrm{B}$	(IL_6)	$(\mathrm{A} \supset \mathrm{C}) \supset ((\mathrm{B} \supset \mathrm{C}) \supset ((\mathrm{A} \vee \mathrm{B}) \supset \mathrm{C}))$
(IL_7)	$\mathrm{A} \supset (\mathrm{A} \vee \mathrm{B})$	(IL_8)	$\mathrm{B} \supset (\mathrm{A} \vee \mathrm{B})$
(IL_9)	$(\mathrm{A} \supset \mathrm{B}) \supset ((\mathrm{A} \supset \neg\mathrm{B}) \supset \neg\mathrm{A})$	(IL_{10})	$\neg\mathrm{A} \supset (\mathrm{A} \supset \mathrm{B})$

(MP) From A and $\mathrm{A} \supset \mathrm{B}$ deduce B.

Figure 1: Axioms for ISCI.

Example 1. *The following is an* $\mathsf{H}_{\mathsf{ISCI}}$ *proof that* $\approx$ *is commutative, more precisely, we show* $\mathrm{A} \approx \mathrm{B} \vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{B} \approx \mathrm{A}$*:*

(1)	$\mathrm{A} \approx \mathrm{B}$	*assumption*
(2)	$\mathrm{B} \approx \mathrm{B}$	$\approx_1$
(3)	$((\mathrm{B} \approx \mathrm{B}) \wedge (\mathrm{A} \approx \mathrm{B})) \supset ((\mathrm{B} \approx \mathrm{A}) \approx (\mathrm{B} \approx \mathrm{B}))$	$\approx_4$
(4)	$(\mathrm{B} \approx \mathrm{B}) \supset ((\mathrm{A} \approx \mathrm{B}) \supset ((\mathrm{B} \approx \mathrm{B}) \wedge (\mathrm{A} \approx \mathrm{B})))$	IL_3
(5)	$(\mathrm{A} \approx \mathrm{B}) \supset ((\mathrm{B} \approx \mathrm{B}) \wedge (\mathrm{A} \approx \mathrm{B}))$	MP 2, 4
(6)	$(\mathrm{B} \approx \mathrm{B}) \wedge (\mathrm{A} \approx \mathrm{B})$	MP 1, 5
(7)	$(\mathrm{B} \approx \mathrm{A}) \approx (\mathrm{B} \approx \mathrm{B})$	MP 3, 6
(8)	$((\mathrm{B} \approx \mathrm{A}) \approx (\mathrm{B} \approx \mathrm{B})) \supset ((\mathrm{B} \approx \mathrm{B}) \supset (\mathrm{B} \approx \mathrm{A}))$	$\approx_3$
(9)	$(\mathrm{B} \approx \mathrm{B}) \supset (\mathrm{B} \approx \mathrm{A})$	MP 7, 8
(10)	$\mathrm{B} \approx \mathrm{A}$	MP 2, 9

□

3. Semantics for ISCI

In this work we use the Topological Beth (TB) semantics for ISCI and briefly review its main concepts and results introduced in [4].

Definition 2. *Let* $\mathbf{M}$ *be a set of elements, called* worlds*, such that* $\omega, \pi \in \mathbf{M}$ *and* $\omega \neq \pi$*. A* TB frame *is a bounded distributive lattice* $\mathcal{F} = (\mathbf{M}, \leqslant, \sqcup, \omega, \sqcap, \pi)$ *with* ω *and* π *as least and greatest elements respectively.*

Definition 3. *A* TB pre-model *is a triple* $\mathcal{M} = (\mathcal{F}, [\cdot], \Vdash)$*, where* $\mathcal{F}$ *is a* TB *frame, and* $[\cdot]$ *is a* valuation function *from* $\mathbf{M}$ *to* $\wp(\mathbf{P} \cup \mathbf{F}_\approx)$*, such that for all worlds* m *and* n*:*

$(\mathcal{M}_\pi)$ $[\pi] = \mathbf{P} \cup \mathbf{F}_\approx$*,*

$(\mathcal{M}_\mathrm{K})$ *if* $\mathrm{m} \leqslant \mathrm{n}$ *then* $[\mathrm{m}] \subseteq [\mathrm{n}]$*,*

$(\mathcal{M}_{\approx_1})$ $\mathrm{A} \approx \mathrm{A} \in [\mathrm{m}]$*,*

$(\mathcal{M}_{\approx_4})$ *for all* $\otimes \in \{\wedge, \vee, \supset, \approx\}$*, if* $\mathrm{A} \approx \mathrm{B}, \mathrm{C} \approx \mathrm{D} \in [\mathrm{m}]$ *then* $\mathrm{A} \otimes \mathrm{C} \approx \mathrm{B} \otimes \mathrm{D} \in [\mathrm{m}]$*.*

The forcing relation $\Vdash$ *is inductively defined as the smallest relation on* $\mathbf{M} \times \mathbf{F}$ *such that:*

- $\mathrm{m} \Vdash \mathrm{p}$ *iff* $\mathrm{p} \in [\mathrm{m}]$*,*
- $\mathrm{m} \Vdash \mathrm{A} \approx \mathrm{B}$ *iff* $\mathrm{A} \approx \mathrm{B} \in [\mathrm{m}]$*,*
- $\mathrm{m} \Vdash \bot$ *iff* $\pi \leqslant \mathrm{m}$*,*
- $\mathrm{m} \Vdash \mathrm{A} \wedge \mathrm{B}$ *iff* $\mathrm{m} \Vdash \mathrm{A}$ *and* $\mathrm{m} \Vdash \mathrm{B}$*,*

- $\mathrm{m} \Vdash \mathrm{A} \supset \mathrm{B}$ *iff for all* $\mathrm{n} \in \mathbf{M}$*, if* $\mathrm{n} \Vdash \mathrm{A}$ *then* $\mathrm{m} \sqcup \mathrm{n} \Vdash \mathrm{B}$*,*
- $\mathrm{m} \Vdash \mathrm{A} \vee \mathrm{B}$ *iff for some* $\mathrm{n}_1, \mathrm{n}_2 \in \mathbf{M}$ *such that* $\mathrm{n}_1 \sqcap \mathrm{n}_2 \leqslant \mathrm{m}$*,* $\mathrm{n}_1 \Vdash \mathrm{A}$ *and* $\mathrm{n}_2 \Vdash \mathrm{B}$*.*

A TB model *is a* TB *pre-model satisfying the admissibility condition:*

$(\mathcal{M}_{\approx_3})$ *if* $\mathrm{m} \Vdash \mathrm{A} \approx \mathrm{B}$ *then* $\mathrm{m} \Vdash \mathrm{B} \supset \mathrm{A}$*.*

Let us remark that $\mathcal{M}_\pi$ implies that all TB models have a world π that forces all formulas including $\bot$. As usual, a formula A is *true* (or *satisfied*) in a TB model $\mathcal{M}$, written $\mathcal{M} \vDash \mathrm{A}$, iff $\mathrm{m} \Vdash \mathrm{A}$ for all worlds m in $\mathcal{M}$ and *valid*, written $\vDash$A, iff it is true in all models. It is routine to show that $\mathcal{M}_\pi$ and $\mathcal{M}_\mathrm{K}$ extend from propositional letters and equations to all formulas. $\mathcal{M}_\mathrm{K}$ is the well-known Kripke monotonicity condition, which applies to equations in our setting.

Definition 4. *Let* $\mathcal{M} = (\mathcal{F}, [\cdot], \Vdash)$ *be a* TB *model.* $\mathcal{M}$ *is* regular *iff for all formulas* A*, if* $\mathrm{m} \Vdash \mathrm{A}$ *for some world* m*, then there exists a world* $\mathrm{m_A}$*, called* A-minimal*, such that* $\mathrm{m_A} \Vdash \mathrm{A}$ *and for all worlds* n*,* $\mathrm{n} \Vdash \mathrm{A}$ *implies* $\mathrm{m_A} \leqslant \mathrm{n}$*. We write* $\vDash_\mathrm{r}$ *for the restriction of validity to the class of* regular TB *models.*

Theorem 1 (Adequacy of regular Beth models). *Regular* TB *models are* $\mathsf{H}_{\mathsf{ISCI}}$*-sound and* $\mathsf{H}_{\mathsf{ISCI}}$*-complete: if* $\vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{A}$ *then* $\vDash_\mathrm{r} \mathrm{A}$*, and if* $\vDash_\mathrm{r} \mathrm{A}$ *then* $\vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{A}$*.*

4. Labelled Deduction for ISCI

Since our prover AutoPSI implements the $\mathsf{L}_{\mathsf{ISCI}}$ labelled calculus defined in [4] (more precisely the $\mathsf{L}^2_{\mathsf{ISCI}}$ variant of the calculus), let us briefly recall its basic concepts.

The set $\mathbf{L}$ of *labels* is the union of the set $\mathbb{N}$ with all of its finite subsets. We use the (possibly subscripted or primed) letters $\mathrm{a}, \mathrm{b}, \mathrm{c}$ to denote singletons and save the letters $\mathrm{x}, \mathrm{y}, \mathrm{z}$ to denote arbitrary labels. A label x is a *sublabel* of a label y if $\mathrm{x} \subseteq \mathrm{y}$.

We work with a labelling algebra $\mathcal{L}$ defined as the lattice $(\mathbf{L}, \subseteq, \cup, \emptyset, \cap, \mathbb{N})$, where join $\cup$ and meet $\cap$ are standard set union and intersection. We consider that $\cup$ binds stronger than $\cap$ and we shall frequently write xy instead of $\mathrm{x} \cup \mathrm{y}$ ($\mathrm{xx'} \cap \mathrm{yy'}$ should therefore be read as $(\mathrm{x} \cup \mathrm{x'}) \cap (\mathrm{y} \cup \mathrm{y'})$). In this paper, we shall only use examples with label letters built from the subset $\{ 1, \ldots, 9 \}$. Therefore, we shall use the more concise notation 13 to unambiguously refer to $\{ 1, 3 \}$ (and not to the singleton $\{ 13 \}$).

Definition 5. *A* labelled formula *is a pair* (C, x)*, written* $\mathrm{C} : \mathrm{x}$*, where* C *is a formula and* x *is a label. A* labelled sequent *is a pair* (Γ, Δ)*, written* $\Gamma \vdash \Delta$*, where* Γ, Δ *are* sets *of labelled formulas.*

Given a set Δ of labelled formulas, the notation $\mathrm{x} \sqsubseteq \Delta$ means that $\mathrm{x} \subseteq \mathrm{y}$ for some labelled formula $\mathrm{A} : \mathrm{y}$ occurring in Δ. We write $\lceil \Delta \rceil$ for the set of all labels that are maximal in Δ: $\lceil \Delta \rceil = \{ \mathrm{z} \sqsubseteq \Delta \mid$ if $\mathrm{u} \sqsubseteq \Delta$ and $\mathrm{z} \subseteq \mathrm{u}$ then $\mathrm{u} = \mathrm{z} \}$. Let $\mathrm{s} = \Gamma \vdash \Delta$ be a labelled sequent. A label x is *right maximal in* s if $\mathrm{x} \in \lceil \Delta \rceil$ and s is *right connected* iff $(\forall\, \mathrm{A} : \mathrm{x} \in \Gamma)(\mathrm{x} \sqsubseteq \Delta)$.

The labelled calculus $\mathsf{L}_{\mathsf{ISCI}}$ deals with labelled sequents $\Gamma \vdash \Delta$ where Δ is not allowed to be empty. Some of the rules ($\bot_\mathrm{L}$, $\vee_\mathrm{L}$, $\approx_{\mathrm{LL}}$, $\approx_{\mathrm{LR}}$) have two principal formulas. In this case, one is called *primarily principal* and the other *secondarily principal.* In $\approx_{\mathrm{LL}}$ and $\approx_{\mathrm{LR}}$, the equation that provides the substitution is primarily principal while the formula in which the substitution occurs is secondarily principal. In $\bot_\mathrm{L}$ and $\vee_\mathrm{L}$, the primarily principal formula is the one in the antecedent of the conclusion.

The rule $\approx_{\mathrm{LL}}$ replaces (some, possibly all) occurrences of C in D with B and we write $\mathrm{D}^\mathrm{C}_\mathrm{B}$ as a shorthand for $\mathrm{D}[\mathrm{C}/\mathrm{B}]$. Similarly for $\approx_{\mathrm{LR}}$ with A instead of D. We call such substitutions *sentential substitutions.*

Definition 6. *A formula* A *is a* theorem of *(or* derivable in*)* $\mathsf{L}_{\mathsf{ISCI}}$*, written* $\vdash_{\mathsf{L}_{\mathsf{ISCI}}} \mathrm{A}$*, if* $\vdash \mathrm{A} : \emptyset$ *is derivable in* $\mathsf{L}_{\mathsf{ISCI}}$*.*

Identity Rules:

$$\frac{}{\Gamma, p : x \vdash \Delta, p : y}\,\mathrm{id}_p(x \subseteq y) \qquad \frac{}{\Gamma, A \approx B : x \vdash \Delta, A \approx B : y}\,\mathrm{id}_{\approx}(x \subseteq y)$$

Core Intuitionistic Rules:

$$\frac{\Gamma, B \supset C : x \vdash \Delta, B : z \quad \Gamma, B \supset C : x, C : xz \vdash \Delta}{\Gamma, B \supset C : x \vdash \Delta}\supset_L(xz \sqsubseteq \Delta)$$

$$\frac{\Gamma, A : a \vdash \Delta, B : ya}{\Gamma \vdash \Delta, A \supset B : y}\supset_R \quad \frac{\Gamma, B : x, C : x \vdash \Delta}{\Gamma, B \wedge C : x \vdash \Delta}\wedge_L \quad \frac{\Gamma \vdash \Delta, A : y \quad \Gamma \vdash \Delta, B : y}{\Gamma \vdash \Delta, A \wedge B : y}\wedge_R$$

Disjunction and Falsity Rules:

$$\frac{}{\Gamma, \bot : x \vdash \Delta, A : y}\bot_L(x \subseteq y) \quad \frac{\Gamma \vdash \Delta, A_1 : y, A_2 : y}{\Gamma \vdash \Delta, A_1 \vee A_2 : y}\vee_R$$

$$\frac{\Gamma, B \vee C : x, B : xa_1 \vdash \Delta, A : ya_1 \quad \Gamma, B \vee C : x, C : xa_2 \vdash \Delta, A : ya_2}{\Gamma, B \vee C : x \vdash \Delta, A : y}\vee_L(x \subseteq y)$$

Sentential Identity Rules:

$$\frac{\Gamma, B \approx C : x, C \supset B : x \vdash \Delta}{\Gamma, B \approx C : x \vdash \Delta}\approx_{L3}$$

$$\frac{\Gamma, B \approx C : x, D : x, D^C_B : x \vdash \Delta}{\Gamma, B \approx C : x, D : x \vdash \Delta}\approx_{LL} \qquad \frac{\Gamma, B_1 \approx B_2 : x, B_i \approx B_i : x \vdash \Delta}{\Gamma, B_1 \approx B_2 : x \vdash \Delta}\approx_{LL'}$$

$$\frac{\Gamma, B \approx C : x \vdash \Delta, A : y, A^C_B : y}{\Gamma, B \approx C : x \vdash \Delta, A : y}\approx_{LR}(x \subseteq y) \qquad \frac{}{\Gamma \vdash \Delta, A \approx A : y}\approx_R$$

Maximality Rule:

$$\frac{\Gamma, B \supset C : x \vdash \Delta, B : xz \quad \Gamma, B \supset C : x, C : xz \vdash \Delta}{\Gamma, B \supset C : x \vdash \Delta}\supset_L(xz \in \lceil \Delta \rceil)$$

Eigenvariable conditions: In $\supset_R$ and $\vee_L$, a, a_1, a_2 are fresh singletons and $a_1 \neq a_2$.

Figure 2: Rules for the $\mathsf{L}_{\mathsf{ISCI}}$ Labelled Sequent Calculus.

A very important feature of $\mathsf{L}_{\mathsf{ISCI}}$, stemming from the use of Topological Beth semantics instead of Kripke semantics, is that it remains sound if the eigenvariable conditions are dropped. Therefore, when a labelled formula $C : x$ requiring the introduction of fresh labels has to be expanded, one can reuse the labels that were generated during the first expansion of the formula $C : x$ (or of any formula of the form $C : y$). The use of $\mathsf{L}_{\mathsf{ISCI}}$ without the eigenvariable is called *liberalized* $\mathsf{L}_{\mathsf{ISCI}}$. Since AutoPSI is an implementation of liberalized $\mathsf{L}_{\mathsf{ISCI}}$, we only consider liberalized derivations in the remainder of the paper.

Theorem 2 (Liberalized soundness). *If* A *is provable in liberalized* $\mathsf{L}_{\mathsf{ISCI}}$ *then* $\vdash_{\mathsf{H}_{\mathsf{ISCI}}} A$.

Example 2. *Let us consider the following partial derivation for the non-valid formula* $((p \vee q) \supset p) \vee ((p \vee q) \supset q)$*, where the second instance of* $\supset_R$ *reuses the label* 1 *introduced by the first instance:*

$$\Pi\left\{\begin{array}{l}\dfrac{\dfrac{p \vee q : 1, p \vee q : 1 \vdash p : 1, q : 1}{p \vee q : 1 \vdash p : 1, (p \vee q) \supset q : \emptyset}\supset_R}{\vdash (p \vee q) \supset p : \emptyset, (p \vee q) \supset q : \emptyset}\supset_R}{\vdash ((p \vee q) \supset p) \vee ((p \vee q) \supset q) : \emptyset}\vee_R\end{array}\right.$$

With a standard Kripke rule for left disjunction that would simply propagate the labels, we would get a proof for a non-valid formula as follows:

$$\dfrac{\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{p}:1\vdash\mathrm{p}:1,\mathrm{q}:1}\,\mathrm{id_p}\quad\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:1\vdash\mathrm{p}:1,\mathrm{q}:1}\,\mathrm{id_p}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{p}\vee\mathrm{q}:1\vdash\mathrm{p}:1,\mathrm{q}:1}\,\vee^{\mathrm{K}}_{\mathrm{L}}$$

On the contrary, in $\mathsf{L}_{\mathsf{ISCI}}$*, applying the Beth rule for left disjunction we get*[1]*:*

$$\dfrac{\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{p}:12\vdash\mathrm{p}:12}\,\mathrm{id_p}\quad\dfrac{\overset{\mathrm{s}}{\text{- -}}\qquad\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13\vdash\mathrm{q}:13}\,\mathrm{id_p}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:12\quad\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13\vdash\mathrm{p}:13,\mathrm{q}:1}\,\vee_{\mathrm{L}}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{p}\vee\mathrm{q}:1\vdash\mathrm{p}:1,\mathrm{q}:1}\,\vee_{\mathrm{L}}$$

The left premiss of the topmost instance of $\vee_{\mathrm{L}}$ *gives rise to the sequent* s*, in which* $\mathrm{p}\vee\mathrm{q}:1$ *can be reexpanded using either* $\mathrm{p}:12$*, or* $\mathrm{q}:13$ *in the succedent. Both choices reintroduce* s *as a premiss of* $\vee_{\mathrm{L}}$ *making it impossible to reach an axiom as depicted below.*

$$\dfrac{\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:123,\mathrm{q}:12}\,\mathrm{id_p}\quad\dfrac{\overset{\mathrm{s}}{\text{- -}}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:12}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:12}\,\vee_{\mathrm{L}}$$

$$\dfrac{\dfrac{\overset{\mathrm{s}}{\text{- -}}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:12}\quad\dfrac{}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:123}\,\mathrm{id_p}}{\mathrm{p}\vee\mathrm{q}:1,\mathrm{q}:13,\mathrm{p}:12\vdash\mathrm{p}:13,\mathrm{q}:12}\,\vee_{\mathrm{L}}$$

□

Let us assume some fixed strict total Nœtherian order $\lhd$ on $\mathbf{F}$ such that:

- $\bot\lhd\mathrm{A}$ for all $\mathrm{A}\neq\bot$,
- if $|\mathrm{A}|<|\mathrm{B}|$ then $\mathrm{A}\lhd\mathrm{B}$, and
- if $\mathrm{A}\lhd\mathrm{B}$ then $\mathrm{C}[\mathrm{A}/\mathrm{B}]\lhd\mathrm{C}$.

where $|\mathrm{C}|$ denotes the size of a formula C defined as the number of its connectives.

Definition 7. $\mathsf{rL}_{\mathsf{ISCI}}$ *is* $\mathsf{L}_{\mathsf{ISCI}}$ *under the following restrictions:*

(R1) *Only equations can be secondarily principal for* $\approx_{\mathrm{LR}}$ *and* $\approx_{\mathrm{LL}}$*.*

(R2) *Equations of the form* $\mathrm{A}\approx\mathrm{A}$ *are never (primarily or secondarily) principal for* $\approx_{\mathrm{LR}}$ *and* $\approx_{\mathrm{LL}}$*.*

(R3) $\approx_{\mathrm{LR}}$ *only performs strictly decreasing substitutions, i.e. if the rule replaces* C *with* B *in* A *then* $\mathrm{A}^{\mathrm{C}}_{\mathrm{B}}\lhd\mathrm{A}$*.*

(R4) $\approx_{\mathrm{LR}}$ *and* $\approx_{\mathrm{LL}}$ *only perform uniform substitutions, i.e., they make all possible replacements in the secondarily principal formula.*

(R5) $\approx_{\mathrm{LR}}$ *and* $\approx_{\mathrm{LL}}$ *only perform substitutions that preserve the main connective of the secondarily principal formula.*

Restrictions R1, R2 and R3 are required for the cut-elimination proof developped in [4]. The other restrictions are not mandatory but they simplify the implementation of the AutoPSI theorem prover. Let us remark that R1 guarantees that sentential substitutions should only occur inside equations, while R5 guarantees that equations should remain equations after a sentential substitution.

Since all of the axioms and rules of $\mathsf{H}_{\mathsf{ISCI}}$ are derivable in $\mathsf{rL}_{\mathsf{ISCI}}$ and since cut can be eliminated from $\mathsf{rL}_{\mathsf{ISCI}}$, we have the following completeness result:

Theorem 3 (Completeness under sentential restrictions). *If* $\vdash_{\mathsf{H}_{\mathsf{ISCI}}}\mathrm{A}$ *then* A *is provable in* $\mathsf{rL}_{\mathsf{ISCI}}$*, i.e., in* $\mathsf{L}_{\mathsf{ISCI}}$ *with the restrictions of Definition 7.*

[1] Irrelevant formulas are omitted to keep the proof in the page width.

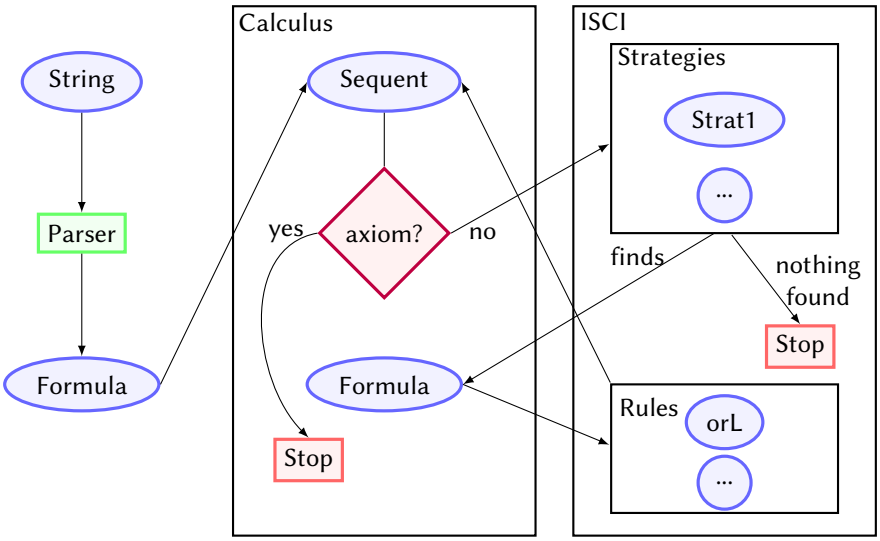


Figure 3: Execution diagram

The completeness result can be further extended to proofs in which rules with principal formulas occurring in the succedent (right principal) are only allowed to be applied if their right principal formulas have a right maximal label. Such a rule application strategy is called *right maximality* Finally, a last important property of $\mathsf{L}_{\mathsf{ISCI}}$ and $\mathsf{rL}_{\mathsf{ISCI}}$ is that the set of provable formulas remains the same if one replaces the original rule for right implication with the maximality rule depicted in Fig. 2.

Theorem 4 (Completeness under right maximality). *If* $\vdash_{\mathsf{H}_{\mathsf{ISCI}}} \mathrm{A}$ *then* A *is provable in* $\mathsf{L}_{\mathsf{ISCI}}$ *or in* $\mathsf{rL}_{\mathsf{ISCI}}$ *both restricted to the right maximality strategy.*

5. AutoPSI: a Prover for ISCI

In this section we present AutoPSI, an implementation of liberalized $\mathsf{rL}_{\mathsf{ISCI}}$ written in Java. The parsing of the formulas is generated by ANTLR from the grammar described in Section 2. The Java application consists in 31 classes, 4 of them being automatically generated by ANTLR.
AutoPSI is available at https://homepages.loria.fr/dmery/autopsi. Let us note that a screen capture of an AutoPSI session is given in Appendix A.

5.1. Prover Architecture

The global architecture of AutoPSI is depicted in the execution diagram in Fig. 3. The input of the prover is a string representing the formula to prove, using the following grammar :

- FORMULA :
  - (FORMULA)
  - FORMULA | FORMULA
  - FORMULA&FORMULA
  - FORMULA->FORMULA
  - FORMULA=FORMULA
  - ATOM
  - F (for $\bot$)
- ATOM : [a-z]*

Algorithm 1: Global Approach

Input: $s : string$
Data: $S : Sequent(setof formulas)$;
$F : Formula$;
$P : Prooftree$
Output: $V : boolean$

```
1 S ← ∅;
2 F ← Parse(s);
3 S ← S ∪ {F}; // Right-hand side of the sequent
4 P ← Compute(S, ∅); // Returns a proof tree
5 V ← IsAProof(P); // True iff all leaves are axioms
6 return V;
```

This string is parsed, creating an instance of the class Formula which is a syntactic tree of the formula. This formula is put into a sequent, which is a list of signed formulas (positive for the left-hand side, negative for the right-hand side). Then the majority of the computation is done by the class Calculus, which for every sequent starts by checking if it is an axiom, if it is an axiom then the computation stops, if not we need to apply a rule. The program then calls for a choice for the next formula in the sequent to decompose. This choice is made by a strategy that gives priorities to the available formulas. Once a formula is chosen, the program can check which rule to apply to the sequent depending on the formula. The application of this rule creates one or two new sequents in which the chosen formula is marked as used (except for the positive implications which can be used several times).

The Calculus class is designed to work with abstract classes for formulas, and interfaces for rules and strategies. This should allow us to reuse the core of the system for other sequent calculi.

Since AutoPSI is an implementation of $\mathsf{L}_{\mathsf{ISCI}}$, we need to deal with labelled formulas. Labels in $\mathsf{L}_{\mathsf{ISCI}}$ are sets of integers that are implemented in AutoPSI as instances of the HashSet class in Java. This allows us to easily perform the inclusion tests required as side conditions by some of the rules like $\supset_{\mathrm{L}}$, $\vee_{\mathrm{L}}$, the axiom rules and the sentential identity rules. From a technical point of view, inclusion is simply checked as inclusion between Java HashSet instances.

The mitigation of the eigenvariable condition is implemented in a singleton class that keeps track of all currently generated singleton labels. The singleton class allows singleton labels to be reused by considering them as minimal w.r.t. the formula they were firstly introduced with. Minimality is achieved by managing a dictionary whose keys are formulas and values are label letters. Whenever a formula requiring a fresh singleton label needs to be introduced in a sequent, for instance when applying the rule $\vee_{\mathrm{L}}$ to a labelled formula $\mathrm{A} \vee \mathrm{B} : \mathrm{x}$, we first check whether the dictionary already contains a key-value pair $\mathrm{A} : \mathrm{a_i}$ (meaning that $\mathrm{a_i}$ is A-minimal). If so, the singleton $\mathrm{a_i}$ is reused and $\mathrm{A} : \mathrm{a_i}$ is inserted in the antecedent of the current sequent. Otherwise, a fresh label letter $\mathrm{a_j}$ is generated by the singleton class and associated with A is the dictionary for later reuse before inserting $\mathrm{A} : \mathrm{a_j}$ in the current sequent.

5.2. Proof Search, Optimizations and Strategies

Given a formula as input, AutoPSI explores the proof search space using a depth-first search policy. During the exploration, the prover keeps track of all the rules that have been applied previously (as well as the formulas on which they were applied) in a tree structure called the rule-tree. To reduce the memory footprint, the sequents themselves are not explicitly stored in the rule-tree but can be recovered from the input formula by replaying all the rules up to a given point. If all the leaves of the rule-tree are axioms (zero-premiss rules) the input formula is deemed valid. In this case, the rule-tree can be converted to an actual proof tree (with all of its intermediate sequents as its nodes) if needed. Otherwise, if all possible rules have already been applied and all backtracking points have been exhausted, the input formula is deemed non-valid. The global approach and the detailed pseudo code of the proof

Algorithm 2: Compute

Input: $S : Sequent(Setoformulas)$;
$P : ProofTree$
Data: $S_1, S_2 : Sequent(setoformulas)$;
$F, FR : Formula$;
$P_1, P_2 : Prooftree$;
$Rules : Rule[]$;
$R : Rule$
Output: $P : Prooftree$

```
1  if S is not an axiom then
2      F ← Choose(S); // A formula is chosen for the derivation
3      Rules ← Rule(F);
4      foreach R in Rules do
5          if R needs a choice on the right then
6              i ← 0;
7              while P1 is null and i < Size(SR) do
                   // We check all possible choices until we find a proof
8                  FR ← SR(i); // of the right formulas of S
9                  < S1, S2 >← Apply(R, S, F, FR);
10                 P1 ← Compute(P, S1);
11                 if IsAProof(P1) then
12                     P2 ← Compute(P, S2);
13                     if IsAProof(P2) then
14                         P ← Compose(P1, P2, P); // We merge the proof trees
15                     end
16                     else
17                         P1, P2 ← null;
18                     end
19                 end
20                 else
21                     P1, P2 ← null;
22                 end
23                 i ← i + 1;
24             end
25             return P;
26         end
27         else
               // R does not need a choice
28             < S1, S2 >← Apply(R, S, F, FR);
29             P1 ← Compute(∅, S1);
30             P2 ← Compute(P, S2);
31             P ← Compose(P1, P2, P); // We merge the proof trees
32             return P;
33         end
34     end
35 end
36 else
       // S is an Axiom
37     return P;
38 end
```

search procedure are given in Algorithms 1 and 2. In Algorithm 2, the only data that we keep track of is the order of application of the rules in the proof, that we call a ProofTree. This allows the prover to only deal with one sequent at a time, while still having the information needed to build the proof if asked to. The while loop starting on line 7 states that we check every possible choice for the proof until we either find a proof, or we have checked every suitable formula for this rule application. For example, if the choices for the secondarily principal formula are between formulas A and B, then we start with formula A first. If the proof is completed, we stop there, if not, the ProofTree will be null, then the loop will go on and try with B. When the proof is completed, the use of the predicate Compose on lines 14 and 31 will merge the ProofTrees (the two trees created by the rule application, P1 and P2, and the former one that represents the proof below, P, into the new ProofTree P) and we will get an axiom at the top of the ProofTree. Then the call to the procedure IsAProof in line 5 of algorithm 1 will return true.

AutoPSI implements all of the restrictions described in Definition 7 and further restricts $\approx_{\mathrm{LL}}$ to sentential substitutions bounded by the size of the initial formula to prove which is called the *degree* of the proof. Let us remark that such a restriction is proven complete for SCI in [5]. In the case of ISCI the completeness result is only achieved, via counter-model construction, for a fragment where formulas are syntactically restricted to implications and identities only [6], but not for the full logic. A current limitation of AutoPSI is that it cannot generate a counter-model in case of non-validity.

5.2.1. Optimizations

Following the terminology of one-sided sequents, we assign a *polarity* to each formula occurring in a labelled sequent: *positive* if it occurs in the antecedent and *negative* if it occurs in the succedent.

A first optimization, called *subsumption*, takes advantage of Kripke monotonicity (condition $\mathcal{M}_{\mathrm{K}}$ of Definition 3) to subsume formulas on both sides of a sequent and thus prevent them from being expanded. More precisely, a positive formula $\mathrm{A} : \mathrm{y}$ is considered subsumed (and thus prevented from expansion) if the current sequent already contains a positive formula $\mathrm{A} : \mathrm{x}$ such that $\mathrm{x} \subseteq \mathrm{y}$. Similarly for negative formulas such that $\mathrm{y} \subseteq \mathrm{x}$. Since we create a finite number of formulas and labels, subsumption is not needed for termination. That is because the prover works with sets of formulas and if a formula is already in the set, it is not added to the sequent, but subsumption will limit the number of useless decompositions in the proof.

A second optimization is that we put an upper bound on the number of times positive implications can be reused to ensure that they do not get expanded infinitely often by $\supset_{\mathrm{L}}$. This upper bound is called "Time to Live" (TTL) and is implemented as a counter that is equal to the sum of the number of negative implications and twice the number of positive disjunctions occurring in a sequent as those rules are the only ones that might generate fresh singletons. $\mathsf{L}_{\mathsf{ISCI}}$ does not enjoy the subformula property (in its strict acception) since sentential substitutions might generate subformulas that did not occur in the initial formula to prove. Therefore, the TTL is updated dynamically during the proof search process when new singletons are generated.

Theorem 5. *The proof system* $\mathsf{L}_{\mathsf{ISCI}}$ *in which positive implications are not used more times than their TTL is complete.*

Proof. When a positive implication is decomposed, there is a list of candidates for the formula on the right-hand side, which are the formulas with bigger labels than the implication being decomposed. In this selection of candidates, there are maximal formulas, i.e. formulas that have labels that are not smaller than any other label on the right-hand side. The prover will try with one formula and if it does not find a proof, it will backtrack to try with another candidate. Then it will eventually use a maximal formula as a secondarily principal formula. Since the system that uses maximality is complete, this step does not prevent completeness. Then, if the implication is decomposed again in the proof, the prover will try a new candidate if and only if there is a new label, that means if a new label atom has been introduced since the last decomposition. Then, each positive implication can be decomposed at most a number of times that is equal to the number of labels created in the proof, that is the formula's TTL. □

5.2.2. Strategies

At each step of the exploration, the prover needs to choose the next formula to be expanded (called the principal formula). This task is devoted to an object called a strategy which implements the Strategy interface. Every time a rule is decomposed, the principal formula is marked as being used, with the exception of the implications on the left-hand side of the sequent that can be used several times. The rule application strategy used in AutoPSI is to delay choices as long as possible. There are two kinds of choices: the ones that involve the guessing of labels satisfying some side conditions as it is the case for the rule $\supset_{\mathrm{L}}$ and the ones that involve the choice of a secondarily principal formula as it is the case for the rules $\vee_{\mathrm{L}}$, $\approx_{\mathrm{LL}}$ and $\approx_{\mathrm{LR}}$. Making the right choices for secondarily principal formulas is the biggest performance issue that the prover currently faces. The number of suitable formulas can be large and when a formula is not valid we have to explore every possible choice before failing eventually.

Our first strategy is called "First One Strategy". In this strategy, the prover decomposes formulas in their order of apparition in the sequent, the older ones having priority. This strategy is not complete.

Our second strategy is called "New Labels First". It gives precedence to the rules that introduce new labels in the proof, such as $\supset_{\mathrm{R}}$ and $\vee_{\mathrm{L}}$, in order to have them ready for occurrences of positive implications and right-handed substitutions that need bigger labels in their right-principal formula. The rules that require choices and do not create labels, like $\supset_{\mathrm{L}}$ and all identity rules are chosen last, in order to delay choices higher in the proof. For the choice of secondarily principal formulas, the candidates are ordered by increasing size, with the hope that a small formula should be less likely to create a big number of branches and nodes. Therefore, in the case that such a choice should eventually fail, backtracking to the next one would not cost as much as for a bigger formula.

A third strategy is called "Partial Δ-Maximality". It uses the same priorities as the "New Labels First" strategy, but it only allows the rules $\supset_{\mathrm{L}}$, $\vee_{\mathrm{L}}$, $\approx_{\mathrm{LL}}$ and $\approx_{\mathrm{LR}}$ to be expanded with secondarily principal formulas having right-maximal labels. It is therefore an implementation of the *maximality strategy* discussed in Section 4. Choosing formulas with right maximal labels reduces the number of candidates. For other rules, the strategy can still use non left-maximal formulas.

More clever strategies for the choice of secondarily principal formulas shall be studied as future work. One improvement to make better choices, inspired by connection methods, would be to aim for candidates that actually contain atoms that could be complementary with the primarily principal formula. One difficulty w.r.t. connection methods is that $\mathsf{L}_{\mathsf{ISCI}}$ does not enjoy the subformula property, so that complementarity cannot be precomputed once and for all before starting the proof search process and should be done on the fly.

Example 3. *Let us consider the formula* $\neg\neg(\mathrm{A} \vee \neg\mathrm{A})$. *Negation is not a native rule in* $\mathsf{L}_{\mathsf{ISCI}}$*, so our starting formula is in fact* $((\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot) \supset \bot$*, which contains two negative implications and no positive disjunctions. Therefore,* AutoPSI *should be allowed to expand the same positive implication at most twice.*

After the creation of the labelled sequent $\vdash ((\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot) \supset \bot : \emptyset$*, only one formula can be selected by the rule application strategy: the one in the succedent. After the rule* $\supset_{\mathrm{R}}$ *is applied we get the sequent* $(\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1 \vdash \bot : 1$ *(for conciseness we do not keep expanded formulas). The only selectable formula is a positive implication. After* $\supset_{\mathrm{L}}$ *is applied we get the following premises.*

$$\frac{(\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1 \vdash \bot : 1, \mathrm{A} \vee (\mathrm{A} \supset \bot) : 1 \quad (\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1, \bot : 1 \vdash \bot : 1}{(\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1 \vdash \bot : 1} \supset_{\mathrm{L}}$$

The second premiss is the axiom $\bot_{\mathrm{L}}$ *so its exploration stops successfully. The first one still needs to be explored further. We have now the choice between two rules,* $\supset_{\mathrm{L}}$ *(still usable) and* $\vee_{\mathrm{R}}$*. The rule application strategy puts the priority on* $\vee_{\mathrm{R}}$*, resulting in the sequent* $(\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1 \vdash \bot : 1, \mathrm{A} : 1, \mathrm{A} \supset \bot : 1$*.*

Now have the choice between $\supset_{\mathrm{L}}$ *or* $\supset_{\mathrm{R}}$ *and the rule application strategy selects* $\supset_{\mathrm{R}}$*. The resulting sequent is* $(\mathrm{A} \vee (\mathrm{A} \supset \bot)) \supset \bot : 1, \mathrm{A} : 2 \vdash \bot : 1, \mathrm{A} : 1, \bot : 12$ *which is not an axiom yet but we can reuse the rule* $\supset_{\mathrm{L}}$ *to get two subproofs reaching axioms:*

$$\Pi_1 \left\{ \dfrac{\dfrac{}{(A \vee (A \supset \bot)) \supset \bot : 1, A : 2 \vdash \bot : 1, A : 1, \bot : 12, A \supset \bot : 12, A : 12}\,\mathrm{id}}{(A \vee (A \supset \bot)) \supset \bot : 1, A : 2 \vdash \bot : 1, A : 1, \bot : 12, A \vee (A \supset \bot) : 12}\,\vee_R \right.$$

$$\Pi_2 \left\{ \dfrac{}{(A \vee (A \supset \bot)) \supset \bot : 1, A : 2, \bot : 12 \vdash \bot : 1, A : 1, \bot : 12}\,\bot_L \right.$$

$$\dfrac{\Pi_1 \qquad\qquad \Pi_2}{(A \vee (A \supset \bot)) \supset \bot : 1, A : 2 \vdash \bot : 1, A : 1, \bot : 12}\,\supset_L$$

□

5.3. Tests and Benchmarks

Several indicators are built in AutoPSI to evaluate its efficiency. The first one is obviously the validity status (St) of the input formula (T for valid, F for invalid). The second one is the execution time measured in milliseconds (Tms). The third and fourth ones are the maximum depth and the size of the search space explored by the prover respectively measured as the maximum depth (D) and the number of nodes (N) in the rule-tree structure. We also measure the number of branches created (B). For valid formulas we also count the number of nodes in the final proof (Np) and its depth (Dp). In the case of a valid formula, the ratio between the size of the proof Np and the size of the search space N gives an account of the optimality of the proof search strategy: if N is significantly greater than Np then AutoPSI explored many wrong choices before making the right ones.

As AutoPSI is the first prover for ISCI, we do not have any other tool to compare it with. There is no substantial test base for ISCI, we thus derive our tests from the Φ formula introduced in [5], where a tableau prover for SCI (an extension of classical logic with the identity) is described:

$$\Phi \equiv (((q \approx p) \supset (p \supset r)) \approx ((p \supset (p \Leftrightarrow p)) \approx p)) \supset (((r \wedge p) \Leftrightarrow (p \approx p)) \vee ((p \wedge p) \vee \neg q))$$

We recall that negation $\neg A$ and logical equivalence $A \Leftrightarrow B$ have no specific rules in AutoPSI since they are only shorthands for $A \supset \bot$ and $(A \supset B) \wedge (B \supset A)$.

Φ is valid in SCI but it is not valid in ISCI, therefore we will try to prove $\neg\Phi$ and $\neg\neg\Phi$ as well, with the later expected to be valid thanks to the double negation. We will also study Ψ, which is Ψ but with identities instead of equivalences:

$$\Psi \equiv (((q \approx p) \supset (p \supset r)) \approx ((p \supset (p \approx p)) \approx p)) \supset (((r \wedge p) \approx (p \approx p)) \vee ((p \wedge p) \vee \neg q))$$

This transformation is not valid since two equivalent formulas are not necessarily identical. The column TTL stands for "Time To Live" and indicates the maximum number of expansions allowed for positive implications. Finally, we add χ:

$$\chi \equiv (((p \wedge (q \supset q_2)) \approx (p \wedge (q \supset q_2))) \wedge (((p \wedge (q \supset q_2)) \approx (r \vee p_2)) \approx (p \wedge (q \supset q_2)))) \supset ((p \wedge (q \supset q_2)) \approx (r \vee p_2))$$

Since ISCI is a conservative extension of IL, we complete our test base with three purely intuitionistic formulas:

$$\begin{aligned} F_1 &\equiv ((((p \Leftrightarrow q) \vee (p \Leftrightarrow r)) \vee (q \Leftrightarrow r)) \supset ((p \wedge q) \wedge r)) \supset ((p \wedge q) \wedge r) \\ F_2 &\equiv (\neg\neg(\neg p \supset q)) \supset ((\neg p \supset \neg q) \supset p) \\ F_3 &\equiv \neg\neg((((q \supset p) \supset (p \supset r)) \supset ((p \supset ((p \supset p) \wedge (p \supset p))) \supset p)) \\ &\quad \supset((((r \wedge p) \supset (p \supset p)) \wedge (p \supset p) \supset (r \wedge p))) \vee ((p \wedge p) \vee \neg q)) \end{aligned}$$

Table 2 summarizes the results of running AutoPSI on the ISCI test base in different conditions. Firstly without the subsumption optimization and without the maximality strategy, then with the subsumption optimization but without the maximality strategy, and finally with both the subsumption optimization and the maximality strategy.

It is clear that the subsumption optimization drastically improves the efficiency of the prover. Without it, most of the formulas in the test base cannot be decided in a reasonable amount of time. What the formula $\neg\neg\Phi$ shows us is that we explore too much, and that is mainly because of the rules $\supset_L$, $\vee_L$, $\approx_{LL}$ and $\approx_{LR}$, since we need to test all of their potential secondarily principal formulas. It would be a

Form	St	TTL	Conditions	T(ms)	D	N	B	DP	NP
Φ	False	10	(1)	-	-	-	-	-	-
			(2)	156.88	20	12185	6086	-	-
			(3)	52.00	20	30476	1734	-	-
$\neg\Phi$	False	9	(1)	10.57	11	1025	512	-	-
			(2)	2.29	11	219	109	-	-
			(3)	2.72	11	219	109	-	-
$\neg\neg\Phi$	-	-	(1)	-	-	-	-	-	-
			(2)	-	-	-	-	-	-
			(3)	-	-	-	-	-	-
$\neg\neg\neg\Phi$	False	10	(1)	-	-	-	-	-	-
			(2)	15461	24	597142	297429	-	-
			(3)	9409	24	266422	133208	-	-
Ψ	False	6	(1)	5.44	13	321	306	-	-
			(2)	3.9	13	539	265	-	-
			(3)	0.46	13	57	26	-	-
$\neg\Psi$	False	7	(1)	1.34	9	231	115	-	-
			(2)	0.58	9	83	41	-	-
			(3)	0.07	9	83	41	-	-
$\neg\neg\Psi$	False	7	(1)	-	-	-	-	-	-
			(2)	14071	27	749390	353560	-	-
			(3)	2412	27	84720	27	-	-
χ	True	8	(1)	0.12	6	9	1	6	7
			(2)	0.07	5	7	1	5	6
			(3)	0.60	5	6	1	5	6
$\neg\chi$	False	8	(1)	-	-	-	-	-	-
			(2)	12172	21	851894	483465	-	-
			(3)	17836	21	851517	425757	-	-
$\neg\neg\chi$	True	9	(1)	23.10	48	92	10	48	58
			(2)	43.3	15	1556	748	7	9
			(3)	4.84	15	144	69	7	9
F_1	True	8	(1)	-	-	-	-	-	-
			(2)	121.21	35	5697	1894	35	1374
			(3)	41.90	38	1512	338	38	1512
F_2	False	6	(1)	-	-	-	-	-	-
			(2)	22470	26	2689430	1211098	-	-
			(3)	514.00	25	52751	26369	-	-
F_3	True	12	(1)	-	-	-	-	-	-
			(2)	62831	80	790190	271515	80	246895
			(3)	18.3	30	698	207	30	698

D = max depth explored, N = Number of nodes explored, B = Branches created,
DP = Depth of the proof, NP = Nodes in the proof
(1) = Without subsumption and without maximality
(2) = With subsumption and without maximality
(3) = With subsumption and with maximality

Table 2
AutoPSI tested in several conditions.

great improvement to find an efficient heuristic for guessing the right secondarily principal formulas since exploring too many wrong choices might take too long for big formulas. One formula we can not solve right now is $\neg\neg\Phi$, which is not valid but the prover must explore too many branches to terminate in a reasonable amount of time.

Despite not being able to solve $\neg\neg\Phi$, the strategy improves the prover on most of the formulas. The only formula where we have a worst case in execution time is $\neg\chi$, because despite having a smaller search tree, the management of maximal formulas is time consuming and here it is not compensated

enough to improve the execution. This is the same reason why we improve the execution time of $\neg\neg\neg\Phi$ by only two seconds despite dividing the size of the search tree by two. All other formulas are improved by the maximality strategy. In particular, tests done on purely intuitionistic formulas show that maximality is an interesting optimization for our prover.

Since ISCI is a conservative extension of IL, AutoPSI is also an automated theorem prover for IL. However, let us remark that AutoPSI is currently not a competitive tool for IL, even when compared to our old STRIP prover [7]. The inefficiency comes from the use of Topological Beth semantics and its requirement to have rules with secondarily principal formulas. For IL, Kripke semantics is simpler and more efficient since one has the subformula property and various ways to tame the introduction of new labels. AutoPSI is firstly and mainly designed for ISCI, which does not enjoy the subformula property so that Topological Beth semantics is for the moment the easiest way to allow label reuse.

Conclusion and future work

The prover AutoPSI is the first automated prover for the logic ISCI. AutoPSI is based on the system $\mathsf{L}_{\mathsf{ISCI}}$, a labelled multi-conclusioned sequent calculus system, based on the Topological Beth semantics. The prover uses properties of both $\mathsf{L}_{\mathsf{ISCI}}$ and the Topological Beth semantics to enhance its performances. These properties are the regularity of the Topological Beth models, allowing the prover to have a finite search space, the maximality of the proof system, which enables the prover to terminate. Other properties are optimizations, like the restrictions on the identity rules allowed by our proof system, the subsumption of formulas that are not useful in the proof, and the maximality property of our proof system that allows to disregard some formulas that are not maximal in some steps of the proof. The last two optimizations are tested on a test base of several ISCI and IL formulas. The tests demonstrate the usefulness of such optimizations, as well as the difficulties of the prover on the restriction to IL. These difficulties are explained by the design of the proof system, built to deal with the identity operator.

Future works will focus on more optimizations for AutoPSI, mostly by implementing and testing other strategies. One strategy that could be really interesting is a strategy that uses the full maximality property, on every step of the proof and not just some steps. Such a strategy could drastically improve the prover performances.

References

[1] R. Suszko, Abolition of the Fregean axiom, in: Logic Colloquium, 1975, pp. 169–239. Springer.

[2] P. Lukowski, Intuitionistic sentential calculus with identity, Bulletin of the Section of Logic 19 (1990) 92–99.

[3] S. Chlebowski, D. Leszcyńska-Jasion, An Investigation into Intuitionistic Logic with Identity, Bulletin of the Section of Logic 48 (2019) 259–283.

[4] D. Galmiche, M. Gawek, D. Méry, Beth semantics and labelled deduction for intuitionistic sentential calculus with identity, in: 6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, LIPIcs 195, Buenos Aires, Argentina, 2021, p. 13:1–13:21.

[5] J. Golińska-Pilarek, T. Huuskonen, M. Zawidzki, Tableau-based decision procedure for non-fregean logic of sentential identity, in: 28th Int. Conference on Automated Deduction, CADE 2021, LNAI 12699, 2021, pp. 41–57.

[6] A. Tomczyk, D. Leszcyńska-Jasion, Decidability of Intuitionistic Sentential Logic with Identity via Sequent Calculus, in: 10th International Conference on Non-Classical Logics, Theory and Applications, NCL 2022, volume 358 of *EPTCS*, 2022, pp. 136–149.

[7] D. Larchey-Wendling, D. Méry, D. Galmiche, STRIP: Structural sharing for efficient proof-search, in: First International Joint Conference on Automated Reasoning, IJCAR 2001, LNCS 2083, Siena, Italy, 2001, pp. 696–700.

A. AutoPSI **Session Capture**

```
bhornbec@hapi:~/IdeaProjects/ISCIProver/out/artifacts/ISCIProver_jar$ java -jar ISCIProver.jar
subsumption
Subsumption off
strategy
Current strategy : Partial Delta Maximality
Choose new Strategy
1: Partial Delta Maximality
2: New Label First
3: First Formula
2
Strategy changed
verbose
verbose mode on
prove ((((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))))
(((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))) is false
ttl 6
Execution in 38.0ms
621 premises created
Max depth explored 13
Number of branches explored 306
------------------------------------------------------------
subsumption
Subsumption on
prove ((((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))))
(((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))) is false
ttl 6
Execution in 45.0ms
539 premises created
Max depth explored 13
Number of branches explored 265
------------------------------------------------------------
strategy
Current strategy : New Label First
Choose new Strategy
1: Partial Delta Maximality
2: New Label First
3: First Formula
1
Strategy changed
prove ((((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))))
(((q=p)->(p->r))=((p->(p=p))=p))->(((r&p)=(p=p))|((p&p)|(q->F))) is false
ttl 6
Execution in 7.0ms
57 premises created
Max depth explored 13
Number of branches explored 26
------------------------------------------------------------
```

Implementing the Fatio Protocol for Multi-Agent Argumentation in LogiKEy

Luca Pasetto[1,*], Christoph Benzmüller[2,3]

[1]University of Luxembourg
[2]Otto-Friedrich-Universität Bamberg
[3]Freie Universität Berlin

Abstract

Fatio is a dialogue protocol that has been proposed to extend the language for agent communications FIPA ACL with locutions for dialectical argumentation. Its syntax is composed of five agent locutions, and its axiomatic semantics is defined in terms of the mental states (beliefs and desires) of the participating agents, that are influenced by what is uttered. LogiKEy is a framework and methodology for the design and engineering of ethico-legal reasoners which is based on semantical embeddings of logics and logic combinations in expressive classical higher-order logic (HOL). In this paper, we explore a modelling and present a first implementation of the axiomatic semantics of Fatio in the Isabelle/HOL proof assistant system following the LogiKEy methodology.

Keywords

Proof assistants, Isabelle/HOL, Knowledge representation and reasoning, Automated theorem proving, Semantical embedding, Higher-order logic, Dialogue, Argumentation, Multi-Agent systems

1. Introduction

Recent advancements in dialogue systems based on language models are pushing the boundaries of what is possible in AI-human interaction. The rapid evolution of these technologies can set up the stage for more intuitive and helpful AI systems in everyday life, but at the same time presents a variety of risks. This motivates research on formal dialogue systems, based on logical languages, for which it is possible to formally (and eventually also mechanically) prove relevant properties. In this paper, we explore an *Isabelle/HOL* modelling of *Fatio*, a formal dialogue system based on argumentation between multiple agents by following the *LogiKEy* [1] framework and methodology. For an overview of argumentation-based dialogue, the interested reader can refer to [2].

LogiKEy [1] is a framework and methodology that can be used for the design, engineering and experimentation of reasoning systems, with a focus on ethico-legal applications [3] and normative reasoning [4]. The formal framework of LogiKEy is based on *shallow semantical embeddings* (SSE) [5] of combinations of various logics in classical higher-order logic (HOL). This meta-logical approach allows the use of off-the-shelf theorem provers and model finders for HOL, and therefore aids designers of ethical intelligent agents by allowing them to employ existing technologies instead of having to face the cumbersome task of developing completely new tools. Additionally, continuous improvements to the theorem provers enhance the reasoning capabilities within LogiKEy without needing extra adjustments. The methodology is represented in Figure 1: the relevant object logics are encoded in HOL and can in turn be used to express domain theories, with the purpose of successively experimenting with concrete applications and examples.

The rest of the paper is structured as follows: Section 2 describes the Fatio protocol as in [6], Section 3 shows its LogiKEy implementation in the Isabelle/HOL proof assistant system, and Section 4 discusses the relevant results and the remaining challenges.

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France
*Corresponding author.
luca.pasetto@uni.lu (L. Pasetto); christoph.benzmueller@uni-bamberg.de (C. Benzmüller)
0000-0003-1036-1718 (L. Pasetto); 0000-0002-3392-3093 (C. Benzmüller)

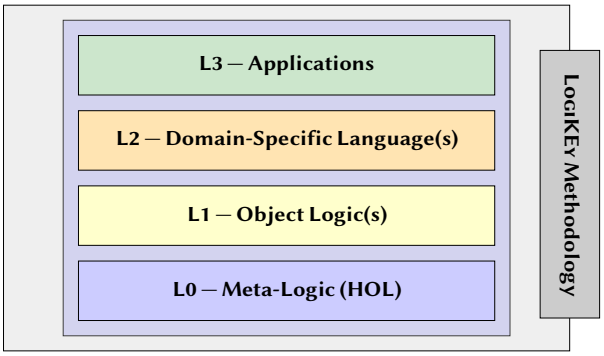


Figure 1: LogiKEy development methodology

2. The Fatio Protocol

This section summarizes the Fatio protocol for dialectical argumentation between agents that has been introduced in [6]. The original paper motivates the protocol as an extension of the language FIPA ACL [7] for agent interactions with the goal of addressing the criticism that it lacks a proper theory of argumentation. The authors therefore provide a set of locutions for the requesting and providing of reasons. The intuition is that agents that are proponents or opponents of a claim aim to establish the truth through various locutions or dialectical moves. The protocol is defined with a syntax, an axiomatic semantics, and an operational semantics.

2.1. Syntax

The general syntax for utterances is $\texttt{illocution}(P_i, \varphi)$ or $\texttt{illocution}(P_i, P_j, \varphi)$, where $\texttt{illocution}$ is an illocutionary act, P_i is an identifier for the agent making the utterance (the speaker), P_j (with $P_j \neq P_i$) denotes an agent at whom the utterance is directed, and φ is the content of the utterance. Utterances are intended to be made to the entire group involved in the dialogue. For the content of the utterance, any agreed formal language can be used, but the authors assume that the content layer is represented in a propositional language, with lower-case Greek letters as propositions. The set of well-formed content formulae, closed under the usual connectives, is denoted as $\mathcal{C}$. Since the protocol is used to exchange justifications for claims, some utterances also contain content comprising arguments (e.g., premises and inference-rules), which are represented by upper-case Greek letters. The set of well-formed argument formulae, closed under the usual connectives, is denoted as $\mathcal{A}$, with $\mathcal{C} \subset \mathcal{A}$. If $\varphi \in \mathcal{C}$ is a proposition in the content language and $\Phi \in \mathcal{A}$ is a justification, $\Phi \vdash^{+} \varphi$ is written to indicate that Φ is an argument in support of φ, and $\Phi \vdash^{-} \varphi$ to indicate that Φ is an argument against φ. In order to keep track of commitments to provide justifications for utterances, the concept of a dialectical obligation is introduced and is used to refer to previous commitments in the dialogue. The syntax is made up of five locutions or dialectical moves [6]:

- **F1 – `assert`(P_i, φ)**: A speaker P_i asserts a statement φ. By doing so, P_i creates a dialectical obligation to provide justification for φ, if subsequently required by another participant.
- **F2 – `question`(P_j, P_i, φ)**: A speaker P_j questions a prior utterance of $\texttt{assert}(P_i, \varphi)$ and seeks a justification for φ. The speaker P_j of the question creates no dialectical obligations.
- **F3 – `challenge`(P_j, P_i, φ)**: A speaker P_j challenges a prior utterance of $\texttt{assert}(P_i, \varphi)$ and seeks a justification for φ. Unlike a question, P_j also creates a dialectical obligation on himself to provide a justification for not asserting φ, such as an argument against φ, if questioned or challenged. Therefore, $\texttt{challenge}(P_j, P_i, \varphi)$ is considered stronger than $\texttt{question}(P_j, P_i, \varphi)$.
- **F4 – `justify`(P_i, $\Phi \vdash^{+} \varphi$)**: A speaker P_i, who had previously uttered $\texttt{assert}(P_i, \varphi)$ and was then questioned or challenged, is able to provide a justification Φ for the initial statement φ.

- **F5 – `retract`(P_i, φ)**: A speaker P_i, who had previously made an assertion `assert`(P_i, φ) or justification `justify`(P_i, $\Phi \vdash^+ \varphi$), can withdraw this statement with the utterance of `retract`(P_i, φ) or `retract`(P_i, $\Phi \vdash^+ \varphi$), respectively. This removes the earlier dialectical obligation on P_i to justify φ or Φ, if questioned or challenged.

The locutions are also subject to combination rules, introduced in [8], that constrain when an utterance can be made within the dialogue, ensuring structured interaction based on assertions, questions, challenges, justifications, and retractions. For instance, they ensure that the utterance `assert`(P_i, φ) may be made at any time, and that immediately following an utterance of `question`(P_j, P_i, φ) or `challenge`(P_j, P_i, φ), the speaker P_i of `assert`(P_i, φ) must reply with `justify`(P_i, $\Phi \vdash^+ \varphi$), for some $\Phi \in \mathcal{A}$.

2.2. Axiomatic Semantics

The axiomatic semantics gives a set of axioms in terms of pre-conditions and post-conditions for each locution of Fatio, and it involves beliefs and desires of the participating agents. Mental states such as beliefs and desires are used in order to be consistent with the axiomatic semantics of FIPA ACL. The concept of a dialectical obligation is also clarified at this point, as the authors introduce, for each participant P_i, one *dialectical obligations store* $DOS(P_i)$ to record dialectical obligations. The contents of $DOS(P_i)$ are triples (P_i, X, Y), where P_i is a participant, $X \in \mathcal{C}$ or $X \in \mathcal{A}$, and $Y \in \{+, -\}$. The intuition is that $(P_i, \varphi, +) \in DOS(P_i)$ indicates that participant P_i has a dialectical obligation to provide a justification in support of φ. Two classes of modal operators are used in the semantics: one for beliefs B_i and one for desires D_i, where i is an agent identifier. Also, an element from FIPA's action language is used: `Done` [*illocution*(P_i, φ), *pre-con*] indicates that *illocution*(P_i, φ) has been uttered by participant P_i with content φ, and with pre-conditions *pre-con* true. For the purposes of the current implementation, we do not consider *pre-con* and use `Done` [*illocution*(P_i, φ)]. The pre-conditions and post-conditions are then [6]:

- `assert`(P_i, φ)
  - **Pre-conditions:** A speaker P_i does not have yet the dialectical obligation to provide justification for φ, and P_i desires that each participant $P_j (j \neq i)$ believes that P_i believes the proposition $\varphi \in C$. Formally, $((P_i, \varphi, +) \notin DOS(P_i)) \wedge (\forall j \neq i)(D_i B_j B_i \varphi)$.
  - **Post-conditions:** Each participant $P_k (k \neq i)$, believes that participant P_i desires that each participant $P_j (j \neq i)$ believe that P_i believes φ. Moreover, the dialectical obligation $(P_i, \varphi, +)$ is added to $DOS(P_i)$.
    Formally, $(P_i, \varphi, +) \in DOS(P_i) \wedge (\forall k \neq i)(\forall j \neq i)(B_k D_i B_j B_i \varphi)$.
- `question`(P_j, P_i, φ)
  - **Pre-conditions:** Some participant $P_i (i \neq j)$ has a dialectical obligation to support φ and participant P_j desires that each other participant $P_k (k \neq j)$, believes that P_j desires that P_i utter a `justify` locution for φ. Formally, $\exists i (i \neq j)$ $((P_i, \varphi, +) \in DOS(P_i) \wedge (\forall k \neq j) D_j B_k D_j (\exists \Delta \in \mathcal{A})$ `Done` $[\texttt{justify}(P_i, \Delta \vdash^+ \varphi)])$.
  - **Post-conditions:** Participant P_i must utter a `justify` locution. There is no change on the dialectical obligations. Formally, $(\exists \Delta \in \mathcal{A})$ `Done` $[\texttt{justify}(P_i, \Delta \vdash^+ \varphi)]$.
- `challenge`(P_j, P_i, φ)
  - **Pre-conditions:** Some participant $P_i (i \neq j)$ has a dialectical obligation to support φ. Participant P_j desires that each other participant $P_k (k \neq j)$, believe both that P_j desires that P_i utter a `justify`($P_i, \Delta \vdash^+ \varphi$) locution for some $\Delta \in A$ and that P_j does not believe φ. Formally, $\exists i (i \neq j)$ $((P_i, \varphi, +) \in DOS(P_i) \wedge (\forall k \neq j)(D_j B_k \neg B_j \varphi)$
    $\wedge (\forall k \neq j) D_j B_k D_j (\exists \Delta \in \mathcal{A})$ `Done` $[\texttt{justify}(P_i, \Delta \vdash^+ \varphi)])$.
  - **Post-conditions:** Participant P_i must utter a `justify` locution. Moreover, the dialectical obligation $(P_j, \varphi, -)$ is added to $DOS(P_j)$. Formally, $((P_j, \varphi, -) \in DOS(P_j)$ $\wedge (\exists \Delta \in \mathcal{A})$ `Done` $[\texttt{justify}\,(P_i, \Delta \vdash^+ \varphi)])$.

- $\texttt{justify}(P_i, \Phi \vdash^{+} \varphi)$
  - **Pre-conditions:** A speaker P_i has a dialectical obligation to support φ. Another speaker $P_j (j \neq i)$ has uttered a $\texttt{question}(P_j, P_i, \varphi)$ or a $\texttt{challenge}(P_j, P_i, \varphi)$ locution. Furthermore, P_i desires that each participant $P_k (k \neq i)$ believes that P_i believes that $\Phi \in A$ is an argument for φ. Formally, $((P_i, \varphi, +) \in DOS(P_i)) \wedge (\texttt{Done}\left[\texttt{question}\left(P_j, P_i, \varphi\right)\right] \vee \texttt{Done}\left[\texttt{challenge}\left(P_j, P_i, \varphi\right)\right) \wedge (\exists \Phi \in \mathcal{A})\, (\forall k \neq i)\, D_i B_k B_i\, (\Phi \vdash^{+} \varphi)$.
  - **Post-conditions:** Each participant $P_k (k \neq i)$ believes that P_i desires that each participant $P_j (j \neq i)$ believes that P_i believes that $\Phi \in A$ is an argument for φ. Moreover, the dialectical obligation $(P_i, \Phi, +)$ is added to $DOS(P_i)$. Formally, $((P_i, \Phi, +) \in DOS(P_i)) \wedge (\forall k \neq i)\, (\forall j \neq i)\, B_k D_i B_j B_i\, (\Phi \vdash^{+} \varphi)$.
- $\texttt{retract}(P_i, \varphi)$
  - **Pre-conditions:** For the proposition $\varphi \in C$, it is required that $(P_i, \varphi, +) \in DOS(P_i)$. Participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer believes φ. Additionally, P_i desires that each participant $P_j (j \neq i)$ understands that P_i no longer maintains a disbelief in φ. Formally, $((P_i, \varphi, +) \in DOS(P_i) \wedge (\forall j \neq i)\, D_i B_j \neg B_i \varphi) \vee ((P_i, \varphi, -) \in DOS(P_i) \wedge (\forall j \neq i)\, D_i B_j \neg\neg B_i \varphi)$.
  - **Post-conditions:** Depending on the case outlined in the pre-conditions, either each participant $P_k (k \neq i)$ believes that participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer believes φ, or each participant $P_k (k \neq i)$ believes that participant P_i desires that each participant $P_j (j \neq i)$ believes that P_i no longer disbelieves φ. Moreover, either $(P_i, \varphi, +)$ or $(P_i, \varphi, -)$ is removed from $DOS(P_i)$. Formally, $((P_i, \varphi, +) \notin DOS(P_i) \wedge (\forall k \neq i)\, (\forall j \neq i)\, B_k D_i B_j \neg B_i \varphi) \vee ((P_i, \varphi, -) \notin DOS(P_i) \wedge (\forall k \neq i)\, (\forall j \neq i)\, B_k D_i B_j \neg\neg B_i \varphi)$.

2.3. Operational Semantics

The operational semantics is given in terms of internal decision mechanisms for agents that in the protocol are not specified and are left to be implemented, for example through reasoning methods based on formal argumentation. For instance, a decision mechanism is used by an agent to decide whether to assert a given statement at a given point of the dialogue, and another decision mechanism is used by an agent with a dialectical obligation to justify a statement or argument to identify the best possible justification at a given point of the dialogue. We do not go into more details of the operational semantics here, as for now we only focused on the encoding of the axiomatic semantics.

3. Fatio in LogiKEy

The possibility of using the existing LogiKEy infrastructure has allowed for rapid prototyping of the Fatio protocol in the Isabelle/HOL proof assistant system [9]. Figure 2 shows the layers of the LogiKEy methodology applied to Fatio.

The layers are as follows:

- **L0** is the underlying Meta-Logic (HOL) layer;
- **L1** represents the combination of object logics: (L1.1) a deeply embedded logic to be used to express the content of the Fatio locutions and (L1.2) a shallowly embedded logic to represent the axiomatic semantics in terms of beliefs and desires;
- **L2** is about domain-specific languages: (L2.1) is the encoding of world knowledge using L1.2 and (L2.2) is the actual language of Fatio, based on agents uttering locutions; and
- **L3** is the layer for dialogue applications, that use in turn the L2 languages.

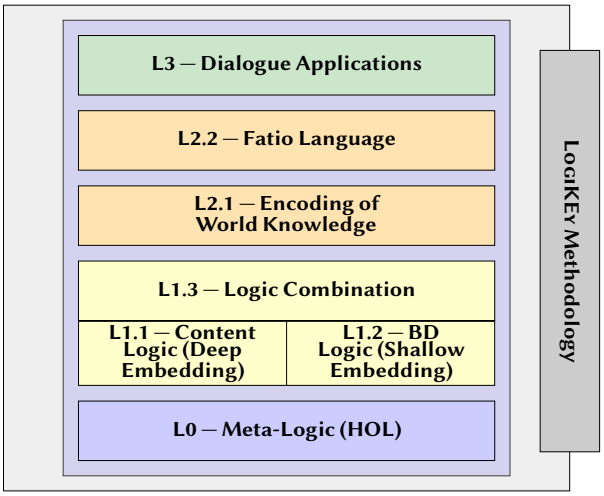


Figure 2: LogiKEy development methodology for Fatio

The architecture shows that at layer L1 we have both a shallow and a deep embedding of logics, but additionally the situation is even more complex, as the two logics are also nested within each other. We now go through the main parts of the implementation and describe choices and challenges.[1]

We begin with layer L1.2, showing the higher-order multimodal logic that we then use to represent beliefs and desires of agents. Here, σ is the type of world depended formulas, and complex formulas are built by following the semantics of the logical connectives.

```
typedecl i (*Type of possible worlds.*)
typedecl μ (*Type of individuals.*)
type_synonym σ="(i⇒bool)" (*Type of world depended formulas (truth sets).*)
type_synonym α="(i⇒i⇒bool)" (*Type of accessibility relations between worlds.*)
(*Lifted HOMML connectives: they operate on world depended formulas (truth sets).*)
definition mtop :: "σ" ("⊤") where "⊤ ≡ λw. True"
definition mbot :: "σ" ("⊥") where "⊥ ≡ λw. False"
definition mneg :: "σ⇒σ" ("¬_"[52]53)  where "¬φ ≡ λw. ¬φ(w)"
definition mand :: "σ⇒σ⇒σ" (infixr"∧"51) where "φ∧ψ ≡ λw. φ(w)∧ψ(w)"
definition mor  :: "σ⇒σ⇒σ" (infixr"∨"50) where "φ∨ψ ≡ λw. φ(w)∨ψ(w)"
definition mimp :: "σ⇒σ⇒σ" (infixr"→"49) where "φ→ψ ≡ λw. φ(w)⟶ψ(w)"
definition mequ :: "σ⇒σ⇒σ" (infixr"↔"48) where "φ↔ψ ≡ λw. φ(w)⟷ψ(w)"
definition mall :: "('a⇒σ)⇒σ" ("∀") where "∀Φ ≡ λw.∀x. Φ(x)(w)"
definition mallB:: "('a⇒σ)⇒σ" (binder"∀"[8]9) where "∀x. φ(x) ≡ ∀φ"
definition mexi :: "('a⇒σ)⇒σ" ("∃") where "∃Φ ≡ λw.∃x. Φ(x)(w)"
definition mexiB:: "('a⇒σ)⇒σ" (binder"∃"[8]9) where "∃x. φ(x) ≡ ∃φ"
definition mbox :: "α⇒σ⇒σ" ("□_ _") where "□ r φ ≡ (λw. ∀v. r w v ⟶ φ v)"
definition mdia :: "α⇒σ⇒σ" ("◇_ _") where "◇ r φ ≡ (λw. ∃v. r w v ∧ φ v)"
```

This logic is shallowly embedded, and after defining a datatype for speakers (for simplicity, at this moment we list only three possible speakers a, b and c) we define the two modal operators B_i and D_i for each speaker i.

```
datatype Speaker = a | b | c

consts BeliefRelationOf::"Speaker⇒α" ("@ᴮ_")
consts DesireRelationOf::"Speaker⇒α" ("@ᴰ_")

definition BeliefOfSpeaker::"Speaker⇒σ⇒σ" ("B_ _") where
  "Bₓ φ ≡ (□ (@ᴮ x) φ)"
definition DesireOfSpeaker::"Speaker⇒σ⇒σ" ("D_ _") where
  "Dₓ φ ≡ (□ (@ᴰ x) φ)"
```

[1]The code of the Isabelle/HOL implementation that we describe here is available at https://github.com/cbenzmueller/LogiKEy/tree/master/Fatio/v1

At layer L1.1 (on the left below) we have a deeply embedded propositional logic, with formulas of type `Formula`. This logic is then used inside the Fatio language of type `FatioL` to represent the content of the Fatio locutions: we show the syntax of the Fatio language at layer L2.2 (on the right below), with the five above mentioned kinds of locutions and content of type `Formula`.

```
datatype AtomExpression = p | q | r | n

datatype Formula =    —‹Formulas›
  Atom AtomExpression ("_ᵃ")
  | Bot ("⊥")
  | Neg Formula ("¬")
  | And Formula Formula (infixr "∧" 97)
  | Or Formula Formula (infixr "∨" 93)
  | Imp Formula Formula (infixr "⟶" 95)
```

```
datatype Sign = Plus ("+") | Minus ("-")

datatype FatioL =    —‹Fatio Language›
  Assert Speaker Formula ("assert[_,_]")
  | Question Speaker Speaker Formula ("question[_,_,_]")
  | Justify Speaker Formula Sign Formula ("justify[_,_⊢_,_]")
  | Challenge Speaker Speaker Formula ("challenge[_,_,_]")
  | Retract Speaker Formula Sign ("retract[_,_,_]")
```

In the axiomatic semantics, the beliefs and desires of the agents are about formulas that have been uttered during the dialogue using the content language, that is, the deeply embedded one. However, the axiomatic semantics is expressed in the shallowly embedded higher-order multimodal logic. Therefore, at layer L1.3 it is necessary to have a mapping between the deeply embedded content formulas (L1.1) and the shallowly embedded higher-order multimodal logic formulas (L1.2). This is built starting from the atoms of the languages with `mkHOMMLatom` (on the left below) and arriving to the complex formulas with the function `Map1` specified on the relative connectives (on the right below). We also map to the higher-order multimodal logic the special formulas that appear in the axiomatic semantics for `Done` $[\mathit{illocution}(P_i, \varphi)]$ (with `MapDone`) and for $\Phi \vdash^{+} \varphi$ in `justify`$(P_i, \Phi \vdash^{+} \varphi)$ (with `MapEntailment`).

```
(* mappings to shallow embedding *)
consts mkHOMMLatom::"AtomExpression⇒σ"
consts MapDone::"FatioL⇒σ"
consts MapEntailment::"Sign⇒Formula⇒Formula⇒σ"
```

```
(* mapping of Formula, from deep to shallow *)
fun Map1::"Formula⇒σ" ("{_}") where
  "Map1 ((Atom x)) = mkHOMMLatom (x)"
| "Map1 (Bot) = mbot"
| "Map1 (Neg φ) = (¬(Map1 φ))"
| "Map1 ((And φ ψ)) = ((Map1  (φ)) ∧ (Map1 (ψ)))"
| "Map1 ((Or φ ψ)) = ((Map1  (φ)) ∨ (Map1 (ψ)))"
| "Map1 ((Imp φ ψ)) = ((Map1  (φ)) ⟶ (Map1 (ψ)))"
```

We show then some experiments to clarify the differences between the deep and the shallow embedding. First, while for the deeply embedded logic of L1.1 we can find a counterexample to show that it does not hold that $p \rightarrow\ p = \neg\bot$, when this is mapped to the shallowly embedded logic of L1.2 we can prove that it holds that `Map1`$(p \rightarrow\ p) =$ `Map1`$(\neg\bot)$ (on the left below). Indeed, the deep embedding is motivated by the need to differentiate two same Fatio locutions that involve different but semantically equivalent formulas of the content language, while with the shallow embedding we cannot distinguish between formulas like $p \rightarrow\ p$ and $\neg\bot$. We can find a counterexample to show that it does not hold that `assert`$(a, p \rightarrow\ p) =$ `assert`$(a, \neg\bot)$, but also there is a counterexample to show that it does not hold that `Done` $[\mathit{assert}(a, p \rightarrow\ p)] =$ `Done` $[\mathit{assert}(a, \neg\bot)]$ (on the right below). This is the intended behaviour for `Done`, as in this case we might be in a dialogue where agent a has asserted $\neg\bot$ but not $p \rightarrow\ p$.

```
lemma "(Map1 ((Atom p) ⟶ (Atom p))) = (Map1 ( ¬⊥ ))"
  by (simp add: mbot_def mimp_def mneg_def)

lemma "((Atom p) ⟶ (Atom p)) = ( ¬⊥ )"
  nitpick
  oops

Nitpicking formula...
Nitpick found a counterexample for card i = 1:
```

```
lemma "(assert[a, (Atom p) ⟶ (Atom p)]) = (assert[a, ¬⊥])"
  nitpick
  oops

lemma "MapDone (assert[a, (Atom p) ⟶ (Atom p)]) = MapDone (assert[a, ¬⊥])"
  nitpick
  oops

Proof state
Nitpicking formula...
Nitpick found a counterexample for card i = 1:
```

We proceed now with the modelling of the axiomatic semantics of the protocol. The first ingredient is the *dialectical obligations store* DOS, that we represent as a list of triples (`Speaker`, `Formula`, `Sign`). The function to update DOS when a Fatio locution is executed is then shown below here on the right: `assert`, `justify`, and `challenge` add an element to DOS, `retract` removes an element from DOS, while `question` does not change DOS.

```
(* DOS Entry *)
datatype DOS = Entry Speaker Formula Sign
```

```
(* DOS update *)
fun
  DOSUpdate::"FatioL ⇒ DOS list ⇒ DOS list"
  where
    "(DOSUpdate assert[i,φ] dos) =
      ((Entry i φ +) # dos)"
  | "(DOSUpdate question[j,i,φ] dos) =
        dos"
  | "(DOSUpdate justify[i, ψ ⊢ˢ φ] dos) =
      ((Entry i ψ +) # dos)"
  | "(DOSUpdate challenge[j,i,φ] dos) =
      ((Entry j φ -) # dos)"
  | "(DOSUpdate retract[i,φ,S] dos) =
      ((remove1 (Entry i φ S) dos))"
```

For layer L1.3, the next step is to encode the pre-conditions of each locution. In order to do that, the formal definitions of the original paper should be clear and not leave space for ambiguity. If we consider the pre-conditions of `question` and `challenge`, we have that $\exists\Delta$ appears after the modal operators $D_jB_kD_j$ in $(\forall k \neq j)D_jB_kD_j\,(\exists\Delta \in \mathcal{A})\,\mathrm{Done}\,[\mathtt{justify}(P_i, \Delta \vdash^{+} \varphi)]$. It is not completely clear if the authors actually intended the modal operators to operate on a quantified formula, or if they intended the existential quantifier as a meta-language element. To better understand the situation, we experimented with the expression and tried to move the existential quantifier outside of the modal operators, obtaining $(\forall k \neq j)\,(\exists\Delta \in \mathcal{A})\,D_jB_kD_j\mathrm{Done}\,[\mathtt{justify}(P_i, \Delta \vdash^{+} \varphi)]$. Indeed, the two formulas are not equivalent, and we found out that while the second one implies the first one (`lemma ent3`), the inverse does not hold (counterexample for `lemma ent2`). For now, we decided to follow strictly the contents of the paper, so we kept the first formula for the pre-conditions, but this might be a knowledge representation choice that can be changed in favor of the formula that better encodes the desired effects.

```
lemma ent2:"(⊨BD DjBkDj (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))) ⟶
            (∃Δ. (⊨BD DjBkDj (MapDone justify[i, Δ ⊢⁺ φ])))"
  unfolding FatioDefs
  unfolding Defs
  nitpick
  oops

lemma ent3:"(∃Δ. (⊨BD DjBkDj (MapDone justify[i, Δ ⊢⁺ φ])) ⟶
            (⊨BD DjBkDj (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ]))))"
  unfolding FatioDefs
  unfolding Defs
  by simp
```

We can then proceed with the encoding of the pre-conditions. We write function `PreCond` that checks whether the pre-conditions of a locution are satisfied, given a current *dialectical obligations store* DOS and a current set Γ of shallowly embedded formulas that represent the encoded world knowledge of layer L2.1.

```
fun PreCond::"FatioL ⇒ (DOS list) ⇒ (σ ⇒ bool) ⇒ bool"
  where
    "(PreCond assert[i,φ] dos Γ) =
     (¬(List.member dos (Entry i φ +)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DᵢBⱼBᵢ {φ})))"
  | "(PreCond question[j,i,φ] dos Γ) =
      (¬(j = i) ∧ (List.member dos (Entry i φ +)) ∧
        (∀k. ¬(k = j) ⟶ ( (Γ ⊨ DⱼBₖDⱼ (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ])))))))"
  | "(PreCond justify[i, ψ ⊢ˢ φ] dos Γ) =
      ((List.member dos (Entry i φ S)) ∧
       ( ∃j. ( (Γ ⊨ (MapDone question[j,i,φ])) ∨ (Γ ⊨ (MapDone challenge[j,i,φ])) )
       ∧
       (∀k. ¬(k = i) ⟶ (Γ ⊨ DᵢBₖBᵢ (MapEntailment S ψ φ) )) ))"
  | "(PreCond challenge[j,i,φ] dos Γ) =
      (¬(j = i) ∧ (List.member dos (Entry i φ +)) ∧
        (∀k. ¬(k = j) ⟶ ((Γ ⊨ DⱼBₖ ¬Bⱼ {φ}) ∧ (Γ ⊨ DⱼBₖDⱼ (∃Δ.(MapDone justify[i, Δ ⊢⁺ φ])))))))"
  | "(PreCond retract[i,φ,S] dos Γ) =
      (if S=+ then
        ((List.member dos (Entry i φ +)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DᵢBⱼ ¬Bᵢ {φ})))
      else
        ((List.member dos (Entry i φ -)) ∧ (∀j. ¬(j = i) ⟶ (Γ ⊨ DᵢBⱼ ¬¬Bᵢ {φ})))
    )"
```

The post-conditions are encoded with the GammaAdd function, that takes a locution and gives the set of formulas representing its post-conditions. At this point, in the axiomatic semantics of the paper the post-conditions of `question` and `challenge` contain a formula $(\exists \Delta \in \mathcal{A})$ Done $[\texttt{justify}(P_i, \Delta \vdash^+ \varphi)]$, which means that the relevant `justify` locution is uttered. However, we think that this would be better handled by combination rules or operational semantics, and the `justify` locution cannot have been uttered already when the post-conditions of `question` and `challenge` are imposed. For this reason, we simplified the post-conditions of these two locutions.

```
fun GammaAdd::"FatioL ⇒ (σ ⇒ bool)"
  where
    "(GammaAdd assert[i,φ]) =
      (λx. (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BₖDᵢBⱼBᵢ {φ})))"
  | "(GammaAdd question[j,i,φ]) =
      (λx. True)" (* simplified *)
  | "(GammaAdd justify[i, ψ ⊢ˢ φ]) =
      (λx. (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BₖDᵢBⱼBᵢ (MapEntailment S ψ φ) )))"
  | "(GammaAdd challenge[j,i,φ]) =
      (λx. True)" (* simplified *)
  | "(GammaAdd retract[i,φ,S]) =
      (λx. (if S=+ then
              (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BₖDᵢBⱼ ¬Bᵢ {φ}))
            else
              (∀k j. (¬(k = i) ∧ ¬(j = i)) ⟶ (x =  BₖDᵢBⱼ ¬¬Bᵢ {φ}))
     ))"
```

The actual evolution of the set of formulas Γ is encoded by GammaUpdate, that outputs the updated set of formulas Γ' given a current locution and a current set of formulas Γ. It does so by including (1) the new formulas added with GammaAdd, (2) the formulas that are already in Γ and are consistent with (1), and (3) the relevant MapDone formula for the current locution.

```
fun
  GammaUpdate::"FatioL ⇒ (σ ⇒ bool) ⇒ (σ ⇒ bool)"
  where
    "(GammaUpdate statement Γ) =
     (λx. (Γ x ∧ (GammaKeep statement x)) ∨ (GammaAdd statement x) ∨ (x = MapDone statement) )"
```

A `FatioState` is defined as a triple (`FatioL list`, *DOS* `list`, Γ). The function `FatioCheck` goes from one `FatioState` to the next by executing its first locution, and the function `FatioCheckRec` executes an entire dialogue by a series of recursive calls, and outputs `True` if the final state is successful, that is, the pre-conditions of each executed locution are satisfied after having imposed the post-conditions of the previous locution.

```
type_synonym FatioState = "(FatioL list)×(DOS list)×(σ ⇒ bool)"
fun FatioCheck::"FatioState ⇒ FatioState" where
  "FatioCheck ([],dos,Γ) = ([],dos,Γ)"
|"FatioCheck ((FL # FList), dos, Γ) =
    (if (PreCond FL dos Γ)
      then (FList, (DOSUpdate FL dos), (GammaUpdate FL Γ) )
      else ([],[], (λx. False)) )"

fun FatioCheckRec::"FatioState ⇒ bool" where
  "FatioCheckRec ([],dos,Γ) = successfulResult ([],dos,Γ)"
|"FatioCheckRec ((FL # FList), dos, Γ) =
FatioCheckRec (FatioCheck ((FL # FList), dos, Γ))"
```

4. Discussion

We presented a first modelling and implementation of the Fatio protocol in the LogiKEy framework. The next step is obviously to carry out more extensive experiments on it, in order to test it and possibly to improve it accordingly. At this point, we ran only a few experiments with some small dialogues. While the reasoning system is promising, and up to our knowledge it is the first implementation of Fatio that actually uses the intended modal logic semantics, the modelling is not simple and there are a few sources of complexity that emerged:

- *complexity of the relationship between the embeddings of the logics* (the deep embedding is being mapped to the shallow embedding, and the Fatio language, that uses the deeply embedded logic, is again being mapped to the shallow embedding);
- *complexity in terms of how Isabelle/HOL can cope with the modelling*, as
  - we had a few technical issues related to the parsing of the formulas combining different logics, and in order to solve them at this point we are using a large quantity of brackets to speed-up the disambiguation process, and
  - we had some issues regarding the responsiveness of the theorem provers called by `sledgehammer`, caused by the complexity of the involved formulas;
- *complexity in terms of human usability/readability for experimentation* (the formulas to be used to encode the world knowledge and to satisfy the pre-conditions in the axiomatic semantics involve complex nesting of beliefs and desires that might be hard to grasp for a human user of the protocol, and moreover as already mentioned the formulas have to involve large numbers of brackets to help the parser).

An additional challenge in this kind of work is that in the original paper there are extra-logical aspects that have to be clarified in order to implement the protocol. As it is natural with implementations, there are differences between the paper and the code, and during the implementation things can be simplified or might need to be made more explicit. In our case, for example, we simplified the post-conditions of `question` and `challenge`, and we modelled the *dialectical obligations store* DOS in a slightly simpler way compared to the original paper. At the same time, we had to make (temporary) choices about how to model the `Done` operator and on where to put the existential quantifier in the pre-conditions of `question` and `challenge`. These are cases where the experimental approach proposed with LogiKEy can be impactful, as we made these choice with the support of small experiments.

The point of how to assess the correctness of our method is important to discuss. A pen and paper proof would be possible if we had a proper target semantics, but in the process of implementing the protocol we actually had to make changes to the original semantics. Therefore, a contribution of this paper is in making more clear how the target semantics should look like, so to possibly simplify it. In general, an overall future goal is to simplify the modelling to solve some of the challenges listed above. For instance, as already stated, the relationship between the shallow and the deep embedding is not straightforward. Therefore, streamlining it (e.g., by only using a shallow embedding, even though a more sophisticated one) would help to fix some of the complexities we have with the integration of the different reasoning mechanisms.

Acknowledgments

This work was supported by the Luxembourg National Research Fund (FNR) through the project Logical methods for Deontic Explanations (INTER/DFG/23/17415164/LODEX) and the project Deontic Logic for Epistemic Rights (OPEN O20/14776480).
The authors would like to thank Prof. Leon van der Torre for sharing the enthusiasm about Fatio, and David Fuenmayor and Daniel Kirchner for helpful comments.

References

[1] C. Benzmüller, X. Parent, L. van der Torre, Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support, Artificial Intelligence 287 (2020) 103348. doi:10.1016/j.artint.2020.103348.
[2] E. Black, N. Maudet, S. Parsons, Argumentation-based dialogue, Handbook of Formal Argumentation, Volume 2 (2021).
[3] C. Benzmüller, D. Fuenmayor, B. Lomfeld, Modelling value-oriented legal reasoning in LogiKEy, Logics 2 (2024) 31–78. doi:10.3390/logics2010003, preprint arXiv:2006.12789.
[4] X. Parent, C. Benzmüller, Automated verification of deontic correspondences in Isabelle/HOL – first results, in: C. Benzmüller, J. Otten (Eds.), ARQNL 2022: Automated Reasoning in Quantified Non-Classical Logics. Proceedings of the 4th International Workshop on Automated Reasoning in Quantified Non-Classical Logics (ARQNL 2022) affiliated with the 11th International Joint Conference on Automated Reasoning (IJCAR 2022). Haifa, Israel, August 11, 2022, volume 3326, CEUR Workshop Proceedings, CEUR-WS.org, 2023, pp. 92–108. URL: https://ceur-ws.org/Vol-3326/.
[5] C. Benzmüller, Universal (meta-)logical reasoning: Recent successes, Science of Computer Programming 172 (2019) 48–62. doi:10.1016/j.scico.2018.10.008.
[6] P. McBurney, S. Parsons, Locutions for argumentation in agent interaction protocols, in: R. M. van Eijk, M.-P. Huget, F. Dignum (Eds.), Agent Communication, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 209–225.
[7] Fipa. communicative act library specification, Standard SC00037J, 2002. Foundation for Intelligent Physical Agents.
[8] P. McBurney, S. Parsons, Games that agents play: A formal framework for dialogues between autonomous agents, J. Log. Lang. Inf. 11 (2002) 315–334. URL: https://doi.org/10.1023/A:1015586128739. doi:10.1023/A:1015586128739.
[9] T. Nipkow, L. C. Paulson, M. Wenzel, Isabelle/HOL - A Proof Assistant for Higher-Order Logic, volume 2283 of *Lecture Notes in Computer Science*, Springer, 2002.

Bisequent Calculi for Neutral Free Logic with Definite Descriptions

Andrzej Indrzejczak[1], Yaroslav Petrukhin[1]

[1]University of Lodz, 3/5 Lindleya St, Lodz, 90-131, Poland

Abstract

We present a bisequent calculus (BSC) for the minimal theory of definite descriptions (DD) in the setting of neutral free logic, where formulae with non-denoting terms have no truth value. The treatment of quantifiers, atomic formulae and simple terms is based on the approach developed by Pavlović and Gratzl. We extend their results to the version with identity and definite descriptions. In particular, the admissibility of cut is proven for this extended system.

Keywords

Bisequent Calculus, Cut Elimination, Three-valued Logic, Neutral Free Logic, First-order Logic, Definite Descriptions

1. Introduction

There is a variety of logics called free logics (FL) and differing in many respects. The common feature is that singular terms are free from existential assumptions, i.e. they are not assumed to denote an existing object. On the other hand, in all free logics quantifiers are assumed to have an existential import. One of the main divisions of this kind of logics is based on the treatment of atomic formulae with non-denoting terms. In positive FL they can be true, in negative FL they are always false, in neutral FL (NFL) they are neither true nor false. This makes NFL in a sense a kind of partial or three-valued logic, with the third value expressing the truth-value gap.

We focus on the proof-theoretic perspective on free logics, namely the examination of sequent calculi (SC). In case of positive and negative FL one may find several studies, due to Maffezioli and Orlandelli [16], Pavlović and Gratzl [18], or Indrzejczak [5]. However, in case of NFL the situation is worse in general, and particularly poor in the field of proof theory. In addition to the papers of Woodruff [21] and Lehmann [15], where one may find some kind of natural deduction and tableau system, only Pavlović and Gratzl [19] recently investigated SC for NFL based on strong and weak Kleene logics [11].

We are going to continue the research carried out in [19] in two ways: first, we reformulate the results from [19] using our own approach based on bisequent calculi (BSC), developed in [9, 7]; second, we extend the resulting proof systems to cover identity and definite descriptions (DD). These complex terms are usually divided into proper (denoting a unique object, like 'the present King of England') and improper (like non-denoting 'the present King of France' or not unique 'the author of Principia Mathematica'). Improper DD are very troublesome and it seems that NFL is quite natural place for exploring their behaviour. Usually, DD are formalised by means of term-forming ι-operator applied to a variable x and formula φ to obtain a term $\iota x\varphi$. It is worth noting that there is an alternative treatment of DD based on the application of binary quantifier and recently developed by Kürbis [12, 13, 14]. We follow here a more standard, ι-operator based approach.

The theories of DD in positive or negative free logics are usually based on Lambert axiom (L):

$$\forall y(y = \iota x\varphi \leftrightarrow \forall x(\varphi \leftrightarrow y = x)) \tag{L}$$

where $\iota x\varphi$ is closed and does not have any occurrence of y. It is the weakest principle characterising the behaviour of proper DD. In our research, devoted to NFL, we consider (L) as well but to avoid using $\leftrightarrow$,

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

andrzej.indrzejczak@filhist.uni.lodz.pl (A. Indrzejczak); yaroslav.petrukhin@gmail.com (Y. Petrukhin)

0000-0003-4063-1651 (A. Indrzejczak); 0000-0002-7731-1339 (Y. Petrukhin)

which considerably complicate the set of required rules and proofs in the setting of Kleene's logic, we will use instead two axioms which together express the same minimal theory of DD:

$$\forall y(y = \iota x\varphi \to \varphi[x/y] \wedge \forall x(\varphi \to y = x)) \tag{$L^{\to}$}$$
$$\forall y(\varphi[x/y] \wedge \forall x(\varphi \to y = x) \to y = \iota x\varphi) \tag{$L^{\leftarrow}$}$$

Moreover, we add a restriction: $\iota x\varphi$ contains no other DD inside. There is a possible treatment of this theory admitting nested DD but it leads to the formulation of rules which do not allow us to prove the admissibility of cut.

Various definitions of the logical connectives and quantifiers can be considered in many-valued logics in general, and the choice of them has an influence also on the behaviour of DD. In this study, following [19], we examine two variants of NFL, based on strong and weak Kleene's [11] interpretation of connectives.

The goal of the paper is to provide a uniform proof-theoretic treatment of NFL with identity and DD. Regarding the systems for DD in the setting of other logics, recently a great progress has been made in the works of Fitting and Mendelsohn [1], Orlandelli [17] and Indrzejczak [6, 10, 8]. The paper [4], particularly important for us, is devoted to sequent calculi for the theories of DD based on positive and negative free logics. Currently we are going to continue this research and examine the behaviour of DD in NFL, however this logic requires some nonstandard sequent basis, like the one originally applied in [19]. We prefer to use for this aim a slightly different framework, namely a bisequent calculus (BSC) [9, 7], since it has shown its usefulness in the field of formalisation of three- and four-valued logics. Moreover, the rules of the generalised sequent calculus from [19] are easily translatable into BSC. Summing up: the present paper is a combination and a continuation of the research presented in [18, 5] (SC for negative and positive free logics), [19] (SC for neutral free logics), [4] (SC for DD theories based on negative and positive free logics), and [9, 7] (BSC for many-valued logics).

The structure of the paper is as follows. Section 2 contains some preliminaries related to the language and semantics. Section 3 is devoted to the introduction of the BSC for two variants of NFL based on strong and weak Kleene's logic. Section 4 is concerned with the constructive proof of cut admissibility. Section 5 consists of some concluding remarks and open problems.

2. Preliminaries

Following [19], we consider the subsequent language but with added conjunction, for dealing with (L).

Definition 2.1. *[19, Definition 1.1] The alphabet of the language $\mathscr{L}$ consists of:*

(1) *Terms: $t, s, \ldots$, consisting of:*
   (*a*) *Denumerable list of free individual variables (parameters): $a, b, c, \ldots$*
   (*b*) *Denumerable list of bound individual variables: $x, y, z, \ldots$*
(2) *Denumerable list of n-ary predicates, including a unary predicate E*
(3) $\neg, \to, \wedge, \forall, (,)$.

For our purposes we extend the language $\mathscr{L}$ with identity and ι-operator.

Definition 2.2. *The alphabet of the language $\mathscr{L}_\iota^=$ consists of all symbols of the alphabet of the language $\mathscr{L}$ as well as $=$ and ι.*

The notions of terms and formulae of the languages $\mathscr{L}$ and $\mathscr{L}_\iota^=$ are defined by simultaneous recursion. Note that in $\mathscr{L}_\iota^=$, t, s represent arbitrary terms, including DD. Complexity of formulae and terms is measured by the number of logical symbols (including predicates E and $=$). $\varphi[t_1/t_2]$ is used for the operation of correct substitution of an arbitrary term t_2 for all occurrences of a variable/parameter t_1 in φ, and similarly $\Gamma[t_1/t_2]$ for a uniform substitution in all formulae in Γ. We use a simplified semantics from [19, 18] to provide interpretation of this language.

Definition 2.3 (Neutral structure $\mathcal{S}_{nt}$). *[19, Definition 4.1] [18, Definition 3.1] A neutral structure $\mathcal{S}_{nt}$ is a pair $\langle \mathscr{D}, \mathscr{I} \rangle$, where $\mathscr{D} = a_1, \ldots, b_1, \ldots$ is a countable list of parameters, and $\mathscr{I}$ is an interpretation function on $\mathscr{L}$ ($\mathscr{L}_\iota^=$):*

- $\mathscr{I}(t) = t$, *where* $t \in \mathscr{D}$
- $\mathscr{I}(E) \subseteq \mathscr{D}$,
- $\mathscr{I}(=) = Ref \cup Id$, *closed under symmetry and transitivity, where*
  - $Ref = \{\langle t, t\rangle \mid t \in \mathscr{I}(E)\}$,
  - $Id \subseteq \mathscr{I}(E) \times \mathscr{I}(E)$,
- $\mathscr{I}(P^n) \subseteq \mathscr{I}(E)^n$ *such that if* $\langle s, t\rangle \in \mathscr{I}(=)$, *then* $\langle \ldots, s_i, \ldots\rangle \in \mathscr{I}(P^n)$ *iff* $\langle \ldots, t_i, \ldots\rangle \in \mathscr{I}(P^n)$, *for any* n *and any* $1 \leqslant i \leqslant n$.

Note that identity is in principle treated as other predicates but it cannot be defined in this way since domains of models contain just parameters, so it should be defined as a condition on models like in [18]. $\mathscr{I}$ is extended to cover descriptions $\iota x\varphi$, where φ does not contain other DD, by partial mapping from the set of so restricted DDs to $\mathscr{D}$. In case $\mathscr{I}(\iota x\varphi) = a$, for some $a \in \mathscr{I}(E)$, we say that $\iota x\varphi$ is defined and assume that it satisfies the following condition:

$$\mathscr{I}(\iota x\varphi) = \begin{cases} a \text{ iff } \mathscr{V}^i(\varphi[x/a]) = 1 \text{ and } \mathscr{V}^i(\varphi[x/b]) = 0 \text{ for every } a \neq b \in \mathscr{I}(E); \\ \text{otherwise it is undefined.} \end{cases}$$

There is a matter of debate (cf. [15]) which Kleene's logic, strong or weak, provides better option for interpreting truth-value gaps in NFL. Pavlović and Gratzl [19] consider both options and we follow their approach. Accordingly $\mathscr{V}^i$ may be either weak ($i = w$) or strong ($i = s$), and is defined as follows:

Definition 2.4 (Weak valuation $\mathscr{V}^w$). (*An extended version of [19, Definition 4.2]*) *The truth-value assignment $\mathscr{V}^w$ on the structure $\langle \mathscr{D}, \mathscr{I}\rangle$ is defined as follows:*

$$\mathscr{V}^w(Et) = \begin{cases} 1 & \textit{iff } t \in \mathscr{I}(E), \\ 0 & \textit{iff otherwise}; \end{cases} \tag{1}$$

$$\mathscr{V}^w(P^n(t_1, \ldots, t_n)) = \begin{cases} 1 & \textit{iff } \langle t_1, \ldots, t_n\rangle \in \mathscr{I}(P^n), \\ 1/2 & \textit{iff for some } 1 \leqslant i \leqslant n, t_i \notin \mathscr{I}(E), \\ 0 & \textit{iff otherwise}; \end{cases} \tag{2}$$

$$\mathscr{V}^w(\neg\varphi) = \begin{cases} 1 & \textit{iff } \mathscr{V}^w(\varphi) = 0, \\ 1/2 & \textit{iff } \mathscr{V}^w(\varphi) = 1/2, \\ 0 & \textit{iff } \mathscr{V}^w(\varphi) = 1; \end{cases} \tag{3}$$

$$\mathscr{V}^w(\varphi \wedge \psi) = \begin{cases} 1 & \textit{iff } \mathscr{V}^w(\varphi) = 1 \textit{ and } \mathscr{V}^w(\psi) = 1, \\ 1/2 & \textit{iff } \mathscr{V}^w(\varphi) = 1/2 \textit{ or } \mathscr{V}^w(\psi) = 1/2, \\ 0 & \textit{iff otherwise}; \end{cases} \tag{4}$$

$$\mathscr{V}^w(\varphi \to \psi) = \begin{cases} 0 & \textit{iff } \mathscr{V}^w(\varphi) = 1 \textit{ and } \mathscr{V}^w(\psi) = 0, \\ 1/2 & \textit{iff } \mathscr{V}^w(\varphi) = 1/2 \textit{ or } \mathscr{V}^w(\psi) = 1/2, \\ 1 & \textit{iff otherwise}; \end{cases} \tag{5}$$

$$\mathscr{V}^w(\forall x\varphi) = \begin{cases} 1 & \textit{iff for every } t \in \mathscr{I}(E), \mathscr{V}^w(\varphi[x/t]) = 1, \\ 0 & \textit{iff for some } t \in \mathscr{I}(E), \mathscr{V}^w(\varphi[x/t]) = 0, \\ 1/2 & \textit{iff otherwise.} \end{cases} \tag{6}$$

Definition 2.5 (Strong valuation $\mathscr{V}^s$). *[19, Definition 4.3] The truth-value assignment $\mathscr{V}^s$ on the structure $\langle \mathscr{D}, \mathscr{I}\rangle$ is defined as follows* (*the other cases are defined as in Definition 2.4*)*:*

$$\mathscr{V}^s(\varphi \wedge \psi) = \begin{cases} 0 & \textit{iff } \mathscr{V}^s(\varphi) = 0 \textit{ or } \mathscr{V}^s(\psi) = 0, \\ 1 & \textit{iff } \mathscr{V}^s(\varphi) = 1 \textit{ and } \mathscr{V}^s(\psi) = 1, \\ 1/2 & \textit{iff otherwise.} \end{cases} \tag{4'}$$

$$\mathscr{V}^s(\varphi \to \psi) = \begin{cases} 0 & \text{iff } \mathscr{V}^s(\varphi) = 1 \text{ and } \mathscr{V}^s(\psi) = 0, \\ 1 & \text{iff } \mathscr{V}^s(\varphi) = 0 \text{ or } \mathscr{V}^s(\psi) = 1, \\ 1/2 & \text{iff otherwise.} \end{cases} \tag{5'}$$

For each of these valuations, we can obtain two different logics by taking the set of designated values D either as $\{1\}$ or $\{1, 1/2\}$. However, we examine here only the first route, hence in what follows D always denotes $\{1\}$. The notion of the entailment (consequence) relation is defined as follows.

Definition 2.6. *For any set of formulas* Γ *and any formula* φ*, it holds that*

$$\Gamma \models \varphi \text{ iff for any valuation } \mathscr{V}\text{: if } \mathscr{V}(\Gamma) \subseteq D, \text{ then } \mathscr{V}(\varphi) \in D, \text{ where } D = \{1\}.$$

In what follows the strong Kleene's logic will be referred to as $\mathbf{K_3}$ and the weak one as $\mathbf{K_3^W}$. Although propositional $\mathbf{K_3}$ has an empty set of validities it is not so for related NFL as defined by the semantics above; in particular it holds:

Lemma 2.1. $(L^{\rightarrow})$ *and* $(L^{\leftarrow})$ *are valid in* $\mathbf{K_3}$ *and* $\mathbf{K_3^W}$.

Proof. As an example, let us prove that $(L^{\rightarrow})$ is valid in $\mathbf{K_3}$. Suppose that $(L^{\rightarrow})$ is not valid in $\mathbf{K_3}$, that is $\mathscr{V}^s\big((L^{\rightarrow})\big) \neq 1$. Suppose that $\mathscr{V}^s\big((L^{\rightarrow})\big) = 0$. Then for some $t \in \mathscr{I}(E)$, $\mathscr{V}^s\big(t = \iota x\varphi \to \varphi[x/t] \wedge \forall x(\varphi \to t = x)\big) = 0$. Consequently, $\mathscr{V}^s\big(t = \iota x\varphi\big) = 1$ and $\mathscr{V}^s\big(\varphi[x/t] \wedge \forall x(\varphi \to t = x)\big) = 0$. Therefore, $\mathscr{V}^s\big(\varphi[x/t]\big) = 0$ or $\mathscr{V}^s\big(\forall x(\varphi \to t = x)\big) = 0$. Since $t \in \mathscr{I}(E)$ and $\mathscr{V}^s\big(t = \iota x\varphi\big) = 1$, $\mathscr{V}^s(\varphi[x/t]) = 1$ and $\mathscr{V}^s(\varphi[x/b]) = 0$ for every $t \neq b \in \mathscr{I}(E)$. Since $\mathscr{V}^s(\varphi[x/t]) = 1$, we get $\mathscr{V}^s\big(\forall x(\varphi \to t = x)\big) = 0$. Then for some $u \in \mathscr{I}(E)$, $\mathscr{V}^s\big(\varphi[x/u] \to t = u\big) = 0$. Hence, $\mathscr{V}^s\big(\varphi[x/u]\big) = 1$ and $\mathscr{V}^s\big(t = u\big) = 0$. Since $t \neq u \in \mathscr{I}(E)$, $\mathscr{V}^s(\varphi[x/u]) = 0$. A contradiction.

Suppose that $\mathscr{V}^s\big((L^{\rightarrow})\big) = 1/2$. Let $(H^{\rightarrow})$ denote $t = \iota x\varphi \to \varphi[x/t] \wedge \forall x(\varphi \to t = x)$. Hence, we can infer that either there is $t \notin \mathscr{I}(E)$ and $\mathscr{V}^s\big((H^{\rightarrow})\big) \in \{1, 1/2, 0\}$ or there is $t \in \mathscr{I}(E)$ such that $\mathscr{V}^s\big((H^{\rightarrow})\big) = 1/2$. Suppose that there is $t \notin \mathscr{I}(E)$ and $\mathscr{V}^s\big((H^{\rightarrow})\big) \in \{1, 1/2, 0\}$. However, $\mathscr{I}(=) = Ref \cup Id$, where $Ref = \{\langle t, t\rangle \mid t \in \mathscr{I}(E)\}$ and $Id \subseteq \mathscr{I}(E) \times \mathscr{I}(E)$. Thus, it cannot be the case that there is $t \notin \mathscr{I}(E)$ and $\mathscr{V}^s\big((H^{\rightarrow})\big) \in \{1, 1/2, 0\}$. Hence, there is $t \in \mathscr{I}(E)$ such that $\mathscr{V}^s\big((H^{\rightarrow})\big) = 1/2$. It cannot be the case $\mathscr{V}^s\big(t = \iota x\varphi\big) = 1/2$, otherwise $t \notin \mathscr{I}(E)$ or $\iota x\varphi \notin \mathscr{I}(E)$, but $\mathscr{I}(=) \subseteq \mathscr{I}(E)$. Thus, $\mathscr{V}^s\big(t = \iota x\varphi\big) = 1$ and $\mathscr{V}^s\big(\varphi[x/t] \wedge \forall x(\varphi \to t = x)\big) = 1/2$. Since $t \in \mathscr{I}(E)$ and $\mathscr{V}^s\big(t = \iota x\varphi\big) = 1$, $\mathscr{V}^s(\varphi[x/t]) = 1$ and $\mathscr{V}^s(\varphi[x/b]) = 0$ for every $t \neq b \in \mathscr{I}(E)$. Since $\mathscr{V}^s(\varphi[x/t]) = 1$ and $\mathscr{V}^s\big(\varphi[x/t] \wedge \forall x(\varphi \to t = x)\big) = 1/2$, we have $\mathscr{V}^s\big(\forall x(\varphi \to t = x)\big) = 1/2$. Hence, we can infer that either there is $u \notin \mathscr{I}(E)$ and $\mathscr{V}^s\big(\varphi[x/u] \to t = u\big) \in \{1, 1/2, 0\}$ or there is $u \in \mathscr{I}(E)$ such that $\mathscr{V}^s\big(\varphi[x/u] \to t = u\big) = 1/2$. Due to the properties of $\mathscr{I}(=)$, only the latter option holds. Since $\mathscr{I}(=) \subseteq \mathscr{I}(E)$, $\mathscr{V}^s\big(t = u\big) \neq 1/2$. Thus, since $\mathscr{V}^s\big(\varphi[x/u] \to t = u\big) = 1/2$, we obtain $\mathscr{V}^s\big(\varphi[x/u]\big) = 1/2$ and $\mathscr{V}^s\big(t = u\big) = 0$. Since $t, u \in \mathscr{I}(E)$, $t \neq u$, and $\mathscr{V}^s(\varphi[x/b]) = 0$ for every $t \neq b \in \mathscr{I}(E)$, we infer that $\mathscr{V}^s(\varphi[x/u]) = 0$. But we already have that $\mathscr{V}^s\big(\varphi[x/u]\big) = 1/2$. A contradiction. □

3. Bisequent Calculi for NFL

Pavlović and Gratzl [19] formalise NFL by means of some nonstandard SC, where sequents are of the form: $\Gamma; \Pi \Rightarrow \Sigma; \Delta$. However we prefer to use for this aim bisequent calculus (BSC) which was already applied as a uniform basis for arbitrary three-valued logics in [9]. Similar sequent system (but with a slightly different semantic interpretation) was applied also by Fjellstad [2, 3]. Bisequents are ordered pairs of sequents $\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma$, where $\Gamma, \Delta, \Pi, \Sigma$ are finite (possibly empty) multisets of formulae. The correspondence of bisequents to nonstandard sequents from [19] is indicated by using the same metavariables $\Gamma, \Delta, \Pi, \Sigma$ in both cases. B and S are used as metavariables for bisequents and sequents, respectively.

In both systems of NFL a bisequent $\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma$ is axiomatic iff for some atomic formula φ (including identities and *Et*), either $\varphi \in \Gamma \cap \Sigma$ or $\varphi \in \Gamma \cap \Delta$ or $\varphi \in \Pi \cap \Sigma$. Moreover, bisequents of the form $Et_1, \ldots, Et_n, \Gamma \Rightarrow \Delta, P(t_1, \ldots, t_n) \mid P(t_1, \ldots, t_n), \Pi \Rightarrow \Sigma$ are also axiomatic.

$$(\neg\Rightarrow|)\ \frac{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma} \qquad (\Rightarrow\neg\,|)\ \frac{\Gamma \Rightarrow \Delta \mid \varphi, \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta, \neg\varphi \mid \Pi \Rightarrow \Sigma}$$

$$(|\,\neg\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi \mid \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \neg\varphi, \Pi \Rightarrow \Sigma} \qquad (|\Rightarrow\neg)\ \frac{\varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \neg\varphi}$$

$$(\wedge\Rightarrow|)\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta \mid S}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta \mid S} \qquad (\Rightarrow\wedge\,|)\ \frac{\Gamma \Rightarrow \Delta, \varphi \mid S \qquad \Gamma \Rightarrow \Delta, \psi \mid S}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi \mid S}$$

$$(|\,\wedge\Rightarrow)\ \frac{S \mid \varphi, \psi, \Gamma \Rightarrow \Delta}{S \mid \varphi \wedge \psi, \Gamma \Rightarrow \Delta} \qquad (|\Rightarrow\wedge)\ \frac{S \mid \Gamma \Rightarrow \Delta, \varphi \qquad S \mid \Gamma \Rightarrow \Delta, \psi}{S \mid \Gamma \Rightarrow \Delta, \varphi \wedge \psi}$$

$$(\Rightarrow\rightarrow|)\ \frac{\Gamma \Rightarrow \Delta, \psi \mid \varphi, \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi \mid \Pi \Rightarrow \Sigma} \qquad (|\Rightarrow\rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \psi}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi \rightarrow \psi}$$

$$(\rightarrow\Rightarrow|)\ \frac{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi \quad \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}$$

$$(|\rightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi \mid \Pi \Rightarrow \Sigma \quad \Gamma \Rightarrow \Delta \mid \psi, \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \varphi \rightarrow \psi, \Pi \Rightarrow \Sigma}$$

Figure 1: Rules of propositional $\mathbf{K_3}$

$$(|\,\wedge_w\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta \mid \varphi, \psi, \Pi \Rightarrow \Sigma \qquad \Gamma \Rightarrow \Delta, \varphi \mid \varphi, \Pi \Rightarrow \Sigma \qquad \Gamma \Rightarrow \Delta, \psi \mid \psi, \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \varphi \wedge \psi, \Pi \Rightarrow \Sigma}$$

$$(|\Rightarrow\wedge_w)\ \frac{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi, \psi \qquad \varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \psi \qquad \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi \wedge \psi}$$

$$(\Rightarrow\rightarrow_w|)\ \frac{\Gamma \Rightarrow \Delta, \psi \mid \varphi, \Pi \Rightarrow \Sigma \qquad \Gamma \Rightarrow \Delta, \varphi \mid \varphi, \Pi \Rightarrow \Sigma \qquad \Gamma \Rightarrow \Delta, \psi \mid \psi, \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi \mid \Pi \Rightarrow \Sigma}$$

$$(\rightarrow_w\Rightarrow|)\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma \qquad \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi, \psi \qquad \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}$$

Figure 2: Specific rules of propositional $\mathbf{K_3^w}$

As follows from [9], propositional connectives of $\mathbf{K_3}$ are characterised by the rules from Fig. 1. The propositional connectives of $\mathbf{K_3^w}$ are mostly as in $\mathbf{K_3}$, however in four cases we have three-premiss rules given in Fig. 2. Fig. 3 contains the BSC rules for the universal quantifier as well as the existence predicate in the spirit of the rules given in [19].

Note that the applications of $(\forall\Rightarrow|)$ and $(|\forall\Rightarrow)$ are restricted to parameters as instantiated terms, no DD are allowed as instantiated terms! It saves us from losing the subformula property.

In negative FL a standard identity is replaced by its weaker variant with restricted reflexivity $Et \rightarrow t = t$. In NFL we treat identity similarly so it behaves like other predicates and this is why the four rules for E from Fig. 3 apply also to identities.

Notice that the rule $(| \Rightarrow =)$, required for proving the Leibniz Principle, is like the rule from [4]. We lose unrestricted subformula property but not in the essential way; in fact only terms s, t and atoms A are needed which are already present in the conclusion of this rule. It is of course possible to use other rules, with smaller branching factor and satisfying the subformula property, to obtain the same effect. However, in the presence of rules for DD with identities as principal formulas, it leads to serious troubles with proving cut admissibility. On the other hand, this rule avoids the problems and allows us to prove cut admissibility for reasonably small price. It is also possible to prepare a more refined version of the calculus, like in [6], with the separation of rules for different kinds of identities, but this leads to the proliferation of rules and, because of space restrictions, we present here a simpler form of the calculus. A generalisation of quantifier rules to variants admitting DD as instantiated terms is easily provable since $\forall x\varphi, Ed \vdash \varphi[x/d]$ holds.

How BSC relates to the semantics from Section 2? Following [9, p. 331], we give the subsequent

$$(\Rightarrow\forall\,|)\ \frac{Ea, \Gamma \Rightarrow \Delta, \varphi[x/a] \mid S}{\Gamma \Rightarrow \Delta, \forall x\varphi \mid S} \qquad (|\Rightarrow\forall)\ \frac{Ea, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi[x/a]}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \forall x\varphi}$$

$$(\forall\Rightarrow|)\ \frac{Eb, \forall x\varphi, \varphi[x/b], \Gamma \Rightarrow \Delta \mid S}{Eb, \forall x\varphi, \Gamma \Rightarrow \Delta \mid S} \quad (|\forall\Rightarrow)\ \frac{Eb, \Gamma \Rightarrow \Delta \mid \forall x\varphi, \varphi[x/b], \Pi \Rightarrow \Sigma}{Eb, \Gamma \Rightarrow \Delta \mid \forall x\varphi, \Pi \Rightarrow \Sigma}$$

$$(E\Rightarrow|)\ \frac{Et, P[t], \Gamma \Rightarrow \Delta \mid S}{P[t], \Gamma \Rightarrow \Delta \mid S} \qquad (|\Rightarrow E)\ \frac{Et, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, P[t]}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, P[t]}$$

$$(|E\Rightarrow)\ \frac{Et_1, \dots, Et_n, P(t_1, \dots, t_n), \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta \mid P(t_1, \dots, t_n), \Pi \Rightarrow \Sigma} \qquad (ETr_1)\ \frac{Et, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid Et, \Pi \Rightarrow \Sigma}$$

$$(\Rightarrow E|)\ \frac{Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, P(t_1, \dots, t_n)}{Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta, P(t_1, \dots, t_n) \mid \Pi \Rightarrow \Sigma} \qquad (ETr_2)\ \frac{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, Et}{\Gamma \Rightarrow \Delta, Et \mid \Pi \Rightarrow \Sigma}$$

where a is fresh and b, b_i are arbitrary parameters. Both $P[t]$ and $P(t_1, \dots, t_n)$ denote atoms or identities but not Et, moreover identities of the form $b = d$ are excluded since they are governed by rules from figure 5. In $P[t]$ there is at least one occurrence of t and there may be other terms; in $P(t_1, \dots, t_n)$ there are no other terms.

Figure 3: Rules for universal quantifier and existence predicate

$$(|\Rightarrow=)\ \frac{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, t \approx s \qquad \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, A[x/t] \qquad A[x/s], \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}$$

$$(=\Rightarrow|)\ \frac{t = t, Et, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{Et, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma} \qquad (E\imath\Rightarrow|)\ \frac{a = d, Ed, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{Ed, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}$$

where $A[x/t]$ is an atom, or identity or Et, $t \approx s$ denotes either $t = s$ or $s = t$, a is a fresh parameter and d is an arbitrary description.

Figure 4: Rules for identity

$$(\imath\Rightarrow|\ 1)\ \frac{\varphi[x/c], c = \imath x\varphi, Ec, \Gamma \Rightarrow \Delta \mid S}{c = \imath x\varphi, Ec, \Gamma \Rightarrow \Delta \mid S} \qquad (|\ \imath\Rightarrow 1)\ \frac{Ec, \Gamma \Rightarrow \Delta \mid \varphi[x/c], c = \imath x\varphi, \Pi \Rightarrow \Sigma}{Ec, \Gamma \Rightarrow \Delta \mid c = \imath x\varphi, \Pi \Rightarrow \Sigma}$$

$$(\imath\Rightarrow|\ 2)\ \frac{c = \imath x\varphi, Eb, Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi[x/b] \quad b = c, c = \imath x\varphi, Eb, Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}{c = \imath x\varphi, Eb, Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma}$$

$$(|\ \imath\Rightarrow 2)\ \frac{Eb, Ec, \Gamma \Rightarrow \Delta, \varphi[x/b] \mid c = \imath x\varphi, \Pi \Rightarrow \Sigma \quad Eb, Ec, \Gamma \Rightarrow \Delta \mid b = c, c = \imath x\varphi, \Pi \Rightarrow \Sigma}{Eb, Ec, \Gamma \Rightarrow \Delta \mid c = \imath x\varphi, \Pi \Rightarrow \Sigma}$$

$$(|\Rightarrow \imath)\ \frac{Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi[x/c] \quad Ea, Ec, \varphi[x/a], \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, a = c}{Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, c = \imath x\varphi,}$$

$$(\Rightarrow \imath\,|)\ \frac{Ec, \Gamma \Rightarrow \Delta, \varphi[x/c] \mid \Pi \Rightarrow \Sigma \quad Ea, Ec, \Gamma \Rightarrow \Delta, a = c \mid \varphi[x/a], \Pi \Rightarrow \Sigma}{Ec, \Gamma \Rightarrow \Delta, c = \imath x\varphi \mid \Pi \Rightarrow \Sigma}$$

where a is a fresh parameter.

Figure 5: Rules for DD

definition.

Definition 3.1. $\vDash \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma$ *iff every valuation* $\mathscr{V}$ *satisfies* $\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma$. *The latter holds for* $\mathscr{V}$ *iff for some* φ*: either* ($\varphi \in \Gamma$ *and* $\mathscr{V}(\varphi) \neq 1$) *or* ($\varphi \in \Delta$ *and* $\mathscr{V}(\varphi) = 1$) *or* ($\varphi \in \Pi$ *and* $\mathscr{V}(\varphi) = 0$) *or* ($\varphi \in \Sigma$ *and* $\mathscr{V}(\varphi) \neq 0$). *Clearly* $\nvDash \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma$ *iff for some* $\mathscr{V}$*, all elements of* Γ *are true, all elements of* Δ *are either false or undefined, all elements of* Π *are either true or undefined and all elements of* Σ *are false. In this case we say that* $\mathscr{V}$ *falsifies this sequent.*

As follows from [9], all axiomatic bisequents are valid and all the rules for connectives are both sound (validity-preserving) and invertible. The same holds for the rules from Fig. 3 as follows from [19]. It may be extended to the new rules from Fig. 4 and 5, hence it holds:

THEOREM 3.1 (Soundness). *For any bisequent B, if $\vdash B$, then $\models B$.*

Proof. By induction of the height of the derivation, using the fact that all the rules are sound. As an example, let us illustrate soundness of the rule ($|\Rightarrow \iota$). The proof holds both for $\mathbf{K_3}$ and $\mathbf{K_3^w}$. Recall that a is fresh.

(1) Suppose that $\models Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi[x/c]$.
(2) Thus, for any valuation $\mathscr{V}$, $\mathscr{V}(Ec) \neq 1$, or for some $\gamma \in \Gamma$, $\mathscr{V}(\gamma) \neq 1$, or for some $\delta \in \Delta$, $\mathscr{V}(\delta) = 1$, or for some $\pi \in \Pi$, $\mathscr{V}(\pi) = 0$, or for some $\sigma \in \Sigma$, $\mathscr{V}(\sigma) \neq 0$, or $\mathscr{V}(\varphi[x/c]) \neq 0$.
(3) Suppose that $\models Ea, \varphi[x/a], Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, a = c$.
(4) Hence, for any valuation $\mathscr{V}$, $\mathscr{V}(Ea) \neq 1$, or $\mathscr{V}(\varphi[x/a]) \neq 1$, or $\mathscr{V}(Ec) \neq 1$, or for some $\gamma \in \Gamma$, $\mathscr{V}(\gamma) \neq 1$, or for some $\delta \in \Delta$, $\mathscr{V}(\delta) = 1$, or for some $\pi \in \Pi$, $\mathscr{V}(\pi) = 0$, or for some $\sigma \in \Sigma$, $\mathscr{V}(\sigma) \neq 0$, or $\mathscr{V}(a = c) \neq 0$.
(5) $\not\models Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, c = \iota x\varphi$.
(6) Therefore, there is a valuation $\mathscr{V}$ such that $\mathscr{V}(Ec) = 1$, for any $\gamma \in \Gamma$, $\mathscr{V}(\gamma) = 1$, for any $\delta \in \Delta$, $\mathscr{V}(\delta) \neq 1$, for any $\pi \in \Pi$, $\mathscr{V}(\pi) \neq 0$, and for any $\sigma \in \Sigma$, $\mathscr{V}(\sigma) = 0$, $\mathscr{V}(c = \iota x\varphi) = 0$.
(7) Then we obtain:
    a) $\mathscr{V}(\varphi[x/c]) \neq 0$,
    b) $\mathscr{V}(Ea) \neq 1$, or $\mathscr{V}(\varphi[x/a]) \neq 1$, or $\mathscr{V}(a = c) \neq 0$,
    c) $\mathscr{V}(Ec) = 1$, $\mathscr{V}(c = \iota x\varphi) = 0$.
(8) By the properties of $\mathscr{I}(=)$, we have $c, \iota x\varphi \in \mathscr{I}(E)$.
(9) $\mathscr{I}(\iota x\varphi) \neq c$, that is $\mathscr{V}(\varphi[x/c]) \neq 1$ or $\mathscr{V}(\varphi[x/b]) \neq 0$, for some $c \neq b \in \mathscr{I}(E)$.
(10) Since a is fresh, $\mathscr{V}(\varphi[x/c]) \neq 1$ or $\big(\mathscr{V}(\varphi[x/a]) \neq 0$ and $c \neq a \in \mathscr{I}(E)\big)$.
(11) Suppose that $\mathscr{V}(\varphi[x/c]) \neq 1$. Thus, $\mathscr{V}(\varphi[x/c]) = 1/2$, since also $\mathscr{V}(\varphi[x/c]) \neq 0$. Since $c \in \mathscr{I}(E)$, $\mathscr{V}(\varphi[x/c]) \neq 1/2$. Contradiction.
(12) Suppose that $\mathscr{V}(\varphi[x/a]) \neq 0$ and $c \neq a \in \mathscr{I}(E)$.
(13) Thus, $\mathscr{V}(Ea) = 1$ and $\mathscr{V}(a = c) \neq 1$.
(14) Hence, $\mathscr{V}(\varphi[x/a]) \neq 1$ or $\mathscr{V}(a = c) \neq 0$.
(15) If $\mathscr{V}(\varphi[x/a]) \neq 1$, then $\mathscr{V}(\varphi[x/a]) = 1/2$, since $\mathscr{V}(\varphi[x/a]) \neq 0$. Since $a \in \mathscr{I}(E)$, $\mathscr{V}(\varphi[x/a]) \neq 1/2$. Contradiction.
(16) If $\mathscr{V}(a = c) \neq 0$, then $\mathscr{V}(a = c) = 1/2$, since $\mathscr{V}(a = c) \neq 1$. However, $\mathscr{V}(a = c) \neq 1/2$, because $a, c \in \mathscr{I}(E)$. Contradiction.
(17) Contradiction. Thus, $\models Ec, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, c = \iota x\varphi$.

□

In fact, for the NFL without DD and identity, as defined by rules from Fig. 1, 2, 3, also completeness was proved in [19], and it may be extended in a straightforward way to cover identity, as was done for negative FL in [18]. For DD we already demonstrated that (L) is valid, so we restrict our considerations of adequacy to show that in (complete) BSC for NFL without DD we can prove interderivability of our rules for DD with both forms of (L).

To do that we must restrict our general notion of provability in BSC. Although we defined satisifability and validity for arbitrary bisequents, in fact we are interested in the narrower notion of BSC proof, related to definition of consequence relation, as defined in Section 2. The definition of validity for bisequents shows that the entailment between Γ and φ is expressed in BSC as provability of $\Gamma \Rightarrow \varphi \mid \Rightarrow$. Moreover, although in propositional Kleene's logic the set of validities is empty it is not so in the first-order NFL based on it. In particular, both forms of (L) are valid and will be shown provable.

To save space in the proofs we will usually omit repetitions of formulae which are still active in premisses, according to rule schema, but which are not essential for the proof in question. The following three results hold:

Lemma 3.1 (Substitution). *If* $\vdash_n \Gamma \Rightarrow \Delta \mid \Theta \Rightarrow \Lambda$ *is derivable (where n denotes derivability with height bounded by n), then the sequent* $\vdash_n \Gamma[b/c] \Rightarrow \Delta[b/c] \mid \Theta[b/c] \Rightarrow \Lambda[b/c]$ *is likewise derivable.*

Proof. Similar to the proof in [19]. Note that it is restricted to parameters. □

Lemma 3.2. $\vdash Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta, \varphi(t_1, \dots, t_n) \mid \varphi(t_1, \dots, t_n), \Pi \Rightarrow \Sigma$, *where* $t_1, \dots, t_n$ *are all terms occuring in* φ

Proof. By induction on the complexity of φ. □

Lemma 3.3 (Leibniz Law). *For any formula* φ, *it holds that* $\vdash t_1 = t_2, \varphi[x/t_1] \Rightarrow \varphi[x/t_2] \mid S$.

Proof. By induction on the complexity of φ. □

The proof of $(L^{\leftarrow})$ in strong Kleene logic is as follows, where (E) stands for provable $Ea \Rightarrow \varphi[x/a] \mid \varphi[x/a] \Rightarrow$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Eb \Rightarrow \varphi[x/b] \mid \varphi[x/b] \Rightarrow \qquad Ea, Eb \Rightarrow a = b \mid a = b \Rightarrow}{Ea, Eb \Rightarrow a = b \mid \varphi[x/b], \varphi[x/b] \to a = b \Rightarrow}(|\to\Rightarrow)}{Ea, Eb \Rightarrow a = b \mid \varphi[x/b], \forall x(\varphi \to a = x) \Rightarrow}(|\,\forall\Rightarrow)}{(E) \qquad Ea \Rightarrow a = \iota x\varphi \mid \varphi[x/a], \forall x(\varphi \to a = x) \Rightarrow}(\Rightarrow \iota\,|)}{Ea \Rightarrow a = \iota x\varphi \mid \varphi[x/a] \wedge \forall x(\varphi \to a = x) \Rightarrow}(|\wedge\Rightarrow)}{Ea \Rightarrow \varphi[x/a] \wedge \forall x(\varphi \to a = x) \to a = \iota x\varphi \mid\Rightarrow}(\Rightarrow\to|)}{\Rightarrow \forall y(\varphi[x/y] \wedge \forall x(\varphi \to y = x) \to y = \iota x\varphi) \mid\Rightarrow}(\Rightarrow\forall\,|)$$

For $(L^{\rightarrow})$ let (E') stand for the following proof:

$$\dfrac{Ea \Rightarrow \varphi[x/a] \mid \varphi[x/a] \Rightarrow}{Ea \Rightarrow \varphi[x/a] \mid a = \iota x\varphi \Rightarrow}(|\,\iota \Rightarrow 1)$$

Then:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Eb \Rightarrow \varphi[x/b] \mid \varphi[x/b] \Rightarrow \qquad Ea, Eb \Rightarrow a = b \mid a = b \Rightarrow}{Ea, Eb \Rightarrow a = b \mid \varphi[x/b], a = \iota x\varphi \Rightarrow}(|\,\iota \Rightarrow 2)}{Ea, Eb \Rightarrow \varphi[x/b] \to a = b \mid a = \iota x\varphi \Rightarrow}(\Rightarrow\to|)}{(E') \qquad Ea \Rightarrow \forall x(\varphi \to a = x) \mid a = \iota x\varphi \Rightarrow}(\Rightarrow\forall\,|)}{Ea \Rightarrow \varphi[x/a] \wedge \forall x(\varphi \to a = x) \mid a = \iota x\varphi \Rightarrow}(\Rightarrow\wedge\,|)}{Ea \Rightarrow a = \iota x\varphi \to \varphi[x/a] \wedge \forall x(\varphi \to a = x) \mid\Rightarrow}(\Rightarrow\to|)}{\Rightarrow \forall y(y = \iota x\varphi \to \varphi[x/y] \wedge \forall x(\varphi \to y = x)) \mid\Rightarrow}(\Rightarrow\forall\,|)$$

Proofs in BSC for weak Kleene's logic are more complex since the applications of $(\Rightarrow\to|)$ and $(|\wedge\Rightarrow)$ introduce additional premisses. However, in all respective premisses we obtain simply sequents of the form $Ea, \Gamma \Rightarrow \Delta, \psi(a) \mid \psi(a), \Pi \Rightarrow \Sigma$ or $Ea, Eb, \Gamma \Rightarrow \Delta, \psi(a, b) \mid \psi(a, b), \Pi \Rightarrow \Sigma$ which are provable.

All rules for DD are derivable if we use $(L^{\rightarrow})$ or $(L^{\leftarrow})$ as additional axioms and use cuts which will be proved admissible in the next section. Here is an example. From both premisses of $(\Rightarrow \iota\,|)$ we obtain:

$$\dfrac{Ec, \Gamma \Rightarrow \Delta, \varphi[x/c] \mid \Pi \Rightarrow \Sigma \qquad \dfrac{\dfrac{\dfrac{Ec, Ea, \Gamma \Rightarrow \Delta, c = a \mid \varphi[x/a], \Pi \Rightarrow \Sigma}{Ec, Ea, \Gamma \Rightarrow \Delta, \varphi[x/a] \to c = a \mid \Pi \Rightarrow \Sigma}(\Rightarrow\to|)}{Ec, \Gamma \Rightarrow \Delta, \forall x(\varphi \to c = x) \mid \Pi \Rightarrow \Sigma}(\Rightarrow\forall\,|)}{Ec, \Gamma \Rightarrow \Delta, \varphi[x/c] \wedge \forall x(\varphi \to c = x) \mid \Pi \Rightarrow \Sigma}(|\Rightarrow\wedge)$$

which by cut with $Ec, \varphi[x/c] \wedge \forall x(\varphi \to c = x) \Rightarrow c = \iota x\varphi \mid\Rightarrow$ yields the conclusion of $(\Rightarrow \iota\,|)$. The latter bisequent is proved by cuts on the axiomatic sequent $(L^{\leftarrow})$ i.e. $\Rightarrow \forall y(\varphi[x/y] \wedge \forall x(\varphi \to y = x) \to y = \iota x\varphi) \mid\Rightarrow$ with $Ec, \forall y(\varphi[x/y] \wedge \forall x(\varphi \to y = x) \to y = \iota x\varphi) \Rightarrow \varphi[x/c] \wedge \forall x(\varphi \to c = x) \to c = \iota x\varphi \mid\Rightarrow$ and $\varphi[x/c] \wedge \forall x(\varphi \to c = x) \to c = \iota x\varphi), \varphi[x/c] \wedge \forall x(\varphi \to c = x) \Rightarrow c = \iota x\varphi \mid\Rightarrow$ which are easily provable.

$$(W\Rightarrow|)\ \frac{\Gamma \Rightarrow \Delta \mid S}{\varphi, \Gamma \Rightarrow \Delta \mid S} \qquad (\Rightarrow W|)\ \frac{\Gamma \Rightarrow \Delta \mid S}{\Gamma \Rightarrow \Delta, \varphi \mid S} \qquad (|W\Rightarrow)\ \frac{S \mid \Pi \Rightarrow \Sigma}{S \mid \varphi, \Pi \Rightarrow \Sigma} \qquad (|\Rightarrow W)\ \frac{S \mid \Pi \Rightarrow \Sigma}{S \mid \Pi \Rightarrow \Sigma, \varphi}$$

$$(C\Rightarrow|)\ \frac{\varphi, \varphi, \Gamma \Rightarrow \Delta \mid S}{\varphi, \Gamma \Rightarrow \Delta \mid S} \qquad (\Rightarrow C|)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi \mid S}{\Gamma \Rightarrow \Delta, \varphi \mid S} \qquad (|C\Rightarrow)\ \frac{S \mid \varphi, \varphi, \Pi \Rightarrow \Sigma}{S \mid \varphi, \Pi \Rightarrow \Sigma} \qquad (|\Rightarrow C)\ \frac{S \mid \Pi \Rightarrow \Sigma, \varphi, \varphi}{S \mid \Pi \Rightarrow \Sigma, \varphi}$$

Figure 6: Admissible structural rules

4. Cut Admissibility

In order to show that BSC is cut-free we need to prove the following results:

Lemma 4.1 (Generalisation of axioms). *For any formula φ, the following bisequents are derivable:*

(1) $\varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi$,
(2) $\varphi, \Gamma \Rightarrow \Delta, \varphi \mid \Pi \Rightarrow \Sigma$,
(3) $\Gamma \Rightarrow \Delta \mid \varphi, \Pi \Rightarrow \Sigma, \varphi$.
(4) $Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta, \varphi[t_1, \dots, t_n] \mid \varphi[t_1, \dots, t_n], \Pi \Rightarrow \Sigma$.

Proof. By induction on the complexity of φ. □

Here we present proofs of the admissibility of cut with all forms of cuts from [19]. But first we demonstrate the admissibility of structural rules, like in [19].

Lemma 4.2 (Admissibility of structural rules). *Structural rules from Fig. 6 are height-preserving admissible.*

Proof. By induction on the height of the derivation. □

In fact, the proof of admissibility of contraction presupposes the following:

Lemma 4.3 (Invertibility). *All the rules of the bisequent calculi in question are height-preserving invertible.*

Proof. By induction on the height of the derivation, using Lemma 3.1. □

We also need the following:

Lemma 4.4 (Transfer). *The following rules are height-preserving admissible:*

$$(LTr)\ \frac{\Gamma \Rightarrow \Delta \mid \varphi, \Pi \Rightarrow \Sigma}{\varphi, \Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma} \qquad (RTr)\ \frac{\Gamma \Rightarrow \Delta, \varphi \mid \Pi \Rightarrow \Sigma}{\Gamma \Rightarrow \Delta \mid \Pi \Rightarrow \Sigma, \varphi}$$

Proof. Straightforward extension of the proof by induction on the height of the derivation from [19]. □

In bisequent framework, we have several cut rules[1] listed in Fig. 7.

Theorem 4.1 (Cut admissibility). *The rules (E-Cut), (L-Cut), (O-Cut), (I-Cut), (R-Cut), (3-Cut) are admissible.*

Proof. Admissibility of $(E-Cut)$ and $(L-Cut)$ is proved first in [19], and their proof applies to our system with no changes. The remaining variants of cut are proved in [19] simultaneously, by double induction on the complexity of the cut formula and on the height of the cut (the sum of heights of premises of cut). The extension of their proof to cover our rules is in some cases straightforward so we consider only the most complex situation with triple cut:

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, c = \imath x\varphi \quad c = \imath x\varphi, \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2 \quad Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}{Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}$$

[1]Notice that we considered just two cut rules on the propositional level [9]. However, the first-order case requires more options, which are actually the same as Pavlović and Gratzl have [19]. They follow the strategy of Fjellstad [3] in this respect.

$$\text{(E-Cut)}\ \frac{\Gamma \Rightarrow \Delta \mid \Lambda \Rightarrow \Theta, Et \quad Et, \Pi \Rightarrow \Sigma \mid \Xi \Rightarrow \Omega}{\Gamma, \Pi \Rightarrow \Delta, \Sigma \mid \Lambda, \Xi \Rightarrow \Theta, \Omega}$$

$$\text{(L-Cut)}\ \frac{Et_1, \dots, Et_n, \Gamma \Rightarrow \Delta \mid \Lambda \Rightarrow \Theta, P(t_1, \dots, t_n) \quad Et_1, \dots, Et_n, P(t_1, \dots, t_n), \Pi \Rightarrow \Sigma \mid \Xi \Rightarrow \Omega}{Et_1, \dots, Et_n \Gamma, \Pi \Rightarrow \Delta, \Sigma \mid \Lambda, \Xi \Rightarrow \Theta, \Omega}$$

$$\text{(O-Cut)}\ \frac{\Gamma \Rightarrow \Delta, \varphi \mid \Lambda \Rightarrow \Theta \quad \varphi, \Pi \Rightarrow \Sigma \mid \Xi \Rightarrow \Omega}{\Gamma, \Pi \Rightarrow \Delta, \Sigma \mid \Lambda, \Xi \Rightarrow \Theta, \Omega}$$

$$\text{(I-Cut)}\ \frac{\Gamma \Rightarrow \Delta \mid \Lambda \Rightarrow \Theta, \varphi \quad \Pi \Rightarrow \Sigma \mid \varphi, \Xi \Rightarrow \Omega}{\Gamma, \Pi \Rightarrow \Delta, \Sigma \mid \Lambda, \Xi \Rightarrow \Theta, \Omega}$$

$$\text{(R-Cut)}\ \frac{\Gamma \Rightarrow \Delta \mid \varphi, \Lambda \Rightarrow \Theta \quad \Pi \Rightarrow \Sigma, \varphi \mid \Xi \Rightarrow \Omega}{\Gamma, \Pi \Rightarrow \Delta, \Sigma \mid \Lambda, \Xi \Rightarrow \Theta, \Omega}$$

$$\text{(3-Cut)}\ \frac{\varphi, \Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1 \quad \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2, \varphi \quad \Gamma_3 \Rightarrow \Delta_3, \varphi \mid \varphi, \Lambda_3 \Rightarrow \Theta_3}{\Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}$$

Figure 7: Cut rules

for simplicity let us refer to the premisses of this cut as P_l, P_m, P_r. Since the middle premiss P_m may be derived either by $(\iota \Rightarrow\mid 1)$ or $(\iota \Rightarrow\mid 2)$ and the rightmost premiss P_r may be derived either by $(\Rightarrow \iota \mid)$ (on the left identity) or $(\mid \iota \Rightarrow 1)$ or $(\mid \iota \Rightarrow 2)$ (on the right identity) we have six subcases in total. The leftmost premiss P_l is derivable only by $(\mid\Rightarrow \iota)$ hence we have always:

$$\frac{\dfrac{D_1}{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, \varphi[x/c]} \quad \dfrac{D_2}{Ea, \varphi[x/a], \Gamma_1 \Rightarrow \Delta_1, \mid \Lambda_1 \Rightarrow \Theta_1, c = a}}{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, c = \iota x \varphi}\ (\mid\Rightarrow \iota)$$

1. and let the middle premiss be derived as follows:

$$\frac{\dfrac{D_3}{c = \iota x \varphi, \varphi[x/c], \Gamma_2 \Rightarrow \Delta_2, \mid \Lambda_2 \Rightarrow \Theta_2}}{c = \iota x \varphi, \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2}\ (\iota \Rightarrow\mid)$$

1.1. and the rightmost one:

$$\frac{\dfrac{D_4}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \iota x \varphi \mid c = \iota x \varphi, \varphi[x/c], \Lambda_3 \Rightarrow \Theta_3}}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \iota x \varphi \mid c = \iota x \varphi, \Lambda_3 \Rightarrow \Theta_3}\ (\mid \iota \Rightarrow 1)$$

let A_1 stands for the following:

$$\frac{D_1}{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, \varphi[x/c]}$$

we transform the proof as follows:

$$\dfrac{\dfrac{A_1 \qquad \dfrac{P_l \qquad P_m \qquad \dfrac{D_4}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \iota x \varphi \mid c = \iota x \varphi, \varphi[x/c], \Lambda_3 \Rightarrow \Theta_3}}{Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \varphi[x/c], \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\ \text{(3-Cut)}}{Ec, \Gamma_1, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_1 \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_1, \Theta_2, \Theta_3}\ \text{(I-Cut)}}{Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\ (C)$$

where $(3 - Cut)$ is admissible by induction on the height and $(I - Cut)$ by induction on the degree.
1.2. P_r is derived as follows:

$$\frac{A_2 \qquad A_3}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \iota x \varphi \mid c = \iota x \varphi, \Lambda_3 \Rightarrow \Theta_3}\ (\mid \iota \Rightarrow 2)$$

where A_2 stands for

$$\frac{D_4}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi, \varphi[x/b] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}$$

and A_3 stands for

$$\frac{D_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, c = b, \Lambda_3 \Rightarrow \Theta_3}$$

note that also Eb must occur in Γ_3. We perform:

$(3 - Cut)$ on P_l, P_m and $Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi, \varphi[x/b] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3$ yield $S_4 := Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3, \varphi[x/b] \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$

$(3 - Cut)$ on P_l, P_m and $Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, c = b, \Lambda_3 \Rightarrow \Theta_3$ yield $S_5 := Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3, \mid c = b, \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$

both cuts are admissible by induction on the height. By substitution lemma on D_2 we obtain a proof of $S_2 := Eb, \varphi[x/b], \Gamma_1 \Rightarrow \Delta_1, \mid \Lambda_1 \Rightarrow \Theta_1, c = b$. We finish with:

$$\dfrac{S_5 \qquad \dfrac{S_4 \qquad S_2}{Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid c = b, \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\,(\text{O-Cut})(C)}{Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\,(\text{I-Cut})(C)$$

where both cuts are admissible by induction on the degree.

1.3. P_r is derived as follows:

$$\frac{A_4 \qquad A_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}\,(\Rightarrow \imath\,|)$$

where A_4 stands for

$$\frac{D_4}{Ec, \Gamma_3 \Rightarrow \Delta_3, \varphi[x/c] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}$$

and A_5 stands for

$$\frac{D_5}{Ea', Ec, \Gamma_3 \Rightarrow \Delta_3, c = a' \mid c = \imath x\varphi, \varphi[x/a'], \Lambda_3 \Rightarrow \Theta_3}$$

where a' is eigenvariable different from a. We perform:

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, c = \imath x\varphi \qquad \dfrac{D_4}{Ec, \Gamma_3 \Rightarrow \Delta_3, \varphi[x/c] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}}{Ec, \Gamma_1, \Gamma_3 \Rightarrow \Delta_1, \Delta_3, \varphi[x/c] \mid \Lambda_1, \Lambda_3 \Rightarrow \Theta_1, \Theta_3}\,(\text{I-Cut})$$

and

$$\frac{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, c = \imath x\varphi \qquad A_6 \qquad Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}{Ec, \varphi[x/c], \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\,(3 - Cut)$$

where A_6 stands for

$$\frac{D_3}{c = \imath x\varphi, \varphi[x/c], \Gamma_2 \Rightarrow \Delta_2, \mid \Lambda_2 \Rightarrow \Theta_2}$$

Both cuts are admissible by induction on the height and the resulting sequents by $(O - Cut)$ lead to desired result.

2. Now let the middle premiss be derived as follows:

$$\frac{A_7 \qquad A_8}{Eb, c = \imath x\varphi, \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2}\,(\imath \Rightarrow|)$$

where A_7 stands for

$$\frac{D_3}{Eb, c = \imath x\varphi, \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2, \varphi[x/b]}$$

and A_8 stands for

$$\frac{D_4}{Eb, c = b, c = \imath x\varphi, \Gamma_2 \Rightarrow \Delta_2 \mid \Lambda_2 \Rightarrow \Theta_2}$$

2.1. and the right premiss by:

$$\frac{\dfrac{D_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \varphi[x/c], \Lambda_3 \Rightarrow \Theta_3}}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}\,(\mid \imath \Rightarrow 1)$$

and the sequent resulting by $(3 - Cut)$ is:
$Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$
we transform the proof as follows:

$$\dfrac{\dfrac{A_9 \qquad \dfrac{P_l \qquad P_m \qquad \dfrac{D_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \varphi[x/c], \Lambda_3 \Rightarrow \Theta_3}}{Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \varphi[x/c], \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\,(\text{3-Cut})}{Eb, Ec, \Gamma_1, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_1\Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_1, \Theta_2, \Theta_3}\,(\text{I-Cut})}{Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3}\,(C)$$

where A_9 stands for

$$\frac{D_1}{\Gamma_1 \Rightarrow \Delta_1 \mid \Lambda_1 \Rightarrow \Theta_1, \varphi[x/c]}$$

2.2. If the right premiss is derived:

$$\frac{A_{10} \qquad A_{11}}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}\,(\mid \imath \Rightarrow 2)$$

where A_{10} stands for

$$\frac{D_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi, \varphi[x/b'] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}$$

where A_{11} stands for

$$\frac{D_6}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, c = b', \Lambda_3 \Rightarrow \Theta_3}$$

note that also Eb' must occur in Γ_3 (b' distinct from b).

$(3 - Cut)$ on P_l, P_m and D_5 yields $S_7 := Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3, \varphi[x/b'] \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$

$(3 - Cut)$ on P_l, P_m and D_6 yields $S_8 := Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid c = b', \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$ with cuts admissible by induction on the height.

By substitution lemma on D_2 we get $S_9 := Eb', \varphi[x/b'], \Gamma_1 \Rightarrow \Delta_1, \mid \Lambda_1 \Rightarrow \Theta_1, c = b'$

$(I - Cut)$ and $(O - Cut)$ made on these three sequents yield the desired result.

2.3. Finally let the right premiss be obtained by:

$$\frac{A_{12} \qquad A_{13}}{Ec, \Gamma_3 \Rightarrow \Delta_3, c = \imath x\varphi \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}\,(\Rightarrow \imath \mid)$$

where A_{12} stands for

$$\frac{D_5}{Ec, \Gamma_3 \Rightarrow \Delta_3, \varphi[x/c] \mid c = \imath x\varphi, \Lambda_3 \Rightarrow \Theta_3}$$

and A_{13} stands for

$$\frac{D_6}{Ea', Ec, \Gamma_3 \Rightarrow \Delta_3, c = a' \mid c = \iota x\varphi, \varphi[x/a'], \Lambda_3 \Rightarrow \Theta_3}$$

where a' is eigenvariable different from a.

By substitution lemma on D_6 we get $S := Eb, Ec, \Gamma_3 \Rightarrow \Delta_3, c = b \mid c = \iota x\varphi, \varphi[x/b], \Lambda_3 \Rightarrow \Theta_3$ (the proof has the same height). S by $(I - Cut)$ with P_l gives $S_1 := Eb, Ec, \Gamma_1, \Gamma_3 \Rightarrow \Delta_1, \Delta_3, c = b \mid \varphi[x/b], \Lambda_1\Lambda_3 \Rightarrow \Theta_1, \Theta_3$

$(3 - Cut)$ on P_l, D_3 and P_r yields $S_2 := Eb, Ec, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3, \varphi[x/b]$

$(3 - Cut)$ on P_l, D_4 and P_r yields $S_3 := Eb, Ec, c = b, \Gamma_1, \Gamma_2, \Gamma_3 \Rightarrow \Delta_1, \Delta_2, \Delta_3 \mid \Lambda_1, \Lambda_2, \Lambda_3 \Rightarrow \Theta_1, \Theta_2, \Theta_3$

All these cuts are admissible by induction on the height.

S_1, S_2, S_3 by $(I - Cut)$ and $(O - Cut)$ yield the desired result.

□

5. Conclusion

To the best of our knowledge this paper offers the first proof-theoretic study of NFL with identity and DD. The most important task for the presented variant of NFL is to find a satisfactory solution which is cut free and does not restrict the treatment of DD to terms which do not admit nesting of other DD inside. Such terms like 'the owner of the biggest diamond' should be dealt with in a direct way. Moreover, (L) characterises only the behaviour of proper DD so the present system provides the weakest theory of DD. Since, as we mentioned, NFL seems to be the most natural framework for developing a satisfactory theory of improper DD, the most important task for the future is to develop stronger theories of DD.

Our intention is also to explore alternative approaches to NFL (see [15, 21]) based on different definitions of propositional connectives, viable options concerning the treatment of existence predicate, identity, and alternative interpretations of quantifiers. Moreover, taking into account that careless treatment of DD leads to contradictions, it is an important task to explore its behaviour as founded on paraconsistent logics, in the framework of BSC.

The next step would be to explore the problems of proof search and automatisation for NFL and related systems. It seems that the completeness proof of [19], allowing for building countermodels, can be extended to cover identity and DD by using techniques applied in [10] or [8]. Preparation of analytic tableau systems for this kind of logics and their implementation would be another welcome output, when the theoretical foundations will be firmly established.

Acknowledgements. We would like to thank anonymous reviewers for their valuable comments and suggestions. This research is 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.

References

[1] Fitting, M., Mendelsohn, R., L.: First-Order Modal Logic, 2nd edition, Springer, Cham (2024)

[2] Fjellstad, A.: Non-classical elegance for sequent calculus enthusiasts. Studia Logica, 105(1), 93–119 (2017)

[3] Fjellstad, A.: Structural proof theory for first-order weak Kleene logics. Journal of Applied Non-Classical Logics, 30(3), 272–289 (2020)

[4] Indrzejczak, A.: Free Definite Description Theory — Sequent Calculi and Cut Elimination. Logic and Logical Philosophy. **29**, 505–539 (2020)

[5] Indrzejczak, A.: Free Logics are Cut-Free. Studia Logica. **109**, 859–886 (2021)

[6] Indrzejczak, A.: Russellian Definite Description Theory — a Proof Theoretic Approach. Review of Symbolic Logic, **16 (2)**: 624–649 (2023)
[7] Indrzejczak, A.: Bisequent Calculus for Four-Valued Quasi-Relevant Logics; Cut Elimination and Interpolation. Journal of Automated Reasoning, **67**: paper 37 (2023)
[8] Indrzejczak, A., Kürbis, N.: A Cut-Free, Sound and Complete Russellian Theory of Definite Descriptions. In: Ramanayake, R., Urban, J. (eds) Automated Reasoning with Analytic Tableaux and Related Methods. TABLEAUX 2023. Lecture Notes in Computer Science, vol. 14278, pp. 131–149. Springer, Cham (2023)
[9] Indrzejczak, A. Petrukhin, Y.: A Uniform Formalisation of Three-Valued Logics in Bisequent Calculus. In: International Conference on Automated Deduction, CADE 2023: Automated Deduction — CADE 29. pp. 325–343. Springer, Rome (2023)
[10] Indrzejczak, A., Zawidzki, M.: When Iota meets Lambda. Synthese 201/72 (2023), DOI: 10.1007/s11229-023-04048-y.
[11] Kleene, S. C.: On a notation for ordinal numbers. The Journal of Symbolic Logic. **3**(1), 150–155 (1938)
[12] Kürbis, N., A binary quantifier for definite descriptions in intuitionist negative free logic: Natural deduction and normalization. Bulletin of the Section of Logic. **48**(2), 81–97 (2019)
[13] Kürbis, N., Two treatments of definite descriptions in intuitionist negative free logic. Bulletin of the Section of Logic. **48**(4), 299–317 (2019)
[14] Kürbis, N., Definite descriptions in intuitionist positive free logic. Logic and Logical Philosophy. **30**(2), 327–358 (2021)
[15] Lehmann, S.: Strict Fregean Free Logic. Journal of Philosophical Logic. **23**(3), 307–336 (1994)
[16] Maffezioli, P., Orlandelli, E.: Full cut elimination and interpolation for intuitionistic logic with existence predicate. Bulletin of the Section of Logic. **48**(2), 137–158 (2019)
[17] Orlandelli, E.: Labelled calculi for quantified modal logics with definite descriptions. Journal of Logic and Computation **31**(3) 923–946 (2021)
[18] Pavlović, E., Gratzl, N. A More Unified Approach to Free Logics. J Philos Logic **50**, 117–148 (2021)
[19] Pavlović, E., Gratzl, N. Neutral Free Logic: Motivation, Proof Theory and Models. J Philos Logic **52**, 519–554 (2023)
[20] Russell, B. On Denoting, Mind. **14**, 479–493 (1905)
[21] Woodruff, P.: Logic and Truth Value Gaps, pages 121–142 in K. Lambert (ed.), Philosophical Problems in Logic. Reidel, Dordrecht (1970)

When Epsilon meets Lambda: Extended Leśniewski's Ontology*

Andrzej Indrzejczak

Department of Logic, University of Lodz, Poland

Abstract

Leśniewski's ontology LO is an expressive calculus of names. It provides a basis for mereology but allows also for direct formalisation of reasoning in natural languages. Recently its elementary part was characterised by means of the cut-free sequent calculus GO. In this paper we investigate its extended version ELO which introduces lambda terms to represent complex descriptive names. The hierarchy of three systems is formalised in terms of sequent calculi which satisfy cut elimination and the subformula property.

Keywords

Leśniewski, ontology, calculus of names, sequent calculus, cut elimination

1. Introduction

Despite of the great success of standard first-order languages and their priviliged role in automated deduction, it is often difficult to apply them in a direct and satisfactory way to formalisation of natural languages. The following two features of natural languages are usually discussed in this context: 1) the subject-predicate structure of atomic sentences, characteristic not only for traditional logic but also for modern linguistics with its NP+VP model of sentences applied in generative grammar; 2) the wide class of naming expressions which are used not only to refer to x, but also to convey information about x, and even if they refer to something it is not necessarily the singular reference.

No wonder that several approaches alternative to FOL (first-order logic) were proposed, attempting to obtain a formalisation of arguments in natural languages which is closer to their original structure. One may mention here for example, the calculi of names due to Sommers [25], the variety of relational sylogistics of Moss and Pratt-Hartmann [21], or the logic QUARC of Ben-Yami [2]. Even in the approaches based on the standard first-order languages one may find several proposals related to the second feature of natural languages. Thus the notion of name was extended to non-referring terms in free logics, or the logic of intentional objects of Paśniczek [20], and even to general names (plural reference) in the plural logic of Oliver and Smiley [19]. Not surprisingly, in these approaches a lot of work was devoted to the development of theories of complex names conveying information, like definite descriptions.

One of the oldest approaches of this kind is the calculus of names called Leśniewski's ontology (LO) (see e.g. [24], [15] or [26]). It satisfies both features mentioned above: the subject-predicate

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

andrzej.indrzejczak@filhist.uni.lodz.pl (A. Indrzejczak)

0000-0003-4063-1651 (A. Indrzejczak)

structure of atomic sentences and a wide understanding of names, including empty and general names (like 'Pegasus' or 'an emperor'). LO in the original form was introduced as a formal basis for developing another, better known theory of Leśniewski – mereology [17]. Thus LO was introduced as an alternative to Frege's construction of logic, while mereology was introduced as an alternative to set theory. LO is a theory of the binary predicate ε understood as the formalisation of the Greek 'esti', hence formulae of the form $s\varepsilon t$ express sentences '(the) s is (a/the) t', and their truth conditions are expressed by means of Leśniewski's axiom LA:

$$\forall xy(x\varepsilon y \leftrightarrow \exists z(z\varepsilon x) \wedge \forall z(z\varepsilon x \rightarrow z\varepsilon y) \wedge \forall zv(z\varepsilon x \wedge v\varepsilon x \rightarrow z\varepsilon v))$$

It roughly says that $x\varepsilon y$ holds iff x exists, is y, and is unique. The weak form of LO, called elementary LO (cf. [24]), may be formalised as an extension of an arbitrary axiomatic system for first-order logic (FOL) with added LA. Of course one has to remember that, in spite of the name 'elementary', and the fact that we refer to FOL as the basis, elementary LO is not an elementary theory in the standard sense, since name variables represent also empty and general names. Accordingly, quantifiers have no existential import; this role is taken up by ε.

Recently the elementary LO and its extension with the variety of predicates obtained well-behaved proof-theoretic characterisation in terms of sequent calculi GO and GOP [10]. But there is a problem, at least from the proof-theoretic standpoint, with formalising complex names in LO. We have briefly discussed in [10] the original approach of Leśniewski to the problem and its deficiences. As a result of these problems both GO and GOP were restricted to simple terms only. However, the advantages of having formal tools for dealing with complex names, like definite descriptions, were recognised in many fields, including: proof theory [11, 14, 13], query answering, [3], knowledge representation [1], and many other.

In this paper we focus on the problem of dealing with complex names in LO. To this aim we introduce extended LO (ELO), with lambda terms applied to represent descriptive names. The main idea is to keep two essential features of LO: the subject-predicate structure and the wide notion of name. However, to represent descriptive terms we admit also the application of relational atoms from FOL, in particular inside lambda-terms. Some way of mixing LO with FOL was already considered by Waragai [27] but he introduced special operators for this aim, similarly like Słupecki [24]. The present approach is simpler in the sense that, except the lambda operator, no extra machinery is needed.

Three versions of ELO are considered, differring in the strength of involvement of complex terms in atomic sentences, and characterised by means of sequent calculi which are cut-free and analytic. In section 2 we describe the language and axioms of three variants of ELO, then we focus on the problem of constructing for them well-behaved sequent calculi called GELO. Before proving their adequacy we focus on the characterisation of identity which provides a necessary prerequisite for further formal development. Section 5 presents the adequacy of all variants of GELO and section 6 provides a constructive proof of cut elimination. We close the paper with a few remarks on open problems and possible further developments.

2. Extended Ontology of Leśniewski

The set of logical constants of the language of all variants of ELO consists of connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$), quantifiers ($\forall, \exists$), two special binary predicates ($\varepsilon, \equiv$) and lambda operator λ. We assume a denumerable set of n-ary relational predicate variables $R^n, n > 1$ and name variables divided into bound: $x, y, z, ...$ (possibly with subscripts), and free: $a, b, c, ...$ (also called parameters). Arbitrary terms are denoted as t, s, u (possibly with subscripts), formulae as φ, ψ, χ, their finite multisets as $\Gamma, \Delta, \Pi, \Sigma$. $\varphi[s/t]$ denotes the result of correct substitution of t for all occurrences of s.

The notion of a term and formula is defined by simultaneous recursion. Terms are simple, i.e. name variables, and complex, i.e. lambda terms of the form $\lambda x\varphi$, where φ is a formula. The set of formulae is the set of atoms closed under quantification of name variables and boolean combinations of formulae. What is specific is that there are three kinds of atoms: relational atoms $Rt_1...t_n$, where all arguments are simple terms, i.e. variables, identities $t_1 \equiv t_2$, where both arguments can be simple or complex, and ε-atoms $t_1 \varepsilon t_2$. Similarly as in [10] we apply for simplicity the convention of omitting ε, thus writing st instead of $s\varepsilon t$; it has a deeper sense connected with counting the complexity of terms and formulae. Roughly, the complexity of any term or formula ($c(t), c(\varphi)$) is the number of occurrences of logical constants, except ε. Thus the complexity of relational atoms, as well as of ab is 0, whereas $c(a \equiv b) = 1$. However, in general for ε-atoms and identities we have $c(st) = c(s) + c(t)$ and $c(s \equiv t) = c(s) + c(t) + 1$.

We consider the hierarchy of three languages: weak, medium and strong, depending on what kind of terms are admitted as arguments of ε-atoms $t_1 \varepsilon t_2$:

1. L_w: t_1 simple, t_2 arbitrary;
2. L_m: additionally ε-atoms with both arguments complex;
3. L_s: additionally ε-atoms with t_1 complex and t_2 simple.

So only L_s admits all possible combinations of terms, as in identities.
Note that in the setting of ELO, the axiom LA covers in fact four schemata:

LA_1 $ab \leftrightarrow \exists z(za) \wedge \forall z(za \rightarrow zb) \wedge \forall zv(za \wedge va \rightarrow zv)$:
LA_2 $a\lambda x\psi \leftrightarrow \exists z(za) \wedge \forall z(za \rightarrow z\lambda x\psi) \wedge \forall zv(za \wedge va \rightarrow zv)$;
LA_3 $\lambda x\varphi\lambda x\psi \leftrightarrow \exists z(z\lambda x\varphi) \wedge \forall z(z\lambda x\varphi \rightarrow z\lambda x\psi) \wedge \forall zv(z\lambda x\varphi \wedge v\lambda x\varphi \rightarrow zv)$;
LA_4 $\lambda x\varphi b \leftrightarrow \exists z(z\lambda x\varphi) \wedge \forall z(z\lambda x\varphi \rightarrow zb) \wedge \forall zv(z\lambda x\varphi \wedge v\lambda x\varphi \rightarrow zv)$.

They form a hierarchy of the commitment of complex terms in forming atoms of ELO, representing different strength of expression. Moreover, in the sequent system, they will be dealt with different kinds of rules. Accordingly, we will be talking about three variants of ELO formalised in respective languages:

1. weak ELO$_w$ in L_w satisfying LA_1, LA_2;
2. medium ELO$_m$ in L_m satisfying LA_1, LA_2, LA_3;
3. strong ELO$_s$ in L_s satisfying LA_1, LA_2, LA_3, LA_4.

However, even ELO_s is in a sense too weak for real applications to the analysis of reasoning in natural languages. For example, we are not able to demonstrate the validity of such simple argument as 'Ann is the oldest daughter of Betty. Therefore, she is Betty's daughter.' It may be formalised as $a\lambda x(Dab \wedge \forall y(Dyb \rightarrow Oay))\ /\ a\lambda xDab$ but to derive the conclusion we need some ways of unfolding the content of lambda term. To resolve this problem we introduce a kind of β-conversion (BC) of the form:

$$a\lambda x\varphi \leftrightarrow aa \wedge \varphi[x/a]$$

where aa is added to restrict a to individual names. Similar principles were considered by Waragai [27] and Słupecki [24] for their special operators for making complex terms.

Finally, mainly for technical reasons, we introduce as the primitive notion the predicate of strong identity $\equiv$ axiomatised by the following equivalence SI:

$$t \equiv s \leftrightarrow \forall x(xt \leftrightarrow xs)$$

Summing up, we assume that in each variant of ELO we have BC and SI as axioms added to FOL, and suitable forms of LA, namely: LA_1, LA_2 in LO_w, LA_1, LA_2, LA_3 in LO_m, and LA_1, LA_2, LA_3, LA_4 in LO_s.

3. Sequent Calculi GELO

All variants of ELO will be characterised in terms of sequent calculi called GELO. First we introduce the auxiliary calculus GOI which is the subsystem of the modular extension of GO called GOP (GO with predicates) from [10]. It consists of the rules defined on sequents $\Gamma \Rightarrow \Delta$ and specified in Fig. 1. Formulae displayed in the schemata are active, 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 this rule application. Proofs are finite trees with nodes labelled by sequents. The height of a proof D of $\Gamma \Rightarrow \Delta$ is defined as the number of nodes of the longest branch in D. $\vdash_k \Gamma \Rightarrow \Delta$ means that $\Gamma \Rightarrow \Delta$ has a proof of the height at most k. In general, when presenting proofs, we omit structural rules to save space. Incidentally we use underlining for side formulae and bold type letters for principal formulae of some steps to facilitate reading of proofs.

GOI is cut-free, satisfies the interpolation theorem and LA_1 (the essential rules are $(R), (T), (S), (E)$; see [10, 12]). We assume for further investigations that GOI is the core calculus for obtaining three variants of GELO in their respective languages. But GOI, even if formulated in any of the languages L_w, L_m, L_s, i.e. with added relational atoms and lambda terms, is too weak to obtain any specific results related to complex terms. Moreover, with quantifier rules $(\forall \Rightarrow), (\Rightarrow \exists)$ admitting only parameters as instantiated terms it is incomplete. We could admit arbitrary term t instead of parameter b in these rules, like we did for $(\equiv\Rightarrow), (\Rightarrow\equiv)$ which were also formulated for parameters only in [10], but it destroys the subformula property. Fortunatelly, much better solution is possible.

To obtain GELO_w we have to add to GOI (in L_w) the rules from Fig. 2. The most direct way to obtain the system capable of proving LA_2 is to strengthen the rules $(R), (T), (S), (E)$ in the sense

$$(Cut)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \varphi, \Pi \Rightarrow \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma} \qquad (AX)\ \varphi \Rightarrow \varphi$$

$$(\neg\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi}{\neg\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\neg)\ \frac{\varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg\varphi} \qquad (W\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}$$

$$(\Rightarrow\wedge)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi} \qquad (\wedge\Rightarrow)\ \frac{\varphi, \psi, \Gamma \Rightarrow \Delta}{\varphi \wedge \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow W)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(\vee\Rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \vee \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\vee)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \qquad (C\Rightarrow)\ \frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta}$$

$$(\rightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, \varphi \qquad \psi, \Gamma \Rightarrow \Delta}{\varphi \rightarrow \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\rightarrow)\ \frac{\varphi, \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi} \qquad (\Rightarrow C)\ \frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi}$$

$$(\leftrightarrow\Rightarrow)\ \frac{\Gamma\Rightarrow \Delta, \varphi, \psi \qquad \varphi, \psi, \Gamma\Rightarrow \Delta}{\varphi\leftrightarrow\psi, \Gamma\Rightarrow \Delta} \qquad (\forall\Rightarrow)\ \frac{\varphi[x/b], \Gamma\Rightarrow \Delta}{\forall x\varphi, \Gamma\Rightarrow \Delta} \qquad (\Rightarrow\exists)\ \frac{\Gamma\Rightarrow \Delta, \varphi[x/b]}{\Gamma\Rightarrow \Delta, \exists x\varphi}$$

$$(\Rightarrow\leftrightarrow)\ \frac{\varphi, \Gamma\Rightarrow \Delta, \psi \qquad \psi, \Gamma \Rightarrow \Delta, \varphi}{\Gamma\Rightarrow \Delta, \varphi\leftrightarrow\psi} \qquad (\Rightarrow\forall)\ \frac{\Gamma\Rightarrow \Delta, \varphi[x/a]}{\Gamma\Rightarrow \Delta, \forall x\varphi} \qquad (\exists\Rightarrow)\ \frac{\varphi[x/a], \Gamma\Rightarrow \Delta}{\exists x\varphi, \Gamma\Rightarrow \Delta}$$

$$(\equiv\Rightarrow)\ \frac{\Gamma\Rightarrow \Delta, bt, bs \qquad bt, bs, \Gamma\Rightarrow \Delta}{t \equiv s, \Gamma\Rightarrow \Delta} \qquad (\Rightarrow\equiv)\ \frac{at, \Gamma\Rightarrow \Delta, as \qquad as, \Gamma \Rightarrow \Delta, at}{\Gamma\Rightarrow \Delta, t \equiv s}$$

$$(R)\ \frac{bb, \Gamma\Rightarrow \Delta}{bc, \Gamma\Rightarrow \Delta} \qquad (T)\ \frac{bd, \Gamma\Rightarrow \Delta}{bc, cd, \Gamma\Rightarrow \Delta} \qquad (S)\ \frac{cb, \Gamma\Rightarrow \Delta}{bc, cc, \Gamma\Rightarrow \Delta}$$

$$(E)\ \frac{ab, \Gamma\Rightarrow \Delta, ac \qquad ac, \Gamma\Rightarrow \Delta, ab \qquad cd, \Gamma \Rightarrow \Delta}{bd, \Gamma \Rightarrow \Delta}$$

where a is a fresh parameter (eigenvariable), not present in Γ, Δ and φ, whereas b, c, d are arbitrary parameters, t, s are arbitrary terms.

Figure 1: Calculus GOI

$$(\beta\Rightarrow)\ \frac{\varphi[x/b], \Gamma\Rightarrow \Delta}{b\lambda x\varphi, \Gamma\Rightarrow \Delta} \qquad (\Rightarrow\beta)\ \frac{\Gamma\Rightarrow \Delta, bb \qquad \Gamma\Rightarrow \Delta, \varphi[x/b]}{\Gamma\Rightarrow \Delta, b\lambda x\varphi}$$

$$(\equiv\Rightarrow E)\ \frac{a \equiv t, \Gamma\Rightarrow \Delta}{\Gamma\Rightarrow \Delta} \qquad (\Rightarrow\equiv E)\ \frac{\Gamma\Rightarrow \Delta, b \equiv c \qquad \Gamma\Rightarrow \Delta, \varphi[x/c]}{\Gamma\Rightarrow \Delta, \varphi[x/b]}$$

where a is a fresh parameter (eigenvariable), b, c are arbitrary parameters, $t \in term(\Gamma \cup \Delta)$ [the set of complex terms of $\Gamma \cup \Delta$] in $(\equiv\Rightarrow E)$, and φ in $(\Rightarrow\equiv E)$ is a relational atom.

Figure 2: The rules for GELO_w

of admitting atoms of the form $b\lambda x\varphi$. The identical proofs as those provided in [10] would do the job. But the most direct does not mean the best. If any of $(R), (T), (S), (E)$ admits ε-atoms $b\lambda x\varphi$ it is possible that cut formula of this form is introduced in the left premiss of (Cut) by $(\Rightarrow \beta)$ and in the right premiss by any of $(R), (T), (S), (E)$. In such situation it is not possible to eliminate cut. It is worth emphasizing the important fact: we don't need to modify $(R), (T), (S), (E)$ to obtain LA_2; the rules which apparently characterise only LA_1 are sufficient for this aim (it will be shown in section 5), and it is crucial for proving cut elimination in section 6.

$(\Rightarrow \beta)$ and $(\beta \Rightarrow)$ adequately characterise our principle BC. Two sequents giving by $(\Rightarrow\leftrightarrow)$

the effect of BC are easily provable; on the other hand, two β-rules are easily derivable if such sequents are used as additional axioms.

$(\equiv\Rightarrow E)$ is not much related to the characterisation of $\equiv$ since it is adequately expressed by $(\Rightarrow\equiv), (\equiv\Rightarrow)$, which may be shown in a similar way as in the case of BC versus $(\Rightarrow \beta), (\beta \Rightarrow)$. This rule rather uses $\equiv$ as a vehicle for introducing new parameters representing complex terms. It makes possible to use in our calculi $(\forall \Rightarrow), (\Rightarrow \exists)$ restricted to arbitrary b instead of t, in the way we already exploited for free logics [7] and the Russelian theory of descriptions [11]. As a result, these restricted quantifier rules are sufficiently strong to obtain everything which is provable by means of unrestricted rules admitting arbitrary terms as instances of variables. Formally it may be shown by proving derivability of stronger variants. Here is the case of unrestricted $(\forall \Rightarrow)$:

$$\dfrac{\dfrac{\dfrac{a \equiv t, \varphi[x/a] \Rightarrow \varphi[x/t]}{a \equiv t, \forall x\varphi \Rightarrow \varphi[x/t]}\,(\forall \Rightarrow)}{\forall x\varphi \Rightarrow \varphi[x/t]}\,(\equiv\Rightarrow E) \qquad \varphi[x/t], \Gamma \Rightarrow \Delta}{\forall x\varphi, \Gamma \Rightarrow \Delta}\,(Cut)$$

where the left top sequent is a provable instance of Leibniz Law LL (see section 4). In a similar way we prove derivability of unrestricted $(\Rightarrow \exists)$. On the other hand, $(\equiv\Rightarrow E)$ is easily derivable in the calculus with unrestricted $(\Rightarrow \exists)$:

$$\dfrac{\dfrac{\dfrac{at \Rightarrow at \qquad at \Rightarrow at}{\Rightarrow t \equiv t}\,(\Rightarrow\equiv)}{\Rightarrow \exists x(x \equiv t)}\,(\Rightarrow \exists) \qquad \dfrac{a \equiv t, \Gamma \Rightarrow \Delta}{\exists x(x \equiv t), \Gamma \Rightarrow \Delta}\,(\exists \Rightarrow)}{\Gamma \Rightarrow \Delta}\,(Cut)$$

Since $(\Rightarrow\equiv), (\equiv\Rightarrow)$ deal only with ε-atoms, $(\Rightarrow\equiv E)$ is added to extend the applicability of $\equiv$ to relational atoms. In the effect we get a calculus where $\equiv$ can express Leibniz law (LL) in the unrestricted way. There are several possible rules to obtain this effect (see [8]) and one may think that, for instance, the popular solution due to Negri and von Plato [18] would be more convenient. However, with other kind of rules we face the same problem of the failure of cut elimination as indicated above, in the context of discussion on modified $(R), (T), (S), (E)$ versus $(\Rightarrow \beta)$. To avoid such problems and to allow one to prove cut elimination, this form of the extra rule for $\equiv$ is optimal.

To obtain GELO$_m$ we add the rules from Fig. 3 to GELO$_w$ formulated in L_m. These rules are similar to the rules introduced in [13] to characterise the Russellian theory of definite descriptions with lambda terms. LA is very similar to the Russellian schema of elimination for descriptions, hence this solution works here as well. Eventually to obtain GELO$_s$ we change the language for L_s and relax the proviso concerning t in rules from Fig. 3: t may be an arbitrary term.

Summing up the calculi for three versions of ELO are constructed as follows:

- GELO$_w$ is obtained by addition of the rules from Fig. 2 to GOI in L_w;
- GELO$_m$ is obtained by addition of the rules from Fig. 3 to GELO$_w$ in L_m;
- GELO$_s$ is obtained by relaxing the condition on t in rules from Fig. 3 in L_s.

$$(\lambda \Rightarrow 1)\ \frac{a\lambda x\varphi, at, \Gamma \Rightarrow \Delta}{\lambda x\varphi t, \Gamma \Rightarrow \Delta} \qquad (\lambda \Rightarrow 2)\ \frac{\Gamma \Rightarrow \Delta, c\lambda x\varphi \qquad \Gamma \Rightarrow \Delta, d\lambda x\varphi \qquad cd, \Gamma \Rightarrow \Delta}{\lambda x\varphi t, \Gamma \Rightarrow \Delta}$$

$$(\Rightarrow \lambda)\ \frac{\Gamma \Rightarrow \Delta, c\lambda x\varphi \qquad \Gamma \Rightarrow \Delta, ct \qquad a\lambda x\varphi, b\lambda x\varphi, \Gamma \Rightarrow \Delta, ab}{\Gamma \Rightarrow \Delta, \lambda x\varphi t}$$

where a, b are new parameters (eigenvariable), c, d are arbitrary, t is complex.

Figure 3: The rules for GELO_m

We finish this section with an example of a cut-free proof of the sequent which will be useful in further considerations:

Lemma 1. *The following sequent is cut-free provable in $GELO_w$ and all its extensions:*

$$(\equiv\Rightarrow)\ \frac{(\equiv\Rightarrow)\ \dfrac{b\lambda x\varphi \Rightarrow bc, b\lambda x\varphi \qquad \dfrac{ac \Rightarrow ac}{bc, b\lambda x\varphi, ab \Rightarrow ac}\,(T)}{c \equiv \lambda x\varphi, b\lambda x\varphi, ab \Rightarrow ac, a\lambda x\varphi} \qquad ac, a\lambda x\varphi \Rightarrow a\lambda x\varphi}{(\equiv\Rightarrow E)\ \dfrac{c \equiv \lambda x\varphi, ab, b\lambda x\varphi \Rightarrow a\lambda x\varphi}{ab, b\lambda x\varphi \Rightarrow a\lambda x\varphi}}$$

4. Identity

Before we show the adequacy of our calculi we need to prove some properties of $\equiv$, in particular the provability of the full form of *LL* (Leibniz Law).

Lemma 2. *The following sequents are cut-free provable in all variants of GELO for arbitrary s, t, u:*

1. $\Rightarrow t \equiv t$
2. $s \equiv t \Rightarrow t \equiv s$
3. $s \equiv t, s \equiv u \Rightarrow t \equiv u$
4. $s \equiv t, u \equiv s \Rightarrow u \equiv t$
5. $t \equiv s, s \equiv u \Rightarrow t \equiv u$
6. $t \equiv s, u \equiv s \Rightarrow u \equiv t$

Proof. Case 1 is trivial, by one application of $(\Rightarrow\equiv)$. Case 2:

$$(\equiv\Rightarrow)\ \frac{at \Rightarrow as, at \qquad as, at \Rightarrow as}{(\Rightarrow\equiv)\ \dfrac{at, s \equiv t \Rightarrow as}{s \equiv t \Rightarrow t \equiv s}} \qquad \frac{as \Rightarrow as, at \qquad as, at \Rightarrow at}{as, s \equiv t \Rightarrow at}\,(\equiv\Rightarrow)$$

Case 3:

$$(\equiv\Rightarrow)\ \frac{at \Rightarrow as, at \qquad (\equiv\Rightarrow)\ \dfrac{as \Rightarrow as, au \qquad as, au \Rightarrow au}{as, at, s \equiv u \Rightarrow au}}{(\Rightarrow\equiv)\ \dfrac{s \equiv t, s \equiv u, at \Rightarrow au}{s \equiv t, s \equiv u \Rightarrow t \equiv u}} \qquad s \equiv t, s \equiv u, au \Rightarrow at$$

where the rightmost sequent is provable in symmetric way.

Case 4: For $s \equiv t, u \equiv s \Rightarrow u \equiv t$ the proof is similar.

Cases 5 and 6 are provable in the same way as 3 and 4, since the only difference is that the respective applications of $(\equiv\Rightarrow)$ to $t \equiv s$ give at, as instead of as, at in premisses and the order does not matter. □

Now we are in the position to prove that *LL* holds for all variants of GELO.

Lemma 3. $GELO_w \vdash s \equiv t, \varphi[x/s] \Rightarrow \varphi[x/t]$

Proof. The proof is by induction on the complexity of φ. In the basis we must show that it holds for φ atomic. Since, the previous lemma guarantees the result for identities, and $(\Rightarrow\equiv E)$ for relational atoms, it remains to show that the following cases hold:

1. $s \equiv t, us \Rightarrow ut$
2. $s \equiv t, su \Rightarrow tu$
3. $t \equiv s, us \Rightarrow ut$
4. $t \equiv s, su \Rightarrow tu$

Case 1: u must be simple (the character of s, t does not matter):

$$(\equiv\Rightarrow)\ \frac{us \Rightarrow us, ut \qquad us, ut \Rightarrow ut}{s \equiv t, us \Rightarrow ut}$$

Case 2: s, t are simple; let u be simple (subcase 2.1):

$$(\equiv\Rightarrow)\ \frac{(\equiv\Rightarrow)\ \dfrac{as \Rightarrow as, at \qquad as, at \Rightarrow at}{(E)\ \dfrac{s \equiv t, as \Rightarrow at}{}} \qquad (\equiv\Rightarrow)\ \dfrac{at \Rightarrow as, at \qquad as, at \Rightarrow as}{s \equiv t, at \Rightarrow as} \qquad tu \Rightarrow tu}{s \equiv t, su \Rightarrow tu}$$

Subcase 2.2: let u be complex:

$$\dfrac{su \Rightarrow sc, su \qquad \dfrac{\dfrac{s \equiv t, \underline{as} \Rightarrow \underline{at} \qquad s \equiv t, \underline{at} \Rightarrow \underline{as}}{\mathbf{sc}, su, c \equiv u, s \equiv t \Rightarrow tu}\,(E) \qquad \dfrac{tc \Rightarrow tc, tu \qquad tc, tu \Rightarrow tu}{\underline{tc}, su, c \equiv u \Rightarrow tu}\,(\equiv\Rightarrow)}{c \equiv u, s \equiv t, su \Rightarrow tu}\,(\equiv\Rightarrow)}{s \equiv t, su \Rightarrow tu}\,(\equiv\Rightarrow E)$$

where sequent $s \equiv t, as \Rightarrow at$ is the case 1, already proven, and $s \equiv t, at \Rightarrow as$ is the case 3, which is provable exactly as case 1, according to the observation made by the end of the proof of lemma 2. The same applies to case 4 which is proved in the same way as case 2.

The induction step for non-atomic cases is provable as in FOL. □

Lemma 4. $GELO_m \vdash s \equiv t, \varphi[x/s] \Rightarrow \varphi[x/t]$

Proof. We need to demonstrate the same cases as in the previous lemma but now for atoms which have complex terms as both arguments.

Case 1 with all terms complex:

$$\dfrac{au \Rightarrow \underline{au} \qquad s \equiv t, as \Rightarrow \underline{at} \qquad \dfrac{bu \Rightarrow \underline{bu} \qquad cu \Rightarrow \underline{cu} \qquad \underline{bc} \Rightarrow bc}{\mathbf{us}, \underline{bu}, \underline{cu} \Rightarrow \underline{bc}}\,(\lambda \Rightarrow 2)}{\dfrac{s \equiv t, au, as, us \Rightarrow \mathbf{ut}}{s \equiv t, us \Rightarrow ut}\,(\lambda \Rightarrow 1)}\,(\Rightarrow \lambda)$$

where sequent $s \equiv t, as \Rightarrow at$ is the case 1 of the previous lemma.

Case 2. This time what matters is the character of s and t with u fixed complex. Since the case of s, t both simple was proven in the preceding lemma, there are three subcases:

2.1. all terms complex:

$$(\Rightarrow \lambda)\,\dfrac{s \equiv t, as \Rightarrow \underline{at} \qquad au \Rightarrow \underline{au} \qquad s \equiv t, su, \underline{bt}, \underline{ct} \Rightarrow \underline{bc}}{(\lambda \Rightarrow 1)\,\dfrac{s \equiv t, as, au, su \Rightarrow \mathbf{tu}}{s \equiv t, su \Rightarrow tu}}$$

where $s \equiv t, as \Rightarrow at$ is the case 1 of the previous lemma and $s \equiv t, su, bt, ct \Rightarrow bc$ is proven as follows:

$$\dfrac{bt \Rightarrow bs, bt \qquad \dfrac{ct \Rightarrow cs, ct \qquad \dfrac{bs \Rightarrow \underline{bs} \qquad cs \Rightarrow \underline{cs} \qquad \underline{bc} \Rightarrow bc}{cs, ct, \mathbf{su}, bs \Rightarrow bc}\,(\lambda \Rightarrow 2)}{bs, bt, s \equiv t, su, ct \Rightarrow bc}\,(\equiv\Rightarrow)}{s \equiv t, su, bt, ct \Rightarrow bc}\,(\equiv\Rightarrow)$$

2.2: s simple, t complex:

$$\dfrac{su \Rightarrow sa, su \qquad \dfrac{\dfrac{(\equiv\Rightarrow)\,\dfrac{ss \Rightarrow ss, st \qquad ss, st \Rightarrow st}{s \equiv t, ss \Rightarrow \underline{st}}}{(\Rightarrow \lambda)\,\dfrac{su \Rightarrow \underline{su} \qquad s \equiv t, ss, \underline{bt}, \underline{ct} \Rightarrow \underline{bc}}{s \equiv t, \underline{ss}, su \Rightarrow \mathbf{tu}}}}{\mathbf{sa}, su, s \equiv t \Rightarrow tu}\,(R)}{\dfrac{a \equiv u, s \equiv t, su \Rightarrow tu}{s \equiv t, su \Rightarrow tu}\,(\equiv\Rightarrow E)}\,(\equiv\Rightarrow)$$

where the rightmost sequent is proved as follows:

$$\dfrac{bt \Rightarrow bs, bt \qquad \dfrac{ct \Rightarrow cs, ct \qquad \dfrac{\dfrac{bc \Rightarrow bc}{\underline{sc}, bs \Rightarrow bc}\,(T)}{\mathbf{cs}, ct, \mathbf{ss}, bs \Rightarrow bc}\,(S)}{bs, bt, s \equiv t, ss, ct \Rightarrow bc}\,(\equiv\Rightarrow)}{s \equiv t, ss, bt, ct \Rightarrow bc}\,(\equiv\Rightarrow)$$

2.3. s complex, t simple:

$$\dfrac{au \Rightarrow ac, au \qquad \dfrac{\dfrac{D_1 \qquad D_2 \qquad \dfrac{tc \Rightarrow tc, tu \qquad tc, tu \Rightarrow tu}{\underline{tc}, c \equiv u \Rightarrow tu}\,(\equiv\Rightarrow)}{\mathbf{ac}, au, c \equiv u, s \equiv t, as, su \Rightarrow tu}\,(E)}{c \equiv u, s \equiv t, as, au, su \Rightarrow tu}\,(\equiv\Rightarrow)}{\dfrac{s \equiv t, as, au, su \Rightarrow tu}{s \equiv t, su \Rightarrow tu}\,(\equiv\Rightarrow E)}\,(\lambda \Rightarrow 1)$$

where D_1 is:

$$\dfrac{bt \Rightarrow bs, bt \qquad \dfrac{bs \Rightarrow \underline{bs} \qquad as \Rightarrow \underline{as} \qquad \underline{ba} \Rightarrow ba}{bs, bt, as, \mathbf{su} \Rightarrow ba}\,(\lambda \Rightarrow 2)}{s \equiv t, as, su, \underline{bt} \Rightarrow \underline{ba}}\,(\equiv\!\Rightarrow)$$

and D_2 is:

$$\dfrac{\dfrac{ba, as \Rightarrow bs}{as, ba \Rightarrow bs, bt}\,(\Rightarrow W) \qquad bs, bt \Rightarrow bt}{s \equiv t, as, \underline{ba} \Rightarrow \underline{bt}}\,(\equiv\!\Rightarrow)$$

where $ba, as \Rightarrow bs$ is a generalised transitivity cut-free provable by lemma 1. □

Proving *LL* for GELO_s, i.e. for the cases $\lambda x\varphi b$, is in some cases identical and in some other simpler than in the previous lemma, hence we omit the proof.

5. Adequacy of GELO

To show that all variants of GELO adequately characterise respective forms of ELO we demonstrate that different variants of *LA* are provable and that these rules are derivable if we use respective forms of *LA* as additional axioms. LA_1 was proved in [10] by means of the rules $(R), (S), (T), (E)$, which were in turn shown derivable in the presence of LA_1. These proofs are correct in GELO_w so we only need to prove LA_2:

Lemma 5. *$a\lambda x\psi \leftrightarrow \exists x(xa) \wedge \forall x(xa \rightarrow x\lambda x\psi) \wedge \forall xy(xa \wedge ya \rightarrow xy)$ is provable in $GELO_w$.*

$$\dfrac{\dfrac{a\lambda x\varphi \Rightarrow ab, a\lambda x\varphi \qquad \dfrac{\dfrac{aa \Rightarrow aa}{aa \Rightarrow \exists x(xa)}\,(\Rightarrow \exists)}{ab, a\lambda x\varphi \Rightarrow \exists x(xa)}\,(R)}{b \equiv \lambda x\varphi, a\lambda x\varphi \Rightarrow \exists x(xa)}\,(\equiv\!\Rightarrow)}{a\lambda x\varphi \Rightarrow \exists x(xa)}\,(\equiv\!\Rightarrow E)$$

$$\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow ac, a\lambda x\varphi \qquad \dfrac{bc \Rightarrow bc}{ac, a\lambda x\varphi, ba \Rightarrow bc}\,(T)}{c \equiv \lambda x\varphi, a\lambda x\varphi, ba \Rightarrow bc, b\lambda x\varphi}\,(\equiv\!\Rightarrow) \qquad bc, b\lambda x\varphi \Rightarrow b\lambda x\varphi}{c \equiv \lambda x\varphi, a\lambda x\varphi, ba \Rightarrow b\lambda x\varphi}\,(\equiv\!\Rightarrow)}{a\lambda x\varphi, ba \Rightarrow b\lambda x\varphi}\,(E)}{a\lambda x\varphi \Rightarrow ba \rightarrow b\lambda x\varphi}\,(\Rightarrow\!\rightarrow)}{a\lambda x\varphi \Rightarrow \forall x(xa \rightarrow x\lambda x\varphi)}\,(\Rightarrow \forall)$$

$$\dfrac{\dfrac{a\lambda x\varphi \Rightarrow ab, a\lambda x\varphi \qquad \dfrac{\dfrac{\dfrac{cd \Rightarrow cd}{ca, ad \Rightarrow cd}(T)}{aa, ca, da \Rightarrow cd}(S)}{ab, a\lambda x\varphi, ca, da \Rightarrow cd}(R)}{b \equiv \lambda x\varphi, a\lambda x\varphi, ca, da \Rightarrow cd}(\equiv\Rightarrow)}{b \equiv \lambda x\varphi, a\lambda x\varphi, ca \wedge da \Rightarrow cd}(\wedge\Rightarrow)}{b \equiv \lambda x\varphi, a\lambda x\varphi \Rightarrow ca \wedge da \rightarrow cd}(\Rightarrow\rightarrow)}{b \equiv \lambda x\varphi, a\lambda x\varphi \Rightarrow \forall xy(xa \wedge ya \rightarrow xy)}(\Rightarrow \forall)}{a\lambda x\varphi \Rightarrow \forall xy(xa \wedge ya \rightarrow xy)}(\equiv\Rightarrow E)$$

yield together by $(\Rightarrow \wedge)$ and $(\Rightarrow\rightarrow)$ the left-right implication of LA_2. The other part is proved as follows:

$$\dfrac{\dfrac{\dfrac{ba \Rightarrow ba \qquad \dfrac{\dfrac{b\lambda x\varphi \Rightarrow bc, b\lambda x\varphi \qquad D}{c \equiv \lambda x\varphi, ba, b\lambda x\varphi, \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(\equiv\Rightarrow)}{ba, b\lambda x\varphi, \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(\equiv\Rightarrow E)}{ba, ba \rightarrow b\lambda x\varphi, \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(\rightarrow\Rightarrow)}{ba, \forall x(xa \rightarrow x\lambda x\varphi), \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(\forall\Rightarrow)}{\exists x(xa), \forall x(xa \rightarrow x\lambda x\varphi), \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(\exists\Rightarrow)$$

where D is:

$$\dfrac{D_1 \qquad \dfrac{da \Rightarrow da}{ba, \underline{db} \Rightarrow \underline{da}}(T) \qquad \dfrac{ac \Rightarrow ac, a\lambda x\varphi \qquad ac, a\lambda x\varphi \Rightarrow a\lambda x\varphi}{\underline{ac}, c \equiv \lambda x\varphi \Rightarrow a\lambda x\varphi}(\equiv\Rightarrow)}{\mathbf{bc}, b\lambda x\varphi, c \equiv \lambda x\varphi, ba, \forall xy(xa \wedge ya \rightarrow xy) \Rightarrow a\lambda x\varphi}(E)$$

where D_1 is:

$$\begin{array}{r} (\Rightarrow\wedge)\ \dfrac{ba \Rightarrow ba \qquad da \Rightarrow da}{} \\ (\rightarrow\Rightarrow)\ \dfrac{ba, da \Rightarrow da \wedge ba \qquad db \Rightarrow db}{} \\ (\forall\Rightarrow)\ \dfrac{ba, da, da \wedge ba \rightarrow db \Rightarrow db}{ba, \forall xy(xa \wedge ya \rightarrow xy), \underline{da} \Rightarrow \underline{db}} \end{array}$$

□

As we already noticed it is quite an interesting fact that all that is needed to prove this axiom beyond rules from Fig. 1 (which were sufficient for proving LA_1) are the rules for $\equiv$; even the rules for β-conversion are not required.

The adequacy of GELO$_m$ (and GELO$_s$ too, as the only differences concern the character of t) follows from the next two lemmata:

Lemma 6. *The rules of Fig. 3 are derivable by means of the rules from Fig. 1 and LA_3 used as an additional axiomatic sequent.*

Proof. For $(\lambda \Rightarrow 1)$:

$$\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow a\lambda x\varphi \qquad at \Rightarrow at}{a\lambda x\varphi \to at, a\lambda x\varphi \Rightarrow at}(\to\Rightarrow) \qquad a\lambda x\varphi, at, \Gamma \Rightarrow \Delta}{a\lambda x\varphi \to at, a\lambda x\varphi, \Gamma \Rightarrow \Delta}(Cut)}{\forall x(x\lambda x\varphi \to xt), a\lambda x\varphi, \Gamma \Rightarrow \Delta}(\forall \Rightarrow)}{\forall x(x\lambda x\varphi \to xt), \exists x(x\lambda x\varphi), \Gamma \Rightarrow \Delta}(\exists \Rightarrow)$$

by two cuts with $\lambda x\varphi t \Rightarrow \forall x(x\lambda x\varphi \to xt), \lambda x\varphi t \Rightarrow \exists x(x\lambda x\varphi)$ which are derivable from LA_3. For $(\lambda \Rightarrow 2)$:

$$\dfrac{S \qquad \dfrac{\dfrac{\dfrac{\dfrac{\Gamma \Rightarrow \Delta, b\lambda x\varphi \qquad \Gamma \Rightarrow \Delta, c\lambda x\varphi}{\Gamma \Rightarrow \Delta, b\lambda x\varphi \wedge c\lambda x\varphi}(\Rightarrow \wedge) \qquad bc, \Gamma \Rightarrow \Delta}{b\lambda x\varphi \wedge c\lambda x\varphi \to bc, \Gamma \Rightarrow \Delta}(\to\Rightarrow)}{\forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy), \Gamma \Rightarrow \Delta}(\forall \Rightarrow)}{\lambda x\varphi t, \Gamma \Rightarrow \Delta}(Cut)$$

where S is $\lambda x\varphi t \Rightarrow \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy)$ which is derivable from LA_3. For $(\Rightarrow \lambda)$ first we prove:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow a\lambda x\varphi \qquad b\lambda x\varphi \Rightarrow b\lambda x\varphi}{a\lambda x\varphi, b\lambda x\varphi \Rightarrow a\lambda x\varphi \wedge b\lambda x\varphi}(\Rightarrow \wedge) \qquad ab, bt \Rightarrow at}{a\lambda x\varphi, b\lambda x\varphi, bt, a\lambda x\varphi \wedge b\lambda x\varphi \to ab \Rightarrow at}(\to\Rightarrow)}{a\lambda x\varphi, b\lambda x\varphi, bt, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow at}(\forall \Rightarrow)}{b\lambda x\varphi, bt, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow a\lambda x\varphi \to at}(\Rightarrow\to)}{b\lambda x\varphi, bt, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \forall x(x\lambda x\varphi \to xt)}(\Rightarrow \forall)$$

where the rightmost sequent is proved by lemma 1 (in case of LA_4 the application of (T) is enough).

Eventually by two cuts with the premisses of $(\Rightarrow \lambda)$ we obtain $\forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy), \Gamma \Rightarrow \Delta, \forall x(x\lambda x\varphi \to xt)$. Since from the leftmost and the rightmost premiss of $(\Rightarrow \lambda)$ we can derive $\Gamma \Rightarrow \Delta, \exists x(x\lambda x\varphi)$ and $\Gamma \Rightarrow \Delta, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy)$ respectively, by cuts with $\exists x(x\lambda x\varphi), \forall x(x\lambda x\varphi \to xt), \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t$ (derivable from LA_3) we get $\forall x(x\lambda x\varphi \to xt), \Gamma \Rightarrow \Delta, \lambda x\varphi t$. Two final cuts yield the conclusion of $(\Rightarrow \lambda)$. □

Lemma 7. *$\lambda x\varphi t \leftrightarrow \exists x(x\lambda x\varphi) \wedge \forall x(x\lambda x\varphi \to xt) \wedge \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy)$ is provable in $GELO_m$ with t complex, and in $GELO_s$ with t arbitrary.*

$$\dfrac{\dfrac{a\lambda x\varphi, at \Rightarrow a\lambda x\varphi}{a\lambda x\varphi, at \Rightarrow \exists x(x\lambda x\varphi)}(\Rightarrow \exists)}{\lambda x\varphi t \Rightarrow \exists x(x\lambda x\varphi)}(\lambda \Rightarrow 1)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow \underline{a\lambda x\varphi} \qquad b\lambda x\varphi \Rightarrow \underline{b\lambda x\varphi} \qquad \underline{ab}, bt \Rightarrow at}{a\lambda x\varphi, \underline{b\lambda x\varphi}, \underline{bt}, \mathbf{xt} \Rightarrow at}(\lambda \Rightarrow 2)}{a\lambda x\varphi, \mathbf{xt} \Rightarrow at}(\lambda \Rightarrow 1)}{\lambda x\varphi t \Rightarrow a\lambda x\varphi \to at}(\Rightarrow\to)}{\lambda x\varphi t \Rightarrow \forall x(x\lambda x\varphi \to xt)}(\Rightarrow \forall)$$

where the rightmost sequent is proved by lemma 1 (or by (T) in case of LA_4).

$$\dfrac{\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow \underline{a\lambda x\varphi} \qquad b\lambda x\varphi \Rightarrow \underline{b\lambda x\varphi} \qquad \underline{ab} \Rightarrow ab}{\mathbf{xt}, a\lambda x\varphi, b\lambda x\varphi \Rightarrow ab}\,(\lambda \Rightarrow 2)}{\lambda x\varphi t, a\lambda x\varphi \wedge b\lambda x\varphi \Rightarrow ab}\,(\wedge \Rightarrow)}{\lambda x\varphi t \Rightarrow a\lambda x\varphi \wedge b\lambda x\varphi \to ab}\,(\Rightarrow\to)}{\lambda x\varphi t \Rightarrow \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy)}\,(\Rightarrow \forall)$$

the above proofs yield the left-right part of LA_3 after application of $(\Rightarrow \wedge)$ and $(\Rightarrow\to)$. For the right-left implication we derive:

$$\dfrac{\dfrac{\dfrac{a\lambda x\varphi \Rightarrow a\lambda x\varphi \qquad at, a\lambda x\varphi, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}{a\lambda x\varphi, a\lambda x\varphi \to at, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}\,(\to\Rightarrow)}{a\lambda x\varphi t, \forall x(x\lambda x\varphi \to xt), \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}\,(\forall \Rightarrow)}{\exists x(x\lambda x\varphi), \forall x(x\lambda x\varphi \to xt), \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}\,(\exists \Rightarrow)$$

where the rightmost sequent is proved as follows:

$$\dfrac{\dfrac{a\lambda x\varphi \Rightarrow \underline{a\lambda x\varphi} \qquad at \Rightarrow \underline{at} \qquad \dfrac{\dfrac{\dfrac{b\lambda x\varphi \Rightarrow b\lambda x\varphi \qquad c\lambda x\varphi \Rightarrow c\lambda x\varphi}{b\lambda x\varphi, c\lambda x\varphi \Rightarrow b\lambda x\varphi \wedge c\lambda x\varphi \qquad bc \Rightarrow bc}\,(\Rightarrow \wedge)}{b\lambda x\varphi, c\lambda x\varphi, b\lambda x\varphi \wedge c\lambda x\varphi \to bc \Rightarrow bc}\,(\to\Rightarrow)}{\underline{b\lambda x\varphi}, \underline{c\lambda x\varphi}, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \underline{bc}}\,(\forall \Rightarrow)}{at, a\lambda x\varphi, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \mathbf{xt}}\,(\Rightarrow \lambda)$$

Together, these two lemmata guarantee the adequacy of GELO_m. For GELO_s the proof of the counterpart of lemma 6 is the same, and in the proof of the counterpart of lemma 7 only the last part (see the proof-tree above) requires more involved work:

$$\dfrac{\dfrac{D \qquad \dfrac{a\lambda x\varphi \Rightarrow ab, a\lambda x\varphi \qquad \dfrac{cb \Rightarrow cb}{ab, a\lambda x\varphi, ca \Rightarrow cb}\,(T)}{b \equiv \lambda x\varphi, a\lambda x\varphi, \underline{ca} \Rightarrow \underline{cb}}\,(\equiv\Rightarrow) \qquad \underline{bt}, b \equiv \lambda x\varphi \Rightarrow \lambda x\varphi t}{b \equiv \lambda x\varphi, \mathbf{at}, a\lambda x\varphi, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}\,(E)}{at, a\lambda x\varphi, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy) \Rightarrow \lambda x\varphi t}\,(\equiv\Rightarrow E)$$

where the rightmost leaf is provable as an instance of LL, and D is:

$$\begin{array}{r}(\Rightarrow \wedge)\\(\to\Rightarrow)\\(\forall \Rightarrow)\end{array}\dfrac{\dfrac{\dfrac{b \equiv \lambda x\varphi, cb \Rightarrow c\lambda x\varphi \qquad a\lambda x\varphi \Rightarrow a\lambda x\varphi}{b \equiv \lambda x\varphi, a\lambda x\varphi, cb \Rightarrow c\lambda x\varphi \wedge a\lambda x\varphi \qquad ca \Rightarrow ca}}{b \equiv \lambda x\varphi, a\lambda x\varphi, c\lambda x\varphi \wedge a\lambda x\varphi \to ca, cb \Rightarrow ca}}{b \equiv \lambda x\varphi, a\lambda x\varphi, \forall xy(x\lambda x\varphi \wedge y\lambda x\varphi \to xy), \underline{cb} \Rightarrow \underline{ca}}$$

where the leftmost leaf again is a provable instance of LL.

□

6. Cut Elimination Theorem

Before we focus on the proof of the cut elimination theorem let us note that for all variants of GELO the following result holds:

Lemma 8 (Substitution). *If* $\vdash_k \Gamma \Rightarrow \Delta$, *then* $\vdash_k \Gamma[a/b] \Rightarrow \Delta[a/b]$.

Proof. By induction on the height of a proof. The rules $(E), (\Rightarrow\equiv), (\equiv\Rightarrow E), (\lambda \Rightarrow 1), (\Rightarrow \lambda)$ may require similar relettering like $(\exists \Rightarrow)$ and $(\Rightarrow \forall)$. Note that the proof provides the height-preserving admissibility of substitution and that it is restricted to substitution of parameters for parameters only. □

Let us assume that all proofs are regular in the sense that every parameter a which is fresh by side condition on 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 the substitution lemma.

In [10] the cut elimination theorem was proved for GO and for GOP which covers GOI as its subsystem. Due to the construction of the rules from Fig. 2 and 3, this proof may be extended to GELO_w and GELO_m. It is enough to show that new rules are reductive in the sense of Ciabattoni [5]. Roughly: a pair of introduction rules $(\Rightarrow \star)$, $(\star \Rightarrow)$ for a constant $\star$ is reductive if an application of cut on cut formulae introduced by these rules may be replaced by the series of cuts made on less complex formulae, in particular on their subformulae. This feature of rules enables the reduction of the cut-degree in the proof of cut elimination. The latter notion, and the notion of proof-degree, is defined as follows:

1. The cut-degree $d\varphi$ is the complexity of the cut-formula φ, i.e. the number of connectives, quantifiers and lambda operators occurring in φ.
2. The proof-degree (dD) is the maximal cut-degree in D.

The reductivity of rules is sufficient for our aim on condition that no other rule in the system introduces the principal formula of such rules as active. It was the main reason for restricting $(R), (S), (T), (E)$ to atoms with simple terms as both arguments and for introducing the new rules for atoms with complex terms, as we explained in section 3. The separation of rules for different cases is the key to avoid the problems with elimination of cuts. Note that:

1. if st is strictly atomic, i.e. containing parameters only, it can be principal only in the antecedent of the right premiss of cut, due to $(R), (S), (T), (E)$;
2. if it is of the form $b\lambda x\varphi$, it can be principal in both premisses of cut but only via $(\Rightarrow \beta)$ and $(\beta \Rightarrow)$;
3. if it is of the form $\lambda x\varphi t$, it can be principal in both premisses of cut but only via $(\Rightarrow \lambda)$ and $(\lambda \Rightarrow 1)$ or $(\lambda \Rightarrow 2)$;
4. identity is principal in both premisses of cut only via $(\Rightarrow\equiv)$ and $(\equiv\Rightarrow)$;
5. relational atom is principal only in the succedent of the left premiss via $(\Rightarrow\equiv E)$.

The first and the fourth case are dealt with in the proof of cut elimination in [10]. The fifth case can be dealt with in a similar way as the first, by pushing cut up until it disappears either because in the opposite premiss the atom was introduced by $(W \Rightarrow)$ or it is an axiom. For the remaining cases it is sufficient to prove:

Lemma 9. 1. *The rules* $(\Rightarrow \beta)$ *with* $(\beta \Rightarrow)$ *are reductive in general;*
2. *Both* $(\Rightarrow \lambda)$ *with* $(\lambda \Rightarrow 1)$, *and* $(\Rightarrow \lambda)$ *with* $(\lambda \Rightarrow 2)$ *are reductive in GELO*$_m$.

Proof. The two rules of β-conversion are trivially reductive. It remains to show that the three rules for λ are reductive in GELO$_m$.

Let the right premiss of cut with the principal formula $\lambda x\varphi\lambda y\psi$ be derived by $(\Rightarrow \lambda)$. In case the right premiss is derived by $(\lambda \Rightarrow 1)$ we apply lemma 8 to its premiss to substitute the occurrences of fresh a with c, then we continue:

$$\dfrac{\Gamma \Rightarrow \Delta, c\lambda y\psi \qquad \dfrac{\Gamma \Rightarrow \Delta, c\lambda x\varphi \qquad c\lambda x\varphi, c\lambda y\psi, \Pi \Rightarrow \Sigma}{c\lambda y\psi, \Gamma, \Pi \Rightarrow \Delta, \Sigma}\,(Cut)}{\dfrac{\Gamma, \Gamma, \Pi \Rightarrow \Delta, \Delta, \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\,(C \Rightarrow), (\Rightarrow C)}\,(Cut)$$

Both cuts are of lower degree, hence both rules are reductive.

If the right premiss is derived by $(\lambda \Rightarrow 2)$ we apply lemma 8 to the rightmost premiss of the application of $(\Rightarrow \lambda)$ instead, to substitute the occurrences of fresh a, b with c, d respectively, then we continue:

$$\dfrac{\dfrac{\Pi \Rightarrow \Sigma, d\lambda x\varphi \qquad \dfrac{\Pi \Rightarrow \Sigma, c\lambda x\varphi \qquad c\lambda x\varphi, d\lambda x\varphi, \Gamma \Rightarrow \Delta, cd}{d\lambda x\varphi, \Gamma, \Pi \Rightarrow \Delta, \Sigma, cd}\,(Cut)}{\Gamma, \Pi, \Pi \Rightarrow \Delta, \Sigma, \Sigma, cd}\,(Cut) \qquad cd, \Pi \Rightarrow \Sigma}{\dfrac{\Gamma, \Pi, \Pi, \Pi \Rightarrow \Delta, \Sigma, \Sigma, \Sigma}{\Gamma, \Pi \Rightarrow \Delta, \Sigma}\,(C \Rightarrow), (\Rightarrow C)}\,(Cut)$$

Since all cuts are of lower degree, we are done. □

Combining lemma 9 with the results proved in [10] we obtain the cut elimination theorem for two of the considered systems:

Theorem 1. *Every proof in GELO*$_w$ *and GELO*$_m$ *can be transformed into a cut-free proof.*

What with GELO$_s$? Note that in GELO$_s$ cut may be performed also on the formulae of the form $\lambda x\varphi b$ by means of $(\Rightarrow \lambda)$ and $(\lambda \Rightarrow 1)$, or $(\Rightarrow \lambda)$ and $(\lambda \Rightarrow 2)$. In such cases we are not guaranteed that the transformed proofs contain cuts on formulae of lower degree. However, note that the transformations displayed above in each case replace cuts on formulae of the form $\lambda x\varphi b$ with cuts performed only on formulae of the form $b\lambda x\varphi$. It follows:

Lemma 10. *Every proof in GELO*$_s$ *can be transformed into a proof with no cuts on formulae of the form* $\lambda x\varphi b$.

Since such proofs may be dealt with as proofs in GELO$_w$ or GELO$_m$, we obtain:

Theorem 2. *Every proof in GELO*$_s$ *can be transformed into a cut-free proof.*

And as the consequence of these theorems we obtain:

Corollary 1. If $\vdash \Gamma \Rightarrow \Delta$ in GELO$_w$, GELO$_m$ or GELO$_s$, then it is provable in a proof which is closed under subformulae of $\Gamma \cup \Delta$ and atomic formulae with possibly new parameters.

7. Conclusion

ELO, similarly to LO, is not characterised semantically here. In fact, there are known controversies concerning the proper interpretation of quantifiers for LO (cf. [16, 22]), and for the time being we prefer to avoid these issues, since our aim is to provide a proof-theoretic analysis. However, note that referring to model-theoretic semantics is not the only option. Girard [6] emphasized that a cut-free system with the subformula property is complete in an internal sense. The idea of proof-theoretic semantics (see e.g. [23]) also shows that we can locate meaning in the well-defined rules. It seems that GELO satisfies these requirements sufficiently well. To strengthen this view it would be welcome to prove also the interpolation theorem for GELO, following the lines of proof of this result for GO and GOP in [12]. It is an open problem.

It was noticed in [10] that we can relatively easy obtain the intuitionistic version of GO (called GIO there) by restricting the sequents to single-succedent and changing slightly some of the rules. One may easily modify in this way also GOP from [10] and all variants of GELO introduced in this paper. The crucial point is to replace the present rule ($\equiv\Rightarrow$) with two variants (with Δ empty):

$$(\equiv\Rightarrow 1)\ \frac{\Gamma\Rightarrow \Delta, bt \quad bt, bs, \Gamma\Rightarrow \Delta}{t \equiv s, \Gamma\Rightarrow \Delta} \qquad (\equiv\Rightarrow 2)\ \frac{\Gamma\Rightarrow \Delta, bs \quad bt, bs, \Gamma\Rightarrow \Delta}{t \equiv s, \Gamma\Rightarrow \Delta}$$

It may be easily checked that all proofs we needed to establish adequacy and cut elimination, hold also in the intuitionistic versions, since, even in the places where ($\equiv\Rightarrow$) is applied, there is only one active formula in the succedent. This way we obtain for free also intuitionistic companions of considered calculi. Again, it must be emphasized that, similarly as in the case of 'classical' variants, the background logic is only apparently intuitionistic, since the terms are not restricted to individual ones, and the quantifiers have no existential import.

Because of the lack of space we were not concerned with the problem of expressivity of ELO. To simplify things we considered the calculus as built on the combination of the language of LO with simple language of pure FOL. However, it is possible to modify LO by admitting richer or different languages as the additional component. For example, even if we keep the first-order language, we may admit arbitrary terms as arguments of relational atoms. Or we may use a totally different language, like the languages of description logics, of QUARC, or of relational syllogistics. Of course, in case of mixing LO with other kinds of languages, it may be necessary to extend also the set of rules to cover specific logics different than FOL. Alternatively, we can consider a different approach to extending LO keeping the language of LO as the outer language and restricting the application of the other as the inner language admitted only inside complex terms. Again, because of the additional complications connected with more complex grammar we did not consider such an approach in this short paper. However, it is another promising field for further exploration.

Together with [10] this paper is meant as a theoretical foundation necessary for developing the novel tools in the field of automated deduction. Close resemblance of the structure of ELO to the structure of natural languages may help in the preparation of provers and proof assistants allowing for more direct and efficient processing of the reasoning tasks in natural languages. It is going to be one of the next steps in future research.

7.0.1. Acknowledgements.

I would like to thank the anonymous reviewers and Nils Kürbis for valuable comments. 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.

References

[1] Artale, A., Mazzullo, A., Ozaki, A., Wolter, F.: On Free Description Logics with Definite Descriptions. In: Bienvenu, M., Lakemeyer, G., Erdem, E. (eds.): Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, pp. 63–73. IJCAI Organization (2021).

[2] Ben-Yami, H.: Logic and Natural Language: On Plural Reference and Its Semantic and Logical Significance. Routledge, New York (2004).

[3] Borgida, A., Toman, D., Weddell, G.: On Referring Expressions in Query Answering over First Order Knowledge Bases. In: Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning, pp. 319–328. IJCAI Organization (2016).

[4] Braüner, T.: Hybrid Logic and its Proof-Theory. Springer, Cham (2011).

[5] Ciabattoni, A.: Automated Generation of Analytic Calculi for Logics with Linearity. In: Marcinkowski, J., Tarlecki, A. (eds.): CSL 2004, LNCS vol. 3210, pp. 503–517. Springer, Heidelberg (2004).

[6] Girard, J-Y.: From Foundations to Ludics. The Bulletin of Symbolic Logic. **9**(2), 131–168 (2003).

[7] Indrzejczak, A.: Free Logics are Cut-free. Studia Logica **109**(4) 859–886 (2021).

[8] Indrzejczak, A.: A Novel Approach to Equality. Synthese **199** 4749–4774 (2021).

[9] Indrzejczak, A.: Sequents and Trees. An Introduction to the Theory and Applications of Propositional Sequent Calculi. Birkhäuser (2021).

[10] Indrzejczak, A.: Leśniewski's Ontology – Proof-Theoretic Characterization. In: Blanchette, J., Kovacs, L., Pattinson, D. (eds.) Automated Reasoning, IJCAR 2022, LNAI vol. 13385, pp. 541–558. Springer, Heidelberg (2022).

[11] Indrzejczak, A.: Russellian definite description theory—a proof-theoretic approach. The Review of Symbolic Logic. **16**(2), 624–649 (2023).

[12] Indrzejczak, A.: Leśniewski's Ontology satisfies interpolation. Proceedings of AWPL, Sapporo (2024).

[13] Indrzejczak, A., Kürbis, N.: A Cut-Free, Sound and Complete Russellian Theory of Definite Descriptions. In: Ramanayake, R., Urban, J. (eds) Automated Reasoning with Analytic Tableaux and Related Methods. TABLEAUX 2023. Lecture Notes in Computer Science, vol. 14278, pp. 131–149. Springer, Cham (2023).

[14] Indrzejczak, A., Zawidzki, M.: When Iota meets Lambda. Synthese 201/72 (2023), DOI: 10.1007/s11229-023-04048-y.

[15] Iwanuś, B.: On Leśniewski's Elementary Ontology. Studia Logica **31**(1), 73–119 (1973).

[16] Küng, G., Canty, J.T.: Substitutional quantification and Leśniewskian quantifiers. Theoria **36**, 165–182 (1970).
[17] Leśniewski, S.: Collected Works. Vol. II. Surma, S., Srzednicki, J., Barnett, D.I. Kluwer/PWN (1992).
[18] Negri, S., von Plato, J.: Structural Proof Theory. Cambridge University Press, Cambridge (2001).
[19] Oliver, A., Smiley, T.: Plural Logic. Oxford University Press, Oxford (2016).
[20] Paśniczek, J.: The Logic of Intentional Objects. A Meinongian Version of Classical Logic. Kluwer, Dordrecht (1998).
[21] Pratt-Hatmann, I., Moss, L.,S.: Logics for the Relational Syllogistic. The Review of Symbolic Logic. **2**(4), 647–683 (2023).
[22] Rickey, F.: Interpretations of Leśniewski's Ontology. Dialectica **39**(3), 181–192 (1985).
[23] Schroeder-Heister, P.: Proof-theoretic Semantics. in: Stanford Encyclopedia of Philosophy 2012, https://plato.stanford.edu/entries/proof-theoretic-semantics/.
[24] Słupecki, J.: S. Leśniewski's Calculus of Names. Studia Logica **3**(1), 7–72 (1955).
[25] Sommers, F.: The Logic of Natural Language. Clarendon Press, Oxford (1982).
[26] Urbaniak, R.: Leśniewski's Systems of Logic and Foundations of Mathematics. Springer, Cham (2014).
[27] Waragai, T.: Ontology as a Natural Extension of Predicate Calculus with Identity Equipped with Description. Annals of the Japan Association for Philosophy of Science, 7(5), 233–250 (1990).

On Regular Relations in Parametric Array Theories

Rodrigo Raya[1]

[1]Max-Planck Institute for Software Systems, Kaiserslautern, Germany

Abstract

Parametric array theories are extensions of the quantifier-free theory of arrays with relations that hold component-wise. Unlike more expressive theories of arrays they allow specifying linear cardinality constraints on interpreted sets of indices, a notion close to the Härtig quantifier from model theory. We apply the notion of generalised power of a structure to study the satisfiability problem of parametric array theories. We show that reasoning about component-wise relations, linear cardinality constraints and succinct regular relations can be done efficiently by reduction to propositional satisfiability. We indicate how our techniques can be adapted to theories of trees.

Keywords

decision procedures, satisfiability modulo theories, symbolic automata

1. Introduction

Many abstractions in computer science and mathematics are naturally modelled as collections of objects of certain type. When addressing the problem of automatically verifying properties of such abstractions it becomes crucial to choose an adequate language in which to express the properties of interest.

Our research is influenced by work in the area of deductive software verification [5, 23]. Research in this area led to the development of specialised algorithms that determine the validity of formulas in restricted fragments of first-order logic. The resulting algorithms are today studied in the so-called satisfiability modulo theories (SMT) framework.

Starting with [21], first-order theories under the name of "array theories" have been studied and, along the years, new decidable fragments and applications have been found. Bradley [6] carried out a systematic exploration of a very expressive and decidable fragment known as the "array property fragment". Most notably, this fragment allowed to express the property of an array being ordered while having an efficiently decidable satisfiable problem under mild restrictions. Moreover, Bradley's work showed that minor variations to the fragment's syntax would led to undecidability, by reduction from Hilbert's tenth problem.

In spite of the above, and after Bradley's work, a family of decidable array theories has been used in the verification of so-called array-based systems [17, 26, 13, 27, 15]. This framework has been used to model sequential programs manipulating arrays and lists, as well as parameterised concurrent systems with local and shared variables [3, 1, 20]. These programs are essential in specialised computing scenarios such as data-base driven systems [4] or business processes [15].

In [30, 29, 31, 32], we have investigated the structure of these array theories. We have observed that they extend classical array theories with point-wise relations, which are defined as in the element theory for every component of the arrays. The conclusion of our investigation is that these point-wise relations are the essential difficulty when designing decision procedures for these theories. In particular, we have described how the satisfiability problem of these theories can be reduced in polynomial time to the satisfiability problem of the theory of a power structure [28] and that the latter admits an efficient procedure for eliminating existential quantifiers [30].

Our results are applicable to a variety of array theories from the literature [10, 16, 14, 1] to which we refer to as "parametric" array theories since they often allow to be instantiated with different index and element theories. An interesting feature of parametric array theories is that the componentwise

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

rraya@mpi-sws.org (R. Raya)

0000-0002-0866-9257 (R. Raya)

relations only require one universal quantifier to be expressed. For instance, one may define the addition of two arrays a and b as:

$$a + b = c \text{ if and only if for every } i \in I, a(i) + b(i) = c(i)$$

This is in contrast to other array theories such as the array property fragment [5], which allow properties using several related universally quantified indices. For instance, one may define in this fragment the property of an array being ordered:

$$a \text{ is ordered if and only if for every } i, j \in I, i \leq j \text{ implies } a[i] \leq a[j]$$

While "array properties" in [5] allow several universal quantifiers, this comes at the cost of severe syntactic restrictions. In contrast, parametric array theories offer the possibility of using linear cardinality relations on sets of indices [16, 14, 1, 30][1] as well as constraints on the sums of elements of array variables. These properties are inexpressible in the array property fragment.

These results motivate us to push further our investigation. In this paper, rather than moving to the design of algorithms for first-order theories (which would be justified by the incipient quantifier elimination method of [30]), we choose to further explore the possibilities in the quantifier-free setting which is the one relevant to the satisfiability modulo theories framework.

We take inspiration from the work of Feferman and Vaught [12], who introduced the notion of generalised power of a structure and motivated by the question of decidability of the weak monadic second order theory of one succesor (WS1S), raised by Tarski, discuss generalised powers with this theory of indices in the later sections of their paper. However, deciding WS1S is computationally intractable [34]. Thus, we present the definable relations of the theory in the form of regular expressions. It was proved by Büchi [7] that both formalisms are expressively equivalent. We refer to the relations on sets of indices induced by regular expressions or WS1S formulas as regular relations. [2]

There are several reasons that lead us to think that an extension of array theories with cardinality constraints and regular expressions is worthwhile investigating. First, this extends the work of Alberti, Ghilardi and Pagani [1] since it is well-known that WS1S is more expressive than Presburger arithmetic [35]. Second, this extension allows us to express properties of arrays such as those appearing in array folds logic [9]. While array folds logic only allows folding expressions over one array variable, this restriction does not appear in the fragment that we present. Third, a similar extension but without cardinality constraints has been considered concurrently to our work in [18].

Both in [18] and in our work, it seems that a non-trivial insight for the construction of the decision algorithm is needed. We point out to the reader that this insight is materialised in our paper in the partition variables introduced in Section 4.1. Indeed, since our specifications contain formulas whose interpretations, as sets of indices of the arrays, may overlap, it is essential to ensure that there exists a model adhering to the regular specification regardless of the overlaps in the semantic domain.

Unlike [18], we focus in the case of regular languages which should be more familiar to the readers. Nevertheless, we include a final section pointing out the main ingredients of the extension to regular tree languages. Also, for the sake of clarity, we have focused in cardinality constraints, but it should be clear that an extension to summation constraints is also possible.

Organisation of the paper. The rest of the paper is organised as follows. Section 2 describes generalised powers using specific theories of sets with cardinalities, theories describing their contents, and theories describing regular relations on the indices of these sets. Section 3 describes the satisfiability preserving encoding of arrays in generalised powers. Section 4 gives an algorithm that in polynomial time takes as input a generalised power structure specification and outputs an equivalent formula in

[1] A similar notion appears in the model theory literature under the name of Härtig's quantifier [2].

[2] We had considered regular expressions in our PhD thesis [29]. Here we consider regular expressions over first-order formulas. This formalism has been popularised in recent times under the name of symbolic regular expression and it can also be seen as motivated by Feferman-Vaught's results. This is what we mean by "succinct" regular relations. We also sometimes speak of "ordering" instead of regular relations since regular relations are precisely those expressible in the monadic theory of order [7].

the combination of the quantifier-free theory of Boolean algebra of sets with Presburger arithmetic and the alphabet's theory. Section 5 discusses the applicability of the technique in the setting of theories of trees, connecting to recent work. Section 6 concludes the paper.

2. Generalised powers

Let us start with the definition of generalised power structure as it is given in [12].

Definition 2.1. The generalised power $\mathscr{P}(\mathscr{M}, I)$ of a structure $\mathscr{M} = \langle M, ... \rangle$ is a structure whose carrier set is the set M^I of functions from the (possibly infinite) index set I to the carrier set M of the structure $\mathscr{M}$ and whose relations are interpreted as sets of the form

$$\{(a_1, \dots, a_n) \in (M^I)^n \mid \Phi(S_1, \dots, S_k)\}$$

where n is a natural number, Φ is a Boolean algebra expression over $\mathscr{P}(I)$ using the symbols $\subseteq$, $\cup$, $\cap$ or $\cdot^c$ and each set variable S is interpreted as

$$S = \{i \in I \mid \theta(a_1(i), \dots, a_n(i))\}$$

where θ is a formula in the first-order theory of $\mathscr{M}$.

In the following, we will use the term "arrays" for the functional elements in the carrier from a generalised power $\mathscr{P}(\mathscr{M}, I)$, the term "elements" for the members of the carrier set of the structure $\mathscr{M}$ and the term "indices" for the members of the set I. We will use the notation $a(i)$ when we want to emphasize the algebraic perspective and the notation $a[i]$ when we want to emphasize the connection to array theories. In particular, we will use the latter notation when describing how to translate from parametric array theories to generalised powers.

Nothing prevents us from considering set interpretations of the form

$$S = \{i \in I \mid \psi(i)\}$$

where ψ is a formula that refers only to indices in the set I. In fact, this direction is pursued in [1] where a fragment of the theory of arrays is investigated that corresponds to a generalised power whose set interpretations conflate both the theory of indices and the theory of elements using the quantifier-free fragment of Presburger arithmetic to refer to both. We will use different set interpretations for indices and elements. We will use relations of the following form:

$$F(S_1, \dots, S_k) \wedge R(S_1, \dots, S_k) \wedge C(a_1, \dots, a_k) \tag{1}$$

Here F specifies linear cardinality constraints on the shared set variables $S_1, \dots, S_k$. R specifies the regular relations on the set of indices $S_1, \dots, S_k$. Finally, C specifies the componentwise relations on the arrays $a_1, \dots, a_k$. The precise description of Formula (1) occupies the rest of the section.

2.1. Sets of indices

Formulas F, R and C in (1) use variables $S_1, \dots, S_k$ representing subsets of an index set I. This is explicitly shown in (1) for F and R. The variables $S_1, \dots, S_k$ are omitted from formula C to emphasize the role of componentwise relations on array variables. Thus, the variables $S_1, \dots, S_k$ are used to combine the three theories. This approach to theory combination was pioneered in [37]. As we focus on arrays, $I = \mathbb{N}$. Generalisations to trees are also possible and in that case $I = \{0, 1\}^*$.

2.2. Linear cardinality constraints on the sets of indices

$F(S_1, \ldots, S_k)$ is a formula in the quantifier-free theory of Boolean algebra with Presburger arithmetic (QFBAPA) [25]. The syntax of QFBAPA is given in Figure 1. The top-level symbol F presents the Boolean structure of the formula, A stands for the atomic formulas which can be either Boolean algebra expressions on the sets denoted by the symbol B or Presburger arithmetic restrictions on numbers denoted by the symbol T. The operator dvd stands for the divisibility relation, which is used to ensure that the quantifier-free fragment has the same expressive power as the full first-order theory of Boolean algebra with Presburger arithmetic (BAPA)[24]. $\mathcal{U}$ represents the universal set I. Lowercase x and k represent Boolean and integer variables respectively. The remaining interpretations are standard in the respective theories (Boolean algebra of sets or Presburger arithmetic).

$$
\begin{aligned}
F &::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \\
A &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid T_1 = T_2 \mid T_1 \leq T_2 \mid K \text{ dvd } T \\
B &::= x \mid \emptyset \mid \mathcal{U} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \\
T &::= k \mid K \mid T_1 + T_2 \mid K \cdot T \mid |B| \\
K &::= \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots
\end{aligned}
$$

Figure 1: QFBAPA's syntax

Example 2.1. An example of QFBAPA formula is $|A| > 1 \wedge A \subseteq B \wedge |B \cap C| \leq 2$.

2.3. Componentwise relations on arrays

$C(a_1, \ldots, a_k)$ is a formula specifying componentwise relations. It does so with set interpretations of the form:

$$S_j = \{i \in I \mid \phi_j(\overline{a}(i), \overline{c})\}$$

where $\overline{a}$ denotes a tuple of array variables, $\overline{c}$ denotes a tuple of constants from the element theory and ϕ_j is a formula of the element theory. As in Definition 2.1, $a(i)$ denotes the i-th position of array a and $\overline{a}(i) = (a_1(i), \ldots, a_k(i))$.

Example 2.2. The equality between two arrays a_1 and a_2 can be written in the fragment of (1) as:

$$S = \{i \in I \mid a_1(i) = a_2(i)\} \wedge |S| = |\mathcal{U}|$$

where $\mathcal{U}$ as explained above, represents the universal set I.

2.4. Regular relations on the set of indices

$R(S_1, \ldots, S_k)$ is a formula specifying regular relations in the set of indices. For instance, the array could be specified by the symbolic regular expression:[3]

$$\phi_1(e, \overline{c})(\phi_1(e, \overline{c}) \vee \phi_2(e, \overline{c}))^* \phi_3(e, \overline{c}) \tag{2}$$

This specifies that the first element e of the array satisfies the formula $\phi_1(e, \overline{c})$, then there is a sequence of zero or more elements e satisfying either $\phi_1(e, \overline{c})$ or $\phi_2(e, \overline{c})$ and the last element e satisfies the formula $\phi_3(e, \overline{c})$. Each instantiation of e is different for each witness, while the value of the parameters in $\overline{c}$ must be the same for the whole array. However, this approach conflates the specifications of the indices and the specifications of the elements of the arrays.

[3]There are several possibilities to write regular relations. Here we use regular expressions for economy of notation.

Let us instead write a symbolic version of the regular expression above

$$S_1(S_1 \vee S_2)^* S_3 \tag{3}$$

To relate this symbolic expression and the theory of the elements, we let t be a sequence of bit-strings $t \in (\{0,1\}^3)^*$ and define

$$\begin{aligned} S_1 &= \{i \in I \mid t_1(i) = 1\} \wedge S_1 = \{i \in I \mid \phi_1(a(i), \bar{c})\} \\ S_2 &= \{i \in I \mid t_2(i) = 1\} \wedge S_2 = \{i \in I \mid \phi_2(a(i), \bar{c})\} \\ S_3 &= \{i \in I \mid t_3(i) = 1\} \wedge S_3 = \{i \in I \mid \phi_3(a(i), \bar{c})\} \end{aligned} \tag{4}$$

where t_1, t_2 and t_3 denote, respectively, the first, second and third rows of the sequence and $t_j(i)$ denotes the i-th position in t_j.

Then, satisfiability of (2) is equivalent to satisfiability of

$$\exists t \in (\{0,1\}^3)^*. t \vDash S_1(S_1 \vee S_2)^* S_3 \wedge (4) \tag{5}$$

where $t \vDash R(S_1, \dots, S_k)$ means that the bit-string sequence t satisfies propositionally the regular expression $R(S_1, \dots, S_k)$, that is, there is a word w of propositional formulas over the variables S_1, S_2 and S_3 generated by R such that for each i, $t(i) \vDash w(i)$ propositionally.

Example 2.3. A bit-string sequence t belonging to the language of the symbolic regular expression $S_1(S_1 \vee S_2)^* S_3$ is the following

$$\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}$$

The values of its rows are respectively $t_1 = 11000$, $t_2 = 10110$ and $t_3 = 00011$.

It satisfies the word of propositional formulas $S_1(S_1 \vee S_2)(S_1 \vee S_2)(S_1 \vee S_2)S_3$ which is generated by $S_1(S_1 \vee S_2)^* S_3$.

We call the bit-string sequences t regular tables or simply tables and write $T(R)$ for the set of all tables satisfying the symbolic regular expression R.

3. Encoding of arrays

Let us briefly mention how would the terms of an array theory, most importantly, array "reads" and "writes", be written in the language of generalised powers, while preserving the satisfiability of formulas.

A componentwise specification is written as in Example 2.2.

An array read is a functional term $a[j]$. To encode this term in generalised powers, we introduce the set variable J representing the singleton set $\{j\}$ and require that $|J| = 1$. We then introduce an element theory variable a_j and require that $\{i \in I \mid a(i) = a_j\} \supseteq J$.

An array write is a functional term $store(a, j, v)$. To encode this term in generalised powers, we introduce a new variable b to stand for the term $store(a, j, v)$ and require that $\{i \in I \mid b(i) = v\} \supseteq J$ and $\{i \in I \mid a(i) = b(i)\} \supseteq \mathcal{U} \setminus J$.

Example 3.1. Consider the array formula from [5].

$$i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge store(store(a, i_1, v_1), i_2, v_2)[j] \neq a[j]$$

For each index variable j, i_1, i_2, we introduce set variables J, I_1, I_2 and impose that $I_1 = J$, $I_1 \neq I_2$ and $|I_1| = |I_2| = |J|$.

The term $a[j] = v_1$ is translated into $\{i \in I \mid a(i) = v_1\} \supseteq J$.

We introduce the array variables b for $store(a, i_1, v_1)$ and c for $store(b, i_2, v_2)$. We can then encode the fourth conjunct as $\{i \in I \mid c(i) \neq a(i)\} \supseteq J$. The store operators are encoded as indicated above.

The resulting formula is in the theory of the generalised power and is equisatisfiable to the original.

4. Deciding generalised powers

Let us now give a method to decide satisfiability of formulas of the form (1). These are of the following form:

$$F(S_1, \ldots, S_k) \wedge \exists t \in T(R). \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid \phi_i(\bar{a}(n), \bar{c}) \} = \{ n \in \mathbb{N} \mid t_i(n) = 1 \} \tag{6}$$

The satisfiability problem requires showing that the following formula is true:

$$\begin{aligned} &\exists a \in \mathscr{D}^*, t \in T(R), \bar{c}. \\ &\qquad F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid \phi_i(\bar{a}(n), \bar{c}) \} = \{ n \in \mathbb{N} \mid t_i(n) = 1 \} \end{aligned} \tag{7}$$

where $\mathscr{D}$ is the domain of the array elements.

We show how to compute in polynomial time a formula in the combination of QFBAPA and the existential fragment of the first-order theory of the domain $\mathscr{D}$ such that Formula (7) is equivalent to the computed formula.

Theorem 4.1. There is a polynomial time algorithm that given formula (7) computes an equivalent formula (8) in the combination of QFBAPA and the existential fragment of the first-order theory of $\mathscr{D}$, $Th_{\exists^*}(\mathscr{D})$.

4.1. Construction of the equivalent formula

The equivalent formula has three parts. The first is an existential prefix shared between both QFBAPA and $Th_{\exists^*}(\mathscr{D})$

$$\exists N \leq p(|F|), \exists s \in [m], \sigma : [s] \hookrightarrow [m], \exists \beta_1, \ldots, \beta_N \in \{0,1\}^k.$$

where N is a natural number, p is a polynomial, $|F|$ is the number of symbols used to write F, $[n] := \{1, \ldots, n\}$ abbreviates the set of the first n natural numbers, m is the number of propositional formulas used in M_S, σ is an injection from $[s]$ to $[m]$ and k is the number of set variables used in F.

The second part of the formula is an expression in the theory $Th_{\exists^*}(\mathscr{D})$

$$C(\beta_1, \ldots, \beta_N) := \exists \bar{c}. \bigwedge_{j=1}^{N} \exists e \in D. \phi^{\beta_j}(e, \bar{c})$$

where we use the notation that given the list $\phi_1, \ldots, \phi_k$ of formulas specifying the elements in Formula 6 and given a bit-string $\beta \in \{0,1\}^k$, $\phi^\beta := \bigwedge_{i=1}^{k} \phi_i^{\beta(i)}$.

The third part of the formula is in QFBAPA

$$\begin{aligned} H(c_1, \ldots, c_k, \ldots) := &\rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ &\bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P_i \end{aligned}$$

where ρ is a formula in the existential fragment of Presburger arithmetic of size linear in the size of the regular expression R. ρ describes the Parikh image of R. The description of the Parikh image in terms of linear-size existential Presburger arithmetic formulas is based on a result from [33].

Definition 4.1 (Parikh Image)**.**
The Parikh image of a symbolic regular expression R using propositional letters $L_1, \ldots, L_m$ over the variables $S_1, \ldots, S_k$ is the set

$$\mathsf{Parikh}(R) = \{(|\bar{s}|_{L_1}, \ldots, |\bar{s}|_{L_m}) \mid \bar{s} \in M_S(L_1, \ldots, L_m)\}$$

where $\bar{s}$ is a word of propositional formulas and $|\bar{s}|_{L_i}$ is the number of occurrences of L_i in the symbolic table $\bar{s}$.

Example 4.1. Continuing Example 2.3, we had a symbolic regular expression $S_1(S_1 \vee S_2)^* S_3$ and a word of propositional formulas $S_1(S_1 \vee S_2)(S_1 \vee S_2)(S_1 \vee S_2)S_3$. Thus, one vector in the Parikh image is $(1, 3, 1)$. In general, the Parikh image of $S_1(S_1 \vee S_2)^* S_3$ would contain the vectors $(1, k, 1)$ for each $k \in \mathbb{N}$.

Continuing with the description of the third part of the formula, F is the QFBAPA term in Formula 6, $p_\beta := \cap_{i=1}^{k} S_i^{\beta(i)}$ where $\beta \in \{0,1\}^k$, $p_L := \cup_{\beta \vDash L} p_\beta$ where $\vDash$ is the propositional satisfaction relation and $\beta \in \{0,1\}^k$, $|\cdot|$ denotes the cardinality of the argument set expression, $\dot\cup_{i \in I} S_i$ is the set $\cup_{i \in I} S_i$ where we emphasize that each pair of sets S_i, S_j are disjoint.

Shortening tuples of variables with an overline, we write Formula (8) as

$$\exists N \leq p(|F|), \exists s \in [m], \sigma : [s] \hookrightarrow [m], \exists \overline{\beta} \in \{0,1\}^k . C(\overline{\beta}) \wedge \exists \overline{c}, \overline{P} . H(\overline{c}, \overline{P}, \overline{\beta}) \tag{8}$$

This formula can be computed in polynomial time.

Intuitively, the partition variables P_i determine which propositional formula generated each value of the arrays accepted, so that, even if these formulas overlap, a model corresponding to a run of the automaton can be rebuilt. The reason why we need to check the existence of only one witness per elementary Venn region follows from the fact that we can "replicate" this witness in each of the indices that satisfied the corresponding formula ϕ^β (see also [30, 31]).

4.2. Proof of the theorem

We prove that Formula (6) and Formula (13) are equivalent.

$\Rightarrow$) If Formula (7) is true, then there are sets $S_1, \ldots, S_k$, a finite array a and a table $t \in T(R)$ such that

$$F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid \phi_i(\overline{a}(n), \overline{c}) \,\} = \{\, n \in \mathbb{N} \mid t_i(n) = 1 \,\} \tag{9}$$

Let $\bar{s}$ be the symbolic table corresponding to t, that is, the table made of the propositional formulas generated by $R(S_1, \ldots, S_k)$ such that t satisfies $\bar{s}$.

Define $c_i := |\bar{s}|_{L_i}$ for $i \in \{1, \ldots, m\}$ as the number of occurrences of L_i in the symbolic table $\bar{s}$, $s = |\{\, i \mid c_i \neq 0 \,\}|$, σ mapping the indices in $[s]$ to the indices i of the terms for which c_i is non-zero and $P_i = \left\{\, n \in \mathbb{N} \;\middle|\; \bar{s}(n) = L_{\sigma(i)} \,\right\}$. From the equalities $S_i = \{\, n \in \mathbb{N} \mid t_i(n) = 1 \,\} = \{\, n \in \mathbb{N} \mid \phi_i(\overline{a}(n), \overline{c}) \,\}$ in (9), we can show that the following holds

$$\begin{aligned} &\rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ &\bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \dot\cup_{i=1}^{s} P_i \end{aligned} \tag{10}$$

Formula (10) is in QFBAPA. Following the procedure from [25], we eliminate Boolean algebra expressions and the cardinality operator yielding a system of equations of the form

$$\exists k_1, \ldots, k_p . G \wedge \sum_{j=0}^{2^k - 1} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{11}$$

where G is the existential Presburger arithmetic formula that results from (10) after the elimination and $l_{\beta_j} = |p_{\beta_j}|$.

We remove from the sum those terms corresponding to elementary Venn regions β such that $l_\beta = 0$. This includes regions whose associated formula in the interpreted Boolean algebra $\phi^\beta(a, \bar{c})$ is unsatisfiable,

and regions corresponding to bit-strings not occurring in t. This transformation leaves a reduced set of indices $\mathscr{R}$ participating in the sum:

$$\exists k_1, \ldots, k_p.G \wedge \sum_{\beta \in \{\beta_1, \ldots, \beta_N\} \subseteq \mathscr{R}} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{12}$$

We now give the key auxiliary result from [25] proved in [11].

Definition 4.2. Given a subset $S \subseteq \mathbb{R}^n$, the integer conic hull of S is the set

$$int_{cone}(S) = \left\{ \sum_{i=1}^{t} \lambda_i s_i \Big| t \geq 0, s_i \in S, \lambda_i \in \mathbb{N} \right\}$$

Theorem 4.2. Let $X \subseteq \mathbb{Z}^n$ be a finite set of integer vectors, $b \in int_{cone}(X)$ and $M = \max_{\mathbf{x} \in X} \|\mathbf{x}\|_\infty = \max_{\mathbf{x} \in X} \max\{|x_1|, \ldots, |x_n|\}$ where $|x|$ denotes the absolute value of the number x. There exists a subset $X' \subseteq X$ such that $b \in int_{cone}(X')$ and $|X'| \leq 2n \log_2(4nM)$ where $|X'|$ denotes the cardinality of the set X'.

Using Theorem 4.2, there is a polynomial family of Venn regions $\beta_1, \ldots, \beta_N$ and corresponding cardinalities $l'_{\beta_1}, \ldots, l'_{\beta_N}$ such that:

$$\exists k_1, \ldots, k_p.G \wedge \sum_{\beta \in \{\beta_1, \ldots, \beta_N\} \subseteq \mathscr{R}} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l'_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{13}$$

We can assume that each cardinality variable l'_β is non-zero, since otherwise, we can remove it from the sum. With this assumption, $l'_{\beta_1}, \ldots, l'_{\beta_N}$ lists the cardinalities of a model of (10) which defines the cardinality of the elementary Venn region $S_1'^{\beta_1} \cap \ldots \cap S_k'^{\beta_k}$ to be equal to l'_β if $\beta \in \{\beta_1, \ldots, \beta_N\}$ and zero otherwise. In particular, we have that the following formula, corresponding to the subformula H in (8), holds

$$\begin{gathered} \rho(c_1, \ldots, c_m) \wedge F(S'_1, \ldots, S'_k) \wedge \bigwedge_{i=1}^{k} P'_i \subseteq p'_{L_{\sigma(i)}} \wedge \\ \bigwedge_{i=1}^{s} |P'_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S'_i = \cup_{i=1}^{m} p'_{L_i} = \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P'_i \end{gathered} \tag{14}$$

For each $\beta \in \{\beta_1, \ldots, \beta_N\}$, the formula $\exists e.\phi^\beta(e, \bar{c})$ is true, since $l'_\beta > 0$. Thus, the subformula C in (8) is also true.

$\Leftarrow$) If Formula (8) is true, then there is a natural number $N \leq p(|F|)$ where p is a polynomial, $s \in [m]$, $\beta_1, \ldots, \beta_N \in \{0,1\}^k$, $c_1, \ldots, c_m \in \mathbb{N}$ and sets $S_1, \ldots, S_k, P_1, \ldots, P_s$ such that

$$\begin{gathered} \exists \bar{c}. \bigwedge_{j=1}^{N} \exists e.\phi^{\beta_j}(e, \bar{c}) \wedge \rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ \wedge \bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i = \cup_{i=1}^{N} p_{\beta_i} \end{gathered} \tag{15}$$

From (15) and the definition of ρ, follows that there is a symbolic table $\bar{s}$ generated by the symbolic regular expression R such that $|\bar{s}|_{L_i} = c_i$ for each $L_i \in \{L_1, \ldots, L_m\}$ occurring in $\bar{s}$. Moreover, from

$$\bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i$$

we have that all the sets p_{L_i} are empty except $p_{L_{\sigma(1)}}, \ldots, p_{L_{\sigma(s)}}$.

From the subformula

$$\cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)}$$

follows that we may define P_i to consist of the indices of $\bar{s}$ labelled with the formula $L_{\sigma(i)}$ for each $i \in \{1, \ldots, s\}$. This is because the values in P_i are guaranteed to satisfy the formula $L_{\sigma(i)}$. Note that the values in P_i could satisfy other formulas $L_{\sigma(j)}$. However, the role of the variables P_i is to determine which formula of the regular expression generated the values satisfying $L_{\sigma(i)}$ regardless of whether the witnesses satisfied other formulas too.

From the subformula

$$\exists \bar{c}. \bigwedge_{j=1}^{N} \exists e. \phi^{\beta_j}(e, \bar{c}) \wedge \dot{\cup}_{i=1}^{s} P_i = \cup_{i=1}^{N} p_{\beta_i}$$

it follows that there exist values $e_{\beta_1}, \ldots, e_{\beta_N}$ satisfying the formulas $\phi^{\beta_1}(e, \bar{c}), \ldots, \phi^{\beta_N}(e, \bar{c})$ and moreover, all the elements in p_{β_j} belong to a single set P_i.

We define a table t by substituting in $\bar{s}$ the indices in P_i by the values β_j such that $p_{\beta_j} \subseteq P_i$. Similarly, we build an array a by substituting in $\bar{s}$ the indices in P_i by the values e_{β_j} such that $p_{\beta_j} \subseteq P_i$.

Observe that $F(S_1, \ldots, S_k)$ holds by assumption. Moreover,

$$S_j = \cup_{\{i \mid \beta_i(j)=1\}} p_{\beta_i} = \{\, n \in \mathbb{N} \mid t_j(n) = 1 \,\}$$
$$S_j = \cup_{\{i \mid \beta_i(j)=1\}} p_{\beta_i} = \{\, n \in \mathbb{N} \mid \phi_j(\bar{a}(n), \bar{c}) \,\}$$

Thus, Formula (7) is true. □

4.3. Computational complexity of the combination

Note that Theorem 4.1 also allows to assert at once the complexity of the underlying logical theory.

Corollary 4.1. Let $\mathscr{C}$ be the complexity class to which the satisfiability problem of $Th_{\exists^*}(\mathscr{D})$ belongs. If $\mathscr{C} = \mathrm{P}$ then the satisfiability problem of formulas of the form (6) is in NP. If $\mathscr{C} \supseteq \mathrm{NP}$ then the satisfiability problem of formulas of the form (6) is in $\mathscr{C}$.

5. The case of trees

One can adapt the techniques of Section 4 to the case when the regular specification is given by a parametric tree automaton, thus extending the results of [18]. The main difference with the procedure of Section 4 is that in the case of parametric tree automata one needs to compute the Parikh image of a regular tree language. This is done in a completely analogous way as it is done in Definition 4.1 for parametric finite automata. Note that it is easy to convert from non-deterministic top-down to non-deterministic bottom-up tree automata [8, Theorem 1.6.1]. One can then use the observation of Klaedtke and Rueß [22, Lemma 17] which allows to reduce the problem to the computation of the Parikh image of a context-free grammar. Finally, [36] says that the Parikh image of a context-free grammar can be described by a linear-sized existential Presburger arithmetic formula.

6. Conclusion

We have shown how to extend decision procedures for satisfiability of parametric array fragments with regular constraints. In terms of quantifiers, this shows how to simultaneously support Härtig's quantifiers and WS1S second-order quantification. Our techniques extend previous results of Alberti, Ghilardi and Pagani [1] since the relations expressible in WS1S extend those expressible in Presburger

arithmetic. They also extend recent results in the literature [18], which did not handle the cardinality operator.

Our work mixes ideas from decision algorithms for three different logical theories: quantifier-free BAPA [25], combinations of BAPA with WS1S and WS2S [37] and existential fragment of power structures [30]. However, a crucial technical difficulty has been overcome to make the combination work. This difficulty stems from the fact that while Büchi's automata are over finite alphabets, the corresponding automata (or equivalently, regular expressions) that appear in the Feferman-Vaught framework use first-order formulas in their transitions, since set variables are interpreted. We demonstrated, as our colleagues [18], that one can still use the Parikh image of this symbolic automata to combine theories. Since one is now counting formulas rather than symbols from a finite alphabet, and formulas can overlap in the semantic domain, it becomes necessary to indicate to the QFBAPA constraint which transition of the automaton produced each index. We achieved this by introducing a partition of the sets of indices of the array, where each part corresponds to the indices generated by a given transition.

The implementation of the algorithm could be achieved mixing the techniques of ARCA-SAT [1] with software computing the Parikh image of regular expressions and context-free grammars. For the latter, there exist several implementations, we mention for instance [19], where a fix to the construction original of Verma et alii [36], is also described.

Possible applications of the decision procedure include automatic verification of array manipulating programs in deductive verification systems and model checking of distributed protocols. Nevertheless, due to the number of ideas that are combined in this work, we would not be surprised if further applications are found in the future.

Acknowledgements

The author wishes to express his gratitude to Viktor Kunčak for suggesting us the application of the methods of our thesis to symbolic automata. Anthony Lin and Oliver Markgraf provided useful remarks on the final presentation of the results in this paper. Research supported by the Swiss NSF Project P500PT_222338

References

[1] F. Alberti, S. Ghilardi, and E. Pagani. Cardinality constraints for arrays (decidability results and applications). *Formal Methods in System Design*, 51(3):545–574, December 2017. doi:10.1007/s10703-017-0279-6.

[2] J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Perspectives in Logic. Cambridge University Press, Cambridge, 2017. doi:10.1017/9781316717158.

[3] Roderick Bloem, Ayrat Khalimov, Swen Jacobs, Igor Konnov, Helmut Veith, Josef Widder, and Sasha Rubin. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-031-02011-7.

[4] Mikołaj Bojańczyk, Luc Segoufin, and Szymon Toruńczyk. Verification of database-driven systems via amalgamation. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, PODS '13, pages 63–74, New York, NY, USA, June 2013. Association for Computing Machinery. doi:10.1145/2463664.2465228.

[5] Aaron Bradley and Zohar Manna. *Calculus of computation: decision procedures with applications to verification*. Springer, Berlin, 2007. OCLC: ocn190764844.

[6] Aaron R Bradley. *Safety analysis of systems*. PhD thesis, Stanford University, May 2007.

[7] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. doi:10.1002/malq.19600060105.

[8] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Loding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. INRIA, 2008.

[9] Przemysław Daca, Thomas A. Henzinger, and Andrey Kupriyanov. Array Folds Logic. In *Computer Aided Verification*, Lecture Notes in Computer Science, pages 230–248, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-41540-6_13.

[10] Leonardo de Moura and Nikolaj Bjorner. Generalized, efficient array decision procedures. In *2009 Formal Methods in Computer-Aided Design*, pages 45–52, Austin, TX, November 2009. IEEE. doi:10.1109/FMCAD.2009.5351142.

[11] Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, September 2006. doi:10.1016/j.orl.2005.09.008.

[12] S. Feferman and R. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47(1):57–103, 1959.

[13] Paolo Felli, Alessandro Gianola, and Marco Montali. SMT-based Safety Checking of Parameterized Multi-Agent Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6321–6330, May 2021. Number: 7. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16785, doi:10.1609/aaai.v35i7.16785.

[14] Klaus Freiherr von Gleissenthall. *Cardinalities in Software Verification*. PhD thesis, Technische Universität München, Fakultät für Informatik, 2016.

[15] Alessandro Gianola. *Verification of Data-Aware Processes via Satisfiability Modulo Theories*, volume 470 of *Lecture Notes in Business Information Processing*. Springer Nature Switzerland, Cham, 2023. doi:10.1007/978-3-031-42746-6.

[16] Klaus v. Gleissenthall, Nikolaj Bjørner, and Andrey Rybalchenko. Cardinalities and universal quantifiers for verifying parameterized systems. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, pages 599–613, New York, NY, USA, June 2016. Association for Computing Machinery. doi:10.1145/2908080.2908129.

[17] Arie Gurfinkel, Sharon Shoham, and Yuri Meshman. SMT-based verification of parameterized systems. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 338–348, New York, NY, USA, November 2016. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/2950290.2950330, doi:10.1145/2950290.2950330.

[18] Matthew Hague, Artur Jeż, and Anthony W. Lin. Parikh's Theorem Made Symbolic. *Proceedings of the ACM on Programming Languages*, 8(POPL):65:1945–65:1977, January 2024. URL: https://dl.acm.org/doi/10.1145/3632907, doi:10.1145/3632907.

[19] Matthew Hague and Anthony Widjaja Lin. Synchronisation- and Reversal-Bounded Analysis of Multithreaded Programs with Counters. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 260–276, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-31424-7_22.

[20] Chih-Duo Hong and Anthony W. Lin. Regular Abstractions for Array Systems. *Proceedings of the ACM on Programming Languages*, 8(POPL):638–666, January 2024. URL: https://dl.acm.org/doi/10.1145/3632864, doi:10.1145/3632864.

[21] James Cornelius King. *A program verifier*. PhD thesis, Carnegie-Mellon University, Pittsburgh Pennsylvania USA, September 1969. Section: Technical Reports. URL: https://apps.dtic.mil/sti/citations/AD0699248.

[22] Felix Klaedtke and Harald Rueß. Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints. Technical Report 177, Freiburg University, Institute of Computer Science, 2002.

[23] Daniel Kroening and Ofer Strichman. *Decision Procedures*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition, 2016. doi:10.1007/978-3-662-50497-0.

[24] Viktor Kunčak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *Journal of Automated Reasoning*, 36(3):213–239, April 2006. doi:10.1007/s10817-006-9042-1.

[25] Viktor Kunčak and Martin Rinard. Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In *Automated Deduction – CADE-21*, Lecture Notes in Computer

Science, pages 215–230, Berlin, Heidelberg, 2007. Springer. doi:10.1007/978-3-540-73595-3_15.

[26] Haojun Ma, Aman Goel, Jean-Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A. Sakallah. I4: incremental inference of inductive invariants for verification of distributed protocols. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 370–384, Huntsville Ontario Canada, October 2019. ACM. doi:10.1145/3341301.3359651.

[27] Makai Mann, Ahmed Irfan, Alberto Griggio, Oded Padon, and Clark Barrett. Counterexample-Guided Prophecy for Model Checking Modulo the Theory of Arrays. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 113–132, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-72016-2_7.

[28] Andrzej Mostowski. On direct products of theories. *The Journal of Symbolic Logic*, 17(1):1–31, March 1952. doi:10.2307/2267454.

[29] Rodrigo Raya. *Decision Procedures for Power Structures*. PhD thesis, EPFL, 2023. doi:10.5075/epfl-thesis-10546.

[30] Rodrigo Raya and Viktor Kunčak. NP Satisfiability for Arrays as Powers. In *23rd International Conference on Verification, Model Checking, and Abstract Interpretation*, Lecture Notes in Computer Science, pages 301–318, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-94583-1_15.

[31] Rodrigo Raya and Viktor Kunčak. On algebraic array theories. *Journal of Logical and Algebraic Methods in Programming*, 136:100906, January 2024. doi:10.1016/j.jlamp.2023.100906.

[32] Rodrigo Raya and Viktor Kunčak. Succinct ordering and aggregation constraints in algebraic array theories. *Journal of Logical and Algebraic Methods in Programming*, 140:100978, August 2024. doi:10.1016/j.jlamp.2024.100978.

[33] Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. Counting in Trees for Free. In *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 1136–1149, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-27836-8_94.

[34] Larry Stockmeyer and Albert R. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM*, 49(6):753–784, November 2002. doi:10.1145/602220.602223.

[35] Wolfgang Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages*, pages 389–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. URL: http://link.springer.com/10.1007/978-3-642-59126-6_7, doi:10.1007/978-3-642-59126-6_7.

[36] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the Complexity of Equational Horn Clauses. In *Automated Deduction – CADE-20*, volume 3632, pages 337–352, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11532231_25.

[37] Thomas Wies, Ruzica Piskac, and Viktor Kunčak. Combining Theories with Shared Set Operations. In *Frontiers of Combining Systems*, Lecture Notes in Computer Science, pages 366–382, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-04222-5_23.

A Proof-Theoretical Approach to Some Extensions of First Order Quantification

Loïc Allègre[1], Ophélie Lacroix[2] and Christian Retoré[1]

[1]*LIRMM (Univ Montpellier & CNRS), Montpellier, France*
[2]*Resolve, Copenhagen, Danemark*

Abstract

Generalised quantifiers, which include Henkin's branching quantifiers, have been introduced by Mostowski and Lindström and developed as a substantial topic application of logic, especially model theory, to linguistics with work by Barwise, Cooper, Keenan.

In this paper, we mainly study the proof theory of some non-standard quantifiers as second order formulae. Our first example is the usual pair of first order quantifiers (for all / there exists) when individuals are viewed as individual concepts handled by second order deductive rules. Our second example is the study of a second order translation of the simplest branching quantifier: "A member of each team and a member of each board of directors know each other", for which we propose a second order treatment.

Keywords

proof theory, second order logic, generalised quantifiers, branching quantifiers, individual concepts,

1. Generalisation of Usual First Order Quantification

Common first order quantifiers ∃ and ∀ have been formalised the way they are in standard mathematics by Frege [1] whose logical and philosophical view matches Hilbert desiderata regarding logical foundations of mathematics.[2, 3, 4, 5]

By that time, mathematicians and logicians were making little difference between the interpretations of quantifiers in *the* standard model and their proof rules — before the work of Skolem or Gödel, mathematicians and logicians made little distinction between syntax and semantics, they worked with an interpreted language, see e.g. [2] —— perhaps Hilbert was more demanding regarding quantifiers because he focused on foundations of mathematics [3].

Extensions of usual quantification have mainly been considered for faithfully modelling quantification modes that one finds in ordinary language, like numbers ("three students came to the party"), "most" ("most students came to the party"), percentages ("a third of the students came to the party"), vague quantifiers ("few students came to the party"), etc.

This gave rise to the theory of generalised quantifiers, initially intended for model theory [6, 7], that was intensively developed in connection with linguistics.[8, 9, 10]. In such a setting, the lexical item expressing a quantifier (say "most A are B") is viewed as a function with two

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France
loic.allegre@lirmm.fr (L. Allègre); ophelie.lacroix@resolve.tech (O. Lacroix); christian.retore@lirmm.fr (C. Retoré)
https://sites.google.com/site/ophelielacroixnlp/ (O. Lacroix); http://www.lirmm.fr/~retore (C. Retoré)
0000-0002-2401-9158 (C. Retoré)

arguments that are predicates. This fits in well with the logico-functional view of quantification in Montague semantics [11]. Most (!) generalised quantifiers can be viewed as a second order construction. For instance the interpretation of a generalised quantifier depending on two unary predicates like "most" is interpreted as a second order construction, i.e. is true whenever the pairs of unary predicates it is applied to is a "legal" pair of subsets of the domain, namely a pair of sets (A, B) such that $|A \cap B| > |A \setminus B|$.

The paper is organised as follows. We first provide a reminder on second order logic, because this topic is not so common. Next, we present a second order view of usual first order quantification as second order quantification over individual concepts — and show the two formulations are proved to be equivalent provided the individual concepts are standard (i.e. they may not be empty, as opposed to some Kripke and Muskens variants). Then, after a quick presentation of generalised quantifiers, we study the simplest branching quantifier as a second order construction, and we propose direct rules for this quantifier.

2. A Reminder on Second Order Logic

We briefly remind the reader with basic facts about second order logic, following [12, Chapter 5], and one may also refer to the survey [13].

2.1. Language

A second-order language is based on a first-order language (the first three items in the list below), and extended with an infinitely enumerable set of predicate variables, each of them endowed with an arity (the last item in this list).

- an infinite enumerable set of first order (a.k.a. individual) variables x_i, with i in $\mathbb{N}$
- a set of constants c_i, with i in I (I is often enumerable, but this is not required);[1]
- an enumerable (or finite) set of predicate constants P_i^n with i in $\mathbb{N}$ each of them with an arity n (as in a first order language) – a predicate constant with arity 0 is a proposition;
- an enumerable set of predicate variables X_i^n i in $\mathbb{N}$ each of them with an arity n – a predicate variable with arity 0 is a propositional variable.

Second order formulae are defined "as expected": an atomic formula is $Z^n(u_1, \dots, u_n)$ with Z a predicate constant or a predicate variable of arity n and the u_i being n first order terms (here first order variables or first order constants since we have no functions). Les us call $\mathscr{A}$ the set of atomic formulae . Then the set of second order formulae $\mathscr{F}$ is defined by

$$\mathscr{F} =: : \mathscr{A} \mid \neg\mathscr{F} \mid \mathscr{F} \wedge \mathscr{F} \mid \mathscr{F} \vee \mathscr{F} \mid \mathscr{F} \to \mathscr{F} \mid \forall x_i \mathscr{F} \mid \exists x_i \mathscr{F} \mid \forall^n X_i^n \mathscr{F} \mid \exists^n X_i^n \mathscr{F}$$

where x_i stands for an individual variable while X_i^n stands for a predicate variable of arity n.

The formula $A \Leftrightarrow B$ is just a short-hand for $(A \to B) \wedge (B \to A)$.

[1] The first order language may also include an enumerable set of first order functions, but an n-ary function g in the language can replaced with an $n+1$-ary predicate G with an axiom that F is a function, and at second order there is a formula $F[X]$ saying that the $n+1$-ary predicate X corresponds to an n-ary function.

Although we shall not always write the ^{n} superscript in $\forall^n$ and $\exists^n$, beware that there are different pairs of second order quantifiers ($\exists^n/\forall^n$), one pair for each arity. The occurrences of the variable x_i or X_i^n are bound by the closest $\forall x, \exists x, \forall^n X_i^n, \exists^n X_i^n$ — if any — above them in the formula tree.

2.2. Proof Rules in Natural Deduction

We use natural deduction with standard rules as can be found in [12]. As we limit ourselves to classical logic, an extra principle is needed: *tertium non datur* ($A \vee \neg A$ for all A) or *reductio ad absurdum* (from a deduction with conclusion $\bot$ under hypothesis $\neg A$, conclude A), see e.g. [14]

The proof rules for second order quantifiers, namely the introduction and elimination rules of $\forall^n$ and $\exists^n$ are as expected, they are similar to the rules for first order quantifiers, *mutatis mutandi*:

$$\frac{\begin{array}{c}\vdots\\ \forall^n X_i^n\ T[X_i^n]\end{array}}{T[X_{i,k}^n(t_k^1,\dots,t_k^n)\ :=\phi^n(t_k^1,\dots,t_k^n)]}\ (\forall^n)_E \qquad \frac{\begin{array}{c}\vdots\\ T[X_i^n]\end{array}}{\forall^n X_i^n\ T[X_i^n]}\ (\forall^n)_I$$

$$\frac{\exists^n X_i^n\ T[X_i^n] \qquad \begin{array}{c}\left[T[X_i^n]\right]^k\\ \vdots\\ \psi\end{array}}{\psi}\ (\exists^n)_E^k \qquad \frac{\begin{array}{c}\vdots\\ T[X_{i,k}^n(t_k^1,\dots,t_k^n)\ :=\phi^n(t_k^1,\dots,t_k^n)]\end{array}}{\exists^n X_i^n T[X_i^n]}\ (\exists^n)_I$$

where

1. $T[X_i^n]$ stands for a formula in which the predicate variable X_i^n may occur (but that is not mandatory, as for first order quantification).
2. There should be no free occurrence of X_i^n in the hypotheses of the introduction rule $(\forall^n)_I$ nor in the elimination rule $(\exists^n)_E^k$ — as in the first order $\forall_I$ and $\exists_E$ introduction rules.
3. The obscure notation[2] $T[X_{i,k}^n(t_k^1,\dots,t_k^n)\ :=\phi^n(t_k^1,\dots,t_k^n)]$ requires some explanation. This formula stands for the formula obtained by replacing
   - the k^{th} occurrence $X_{i,k}^n$ of X_i^n which is applied to n terms $(t_k^1,\dots,t_k^n)$
   - with a formula with n free variables applied to the very same terms $(t_k^1,\dots,t_k^n)$

   with the requirement that no originally free variable in $(t_k^1,\dots,t_k^n)$ becomes bound after the application of ϕ to $(t_k^1,\dots,t_k^n)$.
   Here is an example: let $\phi(x,y) = P(x,y) \wedge Q(y,a)$, let $T[X_1^2] = X_{1,1}^2(z,a) \wedge X_{1,2}^2(a,b)$ – mind the second subscript of $X_{1,\bullet}^2$ which indicates the occurrence number (there are two occurrences of the predicate variable X_1^2 in $T[X_1^2]$. Then $T[X_{1,k}^2\ :=\phi(t_k^1,\dots,t_k^n)]$ is

[2]This point is often under explained in the literature.

$(P(x,y) \wedge Q(y,a))[x := z; y := a] \wedge (P(x,y) \wedge Q(y,a))[x := a; y := b]$ that is $(P(z,a) \wedge Q(a,a)) \wedge (P(a,b) \wedge Q(b,a))$. From the definition and the example, it is unsurprising that second order unification is undecidable [15].

4. In the rule $(\exists^n)_E$ the expression $\left[T[X_i^n]\right]^k$ indicates that the hypothesis $T[X_i^n]$ has been cancelled during the $(\exists^n)_E^k$ number k.

Some remarks:

1. this proof system can derive the comprehension axiom:
$$\exists X^n \forall x_1 \dots x_n \left[\varphi(x_1, \dots, x_n) \leftrightarrow X^n(x_1, \dots, x_n)\right]$$
2. equality can be defined à la Leibnitz: $x = y : \forall^1 X^1 \left[X^1(x) \to X^1(y)\right]$ (because of negation there is no need to use $\Leftrightarrow$ in this definition)
3. being equal to x is a property $E_x(y) : \forall^1 X^1 \left[X^1(x) \to X^1(y)\right]$.
4. the Dedekind finiteness, "any injective function is surjective" is definable:
$\forall^2 X^2 \quad ((\forall x \forall y \forall z (X(x,y) \wedge X(x,z) \to y = z)) \wedge (\forall x \forall y \forall z (X(y,x) \wedge X(z,x) \to y = z)))$
$\to (\forall w \exists u\, X(u,w))$

Finally, using implication $\to$, first order $\forall$, and propositional second order $\forall^0$ one can define false, $\bot$, negation $\neg$, the propositional connective $\wedge$, $\vee$, first order existential quantification $\exists$. Adding $\forall^n$ to $\to$, $\forall$, one can also define $\exists^n$. Thus, the expressive power of second order propositional quantifier $\forall^0$ is impressive.

2.3. Standard and Non-standard Models, Completeness

We here follow [16, 13, 12].

A second order model consists in a first order model, i.e. with a domain D, endowed with a set of sets of tuples of length n for each $n \in \mathbb{N}$ in order to interpret predicate variables of arity n: they may vary in a fixed subset A^n of $\mathscr{P}(D^n)$ which is not necessarily the full powerset $\mathscr{P}(D^n)$; for this structure to define a model, it must enjoy the comprehension axiom scheme $\exists^n X^n \forall x_1 \cdots \forall x_n [\phi(x_1, \dots, x_n) \leftrightarrow X^n(x_1, \dots, x_n)]$ where the n-ary predicate variable X^n does not appear in ϕ — in other words the subsets of A^n must include the interpretations of the formulae with n free variables. The comprehension axiom scheme is derivable from the existential introduction rule given above.

By definition, a second-order model is said to be *standard* (or *full*) whenever the subset A^n is $\mathscr{P}(D^n)$ for any arity n. Standard models satisfy the comprehension scheme. They do match intuition: a predicate variable of arity n varies in all possible subsets of D^n. However, neither completeness nor compactness hold when only standard models are considered — indeed second order logic can express the finiteness of the domain D, see *e.g.* [12].

Second order logic may be encoded in first order logic: predicate variables X_i^n are viewed as individual constants interpreted in a domain (that also contains standard individuals), and some additional predicate constants of arity $n + 1$ are needed to mimic the application of a predicate variable of arity n to n terms. Then a second order formula is provable within second

order logic whenever its first order translation is provable in first order logic. So applying first order completeness theorem one gets that a second order formula is provable in second order logic if and only if it is true in all second order models (including the non standard ones). As a consequence of completeness, compactness holds, so one can have a model with at least n elements for each integer n, which is Dedekind finite.

3. A Second Order View of First Order Quantification: Quantifying Over Individual Concepts

3.1. Individual Concepts

Individual concepts view an individual as a formula $\phi[x]$ with a single free variable x such that there is a single individual satisfying the formula $\phi[x]$.[3] This can be said in second order logic: a formula $\phi[x]$ with a single free variable x is said to be an individual concept whenever there is at most one individual x such that $\phi[x]$ and at least one x such that $\phi[x]$ — so it makes exactly one x such that $\phi[x]$. That the concept ϕ is an *individual concept* can actually be expressed in second order logic, where the "=" symbol refers to usual equality (with its usual rules: reflexivity, symmetry, transivity, substition etc. noted "=" in the proofs):[4]

$$C(\phi) := (\forall x \forall y [\phi(x) \wedge \phi(y) \rightarrow x = y]) \wedge \exists z\, \phi(z)$$

Individual concepts are close to Montague semantics and Leibnitz identity (where an individual is identified with the set of all properties it enjoys) [18, 19, 11]. For reasons like possible worlds semantics, see e.g. the discussion in [20, chapter 4], some logicians consider a variant of individual concepts that are possibly empty. This may look strange but if you think individual concepts are some kind of proper name that are part of the logical language, it is hard to tell what a proper name refers to before the reference actually exists. For instance, the individual concept Gödel(x) has no reference in Egyptian times. Hence Kripke in the 70s dropped the existence condition from individual concepts (see [21] for a recent account of those ideas), thus obtaining a formula with a lower logical complexity profile:

$$C_\epsilon(\phi) := \forall x \forall y [\phi(x) \wedge \phi(y) \rightarrow x = y]$$

So $C(\phi)$ can be defined as $C(\phi) := C_\epsilon(\phi) \wedge \exists z\, \phi(z)$.

3.2. First Order Universal Quantification as Universal Quantification Over Non-empty Individual Concepts

The simplest quantifiers one can try to view as a second order construction are clearly the usual first order quantifiers $\forall, \exists$. So let us compare the second order quantification over individual concepts to usual quantification.

[3] This notion of concept is somehow related to concepts in description logics, and the individual concepts that we use correspond to individual names cf. e.g. [17].

[4] The ":=" symbol denotes an abbreviation (or a definition) of a formula, and in proofs, the replacement of an abbreviation by its expansion or vice-versa.

Proposition 1. *First order universal quantification and second order quantification over individual concepts are equivalent:*

1. *given a property $\varphi(X)$ of individual concepts, the following equivalence holds: $\forall x\, \varphi^{\downarrow}(x) \Leftrightarrow \forall X(C(X) \to \varphi(X))$ where $\varphi^{\downarrow}(x) := \exists X(C(X) \wedge X(x) \wedge \varphi(X))$.*
2. *given a property $\psi(x)$ of individuals, the following equivalence holds: $\forall x\, \psi(x) \Leftrightarrow \forall X\Big(C(X) \to \psi^{\uparrow}(X)\Big)$ where $\psi^{\uparrow}(X) := \exists x(X(x) \wedge \psi(x))$.*

Proof. Let us first observe that there is a simple formal proof without assumption that $C(E_x)$ i.e. that "being equal to x" (cf. section 2 item 3) is an individual concept, $E_x(y) : y = x : \forall^1 X^1\ (X^1(x) \to X^1(y))$, and let us call this proof δ, because we are going to use it several times:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[y = x \wedge z = x]}{y = x}\wedge E \quad \dfrac{\dfrac{[y = x \wedge z = x]}{z = x}\wedge E}{x = z}=}{y = z}=}{y = x \wedge z = x \to y = z}\to I}{\forall z(y = x \wedge z = x \to y = z)}\forall^1 I}{\forall y \forall z(y = x \wedge z = x \to y = z)}\forall^1 I}{C(E_x)}:=$$

1. a) Assuming $\forall x\, \varphi^{\downarrow}(x)$ one can prove $\forall X(C(X) \to \varphi(X))$

$$\dfrac{\dfrac{[\forall x \varphi^{\downarrow}(x)] \quad \dfrac{\dfrac{\dfrac{[C(X)]}{\forall x \forall y(X(x) \wedge X(y) \to x = y) \wedge \exists x X(x)}\wedge E}{\exists x X(x)} \quad \dfrac{[X(x)]}{X(x)}\exists^1 E}{\forall x \varphi^{\downarrow}(x) \wedge X(x)}\wedge I}{\dfrac{\varphi^{\downarrow}(x)}{\exists Y C(Y) \wedge Y(x) \wedge \varphi(Y)}:= \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[C(Y) \wedge Y(x) \wedge \varphi(Y)]}{C(Y) \wedge Y(x) \wedge \varphi(Y)}\wedge E}{Y(x) \wedge \varphi(Y)}\wedge E}{\varphi(Y)}:=}{\varphi(X)}}{C(X) \to \varphi(X)}\to I}{\forall X(C(X) \to \varphi(X))}\forall^2 E}\exists^2 E$$

   b) Assuming $\forall X(C(X) \to \varphi(X))$ one can prove $\forall x\, \varphi^{\downarrow}(x)$. In the proof below, we use δ the proof that $E_x(\_) =$ "$\_ = x$" (that is, "being equal to x") is an individual concept.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\delta\\ \vdots\\ C(E_x)\end{matrix}\qquad \dfrac{\dfrac{x=x}{E_x(x)}\qquad \dfrac{\begin{matrix}\delta\\ \vdots\\ C(E_x)\end{matrix}\qquad \dfrac{[\forall X(C(X)\to\varphi(X))]}{C(E_x)\to\varphi(E_x)}\,\forall E}{\varphi(E_x)}\to E}{E_x(x)\land\varphi(E_x)}\land I}{C(E_x)\land E_x(x)\land\varphi(E_x)}\land I}{\exists X(C(X)\land X(x)\land\varphi(X))}\,\exists^2 I}{\varphi^{\downarrow}(x)}\,:=
$$

$$
\dfrac{\varphi^{\downarrow}(x)}{\forall x\varphi^{\downarrow}(x)}\,\forall^1 E
$$

2. a) Assuming $\forall x\,\psi(x)$ one can prove $\forall X\big(C(X)\to\psi^{\uparrow}(X)\big)$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[C(X)]}{\forall x\forall y(X(x)\land X(y)\to x=y)\land\exists x\,X(x)}\,:=}{\exists xX(x)}\land E\qquad [X(x)]}{X(x)}\,\exists^1 E\qquad \dfrac{[\forall x\psi(x)]}{\psi(x)}\,\forall^1 E}{X(x)\land\psi(x)}\land I}{\exists x(X(x)\land\psi(x))}\,\exists^1 I}{\psi^{\uparrow}(X)}\,:=}{C(X)\to\psi^{\uparrow}(X)}\to I}{\forall X(C(X)\to\psi^{\uparrow}(X)}\,\forall^2 I
$$

b) Finally assuming $\forall X\big(C(X)\to\psi^{\uparrow}(X)\big)$ one can prove: $\forall x\,\psi(x)$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}\delta\\ \vdots\\ C(E_x)\end{matrix}\qquad \dfrac{[\forall X(C(X)\to\psi^{\uparrow}(X)]}{C(E_x)\to\psi^{\uparrow}(E_x)}\,\forall^2 E}{\psi^{\uparrow}(E_x)}\to E}{\exists yE_x(y)\land\psi(y)}\,:=}{E_x(y)\land\psi(y)}\,\exists^1 E}{y=x\land\psi(y)}\,:=}{\psi(x)}\land E
$$

$$
\dfrac{\psi(x)}{\forall x\psi(x)}\,\forall^1 I
$$

□

3.3. First Order Existential Quantification as Existential Quantification Over Non-empty Individual Concepts

As for the universal quantification, we have:

Proposition 2. *First order existential quantification and second order quantification over individual concepts are equivalent:*

1. *when ϕ is a property of individual concepts, one has $\exists x\, \varphi^{\downarrow}(x) \Leftrightarrow \exists X(C(X) \wedge \varphi(X))$ where $\varphi^{\downarrow}(x) := \exists X(C(X) \wedge X(x) \wedge \varphi(X))$.*
2. *when ψ is a property of individuals, one has $\exists x\, \psi(x) \Leftrightarrow \exists X\Big(C(X) \wedge \psi^{\uparrow}(X)\Big)$ where $\psi^{\uparrow}(X) := \exists x(X(x) \wedge \psi(x))$*

Proof. At point 2.a) we will also use the proof δ of $C(E_x)$ from the proof of proposition 1.

1. $\exists x\, \varphi^{\downarrow}(x) \Leftrightarrow \exists X(C(X) \wedge \varphi(X))$ where $\varphi^{\downarrow}(x) := \exists X(C(X) \wedge X(x) \wedge \varphi(X)$
   a) Let us prove $\exists X(C(X) \wedge \varphi(X))$ under the assumption $\exists x\, \varphi^{\downarrow}(x)$.

$$\dfrac{[\exists x \varphi^{\downarrow}(x)] \qquad \dfrac{[\varphi^{\downarrow}(x)]}{\exists X(C(X) \wedge X(x) \wedge \varphi(X))}}{\exists X(C(X) \wedge \varphi(X))}\ \exists^1 E \quad := \quad \dfrac{\dfrac{\dfrac{[C(X) \wedge X(x) \wedge \varphi(X)]}{C(X)}\ \wedge E \qquad \dfrac{[C(X) \wedge X(x) \wedge \varphi(X)]}{\varphi(X)}\ \wedge E}{C(X) \wedge \varphi(X)}\ \exists^2 I}{\exists X(C(X) \wedge \varphi(X))}\ \exists^2 E$$

   b) Now let us prove $\exists x\, \varphi^{\downarrow}(x)$ under the assumption $\exists X(C(X) \wedge \varphi(X))$. We first need α, β, γ i.e. the three following proofs :

α :

$$\dfrac{[\exists X(C(X) \wedge X(x) \wedge \varphi(X))] \qquad \dfrac{[C(X) \wedge \varphi(X)]}{C(X)}\ \wedge E}{C(X)}\ \exists^2 E$$

β :

$$\dfrac{[\exists X(C(X) \wedge X(x) \wedge \varphi(X))] \qquad \dfrac{[C(X) \wedge \varphi(X)]}{\varphi(X)}\ \wedge E}{\varphi(X)}\ \exists^2 E$$

γ :

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\alpha\\ \vdots\\ C(X)\end{array}}{\forall x\forall y(X(x)\wedge Y(x)\to x=y)\wedge\exists x X(x)}\ :=}{\exists x X(x)}\ \wedge E \qquad [X(x)]}{X(x)}$$

and the proof we are looking for is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{cc}\alpha & \gamma\\ \vdots & \vdots\\ C(X) & X(x)\end{array}}{C(X)\wedge X(x)}\ \wedge I \qquad \begin{array}{c}\beta\\ \vdots\\ \varphi(X)\end{array}}{C(X)\wedge X(x)\wedge\varphi(X)}\ \wedge I}{\exists X(C(X)\wedge X(x)\wedge\varphi(X))}\ \exists^2 I}{\dfrac{\varphi^{\downarrow}(x)}{\exists x\varphi^{\downarrow}(x)}\ \exists^1 I}\ :=$$

2. $\exists x\,\psi(x)\Leftrightarrow\exists X\big(C(X)\wedge\psi^{\uparrow}(X)\big)$ where $\psi^{\uparrow}(X)\ :=\exists x(X(x)\wedge\psi(x))$
   a) Let us prove $\exists X\big(C(X)\wedge\psi^{\uparrow}(X)\big)$ under the assumption $\exists x\,\psi(x)$ – δ is the proof of $C(E_x)$ defined in the proof of proposition 1.

$$\dfrac{\dfrac{\begin{array}{c}\delta\\ \vdots\\ C(E_x)\end{array}}{C(E_x)\wedge\exists y(E_x(y)\wedge\psi(y))}\ \wedge I \qquad \dfrac{\dfrac{\dfrac{x=x}{E_x(x)}\qquad\dfrac{[\exists x\psi(x)]\quad[\psi(x)]}{\psi(x)}\ \exists^1 E}{E_x(x)\wedge\psi(x)}\ \wedge I}{\exists y(E_x(y)\wedge\psi(y))}\ \exists^1 I}{\dfrac{\begin{array}{c}\delta\\ \vdots\\ C(E_x)\end{array}\qquad\psi^{\uparrow}(E_x)}{C(E_x)\wedge\psi^{\uparrow}(E_x)}\ \wedge I}\ :=$$
$$\dfrac{C(E_x)\wedge\psi^{\uparrow}(E_x)}{\exists X(C(X)\wedge\psi^{\uparrow}(X)}\ \exists^2 I$$

b) Let us prove $\exists x\,\psi(x)$ under the assumption $\exists X\big(C(X)\wedge\psi^{\uparrow}(X)\big)$

$$\dfrac{\dfrac{\dfrac{[\exists X(C(X)\wedge\psi^{\uparrow}(X))]\quad[C(X)\wedge\psi^{\uparrow}(x)]}{C(X)\wedge\psi^{\uparrow}(X)}\ \exists^2E}{\psi^{\uparrow}(X)}\ \wedge E}{\exists x(X(x)\wedge\psi(x))}\ :=\qquad\dfrac{[X(x)\wedge\psi(x)]}{\psi(x)}\ \wedge E$$
$$\dfrac{\exists x(X(x)\wedge\psi(x))\qquad\qquad\psi(x)}{\psi(x)}\ \exists^1E$$
$$\dfrac{\psi(x)}{\exists x\psi(x)}\ \exists^1I$$

□

3.4. Dealing With Possibly Empty Individual Concepts

When individual concepts are possibly empty the second order formula expressing that X is an individual concept is $C_\epsilon(X)=\forall x\forall y(Xx\wedge Xy\to x=y)$ — the $\exists xX(x)$ left out of our initial definition of individual concepts.

Regarding universal quantification, $\forall X(C_\epsilon(X)\to\varphi(X))$ still entails $\forall x\,\varphi^{\downarrow}(x)$, but $\forall x\,\varphi^{\downarrow}(x)$ does not entail $\forall X(C_\epsilon(X)\to\varphi(X))$ anymore. This is logical: when all individual concepts have a property be they empty or not, all individuals enjoy the corresponding first order property. The converse does not hold: when all individuals enjoy a property, all the non empty concepts enjoy the property, but why should the empty individual concept enjoy this property as well?

Of course regarding existential quantification, that's the opposite. $\exists x\,\varphi(x)$ entails $\exists X(C_\epsilon(X)\wedge\varphi(X)))$ but $\exists X(C_\epsilon(X)\wedge\varphi(X))$ does not entail $\exists x\,\varphi(x)$. When an individual enjoys a property, so does the corresponding individual concept. But when a possibly empty individual concept enjoys a property, it does not entail that an individual enjoys this property, because this individual concept might be an empty individual concept.

So the second order view of usual quantification does not fit in well with possibly empty individual concepts.

4. A Reminder on Generalised and Branching Quantifiers

This reminder mainly relies on the presentation given by Peters and Westerståhl [22], one may also consult the survey [10]. Generalised quantifiers, initially introduced by Mostowski [6] and further developed by Lindström [7] are a generalisation of standard universal and existential quantification. Roughly speaking, generalised quantifiers view quantifiers as relations over relations (or tuples of relations) — those relations are relations on the domain (a.k.a universe, model) of an interpretation: thus, quantifiers are viewed as second-order concepts.

4.1. Generalised Quantifiers

Given k integers $n_1,...,n_k$, a quantifier $\mathcal{Q}$ of type $\langle n_1,...,n_k\rangle$ can be viewed as a function endowing each domain M with a k-ary relation Q_M such that if $(R_1,...,R_k)\in Q_M$, then for all i in between 1 and k, R_i is a n_i-ary relation over elements of M. Let us give some examples.

The usual quantifiers $\forall$ and $\exists$ can be then expressed as simple type $\langle 1\rangle$ quantifiers : $\exists_M = \{A \subseteq M,\ A \neq \emptyset\}$ and $\forall_M = \{A \subseteq M,\ A = M\}$. Thus, the existential quantifier is in every domain the unary relation which holds true for all non-empty predicates, and the universal quantifier is the relation which holds true only for the whole domain M.

Some generalised quantifiers have an equivalent formulation in usual first-order logic, such as the quantifier "at least two": $(\exists_{\geq 2})_M = \{A \subseteq M,\ |A| \geq 2\}$ which can be expressed with the following first-order formula: $\exists x \exists y[x \neq y]$. However this is not always the case. Take for example the type $\langle 1,1\rangle$ quantifier expressing that most A are B: $\mathbf{Most}_M(A,B) \iff |A \cap B| > |A - B|$ which notably cannot be expressed as a first-order formula.

It is worth noting that universal and existential quantification on individual concepts as we presented in Section 3 can also be formulated in terms of generalised quantifiers. To say that all (resp. some) individual concepts satisfy a property φ is in fact a second-order statement about the predicates C ("to be an individual concept") and φ. Hence the second order view of first order quantification that we presented in the previous section can be expressed as generalised quantifiers $\forall_C$ and $\exists_C$ with type $\langle 1,1\rangle$:

$$\forall_C(C,\varphi) \iff C \subset \varphi \qquad \exists_C(C,\varphi) \iff C \cap \varphi \neq \emptyset$$

4.2. Branching Quantifiers

Among generalised quantifiers, branching quantifiers are of particular interest, both for logic and linguistics. Initially introduced by Henkin [23] and much later on studied by Hintikka [24] — independently of generalised quantifiers — branching quantification is a generalisation of classical quantification that allows the expression of independence between some existentially quantified variable and some previously universally quantified variables. This cannot be expressed within usual quantification because quantifiers are supposed to be linearly ordered. The simplest example of such a non-first-order quantifier is the following Henkin quantifier where, as the notation suggests, x' only depends on x, while y', only depends on y.

$$\begin{matrix} \forall x \exists x' \\ \forall y \exists y' \end{matrix} \Big\rangle\ F(x,y,x',y')$$

As proven by Ehrenfeucht (in Henkin [23]), this construction has no first-order equivalent. Notably, it cannot be expressed with a linear quantifier prefix such as $\forall x \exists x' \forall y \exists y'$ or $\forall x \forall y \exists x' \exists y'$, since there would be unwanted dependencies between x' and y, and y' and x.

Although not initially introduced as such, branching quantifiers can in fact be seen as specific generalised quantifiers. Indeed, the (in)dependencies between variables can be expressed using Skolem functions, e.g. the Henkin quantifier above can be written as follows:

$$\exists f \exists g \forall x \forall y\, F(x, f(x), y, g(y))$$

This in turn allows us to translate it as a generalised quantifier, for example here as the type $\langle 4\rangle$ quantifier:

$$H_M = \{R \subseteq M^4 \mid \exists f \exists g \forall x \forall y\, (x, f(x), y, g(y)) \in R\}$$

5. Second Order Proof Rules for Branching Quantifiers

Branching Quantifiers as Second Order Formulae In this part, we focus on the expression of branching quantifiers as second-order constructions. Such quantifiers can be quite complex, so we limit ourselves to studying the simplest branching quantifier. Our main object of study is the typical branching constructions found in natural language in the so-called Hintikka sentences, such as :

(H) *A member of each team and a member of each board of directors know each other*

The branching-quantifier reading of the above English sentence can be formulated within second-order logic:[5]

$$\begin{matrix}\forall x \exists x' \\ \forall y \exists y'\end{matrix} \Big\rangle\ T(x) \wedge B(y) \rightarrow M(x,x') \wedge M(y,y') \wedge K(x',y')$$

As mentioned earlier, this formula can be expressed as a second order formula with existential quantification over functions:

$$(\text{Hfun}) \ : \ \exists f \exists g \forall x \forall y \, [T(x) \wedge B(y)] \rightarrow K(f(x), g(y))$$

Natural Deduction Rules With Binary Predicates This formulation of the Henkin quantifier as a second-order formula with quantification over functions is however not fully satisfactory, for it actually provides a stronger effect than needed: defining f and g as functions implies the unicity of $f(x)$ and $g(y)$ for any given x and y, while the original formula with a branching quantifier only requires that there exists one (possibly more) x' for each x and y' for each y. Thus f and g need not be functions, but only need be non-empty binary predicates — as always with Skolem functions, the choice of $f(x)$ for each x is part of the interpretation of the function symbol.

Therefore, we propose another second-order representation of this reading of the sentence using quantification over predicates instead of quantification over functions:

$$\begin{aligned}(\text{Hpred}) \ : & \exists F \exists G [\forall x \exists x' T(x) \rightarrow F(x,x')] \wedge [\forall y \exists y' B(y) \rightarrow G(y,y')] \\ & \wedge [\forall x \forall x' \forall y \forall y' [T(x) \wedge B(y) \wedge F(x,x') \wedge G(y,y')] \rightarrow K(x',y')]\end{aligned}$$

This formula simply replaces each of the two functions f and g of $(Hfun)$ above with the binary predicates F and G. These two predicates act intuitively as relations that select suitable x' and y', since all we need to ensure is that whenever x' is a valid representative for x (and y' for y), then x' and y' know each other. The binary predicates F and G are required to relate each possible value of their first argument which ought to be in the proper set/predicate (T for x, B for

[5] There is also a first-order reading of this sentence, which the 'each other' (perhaps) makes less perceptible, and which can be expressed within first-order logic: $[\forall x \exists x' \forall y \exists y' \quad T(x) \wedge B(y) \rightarrow M(x,x') \wedge M(y,y') \wedge K(x',y')] \wedge [\forall y \exists y' \forall x \exists x' \quad T(x) \wedge B(y) \rightarrow M(x,x') \wedge M(y,y') \wedge K(x',y')]$. According to Szymanik [25], in two-thirds of cases, the first-order reading is preferred to the branching quantifier reading.

y) to at least one value of their second argument.[6] There is nevertheless a difference between using function as in (Hfun) and (Hpred): in (Hpred) there may well exist several values of x' (resp. y') such that $F(x, x')$ (resp. $G(y, y')$), there is no need to chose one as opposed to (Hfun), where one is explicitly chosen.

The natural deduction rules for second-order logic from section 2.2 give us the introduction and elimination rules for the branching Henkin quantifier. For the sake of readability, let us write from here on:

$$\Phi(F, G, x, x', y, y') = [T(x) \wedge B(y) \wedge F(x, x') \wedge G(y, y')] \rightarrow K(x', y')$$

and

$$\begin{aligned}\Psi(F, G) = {} & [\forall x \exists x' T(x) \rightarrow F(x, x')] \wedge [\forall y \exists y' B(y) \rightarrow G(y, y')] \\ & \wedge [\forall x, \forall x' \forall y \forall y' \Phi(F, G, x, x', y, y')]\end{aligned}$$

The introduction rule is quite straightforward:

$$\frac{\varphi(x, t) \qquad \psi(y, u) \qquad \Phi(\varphi, \psi, x, x', y, y')}{\exists F \exists G[\forall x \exists x' T(x) \rightarrow F(x, x')] \wedge [\forall y \exists y' B(y) \rightarrow G(y, y')] \wedge [\forall x, \forall x' \forall y \forall y' \Phi(F, G, x, x', y, y')]} H_I$$

where φ is a formula with free variables exactly x, x', ψ a formula with free variables exactly y, y', and t, u are terms.

The elimination rule, however, is more complicated due to the use of two second-order eliminations of $\exists^2$:

$$\dfrac{\exists F \exists G \Psi(F, G) \qquad \dfrac{[\exists G \Psi(A, G)]^{(2)} \qquad \dfrac{[\Psi(A, B)]^{(1)} \\ \vdots \\ \varphi}{\varphi} \exists^2 E^{(1)}}{\varphi} \exists^2 E^{(2)}$$

in which A and B must not appear free in φ.

Let us write as above $H(x, x', y, y')$ the branching quantifier that binds the two universally quantified variables x, y and the two existentially quantified variables x', y' e.g. the example above may be written $H(x, x', y, y')\ \Phi(F, G, x, x', y, y')$.

Let $\mathscr{H}$ be the set of closed formulae that can be written with this quantifier and the two usual first order quantifiers $\exists x$ and $\forall y$.

In the near future, we intend to determine whether our direct rules can be used to derive all the formulae in $\mathscr{H}$ that can be derived with the usual rules for second and first quantifiers. It seems plausible to us, because cut-elimination holds for second order logic. [26] However, we are not yet fully certain, and this is one aspect that we intend to clarify in the near future.

[6] Similarly, we could ask that x' and y' are in the proper set/predicate (T for x', B for y') but it is less important. Thus we do not add this precision, which is not needed — unlike the restriction to x and y — and makes the formulae, which are already long enough, considerably longer: Hpred′ : $\exists F \exists G[\forall x \exists x' T(x) \rightarrow F(x, x') \wedge T(x')] \wedge [\forall y \exists y' B(y) \rightarrow G(y, y') \wedge B(y')] \wedge [\forall x \forall x' \forall y \forall y' T(x) \wedge T(x') \wedge B(y) \wedge B(y') \wedge F(x, x') \wedge G(y, y') \rightarrow K(x', y')]$

6. Conclusion

This ongoing work deals with the proof rules of extensions of first order quantification viewed as second order logic constructions which do not need the full expressive power (and complexity) of second order logic.

We first described usual first order quantifiers within the second order logic as quantification over individual concepts and obtained that this description works provided individual concepts are asked to be non empty — unsurprisingly it does not work without this restriction.

Thereafter we focused on the non first order reading of the simplest branching quantifier that one finds in sentences like: *"A member of each team and a member of each board of directors know each other"*. Regarding this (reading of this) quantifier, we proposed a definition of it within second order logic and provided direct introduction and elimination rules for this complex quantifier which is often only described in model-theoretic terms. Later on we shall prove that the complete set of second order rules does not derive more sentences with those connectives than our direct rules.

While we were working on the final version of this article, we realised that Matthias Baaz and Anela Lolic [27] proposed an analytic calculus for Henkin quantifiers. In their paper, Henkin quantifiers are viewed as the existence of functions (i.e. as a particular form of second order quantification), and they proved cut-elimination for this calculus. It is too late for this paper of ours to examine how their account of Henkin quantifiers differs from ours, but this will be be our first aim in continuing our work. A first remark is that $(1) : \exists F\, \forall u\, \exists v\, F(u, v)$ seems slightly weaker than $(2) : \exists f\, F(u, f(u))$: in order to derive (2) from 1, it is necessary to utilise some form of the axiom of choice. Another difference, a very small one actually, is that our rules are natural deduction rules (introduction/elimination rules) and not sequent calculus: sequent-calculus right-rules correspond to natural-deduction introduction-rules but sequent-calculus left-rules do not correspond to natural-deduction elimination-rules.

By way of conclusion, let us mention a prospect that has opened up to us recently. The epsilon calculus [28, 29] may express readings that are close to branching quantifier readings, with epsilon formulas that have no equivalent in first or higher order logic[7]. Indeed, the subnectors epsilon and tau express quantification with some scope ambiguities (under-specification)[8]. However, it is presently too early to say anything definite about this question.

Acknowledgments

We would like to express our gratitude to the anonymous reviewers and to the chairs of ARQNL2024 for their valuable feedback, which has been instrumental in helping us to enhance this article. Furthermore, we would like to extend our gratitude to the programme chairs who accepted some late modifications.

[7]The formula $A(\varepsilon_v B(v))$ is not equivalent to any first order formula — although it is equivalent to $\exists v\, A(v) \wedge B(v)$ when additionally $\exists v\, B(v) \equiv B(\varepsilon_v B(v))$ holds.

[8]As an example of under-specification or scope ambiguity in the ε-calculus, a formula like $G(\varepsilon_u A(u), \tau_w B(w))$ which has no equivalent in first order logic, has some logical relation, depending on whether A and B are $\emptyset$ or D, with the two first-order formulas $\forall u \exists w\, A(u) \implies (B(w) \wedge G(u, w))$ and $\exists w \forall u\, B(w) \wedge (A(u) \implies G(u, w))$.

References

[1] G. Frege, Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, Verlag von Louis Nebert, Halle, 1879.

[2] W. Kneale, M. Kneale, The development of logic, 3rd ed., Oxford University Press, 1986.

[3] W. D. Goldfarb, Logic in the twenties: The nature of the quantifier, J. Symb. Log. 44 (1979) 351–368. URL: https://doi.org/10.2307/2273128. doi:10.2307/2273128.

[4] D. Hilbert, P. Bernays, Grundlagen der Mathematik. Bd. 1., Berlin: Julius Springer. XII, 471 S., 1934. Traduction française de F. Gaillard et M. Guillaume, L'Harmattan, 2001.

[5] D. Hilbert, P. Bernays, Grundlagen der Mathematik. Bd. 2., Springer, 1939. Traduction française de F. Gaillard, E. Guillaume et M. Guillaume, L'Harmattan, 2001.

[6] A. Mostowski, On a generalization of quantifiers, Fundamenta Mathematicae 44 (1957) 12–36.

[7] P. Lindström, First order predicate logic with generalized quantifiers, Theoria 32 (1966) 186–195.

[8] J. Barwise, R. Cooper, Generalized quantifiers and natural language, Linguistics and Philosophy 4 (1981) 159–219. doi:10.1007/BF00350139.

[9] E. Keenan, J. Stavi, A semantic characterization of natural language determiners, Linguistic and Philosophy 9 (1986) 253–326.

[10] D. Westerståhl, Generalized Quantifiers, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Winter 2019 ed., Metaphysics Research Lab, Stanford University, 2019.

[11] R. Montague, The proper treatment of quantification in ordinary english, in: J. Hintikka, J. Moravcsik, P. Suppes (Eds.), Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics, Reidel, Dordrecht, 1973.

[12] D. van Dalen, Logic and Structure, Universitext, fifth ed., Springer-Verlag, 2013.

[13] J. Väänänen, Second-order and Higher-order Logic, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Fall 2021 ed., Metaphysics Research Lab, Stanford University, 2021.

[14] R. Moot, C. Retoré, Classical logic and intuitionistic logic: equivalent formulations in natural deduction, Gödel-Kolmogorov-Glivenko translation, 2016. arXiv:1602.07608.

[15] W. D. Goldfarb, The undecidability of the second-order unification problem, Theor. Comput. Sci. 13 (1981) 225–230. URL: https://doi.org/10.1016/0304-3975(81)90040-2. doi:10.1016/0304-3975(81)90040-2.

[16] L. Henkin, Completeness in the theory of types, J. Symb. Log. 15 (1950) 81–91. URL: https://doi.org/10.2307/2266967.

[17] S. Rudolph, Foundations of description logics, in: A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, P. F. Patel-Schneider (Eds.), Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Tutorial Lectures, volume 6848 of *LNCS*, Springer, 2011, pp. 76–136. doi:10.1007/978-3-642-23032-5_2.

[18] S. Kripke, Identity and necessity, in: M. Munitz (Ed.), Indentity and Individuation, New-York University Press, 1971, pp. 135–164.

[19] S. Kripke, Naming and necessity, in: D. Davidson, G. Harman (Eds.), Semantics of Natural Language, Reidel, 1972, pp. 253–355.

[20] M. Fitting, R. Mendelsohn, First-Order Modal Logic, Springer Netherlands, 1998.

[21] R. Muskens, A theory of names and true intensionality, in: M. Aloni, V. Kimmelman, F. Roelofsen, G. W. Sassoon, K. Schulz, M. Westera (Eds.), Amsterdam Colloquium 2011, volume 7218 of *LNCS/FoLLI*, Springer-Verlag, 2012, pp. 441–449.
[22] S. Peters, D. Westerståhl, Quantifiers in Language and Logic, Clarendon Press, 2006.
[23] L. Henkin, Some remarks on infinitely long formulas, in: Infinitistic Methods: Proceedings of the Symposium on Foundations of Mathematics, Warsaw, 1959, Panstwowe Wydawnictwo Naukowe / Pergamon Press, Warsaw, 1961, pp. 167–183.
[24] J. Hintikka, G. Sandu, Game-theoretical semantics, in: J. van Benthem, A. ter Meulen (Eds.), Handbook of Logic and Language, North-Holland Elsevier, Amsterdam, 1996, pp. 361–410.
[25] J. Szymanik, Branching Quantifiers, Springer International Publishing, Cham, 2016, pp. 143–162. doi:10.1007/978-3-319-28749-2_9.
[26] S. Hetzl, A. Leitsch, D. Weller, CERES in higher-order logic, Ann. Pure Appl. Log. 162 (2011) 1001–1034. doi:10.1016/J.APAL.2011.06.005.
[27] M. Baaz, A. Lolic, Towards a proof theory for Henkin quantifiers, J. Log. Comput. 31 (2021) 40–66. URL: https://doi.org/10.1093/logcom/exaa071.
[28] J. Avigad, R. Zach, The epsilon calculus, in: E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Center for the Study of Language and Information, 2008. URL: http://plato.stanford.edu/.
[29] S. Chatzikyriakidis, F. Pasquali, C. Retoré, From logical and linguistic generics to Hilbert's tau and epsilon quantifiers, IfCoLog Journal of Logics and their Applications 4 (2017) 231–255.