Lecture Notes in Artificial Intelligence 1847

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G.Goos, J. Hartmanis, and J. van Leeuwen

Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Roy Dyckhoff (Ed.)

Automated Reasoning with Analytic Tableaux and Related Methods

International Conference, TABLEAUX 2000
St Andrews, Scotland, UK, July 3-7, 2000
Proceedings

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editor

Roy Dyckhoff
University of St Andrews, School of Computer Science
North Haugh, St Andrews, Fife, KY16 9SS, Scotland
E-mail: rd@dcs.st-and.ac.uk

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Automated reasoning with analytic tableaux and related methods :
international conference ; tableaux 2000, St. Andrews, Scotland, UK,
July 3 - 7, 2000 ; proceedings / Roy Dyckhoff (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ;
Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1847 : Lecture notes in
artificial intelligence)
ISBN 3-540-67697-X

CR Subject Classification (1998): I.2.3, F.4.1

ISBN 3-540-67697-X Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10722086 06/3142 5 4 3 2 1 0

Foreword

This volume contains the main papers presented at the International Conference on Analytic Tableaux and Related Methods (TABLEAUX 2000) held on July 3–7, 2000 in St Andrews, Scotland. This conference succeeded other meetings on the same topic held in Lautenbach (1992), Marseille (1993), Abingdon (1994), St Goar (1995), Terrasini (1996), Pont-à-Mousson (1997), Oisterwijk (1998) and Saratoga Springs (1999).

Tableaux and related methods, such as Gentzen calculi, are convenient and effective for automating deduction not just in classical logic but also in various non-standard logics. Examples taken from this meeting alone include temporal, description, non-monotonic, tense, modal, epistemic, fuzzy and intuitionistic logics. Areas of application include verification of software and computer systems, deductive databases, knowledge representation and system diagnosis. The conference brought together researchers interested in all aspects – theoretical foundations, implementation techniques, systems development, experimental comparison and applications – of the automation of reasoning by means of tableaux or related methods.

Each research paper was formally evaluated by three referees, mainly members of the programme committee. From the 42 submissions received, 23 original *research papers* and 2 original *system descriptions* were chosen by the programme committee for presentation at the conference and for inclusion in these proceedings, together with the 3 invited lectures. Also included are a summary of the non-classical systems *Comparison* (*TANCS*), descriptions of the Comparison entries, and the titles and authors of *position papers* and *tutorials*, which were also presented at the conference. Eight systems designers expressed interest for *TANCS*: two abandoned submission plans because of inefficiency and one because of unsoundness. All submitted systems were installed (a most painful process) and checked for soundness: submitted data were sampled for coherence.

Acknowledgements First, I thank my colleagues Andrew Adams, Helen Bremner, Martin Dunstan, Brian McAndie and Joy Thomson, who helped with secretarial and technical aspects of the conference, Second, I thank my students Hanne Gottliebsen and Mairead Ní Eineachain for their practical help. Third, I thank the PC members and other referees for their tireless work reviewing papers. In particular, I thank Fabio Massacci (for his energetic organisation of the Comparison) and Neil Murray (for helpful advice and information about last year's meeting). Fourth, I thank Peter Schmitt for his encouragement to organise the meeting at St Andrews and for his guiding role in the development of the TABLEAUX series. Finally, I thank the authors (e.g. for their patience with my strictures about style and typography), the speakers, the tutorial organisers, the Comparison entrants, and, last but not least, the sponsors.

April 2000 Roy Dyckhoff

Previous Tableaux Workshops/Conferences

1992	Lautenbach, Germany	1993	Marseille, France
1994	Abingdon, England	1995	St. Goar, Germany
1996	Terrasini, Italy	1997	Pont-à-Mousson, France
1998	Oisterwijk, The Netherlands	1999	Saratoga Springs, USA

Invited Speakers

Franz Baader	RWTH Aachen, Germany
Melvin Fitting	CUNY, New York City, U.S.A.
Alasdair Urquhart	Toronto University, Canada

Programme Chair & Local Arrangements

Roy Dyckhoff
University of St Andrews, St Andrews, Scotland

Programme Committee

M. D'Agostino	University of Ferrara, Italy
N. Arai	Hiroshima City University, Japan
P. Baumgartner	University of Koblenz, Germany
B. Beckert	University of Karlsruhe, Germany
K. Broda	Imperial College, London, England
R. Dyckhoff	St Andrews University, Scotland
A. Felty	University of Ottawa, Canada
C. Fermüller	Technical University of Vienna, Austria
U. Furbach	University of Koblenz, Germany
D. Galmiche	LORIA, Nancy, France
R. Goré	Australian National University, Australia
J. Goubault-Larrecq	GIE Dyade, France
R. Hähnle	Chalmers University, Sweden
J. Hodas	Harvey Mudd College, California, U.S.A.
I. Horrocks	Manchester University, England
C. Kreitz	Cornell University, U.S.A.
R. Letz	Technical University of Munich, Germany
F. Massacci	Siena University, Siena, Italy
D. Miller	Pennsylvania State University, U.S.A.
U. Moscato	University of Milan, Italy
N. Murray	University at Albany - SUNY, U.S.A.
J. Pitt	Imperial College, London, England
H. Wansing	Dresden University of Technology, Germany

Referees

Each submitted paper was refereed by three members of the programme committee. In some cases, they consulted specialists who were not on the committee. We gratefully mention their names.

Wolfgang Ahrendt
Alessandro Avellone
Vincent Balat
Ross Brady
Juergen Dix
Mauro Ferrari
Guido Fiorino
Tim Geisler
Ian Gent
Guido Governatori
Andreas Jakoby
Gerhard Lakemeyer
Dominique Larchey-Wendling
Mario Ornaghi
Greg Restall
Mark Reynolds
Gernot Salzer
Renate Schmidt
Viorica Sofronie-Stokkermans
Alasdair Urquhart

Sponsors

British Logic Colloquium
Compulog Network
London Mathematical Society

Abstracts of Tutorials

The regular conference program included the presentation of 2 tutorials: details appear in the same volume as the "Position Papers" (see below).

Empirical Methods for Artificial Intelligence and Computer Science
Paul Cohen, Ian Gent and Toby Walsh

Rasiowa-Sikorski Deduction Systems: Foundations and Applications in CS Logics
Beata Konikowska

Position Papers

The regular conference program included the presentation of 8 short position papers. Informal proceedings containing these papers appeared as the internal scientific report "Position Papers, TABLEAUX 2000", School of Computer Science, University of St Andrews, St Andrews, Scotland.

How to find symmetries hidden in combinatorial problems.
Noriko H. Arai and Ryuji Masukawa

Labelled modal sequents.
Guido Governatori and Antonino Rotolo

Multiple sequent calculus for tense logics.
Andrzej Indrzejczak

Minimal model generation with factorization and constrained search.
Miyuki Koshimura, Megumi Kita and Ryuzo Hasegawa

Inference for non-Horn regular multiple-valued logics.
James J. Lu, Neil V. Murray and Erik Rosenthal

Termination in intuitionistic connection-driven search.
Arild Waaler

Decidable relevant logic ER and its tableau method based decision procedure.
Noriaki Yoshiura and Naoki Yonezaki

IPAL: An interactive prover for algorithmic logic.
Anna Zalewska

Table of Contents

Invited Lectures

Comparison

Research Papers

System Descriptions

Tableau Algorithms for Description Logics

Franz Baader and Ulrike Sattler

Theoretical Computer Science, RWTH Aachen, Germany

Abstract. Description logics are a family of knowledge representation formalisms that are descended from semantic networks and frames via the system KL-ONE. During the last decade, it has been shown that the important reasoning problems (like subsumption and satisfiability) in a great variety of description logics can be decided using tableau-like algorithms. This is not very surprising since description logics have turned out to be closely related to propositional modal logics and logics of programs (such as propositional dynamic logic), for which tableau procedures have been quite successful.
Nevertheless, due to different underlying intuitions and applications, most description logics differ significantly from run-of-the-mill modal and program logics. Consequently, the research on tableau algorithms in description logics led to new techniques and results, which are, however, also of interest for modal logicians. In this article, we will focus on three features that play an important rôle in description logics (number restrictions, terminological axioms, and role constructors), and show how they can be taken into account by tableau algorithms.

1 Introduction

Description logics (DLs) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames [44,37], in that they are equipped with a formal, *logic*-based semantics, which can, e.g., be given by a translation into first-order predicate logic.

Knowledge representation systems based on description logics (DL systems) provide their users with various inference capabilities that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* algorithm allows one to determine subconcept-superconcept relationships: C is subsumed by D iff all instances of C are also instances of D, i.e., the first description is always interpreted as a subset of the second description. In order to ensure a reasonable and predictable behaviour of a DL system, the subsumption problem for the DL employed by the system should at least be

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 1–18, 2000.

decidable, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues in DL research. Roughly, this research can be classified into the following four phases.

Phase 1: First system implementations. The original KL-ONE system [12] as well as its early successor systems (such as BACK [43], K-REP [36], and LOOM [35]) employ so-called structural subsumption algorithms, which first normalise the concept descriptions, and then recursively compare the syntactic structure of the normalised descriptions (see, e.g., [38] for the description of such an algorithm). These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption relationships (though this fact was not necessarily known to the designers of the early systems).

Phase 2: First complexity and undecidability results. Partially in parallel with the first phase, the first formal investigations of the subsumption problem in DLs were carried out. It turned out that (under the assumption P $\neq$ NP) already quite inexpressive DLs cannot have polynomial subsumption algorithms [10,39], and that the DL used by the KL-ONE system even has an undecidable subsumption problem [49]. In particular, these results showed the incompleteness of the (polynomial) structural subsumption algorithms. One reaction to these results (e.g., by the designers of BACK and LOOM) was to call the incompleteness of the subsumption algorithm a feature rather than a bug of the DL system. The designers of the CLASSIC system [42,9] followed another approach: they carefully chose a restricted DL that still allowed for an (almost) complete polynomial structural subsumption algorithm [8].

Phase 3: Tableau algorithms for expressive DLs and thorough complexity analysis. For expressive DLs (in particular, DLs allowing for disjunction and/or negation), for which the structural approach does not lead to complete subsumption algorithms, tableau algorithms have turned out to be quite useful: they are complete and often of optimal (worst-case) complexity. The first such algorithm was proposed by Schmidt-Schauß and Smolka [50] for a DL that they called $\mathcal{ALC}$ (for "attributive concept description language with complements").[1] It quickly turned out that this approach for deciding subsumption could be extended to various other DLs [28,26,4,1,23] and also to other inference problems such as the instance problem [24]. Early on, DL researchers started to call the algorithms obtained this way "tableau-based algorithms" since they observed that the original algorithm by Schmidt-Schauß and Smolka for $\mathcal{ALC}$, as well as subsequent algorithms for more expressive DL, could be seen as specialisations of the tab-

[1] Actually, at that time the authors were not aware of the close connection between their rule-based algorithm working on constraint systems and tableau procedures for modal and first-order predicate logics.

leau calculus for first-order predicate logic (the main problem to solve was to find a specialisation that always terminates, and thus yields a decision procedure). After Schild [47] showed that $\mathcal{ALC}$ is just a syntactic variant of multi-modal K, it turned out that the algorithm by Schmidt-Schauß and Smolka was actually a re-invention of the known tableau algorithm for K.

At the same time, the (worst-case) complexity of a various DLs (in particular also DLs that are not propositionally closed) was investigated in detail [20,21,19].

The first DL systems employing tableau algorithms (KRIS [5] and CRACK [13]) demonstrated that (in spite of their high worst-case complexity) these algorithms lead to acceptable behaviour in practice [6]. Highly optimised systems such as FaCT [30] have an even better behaviour, also for benchmark problems in modal logics [29,31].

Phase 4: Algorithms and efficient systems for very expressive DLs. Motivated by applications (e.g., in the database area), DL researchers started to investigate DLs whose expressive power goes far beyond the one of $\mathcal{ALC}$ (e.g., DLs that do not have the finite model property). First decidability and complexity results for such DLs could be obtained from the connection between propositional dynamic logic (PDL) and DLs [47]. The idea of this approach, which was perfected by DeGiacomo and Lenzerini, is to translate the DL in question into PDL. If the translation is polynomial and preserves satisfiability, then the known EXPTIME-algorithms for PDL can be employed to decide subsumption in exponential time. Though this approach has produced very strong complexity results [16,17,18] it turned out to be less satisfactory from a practical point of view. In fact, first tests in a database application [33] showed that the PDL formulae obtained by the translation technique could not be handled by existing efficient implementations of satisfiability algorithms for PDL [41]. To overcome this problem, DL researchers have started to design "practical" tableau algorithms for *very* expressive DLs [32,33].

The purpose of this article is to give an impression of the work on tableau algorithms done in the DL community, with an emphasis on features that, though they may also occur in modal logics, are of special interest to description logics. After introducing some basic notions of description logics in Section 2, we will describe a tableau algorithm for $\mathcal{ALC}$ in Section 3. Although, from the modal logic point of view, this is just the well-known algorithm for multi-modal K, this section will introduce the notations and techniques used in description logics, and thus set the stage for extensions to more interesting DLs. In the subsequent three section we will show how the basic algorithm can be extended to one that treats number restrictions, terminological axioms, and role constructors of different expressiveness, respectively.

2 Description Logics: Basic Definitions

The main expressive means of description logics are so-called concept descriptions, which describe sets of individuals or objects. Formally, *concept descriptions* are inductively defined with the help of a set of *concept constructors*, starting

Table 1. Syntax and semantics of concept descriptions.

Construct name	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
at-least restriction	$(\geqslant n r.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
at-most restriction	$(\leqslant n r.C)$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

with a set N_C of *concept names* and a set N_R of *role names*. The available constructors determine the expressive power of the DL in question. In this paper, we consider concept descriptions built from the constructors shown in Table 1, where C, D stand for concept descriptions, r for a role name, and n for a non-negative integer. In the description logic $\mathcal{ALC}$, concept descriptions are formed using the constructors negation, conjunction, disjunction, value restriction, and existential restriction. The description logic $\mathcal{ALCQ}$ additionally provides us with (qualified) at-least and at-most *number restrictions*.

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $P \in N_C$ to a set $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

From the modal logic point of view, roles are simply names for accessibility relations, and existential (value) restrictions correspond to diamonds (boxes) indexed by the respective accessibility relation. Thus, any $\mathcal{ALC}$ description can be translated into a multi-modal formula and vice versa. For example, the description $P \sqcap \exists r.P \sqcap \forall r.\neg P$ corresponds to the formula $p \wedge \langle r \rangle p \wedge [r]\neg p$, where p is an atomic proposition corresponding to the concept name P. As pointed out by Schild [47], there is an obvious correspondence between the semantics of $\mathcal{ALC}$ and the Kripke semantics for multi-modal K, which satisfies $d \in C^{\mathcal{I}}$ iff the world d satisfies the formula ϕ_C corresponding to C in the Kripke structure corresponding to $\mathcal{I}$. Number restrictions also have a corresponding construct in modal logics, so-called graded modalities [53], but these are not as well-investigated as the modal logic K.

One of the most important inference services provided by DL systems is computing the subsumption hierarchy of a given finite set of concept descriptions.

Definition 1. *The concept description D* subsumes *the concept description C ($C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$; C is* satisfiable *iff there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$; and C and D are equivalent iff $C \sqsubseteq D$ and $D \sqsubseteq C$.*

In the presence of negation, subsumption can obviously be reduced to satisfiability: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable.[2]

Given concept descriptions that define the important notions of an application domain, one can then describe a concrete situation with the help of the assertional formalism of description logics.

Definition 2. *Let N_I be a set of* individual names. *An* ABox *is a finite set of assertions of the form $C(a)$ (*concept assertion*) or $r(a, b)$ (*role assertion*), where C is a concept description, r a role name, and a, b are individual names.*

An interpretation $\mathcal{I}$, which additionally assigns elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to individual names a, is a model *of an ABox $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ $((a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}})$ holds for all assertions $C(a)$ $(r(a, b))$ in $\mathcal{A}$.*

The Abox $\mathcal{A}$ is consistent *iff it has a model. The individual a is an* instance *of the description C w.r.t. $\mathcal{A}$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{A}$.*

Satisfiability (and thus also subsumption) of concept descriptions as well as the instance problem can be reduced to the consistency problem for ABoxes: (i) C is satisfiable iff the ABox $\{C(a)\}$ for some $a \in N_I$ is consistent; and (ii) a is an instance of C w.r.t. $\mathcal{A}$ iff $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent.

Usually, one imposes the unique name assumption on ABoxes, i.e., requires the mapping from individual names to elements of $\Delta^{\mathcal{I}}$ to be injective. Here, we dispense with this requirement since it has no effect for $\mathcal{ALC}$, and for DLs with number restrictions we will explicitly introduce inequality assertions, which can be used to express the unique name assumption.

3 A Tableau Algorithm for $\mathcal{ALC}$

Given an $\mathcal{ALC}$-concept description C_0, the tableau algorithm for satisfiability tries to construct a finite interpretation $\mathcal{I}$ that satisfies C_0, i.e., contains an element x_0 such that $x_0 \in C_0^{\mathcal{I}}$. Before we can describe the algorithm more formally, we need to introduce an appropriate data structure in which to represent (partial descriptions of) finite interpretations. The original paper by Schmidt-Schauß and Smolka [50], and also many other papers on tableau algorithms for DLs, introduce the new notion of a constraint system for this purpose. However, if we look at the information that must be expressed (namely, the elements of the interpretation, the concept descriptions they belong to, and their role relationships), we see that ABox assertions are sufficient for this purpose.

It will be convenient to assume that all concept descriptions are in *negation normal form* (NNF), i.e., that negation occurs only directly in front of concept names. Using de Morgan's rules and the usual rules for quantifiers, any $\mathcal{ALC}$-concept description can be transformed (in linear time) into an equivalent description in NNF.

Let C_0 be an $\mathcal{ALC}$-concept in NNF. In order to test satisfiability of C_0, the algorithm starts with $\mathcal{A}_0 := \{C_0(x_0)\}$, and applies consistency preserving

[2] This was the reason why Schmidt-Schauß and Smolka [50] added negation to their DL in the first place.

The $\rightarrow_{\sqcap}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x), C_2(x)\}$.

The $\rightarrow_{\sqcup}$-rule
Condition: $\mathcal{A}$ contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{C_2(x)\}$.

The $\rightarrow_{\exists}$-rule
Condition: $\mathcal{A}$ contains $(\exists r.C)(x)$, but there is no individual name z such that $C(z)$ and $r(x, z)$ are in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y), r(x, y)\}$ where y is an individual name not occurring in $\mathcal{A}$.

The $\rightarrow_{\forall}$-rule
Condition: $\mathcal{A}$ contains $(\forall r.C)(x)$ and $r(x, y)$, but it does not contain $C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$.

Fig. 1. Transformation rules of the satisfiability algorithm for $\mathcal{ALC}$.

transformation rules (see Fig. 1) to this ABox. The transformation rule that handles disjunction is *nondeterministic* in the sense that a given ABox is transformed into two new ABoxes such that the original ABox is consistent iff *one of* the new ABoxes is so. For this reason we will consider finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ instead of single ABoxes. Such a set is *consistent* iff there is some i, $1 \leq i \leq k$, such that $\mathcal{A}_i$ is consistent. A rule of Fig. 1 is applied to a given finite set of ABoxes $\mathcal{S}$ as follows: it takes an element $\mathcal{A}$ of $\mathcal{S}$, and replaces it by one ABox $\mathcal{A}'$ or by two ABoxes $\mathcal{A}'$ and $\mathcal{A}''$.

Definition 3. *An ABox $\mathcal{A}$ is called* complete *iff none of the transformation rules of Fig. 1 applies to it. The ABox $\mathcal{A}$ contains a* clash *iff $\{P(x), \neg P(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name P. An ABox is called* closed *if it contains a clash, and* open *otherwise.*

The *satisfiability algorithm for $\mathcal{ALC}$* works as follows. It starts with the singleton set of ABoxes $\{\{C_0(x_0)\}\}$, and applies the rules of Fig. 1 (in arbitrary order) until no more rules apply. It answers "satisfiable" if the set $\widehat{\mathcal{S}}$ of ABoxes obtained this way contains an open ABox, and "unsatisfiable" otherwise. Correctness of this algorithm is an easy consequence of the following lemma.

Lemma 1. *Let C_0 be an $\mathcal{ALC}$-concept in negation normal form.*

1. *There cannot be an infinite sequence of rule applications*

$$\{\{C_0(x_0)\}\} \rightarrow \mathcal{S}_1 \rightarrow \mathcal{S}_2 \rightarrow \cdots .$$

2. *Assume that $\mathcal{S}'$ is obtained from the finite set of ABoxes $\mathcal{S}$ by application of a transformation rule. Then $\mathcal{S}$ is consistent iff $\mathcal{S}'$ is consistent.*

3. *Any closed ABox $\mathcal{A}$ is inconsistent.*
4. *Any complete and open ABox $\mathcal{A}$ is consistent.*

The first part of this lemma (termination) is an easy consequence of the facts that (i) all concept assertions occurring in an ABox in one of the sets $\mathcal{S}_i$ are of the form $C(x)$ were C is a subdescription of C_0; and (ii) if an ABox in $\mathcal{S}_i$ contains the role assertion $r(x, y)$, then the maximal role depth (i.e., nesting of value and existential restrictions) of concept descriptions occurring in concept assertions for y is strictly smaller than the maximal role depth of concept descriptions occurring in concept assertions for x. A detailed proof of termination (using an explicit mapping into a well-founded ordering) for a set of rules extending the one of Fig. 1 can, e.g., be found in [4].

The second and third part of the lemma are quite obvious, and the fourth part can be proved by defining the *canonical interpretation* $\mathcal{I}_\mathcal{A}$ induced by $\mathcal{A}$:

1. The domain $\Delta^{\mathcal{I}_\mathcal{A}}$ of $\mathcal{I}_\mathcal{A}$ consists of all the individual names occurring in $\mathcal{A}$.
2. For all concept names P we define $P^{\mathcal{I}_\mathcal{A}} := \{x \mid P(x) \in \mathcal{A}\}$.
3. For all role names r we define $r^{\mathcal{I}_\mathcal{A}} := \{(x, y) \mid r(x, y) \in \mathcal{A}\}$.

By definition, $\mathcal{I}_\mathcal{A}$ satisfies all the role assertions in $\mathcal{A}$. By induction on the structure of concept descriptions, it is easy to show that it satisfies the concept assertions as well, provided that $\mathcal{A}$ is complete and open.

It is also easy to show that the canonical interpretation has the shape of a finite tree whose depth is linearly bounded by the size of C_0 and whose branching factor is bounded by the number of different existential restrictions in C_0. Consequently, $\mathcal{ALC}$ has the *finite tree model property*, i.e., any satisfiable concept C_0 is satisfiable in a finite interpretation $\mathcal{I}$ that has the shape of a tree whose root belongs to C_0.

To sum up, we have seen that the transformation rules of Fig. 1 reduce satisfiability of an $\mathcal{ALC}$-concept C_0 (in NNF) to consistency of a finite set $\widehat{\mathcal{S}}$ of complete ABoxes. In addition, consistency of $\widehat{\mathcal{S}}$ can be decided by looking for obvious contradictions (clashes).

Theorem 1. *It is decidable whether or not an $\mathcal{ALC}$-concept is satisfiable.*

Complexity issues. The satisfiability algorithm for $\mathcal{ALC}$ presented above may need exponential time and space. In fact, the size of the complete and open ABox (and thus of the canonical interpretation) built by the algorithm may be exponential in the size of the concept description. For example, consider the descriptions C_n ($n \geq 1$) that are inductively defined as follows:

$$C_1 := \exists r.A \sqcap \exists r.B,$$
$$C_{n+1} := \exists r.A \sqcap \exists r.B \sqcap \forall r.C_n.$$

Obviously, the size of C_n grows linearly in n. However, given the input description C_n, the satisfiability algorithm generates a complete and open ABox whose

canonical interpretation is a binary tree of depth n, and thus consists of $2^{n+1}-1$ individuals.

Nevertheless, the algorithm can be modified such that it needs only polynomial space. The main reason is that different branches of the tree model to be generated by the algorithm can be investigated separately, and thus the tree can be built and searched in a depth-first manner. Since the complexity class NPSPACE coincides with PSPACE [46], it is sufficient to describe a nondeterministic algorithm using only polynomial space, i.e., for the nondeterministic $\rightarrow_\sqcup$-rule, we may simply assume that the algorithm chooses the correct alternative. In principle, the modified algorithm works as follows: it starts with $\{C_0(x_0)\}$ and

1. applies the $\rightarrow_\sqcap$- and $\rightarrow_\sqcup$-rules as long as possible and checks for clashes;
2. generates all the necessary direct successors of x_0 using the $\rightarrow_\exists$-rule and exhaustively applies the $\rightarrow_\forall$-rule to the corresponding role assertions;
3. successively handles the successors in the same way.

Since the successors of a given individual can be treated separately, the algorithm needs to store only one path of the tree model to be generated, together with the *direct* successors of the individuals on this path and the information which of these successors must be investigated next. Since the length of the path is linear in the size of the input description C_0, and the number of successors is bounded by the number of different existential restrictions in C_0, the necessary information can obviously be stored within polynomial space.

This shows that the satisfiability problem for $\mathcal{ALC}$-concept descriptions is in PSPACE. PSPACE-hardness can be shown by a reduction from validity of Quantified Boolean Formulae [50].

Theorem 2. *Satisfiability of $\mathcal{ALC}$-concept descriptions is PSPACE-complete.*

The consistency problem for $\mathcal{ALC}$-ABoxes. The satisfiability algorithm described above can also be used to decide consistency of $\mathcal{ALC}$-ABoxes. Let $\mathcal{A}_0$ be an $\mathcal{ALC}$-ABox such that (w.l.o.g.) all concept descriptions in $\mathcal{A}$ are in NNF. To test $\mathcal{A}_0$ for consistency, we simply apply the rules of Fig. 1 to the singleton set $\{\mathcal{A}_0\}$. It is easy to show that Lemma 1 still holds. Indeed, the only point that needs additional consideration is the first one (termination). Thus, the rules of Fig. 1 yield a decision procedure for consistency of $\mathcal{ALC}$-ABoxes.

Since now the canonical interpretation obtained from a complete and open ABox need no longer be of tree shape, the argument used to show that the satisfiability problem is in PSPACE cannot directly be applied to the consistency problem. In order to show that the consistency problem is in PSPACE, one can, however, proceed as follows: In a *pre-completion* step, one applies the transformation rules only to *old* individuals (i.e., individuals present in the original ABox $\mathcal{A}_0$). Subsequently, one can forget about the role assertions, i.e., for each individual name in the pre-completed ABox, the satisfiability algorithm is applied to the conjunction of its concept assertions (see [25] for details).

Theorem 3. *Consistency of $\mathcal{ALC}$-ABoxes is PSPACE-complete.*

4 Number Restrictions

Before treating the qualified number restrictions introduced in Section 2, we consider a restricted form of number restrictions, which is the form present in most DL systems. In *unqualified* number restrictions, the qualifying concept is the top concept $\top$, where $\top$ is an abbreviation for $P \sqcup \neg P$, i.e., a concept that is always interpreted by the whole interpretation domain. Instead of $(\geqslant nr.\top)$ and $(\leqslant nr.\top)$, we write unqualified number restrictions simply as $(\geqslant nr)$ and $(\leqslant nr)$. The DL that extends $\mathcal{ALC}$ by unqualified number restrictions is denoted by $\mathcal{ALCN}$.

Obviously, $\mathcal{ALCN}$- and $\mathcal{ALCQ}$-concept descriptions can also be transformed into NNF in linear time.

4.1 A Tableau Algorithm for $\mathcal{ALCN}$

The main idea underlying the extension of the tableau algorithm for $\mathcal{ALC}$ to $\mathcal{ALCN}$ is quite simple. At-least restrictions are treated by generating the required role successors as new individuals. At-most restrictions that are currently violated are treated by (nondeterministically) identifying some of the role successors. To avoid running into a generate-identify cycle, we introduce explicit inequality assertions that prohibit the identification of individuals that were introduced to satisfy an at-least restriction.

Inequality assertions are of the form $x \neq y$ for individual names x, y, with the obvious semantics that an interpretation $\mathcal{I}$ satisfies $x \neq y$ iff $x^{\mathcal{I}} \neq y^{\mathcal{I}}$. These assertions are assumed to be symmetric, i.e., saying that $x \neq y$ belongs to an ABox $\mathcal{A}$ is the same as saying that $y \neq x$ belongs to $\mathcal{A}$.

The *satisfiability algorithm* for $\mathcal{ALCN}$ is obtained from the one for $\mathcal{ALC}$ by adding the rules in Fig. 2, and by considering a second type of *clashes*:

- $\{(\leqslant nr)(x)\} \cup \{r(x, y_i) \mid 1 \leq i \leq n+1\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n+1\} \subseteq \mathcal{A}$ for $x, y_1, \ldots, y_{n+1} \in N_I$, $r \in N_R$, and a nonnegative integer n.

The nondeterministic $\rightarrow_{\leq}$-rule replaces the ABox $\mathcal{A}$ by finitely many new ABoxes $\mathcal{A}_{i,j}$. Lemma 1 still holds for the extended algorithm (see e.g. [7], where this is proved for a more expressive DL). This shows that satisfiability (and thus also subsumption) of $\mathcal{ALCN}$-concept descriptions is decidable.

Complexity issues. The ideas that lead to a PSPACE algorithm for $\mathcal{ALC}$ can be applied to the extended algorithm as well. The only difference is that, before handling the successors of an individual (introduced by at-least and existential restrictions), one must check for clashes of the second type and generate the necessary identifications. However, this simple extension only leads to a PSPACE algorithm if we assume the numbers in at-least restrictions to be written in base 1 representation (where the size of the representation coincides with the number represented). For bases larger than 1 (e.g., numbers in decimal notation), the number represented may be exponential in the size of the representation. Thus,

The $\rightarrow_{\geq}$-rule
Condition: $\mathcal{A}$ contains $(\geqslant nr)(x)$, and there are no individual names $z_1, \ldots, z_n$ such that $r(x, z_i)$ $(1 \leq i \leq n)$ and $z_i \not\doteq z_j$ $(1 \leq i < j \leq n)$ are contained in $\mathcal{A}$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{r(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \not\doteq y_j \mid 1 \leq i < j \leq n\}$, where $y_1, \ldots, y_n$ are distinct individual names not occurring in $\mathcal{A}$.

The $\rightarrow_{\leq}$-rule
Condition: $\mathcal{A}$ contains distinct individual names $y_1, \ldots, y_{n+1}$ such that $(\leqslant nr)(x)$ and $r(x, y_1), \ldots, r(x, y_{n+1})$ are in $\mathcal{A}$, and $y_i \not\doteq y_j$ is not in $\mathcal{A}$ for some $i \neq j$.
Action: For each pair y_i, y_j such that $i < j$ and $y_i \not\doteq y_j$ is not in $\mathcal{A}$, the ABox $\mathcal{A}_{i,j} := [y_i/y_j]\mathcal{A}$ is obtained from $\mathcal{A}$ by replacing each occurrence of y_i by y_j.

Fig. 2. The transformation rules handling unqualified number restrictions.

we cannot introduce all the successors required by at-least restrictions while only using space polynomial in the size of the concept description if the numbers in this description are written in decimal notation.

It is not hard to see, however, that most of the successors required by the at-least restrictions need not be introduced at all. If an individual x obtains at least one r-successor due to the application of the $\rightarrow_{\exists}$-rule, then the $\rightarrow_{\geq}$-rule need not be applied to x for the role r. Otherwise, we simply introduce *one* r-successor as representative. In order to detect inconsistencies due to conflicting number restrictions, we need to add *another type of clashes:* $\{(\leqslant nr)(x), (\geqslant mr)(x)\} \subseteq \mathcal{A}$ for nonnegative integers $n < m$. The canonical interpretation obtained by this modified algorithm need not satisfy the at-least restrictions in C_0. However, it can easily by modified to an interpretation that does, by duplicating r-successors (more precisely, the whole subtrees starting at these successors).

Theorem 4. *Satisfiability of $\mathcal{ALCN}$-concept descriptions is PSPACE-complete.*

The consistency problem for $\mathcal{ALCN}$-ABoxes. Just as for $\mathcal{ALC}$, the extended rule set for $\mathcal{ALCN}$ can also be applied to arbitrary ABoxes. Unfortunately, the algorithm obtained this way need *not terminate*, unless one imposes a specific strategy on the order of rule applications. For example, consider the ABox

$$\mathcal{A}_0 := \{r(a, a),\ (\exists r.P)(a),\ (\leqslant 1r)(a),\ (\forall r.\exists r.P)(a)\}.$$

By applying the $\rightarrow_{\exists}$-rule to a, we can introduce a new r-successor x of a:

$$\mathcal{A}_1 := \mathcal{A}_0 \cup \{r(a, x),\ P(x)\}.$$

The $\rightarrow_{\forall}$-rule adds the assertion $(\exists r.P)(x)$, which triggers an application of the $\rightarrow_{\exists}$-rule to x. Thus, we obtain the new ABox

$$\mathcal{A}_2 := \mathcal{A}_1 \cup \{(\exists r.P)(x),\ r(x, y),\ P(y)\}.$$

The $\rightarrow_{\text{choose}}$-rule
Condition: $\mathcal{A}$ contains $(\leqslant nr.C)(x)$ and $r(x,y)$, but neither $C(y)$ nor $\neg C(y)$.
Action: $\mathcal{A}' := \mathcal{A} \cup \{C(y)\}$, $\mathcal{A}'' := \mathcal{A} \cup \{\neg C(y)\}$.

Fig. 3. The $\rightarrow_{\text{choose}}$-rule for qualified number restrictions.

Since a has two r-successors in $\mathcal{A}_2$, the $\rightarrow_{\leq}$-rule is applicable to a. By replacing every occurrence of x by a, we obtain the ABox

$$\mathcal{A}_3 := \mathcal{A}_0 \cup \{P(a),\, r(a,y),\, P(y)\}.$$

Except for the individual names (and the assertion $P(a)$, which is, however, irrelevant), $\mathcal{A}_3$ is identical to $\mathcal{A}_1$. For this reason, we can continue as above to obtain an infinite chain of rule applications.

We can easily regain termination by requiring that *generating rules* (i.e., the rules $\rightarrow_\exists$ and $\rightarrow_\geq$) may only be applied if none of the other rules is applicable. In the above example, this strategy would prevent the application of the $\rightarrow_\exists$-rule to x in the ABox $\mathcal{A}_1 \cup \{(\exists r.P)(x)\}$ since the $\rightarrow_{\leq}$-rule is also applicable. After applying the $\rightarrow_{\leq}$-rule (which replaces x by a), the $\rightarrow_\exists$-rule is no longer applicable since a already has an r-successor that belongs to P.

In order to obtain a PSPACE algorithm for consistency of $\mathcal{ALCN}$-ABoxes, the pre-completion technique sketched above for $\mathcal{ALC}$ can also be applied to $\mathcal{ALCN}$ [25].

Theorem 5. *Consistency of $\mathcal{ALCN}$-ABoxes is PSPACE-complete.*

4.2 A Tableau Algorithm for $\mathcal{ALCQ}$

An obvious idea when attempting to extend the satisfiability algorithm for $\mathcal{ALCN}$ to one that can handle $\mathcal{ALCQ}$ is the following (see [53]):

- Instead of simply generating n new r-successors $y_1, \ldots, y_n$ in the $\rightarrow_\geq$-rule, one also asserts that these individuals must belong to the qualifying concept C by adding the assertions $C(y_i)$ to $\mathcal{A}'$.
- The $\rightarrow_{\leq}$-rule only applies if $\mathcal{A}$ also contains the assertions $C(y_i)$ $(1 \leq i \leq n+1)$.

Unfortunately, this does not yield a correct algorithm for satisfiability in $\mathcal{ALCQ}$. In fact, this simple algorithm would not detect that the concept description $(\geqslant 3r) \sqcap (\leqslant 1r.P) \sqcap (\leqslant 1r.\neg P)$ is unsatisfiable. The (obvious) problem is that, for some individuals a and concept descriptions C, the ABox may neither contain $C(a)$ nor $\neg C(a)$, whereas in the canonical interpretation constructed from the ABox, one of the two must hold. In order to overcome this problem, the nondeterministic $\rightarrow_{\text{choose}}$-rule of Fig. 3 must be added [26]. Together with the $\rightarrow_{\text{choose}}$-rule, the simple modification of the $\rightarrow_\geq$- and $\rightarrow_{\leq}$-rule described above yields a correct algorithm for satisfiability in $\mathcal{ALCQ}$ [26].

Complexity issues. The approach that leads to a PSPACE-algorithm for $\mathcal{ALC}$ can be applied to the algorithm for $\mathcal{ALCQ}$ as well. However, as with $\mathcal{ALCN}$, this yields a PSPACE-algorithm only if the numbers in number restrictions are assumed to be written in base 1 representation. For $\mathcal{ALCQ}$, the idea that leads to a PSPACE-algorithm for $\mathcal{ALCN}$ with decimal notation does no longer work: it is not sufficient to introduce just one successor as representative for the role successors required by at-least restrictions. Nevertheless, it is possible to design a PSPACE-algorithm for $\mathcal{ALCQ}$ also w.r.t. decimal notation of numbers [52]. Like the PSPACE-algorithm for $\mathcal{ALC}$, this algorithm treats the successors separately. It uses appropriate counters (and a new type of clashes) to check whether qualified number restrictions are satisfied. By combining the pre-completion approach of [25] with this algorithm, we also obtain a PSPACE-result for consistency of $\mathcal{ALCQ}$-ABoxes.

Theorem 6. *Satisfiability of $\mathcal{ALCQ}$-concept descriptions as well as consistency of $\mathcal{ALCQ}$-ABoxes are PSPACE-complete problems.*

5 Terminological Axioms

DLs systems usually provide their users also with a terminological formalism. In its simplest form, this formalism can be used to introduce names for complex concept descriptions. More general terminological formalisms can be used to state connections between complex concept descriptions.

Definition 4. *A* TBox *is a finite set of terminological axioms of the form $C \doteq D$, where C, D are concept descriptions. The terminological axiom $C \doteq D$ is called* concept definition *iff C is a concept name.*

An interpretation $\mathcal{I}$ is a model of the TBox $\mathcal{T}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for all terminological axioms $C \doteq D$ in $\mathcal{T}$.

The concept description D subsumes *the concept description C* w.r.t. the TBox $\mathcal{T}$ *($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$; C is* satisfiable w.r.t. $\mathcal{T}$ *iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. The Abox $\mathcal{A}$ is* consistent w.r.t. $\mathcal{T}$ *iff it has a model that is also a model of $\mathcal{T}$. The individual a is an* instance of C w.r.t. $\mathcal{A}$ and $\mathcal{T}$ *iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for each model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$.*

In the following, we restrict our attention to terminological reasoning (i.e., the satisfiability and subsumption problem) w.r.t. TBoxes; however, the methods and results also apply to assertional reasoning (i.e., the instance and the consistency problem for ABoxes).

5.1 Acyclic Terminologies

The early DL systems provided TBoxes only for introducing names as abbreviations for complex descriptions. This is possible with the help of acyclic terminologies.

Definition 5. *A TBox is an* acyclic terminology *iff it is a set of concept definitions that neither contains multiple definitions nor cyclic definitions.* Multiple definitions *are of the form* $A \doteq C, A \doteq D$ *for distinct concept descriptions* C, D*, and* cyclic definitions *are of the form* $A_1 \doteq C_1, \ldots, A_n \doteq C_n$*, where* A_i *occurs in* C_{i-1} $(1 < i \leq n)$ *and* A_1 *occurs in* C_n*. If the acyclic terminology* $\mathcal{T}$ *contains a concept definition* $A \doteq C$*, then* A *is called* defined name *and* C *its* defining concept*.*

Reasoning w.r.t. *acyclic terminologies* can be reduced to reasoning without TBoxes by *unfolding* the definitions: this is achieved by repeatedly replacing defined names by their defining concepts until no more defined names occur. Unfortunately, unfolding may lead to an exponential blow-up, as the following acyclic terminology (due to Nebel [39]) demonstrates:

$$\{A_0 \doteq \forall r.A_1 \sqcap \forall s.A_1, \ldots, A_{n-1} \doteq \forall r.A_n \sqcap \forall s.A_n\}.$$

This terminology is of size linear in n, but unfolding applied to A_0 results in a concept description containing the name A_n 2^n times. Nebel [39] also shows that this complexity can, in general, not be avoided: for the DL $\mathcal{FL}_0$, which allows for conjunction and value restriction only, subsumption between concept descriptions can be tested in polynomial time, whereas subsumption w.r.t. acyclic terminologies is coNP-complete.

For more expressive languages, the presence of acyclic TBoxes may or may not increase the complexity of the subsumption problem. For example, subsumption of concept descriptions in the language $\mathcal{ALC}$ is PSPACE-complete, and so is subsumption w.r.t. acyclic terminologies [34]. Of course, in order to obtain a PSPACE-algorithm for subsumption in $\mathcal{ALC}$ w.r.t. acyclic terminologies, one cannot first apply unfolding to the concept descriptions to be tested for subsumption since this may need exponential space. The main idea is that one uses a tableau algorithm like the one described in Section 3, with the difference that it receives concept descriptions containing defined names as input. *Unfolding* is then done *on demand*: if the tableau algorithm encounters an assertion of the form $A(x)$, where A is a name occurring on the left-hand side of a definition $A \doteq C$ in the terminology, then it adds the assertion $C(x)$. However, it does not further unfold C at this stage. It is not hard to show that this really yields a PSPACE-algorithm for satisfiability (and thus also for subsumption) of concepts w.r.t. acyclic terminologies in $\mathcal{ALC}$ [34].

Theorem 7. *Satisfiability w.r.t. acyclic terminologies is PSPACE-complete in* $\mathcal{ALC}$*.*

Although this technique also works for many extensions of $\mathcal{ALC}$ (such as $\mathcal{ALCN}$ and $\mathcal{ALCQ}$), there are extensions for which it fails. One such example is the language $\mathcal{ALCF}$, which extends $\mathcal{ALC}$ by functional roles as well as agreements and disagreements on chains of functional roles (see, e.g., [34] for the definition of these constructors). Satisfiability of concept descriptions is PSPACE-complete for this DL [27], but satisfiability of concept descriptions w.r.t. acyclic terminologies is NEXPTIME-complete [34].

5.2 General TBoxes

For general terminological axioms of the form $C \doteq D$, where C may also be a complex description, unfolding is obviously no longer possible. Instead of considering finitely many such axiom $C_1 \doteq D_1, \ldots, C_n \doteq D_n$, it is sufficient to consider the single axiom $\widehat{C} \doteq \top$, where

$$\widehat{C} := (\neg C_1 \sqcup D_1) \sqcap (C_1 \sqcup \neg D_1) \sqcap \cdots \sqcap (\neg C_n \sqcup D_n) \sqcap (C_n \sqcup \neg D_n)$$

and $\top$ is an abbreviation for $P \sqcup \neg P$.

The axiom $\widehat{C} \doteq \top$ just says that any individual must belong to the concept $\widehat{C}$. The tableau algorithm for $\mathcal{ALC}$ introduced in Section 3 can easily be modified such that it takes this axiom into account: all individuals are simply asserted to belong to $\widehat{C}$. However, this modification may obviously lead to nontermination of the algorithm.

For example, consider what happens if this algorithm is applied to test consistency of the ABox $\mathcal{A}_0 := \{(\exists r.P)(x_0)\}$ modulo the axiom $\exists r.P \doteq \top$: the algorithm generates an infinite sequence of ABoxes $\mathcal{A}_1, \mathcal{A}_2, \ldots$ and individuals $x_1, x_2, \ldots$ such that $\mathcal{A}_{i+1} := \mathcal{A}_i \cup \{r(x_i, x_{i+1}),\ P(x_{i+1}),\ (\exists r.P)(x_{i+1})\}$. Since all individuals x_i $(i \geq 1)$ receive the same concept assertions as x_1, we may say that the algorithms has run into a cycle.

Termination can be regained by trying to detect such cyclic computations, and then blocking the application of generating rules: the application of the rule $\rightarrow_\exists$ to an individual x is *blocked* by an individual y in an ABox $\mathcal{A}$ iff $\{D \mid D(x) \in \mathcal{A}\} \subseteq \{D' \mid D'(y) \in \mathcal{A}\}$. The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new ones. For example, instead of generating a new r-successor for x_2 in the above example, one can simply use the r-successor of x_1. This yields an interpretation $\mathcal{I}$ with $\Delta^{\mathcal{I}} := \{x_0, x_1, x_2\}$, $P^{\mathcal{I}} := \{x_1, x_2\}$, and $r^{\mathcal{I}} := \{(x_0, x_1),\ (x_1, x_2), (x_2, x_2)\}$. Obviously, $\mathcal{I}$ is a model of both $\mathcal{A}_0$ and the axiom $\exists r.P \doteq \top$.

To avoid cyclic blocking (of x by y and vice versa), we consider an enumeration of all individual names, and define that an individual x may only be blocked by individuals y that occur before x in this enumeration. This, together with some other technical assumptions, makes sure that a tableau algorithm using this notion of blocking is sound and complete as well as terminating both for $\mathcal{ALC}$ and $\mathcal{ALCN}$ (see [14,2] for details).

Theorem 8. *Consistency of $\mathcal{ALCN}$-ABoxes w.r.t. TBoxes is decidable.*

It should be noted that the algorithm is no longer in PSPACE since it may generate role paths of exponential length before blocking occurs. In fact, even for the language $\mathcal{ALC}$, satisfiability modulo general terminological axioms is known to be EXPTIME-complete [48].

Blocking does not work for all extensions of $\mathcal{ALC}$ that have a tableau-based satisfiability algorithm. An example is again the DL $\mathcal{ALCF}$, for which satisfiability is decidable, but satisfiability w.r.t. general TBoxes undecidable [40,3].

6 Expressive Roles

The DLs considered until now allowed for atomic roles only. There are two ways of extending the expressivity of DLs w.r.t. roles: adding role constructors and allowing to constrain the interpretation of roles.

Role constructors can be used to build complex roles from atomic ones. In the following, we will restrict our attention to the inverse constructor, but other interesting role constructors have been considered in the literature (e.g., Boolean operators [15] or composition and transitive closure [1,47]). The *inverse* r^- of a role name r has the obvious semantics: $(r^-)^{\mathcal{I}} := \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\}$.

Constraining the interpretation of roles is very similar to imposing frame conditions in modal logics. One possible such constraint has already been mentioned in the previous section: in $\mathcal{ALCF}$ the interpretation of some roles is required to be functional. Here, we will consider transitive roles and role hierarchies. In a DL with *transitive roles*, a subset N_R^+ of the set of all role names N_R is fixed [45]. Elements of N_R^+ must be interpreted by transitive binary relations. (This corresponds to the frame condition for the modal logic K_4.) A *role hierarchy* is given by a finite set of role inclusion axioms of the form $r \sqsubseteq s$ for roles r, s. An interpretation $\mathcal{I}$ satisfies the role hierarchy $\mathcal{H}$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ holds for each $r \sqsubseteq s \in \mathcal{H}$.

DLs with transitive roles and role hierarchies have the nice property that reasoning w.r.t. TBoxes can be reduced to reasoning without TBoxes using a technique called internalisation [3,30,32]. Like in Section 5.2, we may assume that TBoxes are of the form $\mathcal{T} = \{\widehat{C} \doteq \top\}$. In $\mathcal{SH}$, the extension of $\mathcal{ALC}$ with transitive roles and role hierarchies, we introduce a new *transitive* role name u and assert in the role hierarchy that u is a super-role of all roles occurring in $\widehat{C}$ and the concept description C_0 to be tested for satisfiability. Then, C_0 is satisfiable w.r.t. $\mathcal{T}$ iff $C \sqcap \widehat{C} \sqcap \forall u.\widehat{C}$ is satisfiable. Extending this reduction to inverse roles consists simply in making u also a super-role of the inverse of each role occurring in $\widehat{C}$ or C_0 [32]. This reduction shows that a tableau algorithm for $\mathcal{SH}$ must also employ some sort of *blocking* to ensure termination (see Section 5.2).

Things become even more complex if we consider the DL $\mathcal{SHIF}$, which extends $\mathcal{SH}$ by the inverse of roles and functional roles. In fact, it is easy to show that $\mathcal{SHIF}$ no longer has the finite model property, i.e., there are satisfiable $\mathcal{SHIF}$-concept descriptions that are not satisfiable in a finite interpretation [32]. Instead of directly trying to construct an interpretation that satisfies C_0 (which might be infinite), the tableau algorithm for $\mathcal{SHIF}$ introduced in [32,33] first tries to construct a so-called *pre-model*, i.e., a structure that can be "unravelled" to a (possibly infinite) canonical (tree) interpretation. To ensure termination (without destroying correctness), the algorithm employs blocking techniques that are more sophisticated than the one described in Section 5.2. Interestingly, an optimised implementation of this algorithm in the system I-FaCT behaves quite well in realistic applications [33]. A refinement of the blocking techniques employed for $\mathcal{SHIF}$ can be used to prove that satisfiability in $\mathcal{SI}$ (i.e., the extension of $\mathcal{ALC}$ by transitive and inverse roles) is in PSPACE [51,33].

Finally, let us briefly comment on the difference between transitive roles and transitive closure of roles. Transitive closure is more expressive, but it appears that one has to pay dearly for this. In fact, whereas there exist quite efficient implementations for very expressive DLs with transitive roles, inverse roles, and role hierarchies (see above), no such implementations are known (to us) for closely related logics with transitive closure, such as converse-PDL (which is a notational variant of the extension of $\mathcal{ALC}$ by transitive closure, union, composition, and inverse of roles [47]). One reason could be that the known tableau algorithm for converse-PDL [22] requires a "cut" rule, which is massively nondeterministic, and thus very hard to implement efficiently. An other problem with transitive closure is that a blocked individual need no longer indicate "success", as is the case in DLs with transitive roles (see, e.g., the discussion of "good" and "bad" cycles in [1]).

References

1. Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of IJCAI-91*, Sydney, Australia, 1991.
2. Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.
3. Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, and Gert Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *J. of Logic, Language and Information*, 2:1–18, 1993.
4. Franz Baader and Philipp Hanschke. A schema for integrating concrete domains into concept languages. Technical Report RR-91-10, DFKI, Kaiserslautern, Germany, 1991. An abridged version appeared in *Proc. of IJCAI-91*.
5. Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of PDK'91*, volume 567 of *LNAI*, pages 67–86. Springer-Verlag, 1991.
6. Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of KR-92*, pages 270–281. Morgan Kaufmann, 1992.
7. Franz Baader and Ulrike Sattler. Expressive number restrictions in description logics. *J. of Logic and Computation*, 9(3):319–350, 1999.
8. Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
9. Ronald J. Brachman. "Reducing" CLASSIC to practice: Knowledge representation meets reality. In *Proc. of KR-92*, pages 247–258. Morgan Kaufmann, 1992.
10. Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of AAAI-84*, pages 34–37, 1984.
11. Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.
12. Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

13. Paolo Bresciani, Enrico Franconi, and Sergio Tessaris. Implementing and testing expressive description logics: Preliminary report. *Working Notes of the 1995 Description Logics Workshop*, Technical Report, RAP 07.95, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", pages 131–139, Rome (Italy), 1995.
14. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.
15. Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dip. di Inf. e Sist., Univ. di Roma "La Sapienza", 1995.
16. Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of AAAI-94*, pages 205–212. AAAI Press/The MIT Press, 1994.
17. Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In A. G. Cohn, editor, *Proc. of ECAI-94*, pages 411–415. John Wiley & Sons, 1994.
18. Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Proc. of KR-96*, pages 316–327. Morgan Kaufmann, 1996.
19. Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.
20. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR-91*, pages 151–162. Morgan Kaufmann, 1991.
21. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of IJCAI-91*, pages 458–463, Sydney, 1991.
22. Giuseppe De Giacomo and Fabio Massacci. Tableaux and algorithms for propositional dynamic logic with converse. In *Proc. of CADE-96*, pages 613–628, 1996.
23. Philipp Hanschke. Specifying role interaction in concept languages. In *Proc. of KR-92*, pages 318–329. Morgan Kaufmann, 1992.
24. Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of GWAI'90*, volume 251 of *Informatik-Fachberichte*, pages 38–47. Springer-Verlag, 1990.
25. Bernhard Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.
26. Bernhard Hollunder and Franz Baader. Qualifying number restrictions in concept languages. In *Proc. of KR-91*, pages 335–346, 1991.
27. Bernhard Hollunder and Werner Nutt. Subsumption algorithms for concept languages. Technical Report RR-90-04, DFKI, Kaiserslautern, Germany, 1990.
28. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of ECAI-90*, pages 348–353, London, 1990. Pitman.
29. Ian Horrocks. The FaCT system. In Harrie de Swart, editor, *Proc. of TABLEAUX-98*, volume 1397 of *LNAI*, pages 307–312. Springer-Verlag, 1998.
30. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, pages 636–647, 1998.
31. Ian Horrocks and Peter F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.
32. Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

33. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, LNAI. Springer-Verlag, 1999.
34. Carsten Lutz. Complexity of terminological reasoning revisited. In *Proc. of LPAR'99*, volume 1705 of *LNAI*. Springer-Verlag, 1999.
35. Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, 1991.
36. Eric Mays, Robert Dionne, and Robert Weida. K-REP system overview. *SIGART Bulletin*, 2(3), 1991.
37. Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. The MIT Press, 1981. Republished in [11].
38. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *LNAI*. Springer-Verlag, 1990.
39. Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
40. Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.
41. Peter F. Patel-Schneider. DLP. In *Proc. of DL'99*, pages 9–13. CEUR Electronic Workshop Proceedings, 1999. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-22/`.
42. Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
43. Christof Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.
44. M. Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967. Republished in [11].
45. Ulrike Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *Proc. of KI'96*, volume 1137 of *LNAI*. Springer-Verlag, 1996.
46. Walter J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. of Computer and System Science*, 4:177–192, 1970.
47. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, Sydney, Australia, 1991.
48. Klaus Schild. Terminological cycles and the propositional μ-calculus. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of KR-94*, pages 509–520, Bonn, 1994. Morgan Kaufmann.
49. Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proc. of KR-89*, pages 421–431. Morgan Kaufmann, 1989.
50. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
51. Edith Spaan. The complexity of propositional tense logics. In Maarten de Rijke, editor, *Diamonds and Defaults*, pages 287–307. Kluwer Academic Publishers, 1993.
52. Stephan Tobies. A PSPACE algorithm for graded modal logic. In *Proc. of CADE-99*, volume 1632 of *LNCS*. Springer-Verlag, 1999.
53. Wiebe Van der Hoek and Maarten De Rijke. Counting objects. *J. of Logic and Computation*, 5(3):325–345, 1995.

Modality and Databases

Melvin Fitting

Dept. Mathematics and Computer Science
Lehman College (CUNY), Bronx, NY 10468
e-mail: fitting@alpha.lehman.cuny.edu
web page: comet.lehman.cuny.edu/fitting

Abstract. Two things are done in this paper. First, a modal logic in which one can quantify over both objects and concepts is presented; a semantics and a tableau system are given. It is a natural modal logic, extending standard versions, and capable of addressing several well-known philosophical difficulties successfully. Second, this modal logic is used to introduce a rather different way of looking at relational databases. The idea is to treat records as possible worlds, record entries as objects, and attributes as concepts, in the modal sense. This makes possible an intuitively satisfactory relational database theory. It can be extended, by the introduction of higher types, to deal with multiple-valued attributes and more complex things, though this is further than we take it here.

1 Introduction

A few years ago my colleague, Richard Mendelsohn, and I finished work on our book, "First-Order Modal Logic," [2]. In it, non-rigidity was given an extensive examination, and formal treatments of definite descriptions, designation, existence, and other issues were developed. I next attempted an extension to higher-order modal logic. After several false starts (or rather, unsatisfactory finishes) this was done, and a book-length manuscript is on my web page inviting comments, [1]. Carrying out this extension, in turn, led me to rethink the first-order case. There were two consequences. First, I came to realize that the approach in our book could be extended, without leaving the first-order level, to produce a quite interesting logic with a natural semantics and a tableau proof procedure. And second, I realized that this modal logic provided a natural alternative setting for relational databases, which are usually treated using first-order classical logic. In this paper I want to sketch both the modal logic and its application to databases.

In a full treatment of first-order modal logic, one must be able to discourse about two kinds of things: *individual objects* and *individual concepts*. "George Washington" and "Millard Fillmore" denote individual objects, while "the President of the United States" denotes an individual concept, which in turn denotes various individuals at different times. Or again, at the time I am writing this the year is 2000. This particular year is an individual object. "The current year" is an individual concept, and will not always denote 2000. In [2] we had quantifiers ranging over individual objects, and constant symbols with values that

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 19–39, 2000.

were individual concepts. That was a good combination to elucidate a number of well-known philosophical problems, but it is not a full picture. In this paper the formal system presented will have quantifiers over individual objects, and also a second kind of quantifier ranging over individual concepts. Likewise there will be two kinds of constant symbols. The system of [2] can be embedded in the present one. (Of course this is only approximate. In our book we had function symbols, and we do not have them here. There are other differences as well, but the embedability claim is essentially correct.) I'll begin with a presentation of the logic, and then consider its applications to databases.

In a sense, using the modal logic of this paper to supply a semantics for relational databases does not give us anything new. We are able to treat things that, previously, had been treated using classical first-order logic. The modal point of view is substantially different, and hence interesting, but does not expand our concept of relational database. The real significance lies in what comes next, just after the conclusion of this paper. The modal logic presented here is the first-order fragment of a higher-order modal logic, with both extensional and intensional objects at each level. When such a logic is applied to database theory, we get a natural setting within which to model multiple-valued relations, relations having a field whose values are sets of attributes, and more complex things yet. Think of the present paper, then, as providing a different point of view on what is generally understood, and as signaling the approach of an extension, which can be glimpsed down the road.

2 Syntax

The syntax of this version of first-order modal logic is a little more complex than usual, and so some care must be taken in its presentation.

There are infinitely many variables and constants, in each of two categories: individual objects and individual concepts. I'll use lowercase Latin letters x, y, z as object variables, and lowercase Greek letters α, β, γ as concept variables. (Based on the notion that the ancient Greeks were the theoreticians, while the Romans were the engineers.) The Greek letter ϱ, with or without subscripts, represents a variable of either kind. For constants, I'll use lowercase Latin letters such as a, b, c for both kinds, and leave it to context to sort things out.

Definition 1 (Term). *A constant symbol or a variable is a* term*. It is an* object term *if it is an individual object variable or constant symbol. Similarly for* concept terms*.*

If t is a concept term, $\downarrow t$ is an object term. It is called a relativized *term.*

The idea is, if t is a concept term, $\downarrow t$ is intended to designate the object denoted by t, in a particular context. Sometimes I'll refer to $\downarrow$ as the *evaluate at* operator.

Since there are two kinds of variables and constants, assigning an arity to relation symbols is not sufficient. By a *type* I mean a finite sequence of o's and c's, such as $\langle c, o, c\rangle$. Think of an o as marking an object position and a c as marking

a concept position. There are infinitely many relation symbols of each type. In particular there is an equality symbol, $=$, of type $\langle o, o\rangle$. That is, equality is a relation on individual objects. One could also introduce a notion of equality for individual concepts, but it will not be needed here. I allow the empty sequence $\langle\rangle$ as a type. It corresponds to what are sometimes called propositional letters, taking no arguments.

Definition 2 (Formula). *The set of formulas, and their free variables, is defined as follows.*

1. *If P is a relation symbol of type $\langle\rangle$, it is an atomic formula, and has no free variables.*
2. *Suppose R is a relation symbol of type $\langle n_1, n_2, \ldots, n_k\rangle$ and t_1, t_2, $\ldots$, t_k is a sequence of terms such that t_i is an individual object term if $n_i = o$ and is an individual concept term if $n_i = c$. Then $R(t_1, t_2, \ldots, t_k)$ is an atomic formula. Its free variables are the variable occurrences that appear in it.*
3. *if X is a formula, so are $\neg X$, $\Box X$, and $\Diamond X$. Free variable occurrences are those of X.*
4. *If X and Y are formulas, so are $(X \wedge Y)$, $(X \vee Y)$, and $(X \supset Y)$. Free variable occurrences are those of X together with those of Y.*
5. *If X is a formula and ϱ is a variable (of either kind), $(\forall\varrho)X$ and $(\exists\varrho)X$ are formulas. Free variable occurrences are those of X, except for occurrences of ϱ.*
6. *If X is a formula, ϱ is a variable (again of either kind), and t is a term of the same kind as ϱ, $\langle\lambda\varrho.X\rangle(t)$ is a formula. Free variable occurrences are those of X, except for occurrences of ϱ, together with those of t.*

As usual, parentheses will be omitted from formulas to improve readability. Also $=(x, y)$ will be written as $x = y$. And finally, a formula like

$$\langle\lambda\varrho_1.\langle\lambda\varrho_2.\langle\lambda\varrho_3.X\rangle(t_3)\rangle(t_2)\rangle(t_1)$$

will be abbreviated by the simpler expression

$$\langle\lambda\varrho_1, \varrho_2, \varrho_3.X\rangle(t_1, t_2, t_3)$$

and similarly for other formulas involving iterated abstractions.

3 Semantics

I will only formulate an **S5** logic—the ideas carry over directly to other logics, but **S5** is simplest, the ideas are clearest when stated for it, and it is all that is actually needed for databases.

Frames essentially disappear, since we are dealing with **S5**. A model has a set of possible worlds, but we can take every world to be accessible from every other, and so no explicit accessibility relation is needed.

The usual constant/varying domain dichotomy is easily ignored. For first-order modal logics generally, a constant domain semantics can simulate a varying domain version, through the use of an explicit existence predicate and the relativization of quantifiers to it. Here I'll take object domains to be constant—not world dependent—which makes things much simpler.

Since the language has two kinds of terms, we can expect models to have two domains—two sorts, in other words. There will be a domain of individual objects, and a domain of individual concepts. Concepts will be functions, from possible worlds to individual objects. It is not reasonable, or desirable, to insist that *all* such functions be present. After all, if there are countably many possible worlds, there would be a continuum of such concept functions even if the set of individual objects is finite, and this probably cannot be captured by a proof procedure. But anyway, the notion of an individual concept presupposes a kind of coherency for that individual concept—not all functions would be acceptable intuitively. I simply take the notion of individual concept as basic; I do not try to analyze any coherency condition. It is allowed that some, not necessarily all, functions can serve as individual concepts.

Definition 3 (Model). *A* model *is a structure* $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$*, where:*

1. $\mathcal{G}$ *is a non-empty set, of* possible worlds*;*
2. $\mathcal{D}_o$ *is a non-empty set, of* individual objects*;*
3. $\mathcal{D}_c$ *is a non-empty set of functions from* $\mathcal{G}$ *to* $\mathcal{D}_o$*, called* individual concepts*;*
4. $\mathcal{I}$ *is a mapping that assigns:*
   a) *to each individual object constant symbol some member of* $\mathcal{D}_o$*;*
   b) *to each individual concept constant symbol some member of* $\mathcal{D}_c$*;*
   c) *to each relation symbol of type* $\langle\rangle$ *a mapping from* $\mathcal{G}$ *to* $\{false, true\}$*;*
   d) *to each relation symbol of type* $\langle n_1, n_2, \ldots, n_k \rangle$ *a mapping from* $\mathcal{G}$ *to the power set of* $\mathcal{D}_{n_1} \times \mathcal{D}_{n_2} \times \cdots \times \mathcal{D}_{n_k}$*. It is required that* $\mathcal{I}(=)$ *be the constant mapping that is identically the equality relation on* $\mathcal{D}_o$*.*

Some preliminary machinery is needed before truth in a model can be characterized.

Definition 4 (Valuation). *A* valuation v *in a model* $\mathcal{M}$ *is a mapping that assigns to each individual object variable some member of* $\mathcal{D}_o$*, and to each individual concept variable some member of* $\mathcal{D}_c$*.*

Definition 5 (Value At). *Let* $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$ *be a model, and* v *be a valuation in it. A mapping* $(v * \mathcal{I})$ *is defined, assigning a meaning to each term, at each possible world. Let* $\Gamma \in \mathcal{G}$*.*

1. *If* ϱ *is a variable,* $(v * \mathcal{I})(\varrho, \Gamma) = v(\varrho)$*.*
2. *If* c *is a constant symbol,* $(v * \mathcal{I})(c, \Gamma) = \mathcal{I}(c)$*.*
3. *If* $\downarrow t$ *is a relativized term,* $(v * \mathcal{I})(\downarrow t, \Gamma) = (v * \mathcal{I})(t)(\Gamma)$*.*

Item 3 is especially significant. If $\downarrow t$ is a relativized term, t must be a constant or variable of concept type, and so $(v * \mathcal{I})(t)$ has been defined for it in parts 1 and 2, and is a function from worlds to objects. Thus $(v * \mathcal{I})(t)(\Gamma)$ is a member of $\mathcal{D}_o$.

Now the main notion, which is symbolized by $\mathcal{M}, \Gamma \Vdash_v \Phi$, and is read: formula Φ is true in model $\mathcal{M}$, at possible world Γ, with respect to valuation v. To make reading easier, the following special notation is used. Let $\varrho_1, \ldots, \varrho_k$ be variables of any type, and let $d_1, \ldots, d_k$ be members of $\mathcal{D}_o \cup \mathcal{D}_c$, with $d_i \in \mathcal{D}_o$ if the variable ϱ_i is of object type, and $d_i \in \mathcal{D}_c$ if ϱ_i is of concept type. Then

$$\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho_1/d_1, \ldots, \varrho_k/d_k]$$

abbreviates: $\mathcal{M}, \Gamma \Vdash_{v'} \Phi$ where v' is the valuation that is like v on all variables except $\varrho_1, \ldots, \varrho_k$, and $v'(\varrho_1) = d_1, \ldots, v'(\varrho_k) = d_k$.

Here is the central definition. For simplicity, take $\vee$, $\supset$, $\exists$, and $\Diamond$ as defined symbols, in the usual way.

Definition 6 (Truth in a Model). *Let $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I} \rangle$ be a model, and v be a valuation in it.*

1. *If P is of type $\langle\rangle$, $\mathcal{M}, \Gamma \Vdash_v P$ iff $\mathcal{I}(P)(\Gamma) = true$.*
2. *If $R(t_1, \ldots, t_k)$ is atomic, $\mathcal{M}, \Gamma \Vdash_v R(t_1, \ldots, t_k)$ iff $\langle (v * \mathcal{I})(t_1, \Gamma), \ldots, (v * \mathcal{I})(t_k, \Gamma) \rangle \in \mathcal{I}(R)(\Gamma)$.*
3. *$\mathcal{M}, \Gamma \Vdash_v \neg\Phi$ iff $\mathcal{M}, \Gamma \nVdash_v \Phi$.*
4. *$\mathcal{M}, \Gamma \Vdash_v \Phi \wedge \Psi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi$ and $\mathcal{M}, \Gamma \Vdash_v \Psi$.*
5. *$\mathcal{M}, \Gamma \Vdash_v (\forall x)\Phi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi[x/d]$ for all $d \in \mathcal{D}_o$.*
6. *$\mathcal{M}, \Gamma \Vdash_v (\forall \alpha)\Phi$ iff $\mathcal{M}, \Gamma \Vdash_v \Phi[\alpha/d]$ for all $d \in \mathcal{D}_c$.*
7. *$\mathcal{M}, \Gamma \Vdash_v \Box\Phi$ iff $\mathcal{M}, \Delta \Vdash_v \Phi$ for all $\Delta \in \mathcal{G}$.*
8. *$\mathcal{M}, \Gamma \Vdash_v \langle \lambda\varrho.\Phi \rangle(t)$ if $\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho/d]$, where $d = (v * \mathcal{I})(t, \Gamma)$.*

Definition 7 (Validity). *A closed formula X is* valid *in a model if it is true at every world of it.*

A notion of consequence is a little more complicated because, in modal settings, it essentially breaks in two. These are sometimes called *local* and *global* consequence notions. For a set S of formulas, do we want X to be true at every world at which members of S are true (local consequence), or do we want X to be valid in every model in which members of S are valid (global consequence). These have quite different flavors. Fortunately, for **S5**, the situation is somewhat simpler than it is for other modal logics since, to say X is valid in a model is just to say $\Box X$ is true at some world of it. So, if we have a notion of local consequence, we can define a corresponding global consequence notion simply by introducing necessity symbols throughout. So, local consequence is all we need here.

Definition 8 (Consequence). *A closed formula X is a* consequence *of a set S of closed formulas if, in every model, X is true at each world at which all the members of S are true.*

4 Rigidity

An individual concept term t can vary from world to world in what it designates. Call t *rigid* in a model if it is constant in that model, designating the same object at each world. This is a notion that plays an important role in philosophy. For instance Kripke [3], among others, asserts that names are rigid designators. The notion of rigidity can be captured by a formula. Assume c is an individual concept constant symbol in the following.

$$\langle \lambda x. \Box(x =\!\downarrow c)\rangle(\downarrow c) \tag{1}$$

It is quite straightforward to show that (1) is valid in a model if and only if the interpretation of c is rigid in that model. In [2] this, in turn, was shown to be equivalent to the vanishing of the *de re/de dicto* distinction, though this will not be needed here.

One can also speak of *relativized* notions of rigidity. Let us say the interpretation of c is rigid on a particular subset $\mathcal{G}_0$ of the collection $\mathcal{G}$ of possible worlds of a model provided it designates the same object throughout $\mathcal{G}_0$. And let us say c is *rigid relative to* d in a model provided the interpretation of c is rigid on any subset of worlds on which the interpretation of d is rigid. (Of course, this notion applies to individual concept terms that are variables as well. I'm using constant symbols just as a matter of convenience.) The notion of c being rigid relative to d is captured by formula (2).

$$\langle \lambda x, y. \Box[x =\!\downarrow d \supset y =\!\downarrow c]\rangle(\downarrow d, \downarrow c) \tag{2}$$

One can also consider more complicated situations. Formula (3) asserts that c is rigid relative to the rigidity of d and e jointly.

$$\langle \lambda x, y, z. \Box[(x =\!\downarrow d \wedge y =\!\downarrow e) \supset z =\!\downarrow c]\rangle(\downarrow d, \downarrow e, \downarrow c) \tag{3}$$

Finally, one can even say that *all* individual concepts are rigid relative to c. This is done in formula (4). Individual concept quantification is obviously essential here.

$$(\forall \alpha)\langle \lambda x, y. \Box[x =\!\downarrow c \supset y =\!\downarrow \alpha]\rangle(\downarrow c, \downarrow \alpha) \tag{4}$$

5 Databases with a Single Relation

In this section we begin taking a look at relational databases. What we consider is quite basic, and can be found in any textbook on databases—[4] is a good source. Relational databases are commonly reasoned about using classical first-order logic. I want to show that modal logic is also a natural tool for this purpose. For now, only a single relation will be considered—this will be extended later.

The *record* is the basic unit of a relational database, yet it is not a first-class object in the sense that it is not something we can get as an answer to a query. We could get a record number, perhaps, but not a record. We will take the records of a relational database to be the *possible worlds* of a Kripke model. In any standard modal language possible worlds, in fact, cannot be directly spoken of. The accessibility relation will be the usual **S5** one, though there could be circumstances where something more complex might be appropriate.

Entries that fill fields of a relational database generally can be of several data types. To keep things simple, let's say there is just one data type used for this purpose. (In examples I'll use strings.) Looking at this in terms of modal logic, these field entries will be the *individual objects* of a Kripke model.

Next come the attributes themselves. If the database is one listing family relationships, say, and there is an attribute recording "father-of," it has a value that varies from record to record, but in every case that value is what we have taken to be an individual object. In other words, attributes are simply *individual concepts*.

The next question is, what about things like functional dependencies, keys, and so on? Let's begin with the notion of functional dependency. Say we have a relational database in which there are two attributes, call them c and d, and c is functionally dependent on d. Then, if we are at an arbitrary possible world (record) at which c and d have particular values, at any other world at which d has the value it has in this one, c must also have the same value it has in this one. This can be expressed by requiring validity of the following formula, in which we assume c and d of the Kripke model also occur as individual concept constants of the language, and designate themselves.

$$\langle \lambda x, y. \Box [x =\!\downarrow d \supset y =\!\downarrow c] \rangle (\downarrow d, \downarrow c)$$

But this is just formula (2), and says c is rigid relative to d. In this case, a functional dependency can be expressed by a relative rigidity assertion.

More complicated functional dependencies also correspond to relative rigidity formulas. For instance, if c is functionally dependent on $\{d, e\}$, this is expressed by (3).

Next, what about the notion of *keys*? As usually treated, to say an attribute c is a key is to say there cannot be two records that agree on the value of c. We cannot quite say that, since records cannot directly be spoken of. What we can say is that two possible worlds agreeing on the value of c must agree on the values of all attributes. More formally, this is expressed by the validity of the following formula.

$$(\forall \alpha) \langle \lambda x, y. \Box [x =\!\downarrow c \supset y =\!\downarrow \alpha] \rangle (\downarrow c, \downarrow \alpha)$$

Note that this is our earlier formula (4).

Now, what does the design of a relation schema look like from the present modal point of view? We must specify the domain for atomic values of the relation schema. Semantically, that amounts to specifying the set $\mathcal{D}_o$ of a modal

model. Proof-theoretically, it amounts to saying what the individual object constant symbols of the formal language are. (I'll generally assume that constant symbols of the language can also occur in models, and designate themselves.)

Next, we must specify what the attributes for the relation schema are. This amounts to specifying the set $\mathcal{D}_c$ of a model, or the set of individual concept constant symbols of a language.

Finally, we must impose some constraints, such as requiring that some attribute or set of attributes be a key, or that various functional dependencies must hold. This corresponds to taking appropriate relative rigidity formulas as axioms.

6 A Simple Example

In this section I'll show how a simple, standard, example translates into modal language both semantically and proof-theoretically. Consider the relation schema with five attributes: NAME, SSN, BIRTHDATE, JOBNUMBER, and JOBTITLE. It will be assumed that SSN is the primary key, and JOBNUMBER is functionally dependent on JOBTITLE.

We set up a formal language with "NAME," "SSN," "BIRTHDATE," "JOBNUMBER," and "JOBTITLE" as individual concept constant symbols. Then we adopt the following two *constraint axioms*.

1. $\Box(\forall\alpha)\langle\lambda x, y.\Box[x =\!\downarrow\mathtt{SSN} \supset y =\!\downarrow\alpha]\rangle(\downarrow\mathtt{SSN}, \downarrow\alpha)$
2. $\Box\langle\lambda x, y.\Box[x =\!\downarrow\mathtt{JOBTITLE} \supset y =\!\downarrow\mathtt{JOBNUMBER}]\rangle(\downarrow\mathtt{JOBTITLE}, \downarrow\mathtt{JOBNUMBER})$

Table 1 displays a particular relation instance of this relation schema.

Table 1. The relation PERSONS

NAME	SSN	BIRTHDATE	JOBNUMBER	JOBTITLE
Adam	1	01/06/-4004	1	Gardener
Eve	2	01/08/-4004	2	Explorer
Cain	3	10/21/-4004	1	Gardener
Abel	4	11/05/-4003	2	Shepherd
Seth	5	02/04/-3983	2	Explorer

To represent this relation instance as a modal axiomatic theory, we add the following to the constraint axioms above; we call them *instance axioms*.

3. $\Diamond[(\downarrow\mathtt{NAME} = \mathtt{Adam}) \wedge (\downarrow\mathtt{SSN} = 1) \wedge (\downarrow\mathtt{BIRTHDATE} = \mathtt{01/06/-4004}) \wedge (\downarrow\mathtt{JOBNUMBER} = 1) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Gardener})]$
4. $\Diamond[(\downarrow\mathtt{NAME} = \mathtt{Eve}) \wedge (\downarrow\mathtt{SSN} = 2) \wedge (\downarrow\mathtt{BIRTHDATE} = \mathtt{01/08/-4004}) \wedge (\downarrow\mathtt{JOBNUMBER} = 2) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Explorer})]$
5. $\Diamond[(\downarrow\mathtt{NAME} = \mathtt{Cain}) \wedge (\downarrow\mathtt{SSN} = 3) \wedge (\downarrow\mathtt{BIRTHDATE} = \mathtt{10/03/-4004}) \wedge (\downarrow\mathtt{JOBNUMBER} = 1) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Gardener})]$

6. $\Diamond[(\downarrow\mathtt{NAME} = \mathtt{Abel}) \wedge (\downarrow\mathtt{SSN} = 4) \wedge (\downarrow\mathtt{BIRTHDATE} = \mathtt{08/05/-4003}) \wedge (\downarrow\mathtt{JOBNUMBER} = 2) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Shepherd})]$
7. $\Diamond[(\downarrow\mathtt{NAME} = \mathtt{Seth}) \wedge (\downarrow\mathtt{SSN} = 5) \wedge (\downarrow\mathtt{BIRTHDATE} = \mathtt{02/04/-3983}) \wedge (\downarrow\mathtt{JOBNUMBER} = 2) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Explorer})]$

This, of course, assumes individual object constant symbols, "01/06/-4004," "Adam," and so on have been added to the language. I'll also assume these symbols are intended to designate distinct objects. This gives us a (long) list of axioms.

8. $\neg(\mathtt{1} = \mathtt{2})$, $\neg(\mathtt{Adam} = \mathtt{Eve})$, etc.

Corresponding to this, semantically, we have the following **S5** model. There are five possible worlds, one for each of the five rows of Table 1; call them $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$, and Γ_5. $\mathcal{D}_o = \{\mathtt{Adam}, \mathtt{1}, \mathtt{01/06/-4004}, \dots\}$. $\mathcal{D}_c = \{\widehat{\mathtt{NAME}}, \widehat{\mathtt{SSN}}, \widehat{\mathtt{BIRTHDATE}}, \widehat{\mathtt{JOBNUMBER}}, \widehat{\mathtt{JOBTITLE}}\}$, where $\widehat{\mathtt{NAME}}$ is the function that maps Γ_1 to Adam, Γ_2 to Eve, and so on. $\mathcal{I}(\mathtt{Adam}) = \mathtt{Adam}, \dots, \mathcal{I}(\mathtt{NAME}) = \widehat{\mathtt{NAME}}$, and so on.

Finally, here are some sample queries, in modal language. First, who are the explorers? This corresponds to the following, in which appropriate keys (social security numbers) are desired.

$$\langle\lambda x.\Diamond[(\downarrow\mathtt{SSN} = x) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Explorer})]\rangle \tag{5}$$

Suppose we abbreviate formula (5) by Q. Then, on the one hand, $Q(z)$ is true in the modal model we constructed just in case z is 2 or 5. On the other hand, $Q(z)$ is provable in the axiom system we constructed just in case z is 2 or 5.

Here is a second sample query: is there more than one gardener?

$$\begin{aligned}(\exists x)(\exists y)\{&\Diamond[(\downarrow\mathtt{SSN} = x) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Gardener})] \wedge \\ &\Diamond[(\downarrow\mathtt{SSN} = y) \wedge (\downarrow\mathtt{JOBTITLE} = \mathtt{Gardener})] \wedge \\ &\neg(x = y)\}\end{aligned} \tag{6}$$

Formula (6) is derivable from our axioms, and true in our model.

7 Connections

In the example of the previous section we saw that being a consequence of certain axioms and being true in a particular model could coincide. Now we examine to what extent this is generally the case.

Suppose we have a relation schema R and a corresponding set of constraints concerning keys and functional dependencies. Associated with R is a set of *constraint axioms*, which I'll denote $\mathsf{axiom}(R)$, consisting of formulas like (2), (3), and (4). I'll omit an exact definition—axioms 1 and 2 of the example in the previous section is a sufficient guide. Note that these axioms are either quantifier free, or else involve just universal quantifiers, and $\Box$ is the only modal operator.

Next, suppose we have a relation instance r of the relation schema R—a particular set of tuples. Associated with this is a set of *instance axioms*, all of which are quantifier free. Again I omit an exact definition, but axioms 3–7 of the previous section illustrate the notion sufficiently. Finally there are *distinctness axioms*, as in axiom 8 of the previous section. By $\mathsf{axiom}(r)$ I mean the collection of these instance axioms and distinctness axioms, together with the members of $\mathsf{axiom}(R)$. Clearly, to say r is an instance of R and satisfies the constraints, is just to say $\mathsf{axiom}(r)$ is a consistent set of model axioms.

Next, we want a designated modal model to correspond to relation instance r. Again, the example of the previous section serves as a guide. We want the model, denote it by $\mathsf{model}(r)$, meeting the following conditions. The collection of possible worlds $\mathcal{G}$ is the collection of tuples in r. The domain $\mathcal{D}_o$ of individual objects is just the collection of table entries in r. The domain $\mathcal{D}_c$ of individual concepts is the collection of attributes of relation schema R. The interpretation $\mathcal{I}$ maps each table entry (as a constant of the modal language) to itself (as a member of $\mathcal{D}_o$). And $\mathcal{I}$ maps each attribute `ATT` to the function whose value at tuple (possible world) Γ is the entry in the tuple Γ corresponding to `ATT`. The only relation symbol is $=$, which is interpreted as equality on $\mathcal{D}_o$.

Question: what are the connections between $\mathsf{axiom}(r)$ and $\mathsf{model}(r)$? I don't know the most general answer to this, but here is something that accounts for what was seen in the previous section. Note that the queries considered there, formulas (5) and (6), were either quantifier free or involved existential quantifiers of individual object type only. This is significant.

Definition 9. *Call a closed modal formula* simple existential *if it is of the form*

$$(\exists x_1)\cdots(\exists x_n)\Diamond\Phi$$

where Φ is quantifier and modality free, and contains only $=$ as a relation symbol.

Proposition 1. *For a relation instance r and a simple existential sentence X, X is a consequence of $\mathsf{axiom}(r)$ if and only if X is true in $\mathsf{model}(r)$.*

I'll postpone a proof of this to Section 12.3.

8 Partial Concepts

We have taken individual concepts to be *total* functions on the set of possible worlds of a modal model. But there are many circumstances where a more general notion is desirable. "The King of France," for instance, designates an individual at many points in history, but not at all of them. Fortunately, it is straightforward to allow partiality.

Definition 3, of modal model, is changed as follows. From now on $\mathcal{D}_c$ will be a non-empty set of functions, each from some subset of $\mathcal{G}$ to $\mathcal{D}_o$, where that subset can be proper. If Γ is not in the domain of some individual concept f, we will write $f(\Gamma) = \perp$, where $\perp$ is an arbitrary item not a member of $\mathcal{D}_o$. Definition 5,

value at, can be used with no change in wording, but notice that the scope of part 3 has been extended. If $\downarrow t$ is a relativized term, and Γ is not in the domain of $(v * \mathcal{I})(t)$, then $(v * \mathcal{I})(\downarrow t, \Gamma) = (v * \mathcal{I})(t)(\Gamma) = \perp$.

Now Definition 6, truth in a model, must also be modified. I'll follow the pretty obvious general principle that one cannot ascribe properties to what is designated by a non-designating term. In the Definition, item 2 is changed to read as follows.

2. If $R(t_1, \ldots, t_k)$ is atomic,
   a) if any of $(v*\mathcal{I})(t_1, \Gamma), \ldots, (v*\mathcal{I})(t_k, \Gamma)$ is $\perp$ then $\mathcal{M}, \Gamma \nVdash_v R(t_1, \ldots, t_k)$;
   b) otherwise $\mathcal{M}, \Gamma \Vdash_v R(t_1, \ldots, t_k)$ iff $\langle (v * \mathcal{I})(t_1, \Gamma), \ldots, (v * \mathcal{I})(t_k, \Gamma)\rangle \in \mathcal{I}(R)(\Gamma)$.

Also item 8 must be changed.

8. For an abstract $\langle \lambda\varrho.\Phi\rangle(t)$,
   a) if $(v * \mathcal{I})(t, \Gamma) = \perp$, $\mathcal{M}, \Gamma \nVdash_v \langle \lambda\varrho.\Phi\rangle(t)$;
   b) otherwise $\mathcal{M}, \Gamma \Vdash_v \langle \lambda\varrho.\Phi\rangle(t)$ if $\mathcal{M}, \Gamma \Vdash_v \Phi[\varrho/d]$, where $d = (v * \mathcal{I})(t, \Gamma)$.

Notice that, with the definitions modified in this way, $\downarrow t = \downarrow t$ is true at a world of a model if and only if the term t "designates" at that world. This makes possible the following piece of machinery.

Definition 10 (Designation Formula). ***D*** *abbreviates the abstract* $\langle \lambda\alpha.\ \downarrow \alpha = \downarrow\alpha\rangle$.

Clearly $\mathcal{M}, \Gamma \Vdash_v \mathbf{D}(t)$ iff $(v * \mathcal{I})(t, \Gamma) \neq \perp$, where $\mathcal{M} = \langle \mathcal{G}, \mathcal{D}_o, \mathcal{D}_c, \mathcal{I}\rangle$. Thus **D** really does express the notion of designation.

Now our earlier notions of rigidity and relative rigidity must be modified. We say c is rigid if it designates the same thing in all worlds in which it designates at all. This means formula (1) must be replaced with the following, if we want to express a notion of rigidity allowing for partial concepts.

$$\mathbf{D}(c) \supset \langle \lambda x.\Box[\mathbf{D}(c) \supset (x = \downarrow c)]\rangle(\downarrow c) \tag{7}$$

Likewise, c being rigid relative to d must be modified. It should now say: if d designates in two worlds, and designates the same thing, then if c also designates in those worlds, it must designate the same thing at both. This means formula (2) must be replaced with the following.

$$(\mathbf{D}(c) \wedge \mathbf{D}(d)) \supset \langle \lambda x, y.\Box[(\mathbf{D}(c) \wedge \mathbf{D}(d) \wedge x = \downarrow d) \supset (y = \downarrow c)]\rangle(\downarrow d, \downarrow c) \tag{8}$$

Similar changes must be made to the other notions from Section 4. In particular, (4), expressing that every attribute is rigid relative to c, gets expressed as follows.

$$(\forall\alpha)\{(\mathbf{D}(c) \wedge \mathbf{D}(\alpha)) \supset \langle \lambda x, y.\Box[(\mathbf{D}(c) \wedge \mathbf{D}(\alpha) \wedge x = \downarrow c) \supset (y = \downarrow\alpha)]\rangle(\downarrow c, \downarrow\alpha)\} \tag{9}$$

9 Relational Databases More Generally

A relational database generally has more than one relation involved. Now that partial individual concepts are available, this is easy to handle. Suppose we add to the database containing the relation given in Table 1 a second relation, given in Table 2.

Table 2. The relation LOCATION

JOBNUMBER	WHERE
1	Home
2	Away

We allowed, in our modal language, relation symbols of type $\langle\rangle$. Let us introduce two specific ones, PERSONS and LOCATION, intended to distinguish the two relations we now have. The idea is, we will have two kinds of possible worlds, those at which LOCATION is true and those at which PERSONS is true. The first kind of world should correspond to a row of the LOCATION table, and so JOBNUMBER and WHERE should be defined, but nothing else. Similarly for the second kind. This gives us the following new kinds of constraint axioms.

1. $\Box\{\texttt{PERSONS} \supset [\mathbf{D}(\texttt{JOBNUMBER}) \wedge \neg\mathbf{D}(\texttt{WHERE}) \wedge \mathbf{D}(\texttt{NAME}) \wedge \mathbf{D}(\texttt{SSN}) \wedge\mathbf{D}(\texttt{BIRTHDATE}) \wedge \mathbf{D}(\texttt{JOBTITLE})]\}$
2. $\Box\{\texttt{LOCATION} \supset [\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge \neg\mathbf{D}(\texttt{NAME}) \wedge \neg\mathbf{D}(\texttt{SSN}) \wedge\neg\mathbf{D}(\texttt{BIRTHDATE}) \wedge \neg\mathbf{D}(\texttt{JOBTITLE})]\}$

We still want the functional dependencies we had before, but these need to be modified to take partiality of intensional concepts into account. We also want a new dependency saying WHERE is functionally dependent on JOBNUMBER. These take the following form.

3. $\Box(\forall\alpha)\{(\mathbf{D}(\texttt{SSN}) \wedge \mathbf{D}(\alpha)) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{SSN}) \wedge \mathbf{D}(\alpha) \wedge x =\!\downarrow\texttt{SSN}) \supset (y =\!\downarrow \alpha)]\rangle(\downarrow\texttt{SSN}, \downarrow\alpha)\}$
4. $\Box\{(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{JOBTITLE})) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{JOBTITLE}) \wedge x =\!\downarrow\texttt{JOBTITLE}) \supset (y =\!\downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{JOBTITLE}, \downarrow\texttt{JOBNUMBER})\}$
5. $\Box\{(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE})) \supset \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x =\!\downarrow\texttt{WHERE}) \supset (y =\!\downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{WHERE}, \downarrow\texttt{JOBNUMBER})\}$

Next we need the instance axioms. These are quite straightforward.

6. $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Home})]$
7. $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$

8. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Adam}) \wedge (\downarrow\texttt{SSN} = 1) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/06/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$
9. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Eve}) \wedge (\downarrow\texttt{SSN} = 2) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{01/08/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$
10. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Cain}) \wedge (\downarrow\texttt{SSN} = 3) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{10/03/-4004}) \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Gardener})]$
11. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Abel}) \wedge (\downarrow\texttt{SSN} = 4) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{08/05/-4003}) \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Shepherd})]$
12. $\Diamond[\texttt{PERSONS} \wedge (\downarrow\texttt{NAME} = \texttt{Seth}) \wedge (\downarrow\texttt{SSN} = 5) \wedge (\downarrow\texttt{BIRTHDATE} = \texttt{02/04/-3983}) \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{JOBTITLE} = \texttt{Explorer})]$

Finally we assume that all our object constant symbols are distinct.

13. $\neg(1 = 2)$, $\neg(\texttt{Home} = \texttt{Away})$, $\neg(\texttt{Adam} = \texttt{Eve})$, etc.

10 Tableaus

Since consequence issues are important, a sound and complete proof procedure can be useful. Fortunately, standard tableau methods using prefixed formulas adapt quite naturally.

Proofs, and derivations, will be of closed formulas. As usual, in order to handle existential quantifiers, *parameters* will be introduced. We can think of these as being additional constant symbols, added to the language for the purpose of proof construction. Since we have two kinds of quantifiers, object and concept, we will have two kinds of parameters as well. I'll use p^o, q^o, etc. as object parameters, and p^c, q^c, etc. as concept parameters.

A *prefix* for **S5** is simply a positive integer, which we can intuitively think of as the name of a possible world in some model. Unlike in more conventional treatments of modal logic, we must allow not only formulas, but also certain terms to have prefixes. For example, if c is an individual concept constant symbol, its designation in a model will vary from world to world. Think of c with prefix n as the individual object that c designates at the world named by n. To keep notation simple, I'll violate the literal meaning of the word "prefix," and display them as subscripts. Thus c_n is an example of a prefixed concept constant symbol. In our proofs, individual concept constants and individual concept parameters may have prefixes.

A little more formally, by an *extended formula* I mean one that may contain parameters, and in which individual concept constants and parameters may have prefixes. A prefixed concept constant or parameter is considered to be an individual *object* term. All proofs will be of closed formulas, but closed *extended* formulas will appear in proofs.

A *prefixed formula* is a closed extended formula, with a prefix, and here we actually write them as prefixes. Thus, if X is a closed, extended formula, and n is a positive integer, $n\,X$ is a prefixed formula.

As usual, a tableau proof of a sentence X is a tree with $1\,\neg X$ at the root, and meeting certain other conditions which we will specify. Think of the initial

prefixed formula as intuitively asserting there is a world of a model, denoted by 1, at which X is not true. The tableau is constructed using various *branch extension rules*. In them σ is an arbitrary prefix.

Conjunctive Rules

$$\frac{\sigma\, X \wedge Y}{\begin{array}{c}\sigma\, X\\ \sigma\, Y\end{array}} \quad \frac{\sigma\, \neg(X \vee Y)}{\begin{array}{c}\sigma\, \neg X\\ \sigma\, \neg Y\end{array}} \quad \frac{\sigma\, \neg(X \supset Y)}{\begin{array}{c}\sigma\, X\\ \sigma\, \neg Y\end{array}}$$

Double Negation Rule

$$\frac{\sigma\, \neg\neg X}{\sigma\, X}$$

Disjunctive Rules

$$\frac{\sigma\, X \vee Y}{\sigma\, X \,|\, \sigma\, Y} \quad \frac{\sigma\, \neg(X \wedge Y)}{\sigma\, \neg X \,|\, \sigma\, \neg Y} \quad \frac{\sigma\, X \supset Y}{\sigma\, \neg X \,|\, \sigma\, Y}$$

Necessity Rules For any positive integer n that *already occurs* on the branch,

$$\frac{\sigma\, \Box X}{n\, X} \quad \frac{\sigma\, \neg\Diamond X}{n\, \neg X}$$

Possibility Rules If the positive integer n is *new* to the branch,

$$\frac{\sigma\, \Diamond X}{n\, X} \quad \frac{\sigma\, \neg\Box X}{n\, \neg X}$$

Concept Existential Rules In the following, p^c is an individual concept parameter that is *new to the tableau branch*.

$$\frac{\sigma\, (\exists\alpha)\Phi(\alpha)}{\sigma\, \Phi(p^c)} \quad \frac{\sigma\, \neg(\forall\alpha)\Phi(\alpha)}{\sigma\, \neg\Phi(p^c)}$$

Object Existential Rules In the following, p^o is an individual object parameter that is *new to the tableau branch*.

$$\frac{\sigma\, (\exists x)\Phi(x)}{\sigma\, \Phi(p^o)} \quad \frac{\sigma\, \neg(\forall x)\Phi(x)}{\sigma\, \neg\Phi(p^o)}$$

Concept Universal Rules In the following, τ is any individual concept constant symbol or parameter.

$$\frac{\sigma\, (\forall\alpha)\Phi(\alpha)}{\sigma\, \Phi(\tau)} \quad \frac{\sigma\, \neg(\exists\alpha)\Phi(\alpha)}{\sigma\, \neg\Phi(\tau)}$$

Object Universal Rules In the following, τ is any individual object constant symbol or parameter, or *a prefixed individual concept constant symbol or parameter*.

$$\frac{\sigma\, (\forall x)\Phi(x)}{\sigma\, \Phi(\tau)} \quad \frac{\sigma\, \neg(\exists x)\Phi(x)}{\sigma\, \neg\Phi(\tau)}$$

Concept Abstract Rules In the following, τ is an individual concept constant symbol or parameter.

$$\frac{\sigma\, \langle \lambda\alpha.\Phi(\alpha)\rangle(\tau)}{\sigma\, \Phi(\tau)} \qquad \frac{\sigma\, \neg\langle \lambda\alpha.\Phi(\alpha)\rangle(\tau)}{\sigma\, \neg\Phi(\tau)}$$

Object Abstract Rules In the following, τ is an individual object constant symbol, parameter, or *a prefixed individual concept constant symbol or parameter.*

$$\frac{\sigma\, \langle \lambda x.\Phi(x)\rangle(\tau)}{\sigma\, \Phi(\tau)} \qquad \frac{\sigma\, \neg\langle \lambda x.\Phi(x)\rangle(\tau)}{\sigma\, \neg\Phi(\tau)}$$

Before giving the next set of abstract rules, recall that we are allowing individual concepts to be partial functions in models. If τ is an individual concept constant symbol or parameter, what is the status of $\downarrow\tau$? If we are at a world in the domain of the individual concept named by τ, $\downarrow\tau$ should be the individual object designated by that concept at that world, and otherwise it should be $\bot$. Now, if we know $\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)$ is true at a world, it must be that $\downarrow\tau$ is not $\bot$ at that world, since an abstract applied to $\bot$ is false. In such a case we can introduce a name for the object designated by τ at the world; we do this by prefixing (subscripting) τ. On the other hand, if we know $\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)$ is false, it could be that τ does not designate, or it could be that it does, but designates something making Φ false. In such cases we need other information to conclude whether or not $\downarrow\tau$ is $\bot$. This gives us the motivation for the following rules.

Mixed Abstract Rules In the two rules following, τ is an individual concept constant symbol or parameter.

$$\frac{\sigma\, \langle \lambda x.\Phi(x)\rangle(\downarrow\tau)}{\sigma\, \Phi(\tau_\sigma)}$$

In addition, if τ_σ already occurs on the tableau branch, the following rule may be applied.

$$\frac{\sigma\, \neg\langle \lambda x.\Phi(x)\rangle(\downarrow\tau)}{\sigma\, \neg\Phi(\tau_\sigma)}$$

Rules similar to these apply to atomic formulas as well.

Atomic Rules In the two rules following, τ is an individual concept constant symbol or parameter, R is a relation symbol, and $\ldots$ represents a sequence of terms.

$$\frac{\sigma\, R(\ldots, \downarrow\tau, \ldots)}{\sigma\, R(\ldots, \tau_\sigma, \ldots)}$$

And, if τ_σ already occurs on the tableau branch, the following rule may be applied.

$$\frac{\sigma\, \neg R(\ldots, \downarrow\tau, \ldots)}{\sigma\, \neg R(\ldots, \tau_\sigma, \ldots)}$$

Finally, we have the rules for equality. The first one corresponds to the semantic fact that equality is interpreted the same at every world; if individual objects are equal at some world, they are equal at every world.

Equality Transfer Rule If τ_1 and τ_2 are individual object constant symbols or parameters, or prefixed individual concept constant symbols or parameters, and if σ' is a prefix that already occurs on the branch

$$\frac{\sigma\,(\tau_1 = \tau_2)}{\sigma'\,(\tau_1 = \tau_2)}$$

Reflexivity Rule If τ is an individual object constant symbol, parameter, or a prefixed individual concept constant symbol or parameter, and the prefix σ already occurs on the branch,

$$\frac{}{\sigma\,(\tau = \tau)}$$

Substitutivity Rule If τ_1 and τ_2 are individual object constant symbols or parameters, or prefixed individual concept constant symbols or parameters

$$\frac{\sigma\,\Phi(\tau_1) \qquad \sigma'\,(\tau_1 = \tau_2)}{\sigma\,\Phi(\tau_2)}$$

This concludes the presentation of the branch extension rules.

Definition 11 (Closed). *A tableau branch is* closed *if it contains* $\sigma\,X$ *and* $\sigma\,\neg X$ *for some* X*. A tableau is* closed *if every branch is closed.*

Definition 12 (Proof and Derivation). *A sentence* Φ *(without parameters) is* provable *if there is a closed tableau beginning with* $1\,\neg\Phi$*. Likewise,* Φ *has a* derivation *from a set* S *of sentences if there is a closed tableau beginning with* $1\,\neg\Phi$*, in which* $1\,X$ *can be added to any branch at any point, for any* X *that is a member of* S*.*

This concludes the description of the tableau system.

11 A Derivation Example

The example given here is a derivation of

$$\neg\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})] \tag{10}$$

from the axioms of Section 9. It establishes that $\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ cannot be inserted into the database, because it violates an integrity constraint. Before presenting the derivation itself, here is a derived rule that will shorten the presentation.

Derived Rule Suppose τ is an individual concept constant symbol or parameter.

$$\frac{\sigma\ {\downarrow}\tau = a \qquad \sigma\,\mathbf{D}(\tau) \supset \Phi}{\sigma\,\Phi}$$

Think of this as abbreviating the following tableau construction.

$$\begin{array}{l} \sigma\ {\downarrow}\tau = a \quad 1. \\ \sigma\,\mathbf{D}(\tau) \supset \Phi \quad 2. \\ \sigma\,\tau_\sigma = a \quad 3. \end{array}$$

$$\begin{array}{l} \sigma\,\neg\mathbf{D}(\tau) \quad 4. \\ \sigma\,\neg\langle\lambda x.\ {\downarrow}x = {\downarrow}x\rangle(\tau) \quad 6. \\ \sigma\,\neg({\downarrow}\tau = {\downarrow}\tau) \quad 7. \\ \sigma\,\neg(\tau_\sigma = \tau_\sigma) \quad 8. \\ \sigma \quad (\tau_\sigma = \tau_\sigma) \quad 9. \end{array} \qquad \begin{array}{l} \sigma\,\Phi \quad 5. \end{array}$$

Explanation: 3 is from 1 by an atomic rule; 4 and 5 are from 2 by a disjunctive rule; 6 is 4 unabbreviated; 7 is from 6 by a concept abstract rule; 8 is from 7 by an atomic rule, making use of the fact that τ_σ occurs in 3; 9 is by the reflexivity rule. The left branch is closed, and the right branch gives the effect of the conclusion of the derived rule.

Now, the derivation of formula (10) is in Figure 1. The explanation of Figure 1 is as follows: 2 is from 1 by the double negation rule; 3 is from 2 by a possibility rule; 4, 5, and 6 are from 3 by repeated uses of a conjunction rule; 7 introduces axiom 5 of Section 9; 8 is from 7 by a necessity rule; 9 is from 5, 6, and 8 by repeated uses of the derived rule above; 10 is from 9 by a mixed abstract rule; 11 is from 10 by an object abstract rule; 12 is axiom 7 of Section 9; 13 is from 12 by a possibility rule; 14, 15, and 16 are from 13 by repeated conjunction rules; 17 is from 11 by a necessity rule; 18 is from 15, 16, and 17 by the derived rule above (slightly modified); 19 and 20 are from 18 by a disjunctive rule; 21 is from 16 by an atomic rule, as are 22 from 20, 23 from 19, 24 from 5, 25 from 6, and 26 from 15; 27 is from 25 by an equality transfer rule, as is 28 from 24; 29 is from 23 and 27 by a substitutivity rule, as are 30 from 22 and 28, 31 from 27 and 29, and 32 from 26 and 30; 33 is by the reflexivity rule; 34 is by axiom 13 of Section 9; and 35 is from 34 by an equality transfer rule.

As another, and simpler, example, you might try giving a derivation of the following, essentially establishing that Eve is someone who works Away.

$$\begin{array}{l} \langle\lambda x, y.(\exists z)\{\Diamond[\mathtt{PERSONS} \wedge ({\downarrow}\mathtt{NAME} = x) \wedge ({\downarrow}\mathtt{JOBNUMBER} = z)] \\ \qquad \wedge\Diamond[\mathtt{LOCATION} \wedge ({\downarrow}\mathtt{JOBNUMBER} = z) \wedge ({\downarrow}\mathtt{WHERE} = y)]\}\rangle(\mathtt{Eve}, \mathtt{Away}) \end{array} \tag{11}$$

12 Completeness et. al.

In this section I'll sketch soundness and completeness arguments for the tableau system, as well as give a proof for Proposition 1. Nothing is given in much detail, because proofs are straightforward adaptations of what are, by now, fairly standard arguments.

12.1 Soundness

Soundness is by the usual tableau method. One defines a notion of *satisfiability* for prefixed formulas—a set S is satisfiable if there is a model $\mathcal{M}$, a mapping m assigning to each prefix σ a possible world $m(\sigma)$ of $\mathcal{M}$, and a formula Φ is true at world $m(\sigma)$ of $\mathcal{M}$ whenever $\sigma\,\Phi \in S$. A tableau is called satisfiable if the set of prefixed formulas on one of its branches is satisfiable. Then one shows that each tableau rule preserves tableau satisfiability. This requires a case by case check.

Now, if there is a closed tableau for $1\,\neg X$, then X must be valid. For, otherwise, there would be some model in which X was false at some world. It follows that the set $\{1\,\neg X\}$ is satisfiable, so we begin with a satisfiable tableau. Then we can only get satisfiable tableaus, and since we had a closed tableau for $1\,\neg X$ we have the impossible situation of having a closed, satisfiable tableau.

12.2 Completeness

Suppose X is a sentence that has no tableau proof—it must be shown that X is not valid. Again the methodology is standard. Also, while the proof sketch below is just for tableau provability, and not derivability, the argument extends directly. I'm giving the simpler version, for simplicity.

Begin by constructing a tableau for $1\,\neg X$, and do so *systematically*, in such a way that all tableau rules are *fairly* applied. That is, during the tableau construction, any rule that could eventually be applied is. There are many such fair tableau construction procedures—I'll leave the details to you. The result is a tableau that does not close—say it has an open branch θ (König's lemma is needed to guarantee such a branch exists, if the tableau construction is infinite).

Now, construct a model as follows.

1. The set $\mathcal{G}$ of possible worlds is the set of prefixes that occur on branch θ.
2. The domain $\mathcal{D}_o$ of objects is the set consisting of: all individual object constant symbols of the language, all individual object parameters that occur on θ, and all subscripted (prefixed) individual concept constant symbols and parameters that occur on θ.
3. if f is an individual concept constant symbol, or individual concept parameter that occurs on θ, a function $\widehat{f}$ is defined as follows. The domain of $\widehat{f}$ is the set of prefixes σ such that f_σ occurs on θ. And if σ is in the domain of $\widehat{f}$ then $\widehat{f}(\sigma) = f_\sigma$. Note that $\widehat{f}$ maps a subset of $\mathcal{G}$ to $\mathcal{D}_o$. The domain $\mathcal{D}_c$ of concepts is the set of all these $\widehat{f}$.
4. For the interpretation $\mathcal{I}$:

a) $\mathcal{I}$ assigns to each member of $\mathcal{D}_o$ itself.
b) $\mathcal{I}$ assigns to each f that is an individual concept constant or parameter (on θ) the function $\widehat{f}$.
c) $\mathcal{I}$ assigns to a relation symbol P of type $\langle\rangle$ the mapping from $\mathcal{G}$ to $\{false, true\}$ such that $\mathcal{I}(P)(\sigma) = true$ iff $\sigma\, P$ occurs on θ.
d) To make this clause easier to state, I'll use the following notation. If f is an object symbol, set $\widehat{f} = f$. If f is a concept constant or parameter, $\widehat{f}$ has already been defined. Now, $\mathcal{I}$ assigns to a relation symbol R of type $\langle n_1, n_2, \ldots, n_k\rangle$ a mapping on $\mathcal{G}$ such that $\langle\widehat{t_1}, \ldots, \widehat{t_k}\rangle \in \mathcal{I}(R)(\sigma)$ iff $\sigma\, R(t_1, \ldots, t_k)$ occurs on θ.

This completes the definition of a model, call it $\mathcal{M}$. Actually, the equality symbol may not be interpreted by equality, but leaving this aside for the moment, one can show by standard methods involving an induction on formula degree that, for any valuation v:

– If $\sigma\, \Phi$ occurs on θ then $\mathcal{M}, \sigma \Vdash_v \Phi$.
– If $\sigma\, \neg\Phi$ occurs on θ then $\mathcal{M}, \sigma \nVdash_v \Phi$.

The valuation v can be arbitrary because free variables do not occur in tableaus. Since $1\, \neg X$ begins the tableau, it occurs on θ, and hence X is not valid in $\mathcal{M}$ since it is false at world 1.

Finally, one "factors" the model $\mathcal{M}$ using the equivalence relation that is the interpretation of the equality symbol, turning it into a normal modal model. The Equality Transfer Rule tells us that the interpretation of the equality symbol will be the same at every world of $\mathcal{M}$. I'll leave details to you.

12.3 A Sketch of Proposition 1

Assume r is a relation instance and X a simple existential sentence X. All members $\mathsf{axiom}(r)$ are true in $\mathsf{model}(r)$, so if X is a consequence of $\mathsf{axiom}(r)$, then X will be true in $\mathsf{model}(r)$. It is the converse direction that needs work.

Suppose X is not a consequence of $\mathsf{axiom}(r)$. Then X does not have a tableau derivation from $\mathsf{axiom}(r)$. Suppose we now carry out the steps of the completeness argument, from Section 12.2. We begin a tableau for $1\, \neg X$, carry out its construction systematically and, since it is a derivation, we introduce members of $\mathsf{axiom}(r)$ onto the branch during the construction. As usual, I'll omit details.

Members of $\mathsf{axiom}(r)$ that are constraint axioms all involve $\Box$ in a positive location—they do not involve $\Diamond$. The sentence X is simple existential, and so $\neg X$ contains $\Diamond$ in a negative location—it behaves like $\Box$. None of these formulas, then, can invoke applications of a possibility rule. Only the instance axioms of $\mathsf{axiom}(r)$ can do this. So, if there are n members of $\mathsf{axiom}(r)$ that are instance axioms, an open tableau branch will have exactly $n + 1$ different prefixes on it: prefix 1 with which we started, and the n additional prefixes introduced by possibility rule applications to instance axioms.

Now, proceed with the construction of a model, using an open tableau branch, as outlined in Section 12.2. We get a model with $n+1$ worlds, with X is false at

world 1, and a world corresponding to each instance axiom. Because of the form of X (only existential quantifiers and a single possibility symbol), since it is false at world 1, it is also false at every world. If we now consider the submodel in which world 1 is dropped, it is not hard to check that truth values of members of $\mathsf{axiom}(r)$ do not change at remaining worlds, nor does the truth value of X. And the resulting model is (isomorphic to) $\mathsf{model}(r)$.

13 Conclusion

I want to finish by describing two plausible directions for future work, one having to do with the modal logic directly, the other with its applications to databases.

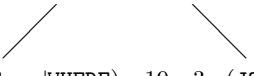
The tableau proof procedure given here used parameters and, as such, is meant for human application. But it should be possible to develop a free-variable version that can be automated. The **S5** modality itself is a kind of quantifier, but it is of a simple nature. Object quantification is essentially classical. Concept quantification may create some difficulty—I don't know. Equality plays a fundamental role, but it is a rather simple one. Perhaps what is needed can be captured efficiently in an automated proof procedure.

The databases considered here were all conventional relational ones. This is what the first-order modal language can handle. But one could consider multiple-valued databases, say, in which entries can be sets. Or for a more complicated example, consider this. Say a record represents a person, and among a person's attributes are these three: `FAVORITE_BOOK`, `FAVORITE_MOVIE`, and `MOST_IMPORTANT`. The first two attributes have the obvious meaning. The `MOST_IMPORTANT` attribute records which that person considers most important in evaluating someone, `FAVORITE_BOOK` or `FAVORITE_MOVIE`. Thus `MOST_IMPORTANT` is an attribute whose value is an attribute. (This example is meant to be easily described, and hence is rather artificial. More realistic examples are not hard to come by.) The modal logic of this paper is really the first-order fragment of a higher-type system, presented in full in [1]. If one uses that, one can easily have sets of objects, or attributes, as entries. Indeed, one can consider much more complex things yet. Of course the proof procedure also becomes more complex, as one would expect. Whether such things are of use remains to be seen.

References

1. Melvin C. Fitting. *Types, Tableaus, and Gödel's God.* 2000. Available on my web site: comet.lehman.cuny.edu/fitting.
2. Melvin C. Fitting and Richard Mendelsohn. *First-Order Modal Logic.* Kluwer, 1998. Paperback, 1999.
3. Saul Kripke. *Naming and Necessity.* Harvard University Press, 1980.
4. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, 1988.

$1\ \neg\neg\Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ 1.
$1\ \Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ 2.
$2\ \texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 1) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})$ 3.
$2\ \texttt{LOCATION}$ 4.
$2\ (\downarrow\texttt{JOBNUMBER} = 1)$ 5.
$2\ (\downarrow\texttt{WHERE} = \texttt{Away})$ 6.
$1\ \Box\{(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE})) \supset$
$\langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x = \downarrow\texttt{WHERE}) \supset$
$(y = \downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{WHERE}, \downarrow\texttt{JOBNUMBER})\}$ 7.
$2\ (\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE})) \supset$
$\langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x = \downarrow\texttt{WHERE}) \supset$
$(y = \downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{WHERE}, \downarrow\texttt{JOBNUMBER})$ 8.
$2\ \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x = \downarrow\texttt{WHERE}) \supset$
$(y = \downarrow\texttt{JOBNUMBER})]\rangle(\downarrow\texttt{WHERE}, \downarrow\texttt{JOBNUMBER})$ 9.
$2\ \langle\lambda x, y.\Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge x = \downarrow\texttt{WHERE}) \supset$
$(y = \downarrow\texttt{JOBNUMBER})]\rangle(\texttt{WHERE}_2, \texttt{JOBNUMBER}_2)$ 10.
$2\ \Box[(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge \texttt{WHERE}_2 = \downarrow\texttt{WHERE}) \supset$
$(\texttt{JOBNUMBER}_2 = \downarrow\texttt{JOBNUMBER})]$ 11.
$1\ \Diamond[\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ 12.
$3\ [\texttt{LOCATION} \wedge (\downarrow\texttt{JOBNUMBER} = 2) \wedge (\downarrow\texttt{WHERE} = \texttt{Away})]$ 13.
$3\ \texttt{LOCATION}$ 14.
$3\ (\downarrow\texttt{JOBNUMBER} = 2)$ 15.
$3\ (\downarrow\texttt{WHERE} = \texttt{Away})$ 16.
$3\ [(\mathbf{D}(\texttt{JOBNUMBER}) \wedge \mathbf{D}(\texttt{WHERE}) \wedge \texttt{WHERE}_2 = \downarrow\texttt{WHERE}) \supset$
$(\texttt{JOBNUMBER}_2 = \downarrow\texttt{JOBNUMBER})]$ 17.
$3\ [(\texttt{WHERE}_2 = \downarrow\texttt{WHERE}) \supset$
$(\texttt{JOBNUMBER}_2 = \downarrow\texttt{JOBNUMBER})]$ 18.



$3\ \neg(\texttt{WHERE}_2 = \downarrow\texttt{WHERE})$ 19.
$3\ (\texttt{WHERE}_3 = \texttt{Away})$ 21.
$3\ \neg(\texttt{WHERE}_2 = \texttt{WHERE}_3)$ 23.
$2\ (\texttt{WHERE}_2 = \texttt{Away})$ 25.
$3\ (\texttt{WHERE}_2 = \texttt{Away})$ 27.
$3\ \neg(\texttt{Away} = \texttt{WHERE}_3)$ 29.
$3\ \neg(\texttt{Away} = \texttt{Away})$ 31.
$1\ (\texttt{Away} = \texttt{Away})$ 33.

$3\ (\texttt{JOBNUMBER}_2 = \downarrow\texttt{JOBNUMBER})$ 20.
$3\ (\texttt{JOBNUMBER}_2 = \texttt{JOBNUMBER}_3)$ 22.
$2\ (\texttt{JOBNUMBER}_2 = 1)$ 24.
$3\ (\texttt{JOBNUMBER}_3 = 2)$ 26.
$3\ (\texttt{JOBNUMBER}_2 = 1)$ 28.
$3\ (1 = \texttt{JOBNUMBER}_3)$ 30.
$3\ (1 = 2)$ 32.
$1\ \neg(1 = 2)$ 34.
$3\ \neg(1 = 2)$ 35.

Fig. 1. Derivation Example

Local Symmetries in Propositional Logic

Noriko H. Arai[1] and Alasdair Urquhart[2] *

[1] Department of Computer Science
Faculty of Information Sciences
Hiroshima City University
3-4-1 Ozuka-higashi
Asaminami-ku
Hiroshima 731-31
JAPAN
narai@cs.hiroshima-cu.ac.jp
[2] Departments of Philosophy and Computer Science
University of Toronto
Toronto, Ontario M5S 1A1
CANADA
urquhart@cs.toronto.edu

Abstract. The symmetry rule in propositional logic allows the exploitation of symmetries present in a problem. In the context of resolution, the rule enables the shortening of refutations by using symmetries present in an initial set of clauses. These symmetries can be local or global. The present paper proves that the local symmetry rule is strictly more powerful than the global symmetry rule. It also exhibits sets of clauses that show exponential lower bounds for the local symmetry rule, where the symmetry group consists of all variable permutations. These examples remain exponentially hard even when the symmetry group is enlarged to include complementation. Examples are exhibited in which resolution with the global symmetry rule has an exponential speed-up with respect to the cutting plane refutation system.

1 Introduction

The symmetry rule arises naturally in proofs of combinatorial principles; in many cases it allows significant shortening of proofs. In proofs of combinatorial theorems like Ramsey's theorem, for example, the occurrence of a phrase such as "... without loss of generality, we may assume that ..." usually signals the exploitation of a symmetry present in the problem. Thus it is natural to consider the addition of this rule to familiar proof systems for propositional logic.

We discuss the efficiency of the rule and some of its variants in the context of the resolution proof system for propositional logic. In an earlier paper [16], the second author proved lower bounds on the size of resolution proofs employing

* The authors gratefully acknowledge the support of the Ministry of Education of Japan and the National Sciences and Engineering Research Council of Canada.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 40–51, 2000.

a global form of the rule; that is to say, the symmetry rule was applicable in cases where there was a global symmetry in the original contradictory set of clauses representing the input to the resolution refutation system. This paper also defined a local form of the symmetry rule, where local symmetries in the input clause sets could be exploited, but no lower bounds were proved for this stronger symmetry rule.

In this paper, we prove superpolynomial lower bounds for the local form of the symmetry rule, and also prove separation results for this form of the rule from the global form of the rule. The symmetry rule can of course be employed in other formulations of propositional logic. The first author has considered the effects of adding a form of the symmetry rule to cut-free Gentzen systems [3,2,1], and has shown that in many cases significant speed-ups can be obtained as compared with the system without the symmetry rule.

2 Definitions and Preliminaries

Before proceeding to consideration of particular proof systems, let us fix notation. We assume an infinite supply of propositional variables and their negations; a variable or its negation is a *literal*. We say that a variable P and its negation $\neg P$ are *complements* of each other; we write the complement of a literal l as $\bar{l}$. A finite set of literals is a *clause*; it is to be interpreted as the disjunction of the literals contained in it. The *width* of a clause C is the number of literals in it, and denoted by $w(C)$. The width of a set Σ of clauses is the maximum width of a clause in Σ.

We shall sometimes write a clause by juxtaposing the literals in it. An *assignment* is an assignment of truth-values to a set of propositional variables; some variables may remain unset under an assignment.

The resolution rule is a simple form of the familiar cut rule. If Al and $B\bar{l}$ are clauses, then the clause AB may be inferred by the resolution rule, *resolving on* the literal l. If either A or B is the empty clause, then we refer to the inference as a *unit resolution*. A *resolution refutation* of a set of clauses Σ is a derivation of the empty clause Λ from Σ, using the resolution rule.

If V is a set of variables, and σ a permutation of V, then for any clause C built from the variables in V, we define the clause $\sigma(C)$ to be the clause resulting from C by applying σ to each variable in C. For a set of clauses Γ, define $\sigma(\Gamma)$ to be $\{\sigma(C) \mid C \in \Gamma\}$.

The symmetry rule was introduced as an extension to the resolution system in a paper by Krishnamurthy [12]. The rule of symmetry allows the following inference. If a clause C has been derived from a set of clauses Γ, and $\sigma(\Gamma) = \Gamma$, then the clause $\sigma(C)$ can be inferred as the next step in the derivation. A proof from a set of clauses Γ in which each step is inferred by resolution from two earlier steps, or by the symmetry rule from an earlier step, is a *symmetric resolution* or *SR-I* proof.

The form of the symmetry rule just defined is designed to exploit global symmetries in a set of clauses. Krishnamurthy also defined a more general form

of the symmetry rule that is able to exploit local symmetries. The local symmetry rule allows the following inference. Suppose that C is a clause derived from a set of clauses Γ, and for every clause A in Γ used in the derivation of C, $\sigma(A)$ is also in Γ. Then the local symmetry rule allows the derivation of $\sigma(C)$. A proof from a set of clauses Γ in which each step is inferred by resolution from two earlier steps, or by the local symmetry rule from an earlier step, is a *locally symmetric resolution proof* or *SR-II* proof.

The symmetry rule can also be generalized in another direction. If L is the set of literals based on a set of n variables, then L is closed under the complementation group, whose elements are the 2^n complementation operations; such an operation interchanges literals and their complements, for some subset of the literals in L. The symmetric group of all permutations of the variables acts in a natural way on the set of literals; hence this group can be enlarged to a group of order $2^n \cdot n!$ by adding the complementation operations. Following Harrison [11, Ch. 5], let us call this enlarged group the *group of permutations and complementations*. We can extend the symmetry rule (in either its global or local form) by allowing the inference of $\sigma(C)$ from C, where σ is an operation in the group of permutations and complementations under which Γ is invariant (or under which Γ is closed, when σ is applied to the appropriate subset of Γ, in the local form of the rule). Let us call the resulting proof systems *SRC-I* and *SRC-II*.

For each of these proof systems, we define the *length* of a proof as the number of inferences in the proof; the *size* of a proof is the number of occurrences of symbols in it. For the inference systems defined above, the two measures of proof complexity are polynomially related. If P is a refutation system, we define the *P complexity* of a set of clauses as the length of a minimal size refutation of this set of clauses in P. Thus the *resolution complexity* of a contradictory set of clauses Σ is the minimal length of a resolution refutation of Σ.

To compare the relative efficiency of refutation systems for sets of clauses, we define a notion of efficient simulation. If S_1 and S_2 are both refutation systems for sets of clauses, then we say that S_1 *p-simulates* S_2 if whenever there is a refutation P_2 of a set of clauses Σ in the system S_2 there is a refutation P_1 of Σ in the system S_1, where the size of P_1 is bounded by a fixed polynomial in the size of P_2. If S_1 and S_2 p-simulate each other, then we say that they are *p-equivalent*.

In proving lower bounds, we shall employ various examples of sets of clauses based on finite graphs. The graph-theoretical terminology used in this paper is that of Bollobás [5]. A *graph* G is an ordered pair of finite sets (V, E) where E is a subset of the set of unordered pairs of V; the set $V = V(G)$ is the set of *vertices* of G, while $E = E(G)$ is the set of *edges* of G. If U, W are subsets of V, then we write $E(U, W)$ for the set of edges joining a vertex in U to a vertex in W. If x, y are vertices, then we write xy for the edge containing x and y. The *degree* of a vertex is the number of vertices adjacent to it. We say that two edges are *adjacent* if they have exactly one vertex in common; a set of edges in a graph is *independent* if no two edges in the set are adjacent. A matching in a graph G

is an independent subset of $E(G)$; the matching is *perfect* if every vertex in G belongs to one of the edges in the matching.

3 Separating Local and Global Symmetry Rules

It seems clear that local symmetry rules are more powerful than their corresponding global versions. It is fairly easy to verify this impression by constructing appropriate sets of clauses. Here we show an exponential speed-up of the local over global symmetry rules by employing sets of clauses based on matchings in graphs.

Given a graph $G = (V, E)$, we can formulate the assertion that G has a perfect matching as a set of clauses, $PM(G)$. Where $x, y \in V$, the propositional variable P_{xy} is to be read as asserting: "Vertex x is matched with vertex y." We identify the variable P_{xy} with the variable P_{yx}. If x is a vertex in V, the disjunction $D_x = \bigvee\{P_{xy} \mid y \text{ adjacent to } x\}$ asserts that x is matched with one of its neighbours. Similarly, the disjunction $E_{xyz} = (\neg P_{xy} \vee \neg P_{xz})$ asserts that x cannot be matched with both y and z. The set of clauses $PM(G)$ contains all the disjunctions D_x, for $x \in V$, together with all the disjunctions E_{xyz}, where $x, y, z \in V$, x is adjacent to y and z, and $y \neq z$. If G is a graph with n vertices, and maximum degree d, then $PM(G)$ has size $O(d^2 n)$.

If G has no perfect matching (for example, if G has an odd number of vertices) then $PM(G)$ is contradictory. Let $K(n+1, n)$ be the complete bipartite graph with $V = V_1 \cup V_2$, $|V_1| = n+1$, $|V_2| = n$, and $E = \{\{x, y\} \mid x \in V_1, y \in V_2\}$. The set of clauses $PM(K(n+1, n))$ is contradictory; the statement of this fact is a formulation of the pigeon-hole principle, so we shall refer to these clauses as the *pigeon-hole clauses* PHC_n. Armin Haken proved the following result about the complexity of resolution refutations of PHC_n.

Theorem 1. *There is a $c > 1$ so that any resolution refutation of PHC_n contains at least c^n distinct clauses.*

Proof. Haken's original proof is in [10]. A simpler proof can be found in [4]. □

The graph $K(n+1, n)$ is highly symmetric; we can use this fact to get a short SR-I refutation of PHC_n (an observation of Krishnamurthy). We formalize the following informal proof: "If there is an injective map f from $\{1, \cdots, n+1\}$ to $\{1, \cdots n\}$, then there is a k, $1 \leq k \leq n$, so that $f(n+1) = k$. By symmetry, we can assume that $k = n$. But then the map f' obtained by removing $\langle n+1, n\rangle$ from f is an injective mapping from $\{1, \cdots, n\}$ into $\{1, \cdots, n-1\}$. Hence the theorem follows by induction on n."

Theorem 2. *There are SR-I refutations of length $(3n+1)n/2$ for the pigeon-hole clauses PHC_n.*

Proof. A detailed proof of the theorem is in Urquhart [16]. □

To defeat the global symmetry rule, we can use the simple idea of attaching "tails" to the vertices of the graph G underlying the set of clauses $PM(G)$. That

is to say, we attach to each vertex of G a tail containing an even number of vertices, where each tail is of a different length. To make this idea precise, let us suppose we are given a graph G with vertices $\{x_1, \ldots, x_n\}$. With the vertex x_i we associate a set of "tail vertices" $\{t_1^i, \ldots, t_{2i}^i\}$. The "graph with tails" G^t constructed from G is defined to be the graph whose vertices consist of the original vertices of G, together with all the new tail vertices. The edges of G^t consists of the original edges of G, together with edges joining x_i to t_1^i, t_1^i to t_2^i, $\ldots$, t_{2i-1}^i to t_{2i}^i, for each vertex x_i of G. Figure 1 illustrates the result of attaching tails to the complete graph K_4.

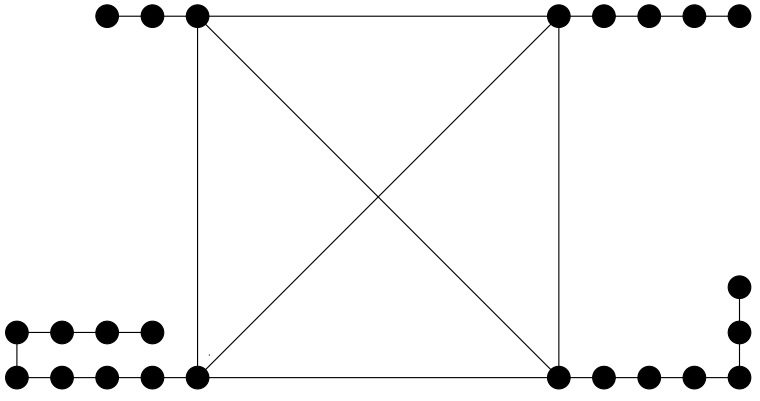


Fig. 1. The graph K_4 with tails added

Lemma 1. *If G is a graph of degree at least 2, then G^t, the result of adding tails to G, has no global symmetries. Hence, PM(G^t) has no non-trivial global symmetries.*

Proof. Any symmetry of G^t must map the vertices at the ends of tails into other vertices of the same type (since they are the only vertices of degree 1). It follows that tails must be mapped into tails. But then the symmetry must be the identity, for otherwise a vertex of degree 2 would be mapped into a vertex of greater degree, since all the tails are of different lengths. The set of clauses $PM(G^t)$ therefore has no non-trivial global symmetries, since any symmetry of the set of clauses would give rise to a symmetry of G. □

We can now separate the local and global symmetry rules by employing the trick of adding tails to the pigeon-hole clauses. That is, let $K(n+1, n)^t$ be the result of adding tails to the complete bipartite graph $K(n+1, n)$. Then the addition of tails defeats the global, but not the local symmetry rule. Let us write PHC_n^t for the set of "pigeon-hole clauses with tails," that is, the set of clauses $PM(K(n+1, n)^t)$. Before proving this result, we prove a simple lemma about restrictions of resolution refutations.

Lemma 2. *Let Σ be a set of clauses, and ϕ an assignment to a subset of the variables in Σ. If $R = C_1, \cdots, C_k$ is a resolution refutation of Σ, then $R\restriction \phi = C_1\restriction \phi, \cdots, C_k\restriction \phi$ contains a subsequence that is a resolution refutation of $\Sigma\restriction \phi$.*

Proof. We prove by induction on the length of the refutation the more general claim: if $C_j\restriction \phi \neq 1$, then $C_1\restriction \phi, \cdots, C_j\restriction \phi$ contains a subsequence that is a resolution proof of a subclause of $C_j\restriction \phi$. □

Theorem 3. *1. There is a $c > 1$ so that any SR-I refutation of PHC_n^t contains at least c^n distinct clauses.*
2. There are SR-II refutations of length $(3n+1)n/2 + O(n)$ of PHC_n^t.

Proof. Since PHC_n^t has no global symmetries, it follows that the global symmetry rule cannot be employed in a refutation of this set of clauses. Thus any SR-I refutation of PHC_n^t must simply be a resolution refutation. Let ϕ be the assignment to the edge variables that results from matching all the tail vertices with other tail vertices (this can be done because all the tails are of even length). The result of applying this restriction to PHC_n^t is the original set of pigeon-hole clauses PHC_n. By Lemma 2, any resolution refutation of PHC_n^t must be at least as long as a resolution refutation of PHC_n. Hence, an exponential lower bound follows by Haken's lower bound for the pigeon-hole clauses (Theorem 1).

To construct short SR-II refutations of PHC_n^t, we can adapt the original short SR-I refutations described in Theorem 2. The clauses associated with the vertices of the original bipartite graph $K(n+1, n)$ in PHC_n^t are identical with the original pigeon-hole clauses PHC_n, except that for each vertex x, the positive clause D_x associated with x contains an extra variable P_x representing the edge attaching x to its associated tail. Clearly this subset of the clauses has the same set of symmetries as PHC_n. Thus we can imitate the short SR-II refutation of PHC_n to obtain a short SR-II proof of the clause $\{P_x |\ x \text{ a vertex in } K(n+1, n)\}$. By repeated application of unit resolution, we can then derive the empty clause in $O(n)$ steps. □

Corollary 1. *SR-I cannot p-simulate SR-II.*

4 Exponential Lower Bounds for SRC-II

In the previous section, and also in the predecessor to this paper [16], the global symmetry rule was defeated by the simple process of constructing sets of clauses with no global symmetries, so that the global symmetry rule could not be applied. This suggests that we could apply the same idea to the local symmetry rule by constructing sets of clauses with no useful local symmetries. In this section, we give a very easy construction that, given an arbitrary set of clauses Σ as input, constructs a new set of clauses Σ^* that is not much larger than the original set, for which the local symmetry rule does not allow any useful speedup. This immediately leads to an exponential lower bound for SRC-II.

Let Σ be a set of clauses $\{C_1, \cdots, C_k\}$ where C_i's are ordered so that $w(C_i) \leq w(C_{i+1})$ for every $1 \leq i \leq k-1$. Let $p_1, \cdots, p_m$ be a set of new variables, where

$m = k(k+1)/2$. Now let the clauses $D_1, \cdots, D_k$ be defined as the sequence $C_1 p_1, C_2 p_2 p_3, \cdots$; that is to say, we add one new variable to C_1, two new variables to C_2, and so on. Note that $w(D_i) \neq w(D_j)$ for any $i \neq j$. Then we define Σ^* to be the set $\{D_1, \cdots, D_k\} \cup \{\neg p_i | 1 \leq i \leq m\}$. Let us write V for the variables in Σ, and V^* for V augmented with the new variables $p_1, \cdots, p_m$.

The new set of clauses consists of k clauses of widths bounded by $k + \max\{w(C_i) | 1 \leq i \leq k\}$, together with m unit clauses. All of the clauses have different widths, with the exception of the unit clauses. It follows that Σ^* has only a very restricted set of local symmetries. Let ϕ be the assignment that sets all the new variables $p_1, \cdots, p_m$ to 0, so that $\Sigma^* \restriction \phi = \Sigma$. If σ is a permutation of V^*, then let us write $\sigma \restriction V$ for σ restricted to the set V of original variables.

Lemma 3. *If σ is a local symmetry of Σ^*, then $\sigma \restriction V$ induces the identity map on the clauses in Σ.*

Proof. If $D_i \in \Sigma^*$, then $\sigma(D_i) = D_i$, because all the clauses in Σ^*, with the exception of the unit clauses, are of different widths. It follows immediately that $\sigma(C_i) = C_i$. □

The next lemma is just a slightly more elaborate version of Lemma 2.

Lemma 4. *If $R = C_1, \cdots, C_k$ is an SRC-II refutation of Σ^*, then $R \restriction \phi = C_1 \restriction \phi, \cdots, C_k \restriction \phi$ contains a subsequence that is a resolution refutation of $\Sigma = \Sigma^* \restriction \phi$.*

Proof. We prove by induction on the length of the refutation the more general claim: if $C_j \restriction \phi \neq 1$, then $C_1 \restriction \phi, \cdots, C_j \restriction \phi$ contains a subsequence that is a resolution proof of a subclause of $C_j \restriction \phi$. For the input clauses, this is immediate, since the unit clauses are set to 1 by the restriction ϕ. Let us assume that C_j is inferred from C_h and C_i by the resolution rule. If the variable resolved on is one of the new variables $p_1, \cdots, p_m$, then one of the premisses must be a unit clause belonging to Σ^*. It follows that $C_j \restriction \phi$ must be identical with one of the premisses $C_h \restriction \phi$ or $C_i \restriction \phi$. If the variable resolved on belongs to V, then $C_j \restriction \phi$ can be inferred from $C_h \restriction \phi$ and $C_i \restriction \phi$ by resolution. Finally, if $\sigma(C_i)$ is inferred by the local symmetry rule from C_i, then by Lemma 3, $\sigma(C_i \restriction \phi) = C_i \restriction \phi$. □

The next theorem follows immediately from Lemma 4.

Theorem 4. *If Σ is a set of clauses, then the SRC-II complexity of Σ^* is bounded from below by the resolution complexity of Σ.*

Theorem 4 guarantees that there exists as many hard examples for SRC-II as for resolution. For example, the sequence of sets of clauses $\{PHC_k^*\}$ obtained from the pigeon-hole clauses is exponentially hard for SRC-II. There are sequences $\{\Sigma_k\}$ of contradictory sets of clauses in 3CNF of size $O(k)$ that require resolution refutations of size $2^{\Omega(k)}$ (for details, see [15,7,4]). Consequently, we obtain the following result.

Corollary 2. *There exists a sequence of contradictory sets of clauses* $\{\Delta_k\}$ *such that any SRC-II refutation of* Δ_k *contains* $2^{\Omega(k)}$ *clauses, though the size of each* Σ_k *is* $O(k^2)$.

The cutting plane system is another refutation system for propositional formulas in conjunctive normal form. In the cutting plane system, the truth values true and false are interpreted by 1 and 0 and propositional formulas are expressed by systems of linear inequalities. The goal (in constructing a refutation) is to derive the inequality $0 \geq 1$. It has its origins in the work of Gomory [9] in integer linear programming. It can also be considered as a generalization of resolution, since it is easy to give an efficient translation from sets of clauses to sets of linear inequalities so that if the original set of clauses has a resolution refutation, the corresponding set of linear inequalities has a cutting plane refutation that is not much longer. The cutting plane system is strictly more powerful than the resolution system, since, for example, the pigeon-hole clauses have short cutting plane refutations; for details see [8].

The cut-free sequent calculus is also a refutational system for propositional formulas when we restrict the system so that every sequent is of the form,

$$C_1, \ldots, C_n \rightarrow \bot$$

where C_i are clauses. The cut-free proofs are expressed either as trees or directed acyclic graphs. When expressed as trees, the cut-free sequent calculus and the analytic tableau system p-simulate each other. When expressed as directed acyclic graphs, it is as efficient as resolution on the class of formulas written in 3CNF. By adding the symmetry rule on the (DAG-like) cut-free sequent calculus, we obtain a system called *simple combinatorial reasoning*. Simple combinatorial reasoning polynomially proves many combinatorial problems including the pigeon-hole clauses [3,2,1].

Before we finish this section, we note that both cutting planes and simple combinatorial reasoning polynomially prove $\{PHC_k^*\}$; SRC-II p-simulates neither of them. In this paper, we only outline the proof. First we show the following lemma.

Lemma 5. *Let* Σ *is a set of clauses of size* n. *Suppose that* Σ *is reduced to another set of clauses* Σ' *by using only unit resolution. If cutting planes has a proof for* Σ' *of size* m, *it has a proof for* Σ *of size* mn^2. *The same thing holds for simple combinatorial reasoning.*

It is easy to show that for any sets of clauses Σ, Σ^* is reducible to Σ by using only unit resolution. More specifically, PHC_k^* is reducible to PHC_k by using unit resolution. Consequently, both cutting planes and simple combinatorial reasoning polynomially proves $\{PHC_k^*\}$.

Theorem 5. *SRC-II p-simulates neither cutting planes nor simple combinatorial reasoning.*

5 Separation of SR-I from Cutting Planes

In the previous section, we gave an exponential lowerbound for SRC-II and show that SRC-II p-simulates neither cutting planes nor simple combinatorial reasoning. In this section, we show that the cutting plane proof system and SRC-II are incomparable in the p-simulation ordering. This result follows from an exponential lower bound proved by Pudlák [13] and our polynomial upper bound for the same sequence of formulas.

We now define the examples used by Pudlák in proving lower bounds for the cutting plane system. We define k-$Clique(n)$ to be the following set of clauses:

1. $\{q_{i,1}, \ldots, q_{i,n}\}$ for $1 \leq i \leq k$,
2. $\{\neg q_{i,m}, \neg q_{j,m}\}$ for $1 \leq m \leq n$ and $1 \leq i < j \leq k$, and
3. $\{\neg q_{i,m}, \neg q_{j,l}, p_{m,l}\}$ for $1 \leq m < l \leq n$ and $1 \leq i, j \leq k$.

The above clauses encode a graph which has n vertices and contains a k-clique as follows. We enumerate all the vertices of the graph $\{1, \ldots, n\}$. The q's encode a function f from $\{1, \ldots, k\}$ to $\{1, \ldots, n\}$. The literal $q_{i,l}$ means that $f(i) = l$. (The intuitive meaning of $f(i) = l$ is that the vertex named i in the graph is actually the vertex named l in the k-clique.) The $p_{m,l}$ encode that there exists an edge between m and l. Hence, the first clause means that the function f is defined for all i $(i = 1, \ldots, k)$. The second clause means that f is one-to-one. The third clause means that if there exists i, j such that $f(i) = m$ and $f(j) = l$, then there exists an edge between m and l. Note that k-$Clique(n)$ corresponds to the positive test graph in the proof of lower bounds for monotone circuits (Razborov [14], Boppana and Sipser [6]). Pudlák's result rests on a generalization of this proof.

We define k'-$Color(n)$ to be the following set of clauses:

1. $\{r_{m,1}, \ldots, r_{m,k'}\}$ for $1 \leq m \leq n$,
2. $\{\neg r_{m,i}, \neg r_{m,j}\}$ for $1 \leq m \leq n$ and $1 \leq i < j \leq k'$, and
3. $\{\neg r_{m,i}, \neg r_{l,i}, \neg p_{m,l}\}$ for $1 \leq m < l \leq n$ and $1 \leq i \leq k'$.

The above clauses encode a graph which is a k'-partite graph as follows. The r's encode a coloring function g from $\{1, \ldots, n\}$ to $\{1, \ldots, k'\}$. The literal $r_{m,i}$ means that the vertex named m is colored by i. Hence, the first clause means that every vertex is colored. The second clause means that none of the vertices has more than one color. The third clause means that the coloring is proper: when the vertices m and l have the same color, then there is no edge between m and l. Note that k'-$Color(n)$ corresponds to the negative test graph.

We now define k-$Test(n)$ to consist of all the clauses in k-$Clique(n)$ together with the clauses in (k–1)-$Color(n)$. The size of k-$Test(n)$ is $O(n^4)$. It is easy to see that k-$Test(n)$ is unsatisfiable: as a matter of fact, if all the clauses in k-$Clique(n)$ is true, then the graph contains a k-clique. A k-clique cannot have a proper $(k-1)$-coloring. This means at least one of the clauses in (k–1)-$Color(n)$ must be false.

Theorem 6. *If* $k = \lfloor \frac{1}{8}(n/\log n)^{2/3} \rfloor$, *then any cutting plane refutation of* k-$Test(n)$ *must contain* $2^{\Omega((n/\log n)^{1/3})}$ *steps.*

Proof. See Pudlák [13], Theorem 6 and Corollary 7. □

The set of clauses k-$Test(n)$ has a very large global symmetry group. This can be seen easily if we give a slightly different description of the set. It can be viewed as defined with respect to three different sets, a set with k members, the set of *indices*, a set with n members, the set of *vertices*, and a set with $k-1$ members, the set of *colors*. Then the set of clauses k-$Clique(n)$ says that there is a one-to-one map from the set of indices onto a k-clique defined on the set of vertices, while (k–1)-$Color(n)$ says that there is a proper $k-1$-coloring of the graph defined on the set of vertices.

The whole set of clauses is invariant under the following operations: any permutation of the index set, any permutation of the vertices, and any permutation of the colors. This very large symmetry group allows us to find short refutations by using the symmetry rule. These short proofs are counterparts of the corresponding short proofs in a cut-free Gentzen calculus with permutation that were constructed by the first author [1].

Theorem 7. k-$Test(n)$ *has a refutation of length* $O(k(n+k))$ *in SR-I.*

Proof. Denote the clauses of the form $\{\neg q_{i,i}, \ldots, \neg q_{k-1,k-1}, \neg q_{k,k}\}$ by C_i for each $1 \leq i \leq k$ and the empty clause by C_{k+1}. Our polynomial-size refutation of k-$Test(n)$ consists in giving short derivations of $C_1, C_2, \ldots, C_{k+1}$.

The following set of clauses expresses that there is no 1-1 and onto function from the set of k objects to the set of $k-1$ objects.

1. $\{r_{m,1}, \ldots, r_{m,k-1}\}$ for $1 \leq m \leq k$,
2. $\{\neg r_{m,i}, \neg r_{m,j}\}$ for $1 \leq m \leq k$ and $1 \leq i < j \leq k-1$, and
3. $\{\neg r_{m,i}, \neg r_{l,i}\}$ for $1 \leq m < l \leq k$ and $1 \leq i \leq k-1$.

Thus, this set of clauses is identical with the pigeon-hole clauses PHC_{k-1}. By Theorem 2, this set has an SR-I refutation of length $O(k^2)$.

Hence, we can derive the clause $\{\neg p_{m,l} \mid 1 \leq m < l \leq k\}$ in $O(k^2)$ steps from the following subset of (k–1)-$Color(n)$:

1. $\{r_{m,1}, \ldots, r_{m,k-1}\}$ for $1 \leq m \leq k$,
2. $\{\neg r_{m,i}, \neg r_{m,j}\}$ for $1 \leq m \leq k$ and $1 \leq i < j \leq k-1$, and
3. $\{\neg r_{m,i}, \neg r_{l,i}, \neg p_{m,l}\}$ for $1 \leq m < l \leq k$ and $1 \leq i \leq k-1$.

By resolving $\{\neg p_{m,l} \mid 1 \leq m < l \leq k\}$ with the clauses

$$\{\neg q_{1,1}, \neg q_{2,2}, p_{1,2}\}, \ldots, \{\neg q_{k-1,k-1}, \neg q_{k,k}, p_{k-1,k}\},$$

we obtain C_1. Intuitively, C_1 says that the set of vertices $1, 2, \ldots, k$ do not form a k-clique if they are properly colored by $(k-1)$ colors.

We now show by induction on i that, starting from the clause C_1, we can derive C_i, for $1 < i \leq k$ in $O(i(n+i))$ steps. Let us assume that we have already

derived C_i, for $i < k$; we show how to derive C_{i+1} in $n + i + 1$ extra steps. By resolving C_i with the input clause $\{q_{i,1}, \ldots, q_{i,n}\}$, we obtain the clause

$$\{q_{i,1}, q_{i,2}, \ldots, q_{i,i-1}, q_{i,i+1}, \ldots, q_{i,n}, \neg q_{i+1,i+1}, \ldots, \neg q_{k,k}\}.$$

Resolving this last clause against the $(n - i)$ input clauses $\{\neg q_{i,j}, \neg q_{j,j}\}$ for $i < j \leq n$, we obtain the clause $\{q_{i,1}, q_{i,2}, \ldots, q_{i,i-1}, \neg q_{i+1,i+1}, \ldots, \neg q_{k,k}\}$. We can apply the symmetry rule to C_i to derive the clauses $\{\neg q_{i,j}, \neg q_{i+1,i+1}, \ldots, \neg q_{k,k}\}$, for $1 < j < i$ (that is to say, we apply the global symmetry that results from interchanging the vertices 1 and j). Using these clauses in $i - 2$ applications of the resolution rule, we finally derive C_{i+1}. This derivation of C_{i+1} from C_i takes $1 + (n - i) + 2(i - 2) = n + i + 1$ inference steps.

It follows that the derivation of C_k from C_1 takes $O(k(n+k))$ steps. Finally, from $C_k = \{\neg q_{k,k}\}$, we can derive $\{\neg q_{k,j}\}$, for $j \neq k$ by the symmetry rule (interchanging the vertices k and j), and hence the empty clause by n resolution steps, starting from the input clause $\{q_{k,1}, \ldots, q_{k,n}\}$. This last part of the derivation takes $O(n)$ steps, so the entire refutation in SR-I takes $O(k(n + k))$ inference steps. □

Corollary 3. *The cutting plane proof system cannot p-simulate SR-I.*

Together with Theorem 5, we obtain the following corollary.

Corollary 4. *The cutting plane system and SRC-II are incomparable in the p-simulation ordering.*

References

1. Noriko H. Arai. No feasible monotone interpolation for cut-free Gentzen type propositional calculus with permutation inference. Forthcoming, *Theoretical Computer Science*.
2. Noriko H. Arai. Tractability of cut-free gentzen type propositional calculus with permutation inference II. Forthcoming, *Theoretical Computer Science*.
3. Noriko H. Arai. Tractability of cut-free gentzen type propositional calculus with permutation inference. *Theoretical Computer Science*, 170:129–144, 1996.
4. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 517–526, 1999.
5. Béla Bollobás. *Graph Theory*. Springer-Verlag, 1979.
6. Ravi Boppana and Michael Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 757–804. MIT Press/Elsevier, 1990.
7. Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35:759–768, 1988.
8. W. Cook, C.R. Coullard, and Gy. Turán. On the complexity of cutting plane proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.
9. Ralph E. Gomory. An algorithm for integer solutions of linear programs. In R.L. Graves and P. Wolfe, editors, *Recent advances in mathematical programming*, pages 269–302. McGraw-Hill, 1963.

10. Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.
11. Michael A. Harrison. *Introduction to switching and automata theory.* McGraw-Hill Book Company, 1965.
12. Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–275, 1985.
13. Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62:981–998, 1997.
14. Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281:798–801, 1985. English translation, *Soviet Mathematics, Doklady*, Vol. 31 (1985), 354-357.
15. Alasdair Urquhart. Hard examples for resolution. *Journal of the Association for Computing Machinery*, 34:209–219, 1987.
16. Alasdair Urquhart. The symmetry rule in propositional logic. *Discrete Applied Mathematics*, 96-97:177–193, 1999.

Design and Results of TANCS-2000 Non-classical (Modal) Systems Comparison*

Fabio Massacci[1] and Francesco M. Donini[2]

[1] Dip. di Ingegneria dell'Informazione — Università di Siena
Via Roma 56, 53100 Siena, Italy — massacci@dii.unisi.it
[2] Dip. di Elettrotecnica ed Elettronica — Politecnico di Bari
Via Re David 200, 70125 Bari, Italy — donini@poliba.it

Abstract. The aim of the TABLEAUX-2000 Non-Classical (Modal) System Comparisons (TANCS-2000) is to provide a set of benchmarks and a standardized methodology for the assessment and comparison of ATP systems in non-classical logics, as it is done for first-order logic with the CADE System Competition. We believe that TANCS can benefit the scientific community in two ways: by promoting the competition among ATP systems and thus yielding novel solutions, and by providing a scientific design for benchmarking non-classical ATP systems.
This paper reports the main ideas behind the design, the benchmarks, the organization, and the rating of the ATP systems of TANCS-2000.

1 Design and Organization of the Comparison

The first Comparison was held in 1998 [1], a second one was held in 1999 [11] and this one continues the series with a focus on expressive modal and description logics (for an introduction to modal logics see [7], for description logics see [3]). The following Automated Theorem Proving (ATP) systems have been submitted this year: MSPASS (based on resolution), *SAT, FaCT, DLP and RACE (based on various optimization of tableaux and Davis-Putnam procedures). Their descriptions can be found in these proceedings.

As in past years, they are compared along two yardsticks: *effectiveness* and *usability*. Effectiveness can be measured on the basis of the type and number of problems solved, the average runtime for successful solutions, the scaling of the prover as problems gets bigger. Usability can be assessed on the basis of availability via web or other sources, portability to various platforms, need for additional software besides the prover itself, ease of installation and use (e.g., visual interfaces), possibility of customizing the search heuristics, etc.

Concerning effectiveness, a sensitive decision is the *choice of benchmark problems* which should offer the possibility to generate enough different samples so

* More details are at http://www.dis.uniroma1.it/~tancs. We thank S. Demri for his valuable contribution, P. Patel-Schneider for beta-testing some benchmarks and all TANCS participants for their constructive suggestions. TANCS problem repository is hosted by the Dipartimento of Informatica e Sistemistica, Università di Roma "La Sapienza", ITALY. Work supported by MURST (project MOSES) and CNR.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 52–56, 2000.

that "benchmark-tailored" techniques will not work, and which are either representative of the difficulties of the underlying satisfiability decision problem or representative of some real-world case.

Another key decision is the *rating of the systems* which cannot just rely on raw running times nor on internal aspects of the algorithm (e.g. Davis-Putnam calls) as we may end up with the impossibility of comparing in any fair way the performance of ATPs using different hardware, operating systems and calculi.

2 Benchmark Problems

Benchmarks are grouped into main divisions and then into categories, as in the CADE System Competition [17], according to the complexity of the corresponding decision problem[1]: a modal PSPACE division, a multi-modal PSPACE one, a global PSPACE division, a modal EXPTIME division.

For each category within a division there are a few *reference problems* which every entrant of the comparison has to try. Then a *C program* can generate all random instances of one's size and choice, as in '99 [11]. Besides parameters such as numbers of clauses and variables, the C program makes it possible to choose between a "plain" version of the benchmark and "modalized" one which hides away propositional reasoning. For details see [11].

The basic benchmark is Unbounded Modal QBF. It was first proposed for TANCS in [11] and its intuition is to encode validity of Quantified Boolean Formulae (QBF) into satisfiability in modal logic K.

In practice we generate a QBF with c clauses, alternation depth equal to d, with at most v variables for alternation. Setting $d = 3$ and $v = 2$, we can generate a QBF like $\forall v_{32} v_{31}.\exists v_{22} v_{21}.\forall v_{12} v_{11}.\exists v_{02} v_{01}.cnf_{c\text{-clauses}}(v_{01} \dots v_{32})$. For each clause we randomly generate k different variables (default 4) and each is negated with probability 0.5. The first and the third variable (if it exists) are existentially quantified, whereas the second and fourth variable are universally quantified. This aims at eliminating trivially unsatisfiable formulae [2]. Other literals are either universal or existentially quantified variables with probability 0.5. The depth of each literal is randomly chosen from 1 to d.

The QBF formula can be translated into modal logic with different encodings:

1. An optimization of Ladner's original translation [10] that does not introduce new variables but still contains many formulae which guarantees that a tree-like model is constructed following the alternating quantifier prefix;
2. a further optimized translation in which the formulae corresponding to the alternation depth are somewhat compiled away;
3. a yet more optimized translation which is fairly close to Schmidt-Schauß and Smolka's reduction of QBF validity into $\mathcal{ALC}$ satisfiability [15].

[1] We recall that deciding modal logic satisfiability is PSPACE complete and EXPTIME-complete if one uses global axioms Fitting-style [4]. However, not necessarily every benchmark set is able to capture these complexity classes.

For every fixed valued of d we can capture hard problems at the $d+1$-th level of the polynomial hierarchy in the same way that is done for 3-SAT problems [16]: we fix the ratio clauses over variables to some value[2] where a phase transition SAT/UNSAT occurs and increase the number of variables. That's better than [6] as we move upward in the complexity chain, yet PSPACE can only be reached by an unbounded and always increasing value of d (but then we can set v=1).

Compared to TANCS-1999, this benchmark has been enhanced to accommodate more expressive modal constructs: benchmark problems generated from QBF have also been encoded using the *inverse* operator on modalities (aka converse in $\mathcal{ALCI}$) and *sequence, union and star* of Converse Propositional Dynamic Logic (or their description logic equivalent). The basic idea behind the encodings is to replace a single relation in modal logic K by a sequence of back-forth-back relations, in some clever way to avoid making most formulae unsatisfiable.

Finally, the benchmark Periodic Modal CNF can capture PSPACE. It encodes periodic satisfiability problems [13] with global axioms Fitting-style [4]. We refer to [11] for further details on this encoding.

3 Usability of Submitted Systems

Since TANCS deals primarily with ATP systems, we considered only usability issues [12] which could be reasonable for ATP. Before the systems were submitted, we envisaged three main characteristics in evaluating usability.

Additional software (beside standard compilers). Two systems needed only a standard C compiler (MSPASS and ⋆SAT), DLP needed free software (New Jersey ML), and two systems (FACT and RACE) needed proprietary software (Allegro Common Lisp, available together with patches through `www.franz.com`). Of the two systems requiring no additional software, MSPASS was the only system we correctly installed on a Linux-based PC and Solaris-based SUN, without any interaction with their creators, while ⋆SAT needed a special *make*. For MSPASS and FACT we needed to contact the authors to obtain the translator from the ATP input syntax to the TPTP syntax (and also hack down the translator).

User interface. Every system provided only a bare command-line interface, with no help available but a "readme" file. This was not a problem, since TANCS deals with theorem provers, not complete systems. However, while a graphical interface would have been clearly a surprise, we believe that a simple textual menu interface might have been helpful to promote usage of these systems.

Proof management. By this, we mean the ability of a system to provide the (even bare) steps followed to reach the conclusion. No submitted system provided us by default the proof of the results it obtained. MSPASS can be set to provide some proof — either a refutation or a saturated set of clauses.

We also considered the level of expertise required to run the system. Of course, we did not expect naive users to be able to run the systems; however, it turned out that because of the characteristics of user interfaces, and additional

[2] The ratio may depend on the number of variables and may not just be a constant as in 3-SAT. See also [2].

Table 1. Benchmark Results on Unbounded Modal QBF

Absolute Running Time in 10msec (Geometric Mean)

	DLP		FaCT+		MSPASS		⋆SAT		RACE	
Benchmark	Time	%Tout	Time	%Tout	Time	%Tout	Time	%Tout	Time	%Tout
K4-C45-V8-D4	80	-	237	-	2453	-	8661	2%	6136	80%
K4-C45-V8-D5	781	-	5236	-	5187	-	38222	41%	9257	81%
K4-C55-V8-D6	554	-	2842	-	10612	-	56135	64%	9509	89%

Normalised Running Time (Geometric Mean)

	MSPASS			DLP			FaCT+			QBF solver		
Benchmark	Tot.	sat	unsat	Tot.	sat	unsat	Tot.	sat	unsat	Tot.	sat	unsat
K4-C45-V8-D4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
K4-C45-V8-D5	2.1	2.1	2.1	9.8	15.1	6.4	22.1	32.1	15.2	1.3	2.3	0.7
K4-C55-V8-D6	4.3	4.3	4.3	7.0	9.1	6.1	12.0	19.9	9.3	1.2	1.1	1.3

software required, no system could have been run by an expert of the problem itself — namely, either an expert in description logics and KRSS, or an expert in modal logics and the TPTP syntax. More expertise in the architecture of systems, operating systems compatibilities and even shell languages was needed.

4 Effectiveness and Performance Analysis

The *normalized running time* is the yardstick used to compare provers. In a nutshell, for every prover we compute the geometric mean time on some reference problems and then normalise the run time of each comparison problem with respect to (i.e. divide by) this reference mean time. Then we obtain a relative ranking which makes it possible to abstract away, at least to a certain extent, machine- and run-dependent characteristics. Notice that the geometric mean time must be used, otherwise we may draw meaningless conclusions [5]. Ability of handling large instances and asymptotic behavior emerge more clearly [8,9].

A compact report of the comparison on some Unbounded Modal QBF problems (using the encoding 3 and composed by approximately 50% of SAT and UNSAT instances) is presented in Table 1 (more details are in the web pages).

The first table reports the absolute running time as submitted by the entrants and the corresponding timeouts. Out of this table, we might be tempted to conclude that DLP is the fastest. But, as we said, this might depend on the compiler, the software etc. The second table shows the normalized time and a new picture: MSPASS is TANCS-00's leader on Unbounded Modal QBF, closely followed by DLP and FaCT+ (a version of FaCT with caching enabled). The last entry is a QBF-solver [14] which solved all problems within a second: modal and description logics ATP systems have still far to go.

DLP and FaCT time increases and then decreases as expected: they are tableau based and satisfiable instances are harder at even levels of $d + 1$ [2].

Extensibility. We want also to mention some other features of the systems, as dealing with more expressive logics may be worth their slow down. All systems

can reason in the basic multi-modal logic K, aka the description logic $\mathcal{ALC}$. Except ⋆SAT, all systems can deal with transitive roles (full transitive closure of roles for DLP), i.e., multi-modal logic S4, and global axioms (aka general TBoxes) and submitted results in the corresponding category. FaCT and MSPASS can deal with inverse modalities and both submitted results using inverse (MSPASS also reported about union and sequential composition).

Concerning problems not among TANCS's benchmarks (yet), FaCT and RACE can reason with qualified number restrictions, aka graded modalities. RACE can also handle ABoxes, which correspond to a restricted use of nominals in modal logics. Finally, we note that MSPASS could, in principle, deal with any feature that can be expressed within first-order logic; its running behavior, though, cannot be predicted for such extensions.

References

1. P. Balsinger and A. Heuerding. Comparison of theorem provers for modal logics. In *Proc. of TABLEAUX-98*, *LNAI* 1397, p. 25–27, Springer Verlag 1998.
2. M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. of AAAI-98*, 1998.
3. F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In *Foundation of Knowledge Representation*, p. 191–236. CSLI-Publications, 1996.
4. M. Fitting. Basic modal logic. In *Handbook of Logic in AI and Logic Programming*, vol. 1, p. 365–448. Oxford Univ. Press, 1993.
5. P. Fleming and J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *CACM*, 29(3):218–221, 1986.
6. E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Proc. of KR-98*, p. 626–635. 1998.
7. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *AIJ*, 54:319–379, 1992.
8. D. Johnson. A theoretician's guide to the experimental analysis of algorithms. Invited talk at AAAI-96. See `http://www.research.att.com/~dsj`, Aug. 1996.
9. D. Johnson and M. Trick, editors. *Cliques, Coloring, Satisfiability: the second DIMACS implementation challenge*, Am. Math. Soc., 1996.
10. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM JoC*, 6(3):467–480, 1977.
11. F. Massacci. Design and Results of Tableaux-99 Non-Classical (Modal) System Competition. In *Proc. of TABLEAUX-99*, *LNAI*, Springer Verlag 1999.
12. D. McGuinness, and P. Patel-Schneider. Usability Issues in Description Logic Systems, *Proc. of DL-97*, pp 84-88, 1997.
13. J. Orlin. The complexity of dynamic languages and dynamic optimization problems. In *Proc. of STOC-81*, p. 218–227, 1981.
14. J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. in *Proc. of IJCAI-99*, p. 1192-1197, 1999.
15. M. Schmidt-Schauß and G. Smolka. Attributive Concept Descriptions with Complements. *AIJ*, 48(1):1–26, 1991.
16. B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *AIJ*, 81(1-2):17–29, 1996.
17. C. Suttner and G. Sutcliffe. The CADE-14 ATP system competition. *JAR*, 21(1):99–134, 1998.

Consistency Testing: The RACE Experience

Volker Haarslev and Ralf Möller

University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

Abstract. This paper presents the results of applying RACE, a description logic system for $\mathcal{ALCNH}_{R^+}$, to modal logic SAT problems. Some aspects of the RACE architecture are discussed in detail: (i) techniques involving caching and (ii) techniques for dealing with individuals.

1 Introduction to the RACE Architecture

The description logic (DL) $\mathcal{ALCNH}_{R^+}$ [5] extends the logic $\mathcal{ALCH}f_{R^+}$ (see [8] for a concept consistency calculus) by adding number restrictions. The inference services supported by RACE for TBoxes and ABoxes are described in [6]. In this paper, due to space restrictions, we assume that the reader is familiar with DLs.

The ABox consistency algorithm implemented in the RACE system is described as a tableaux calculus in [5]. However, optimized search techniques are required in order to guarantee good average-case performance. The RACE architecture incorporates the following standard optimization techniques: dependency-directed backtracking [12] and DPLL-style semantic branching (see [3] for recent results and for an overview of the literature). Among a set of new optimization techniques, the integration of these techniques into DL reasoners for concept consistency has been described in [7] (see also [2] for a discussion of ideas to integrate intelligent backtracking into terminological logics). The implementation of these techniques in the ABox reasoner RACE differs from the implementation of other DL systems (e.g. FaCT or DLP [10]), which provide only concept consistency (and TBox) reasoning. The latter systems have to consider only so-called "labels" (sets of concepts) whereas an ABox prover such as RACE has to explicitly deal with individuals (nominals).

The techniques for TBox reasoning described in [1] (marking and propagation as well as lazy unfolding) are also supported by RACE. As indicated in [4], the architecture of RACE is inspired by recent results on optimization techniques for TBox reasoning [9], namely transformations of axioms (generalized concept inclusions, GCIs), model caching and model merging.

RACE is implemented in Common Lisp and can be copied for research purposes: http://kogs-www.informatik.uni-hamburg.de/~moeller/race/race.html.

2 Results on the TANCS Comparison Problems

The TANCS QBF benchmarks are transformed into concept consistency tests. The runtimes of RACE on the comparison problems with different encodings are documented in Table 1. All tests have been run on a Macintosh Powerbook G3

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 57–61, 2000.

Table 1. Runtimes for the TANCS'2000 comparison problems (runtimes include reading times): S = definitely satisfiable, T = timed out. The percentage of definitely unsatisfiable problems is U = 100 – S – T. The number of instances provided for the TANCS comparison problems of different type is indicated in parentheses.

K	C	V	D	Time (10ms)	S (%)	T (%)
				QBF cnf (4)		
4	10	4	4	2481	100	0
4	10	4	6	10455	0	100
4	20	4	4	4246	75	0
4	20	4	6	10577	0	100
4	30	4	4	776	0	0
4	30	8	4	11443	0	100
				QBF cnf modS4 (64)		
4	20	2	2	2277	14	6
				QBF cnf SSS (8)		
4	10	4	4	23	100	0
4	10	4	6	31	100	0
4	10	8	4	46	100	0
4	10	8	6	82	100	0
4	10	16	4	99	100	0
4	10	16	6	120	100	0
4	20	4	4	48	75	0
4	20	4	6	68	100	0
4	20	8	4	104	100	0
4	20	8	6	296	100	0
4	20	16	4	297	100	0
4	20	16	6	735	100	0
4	30	4	4	74	25	0
4	30	4	6	116	88	0
4	30	8	4	246	100	0
4	30	8	6	733	100	0
4	30	16	4	1647	100	0
4	30	16	6	5206	25	75
4	40	4	4	61	12	0
4	40	4	6	112	25	0
4	40	8	4	414	25	0
4	40	8	6	3344	88	0
4	40	16	4	3661	75	25
4	40	16	6	8932	25	75
4	50	4	4	71	0	0
4	50	4	6	128	0	0
4	50	8	4	716	0	0
4	50	8	6	3306	62	12
4	50	16	4	9075	25	75
4	50	16	6	9896	12	88

K	C	V	D	Time (10ms)	S (%)	T (%)
				QBF cnf Ladn (8)		
4	10	4	4	425	100	0
4	10	4	6	4487	88	12
4	10	8	4	9644	25	75
4	10	8	6	10191	0	100
4	20	4	4	940	100	0
4	20	4	6	9360	12	88
4	20	8	4	10205	0	100
4	20	8	6	10306	0	100
4	30	4	4	857	25	0
4	30	4	6	9710	0	88
4	30	8	4	10276	0	100
4	30	8	6	10433	0	100
4	40	4	4	508	0	0
4	40	4	6	3680	0	38
4	40	8	4	8919	0	88
4	40	8	6	10557	0	100
4	50	4	4	616	12	0
4	50	4	6	1851	0	12
4	50	8	4	10421	0	100
4	50	8	6	10689	0	100
				PSAT cnf (8)		
4	20	4	1	11	100	0
4	20	4	2	29	100	0
4	20	8	1	10	100	0
4	20	8	2	78	100	0
4	30	4	1	14	100	0
4	30	4	2	132	100	0
4	30	8	1	16	100	0
4	30	8	2	2987	62	38
4	40	4	1	23	100	0
4	40	4	2	364	100	0
4	40	8	1	26	100	0
4	40	8	2	10651	12	88
4	50	4	1	30	88	0
4	50	4	2	2088	62	25
4	50	8	1	32	100	0
4	50	8	2	10017	0	100

with 333 MHz. A timeout is set to 100secs. In case of a timeout, the runtime (100secs) is included into the geometric mean. For some of the TANCS'2000 comparison problem types, the set of benchmarks contains harder problems, which are not reported in Table 1. For these problems all instances are timed out, hence the runtime for each problem is 100secs.

The PSAT problems of the TANCS comparison problems are defined with a set of axioms. Axioms are represented as generalized concept inclusions (GCIs). In the RACE system, well-known as well as novel transformations on GCIs are employed in order to reduce runtimes. The goal of the transformations (see [7]) is to reduce the number of GCIs by transforming GCIs in order to derive concept definitions, i.e. a pair of GCIs $\mathsf{A} \sqsubseteq \mathsf{C}$ and $\mathsf{C} \sqsubseteq \mathsf{A}$ or primitive concept definitions $\mathsf{A} \sqsubseteq \mathsf{C}$ (with no $\mathsf{C} \sqsubseteq \mathsf{A}$) such that A is not mentioned on the left-hand side of another GCI in the TBox. Furthermore, as a novel transformation technique, in RACE axioms for declaring the disjointness of atomic concepts and axioms for domain and range restrictions for roles are absorbed. These constructs are treated in a special way by the RACE architecture.

In the PSAT problems the axioms are of the form $\top \sqsubseteq \mathsf{C}$ where C is a complex concept expression (i.e. a complex modal logic formula). Unfortunately, in the

case of PSAT the number of axioms (GCIs) can only be reduced by detecting common subexpressions. The derivation of (primitive) concept definitions is not possible due to the structure of the formulae (see Table 1 for the runtime results).

3 An Important Optimization Technique: Caching

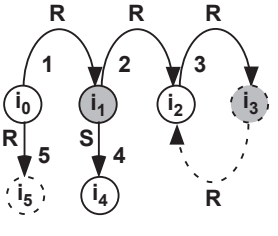
Caching of intermediate computation results is a necessary prerequisite to prove the (in)consistency of many concepts terms [9]. In particular, solutions for some of the TANCS problems mentioned above cannot be computed within the time limit without caching. For instance, the runtimes for the PSAT problems with depth 2 (see Table 1) increase by one order of magnitude.

Caching is not a trivial optimization technique. Solving a consistency problem $\{\mathsf{i}_0 : \mathsf{E}\}$ w.r.t. the following (cyclic) inclusion axioms demonstrates that caching must depend on the context:

$$\mathsf{C} \sqsubseteq (\exists \mathsf{R}.\mathsf{D}) \sqcap (\exists \mathsf{S}.\mathsf{X}) \sqcap \forall \mathsf{S}.(\neg \mathsf{X} \sqcap \mathsf{A}),\ \mathsf{D} \sqsubseteq \exists \mathsf{R}.\mathsf{C},\ \mathsf{E} \sqsubseteq (\exists \mathsf{R}.\mathsf{C}) \sqcup (\exists \mathsf{R}.\mathsf{D})$$

The proof steps are presented in Figure 1. In the figure the sequence of "expansion" steps is indicated with numbers. In step 1, the initial problem $\{\mathsf{i}_0 : \mathsf{E}\}$ is presented. Since there is an axiom for E involving a disjunction we get two constraint systems (see step 2). Let us assume the first alternative is tried first. This leads to a subconstraint system (step 3). The assertion from step 3 is expanded w.r.t. the axioms and we get the constraint system in step 4. The first some-constraint is expanded first. In step 5 the corresponding subconstraint system is considered. The right-hand side of the axiom for D is inserted (step 6). The some-constraint yields another subconstraint system (step 7). Due to the blocking strategy [4], the constraint system in step 7 is not expanded (see the constraint system in step 3 with the "blocking witness" i_1). In Figure 1 the canonical interpretation is indicated with a dashed arrow.

An often-employed strategy is to cache intermediate results, i.e. the satisfiability of D is stored as a so-called pseudo model $\{\mathsf{D}, \exists \mathsf{R}.\mathsf{C}\}$. Let us assume at the end of step 7 a pseudo model for D is stored as indicated above. In our example, there are some proof steps pending. In step 8 the remaining constraints from step 4 are considered. Obviously, the second some-constraint for the role S together with the value restriction for S causes a clash. Therefore, the second



1. $\{\mathsf{i}_0 : \mathsf{E}\}$
2. $\{\mathsf{i}_0 : \mathsf{E}, \mathsf{i}_0 : \exists \mathsf{R}.\mathsf{C}\} \vee \{\mathsf{i}_0 : \mathsf{E}, \mathsf{i}_0 : \exists \mathsf{R}.\mathsf{D}\}$
3. $\{\mathsf{i}_1 : \mathsf{C}\}$
4. $\{\mathsf{i}_1 : \mathsf{C}, \mathsf{i}_1 : \exists \mathsf{R}.\mathsf{D}, \mathsf{i}_1 : \exists \mathsf{S}.\mathsf{X}, \mathsf{i}_1 : \forall \mathsf{S}.\neg \mathsf{X} \sqcap \mathsf{A}\}$
5. $\{\mathsf{i}_2 : \mathsf{D}\}$
6. $\{\mathsf{i}_2 : \mathsf{D}, \mathsf{i}_2 : \exists \mathsf{R}.\mathsf{C}\}$
7. $\{\mathsf{i}_3 : \mathsf{C}\}$
8. $\{\mathsf{i}_4 : \mathsf{X}, \mathsf{i}_4 : \mathsf{A}, \mathsf{i}_4 : \neg \mathsf{X}\}$
9. $\{\mathsf{i}_5 : \mathsf{D}\}$

Fig. 1. Caching example with blocking (see text).

alternative in step 2 has to be considered. The corresponding subconstraint system is presented in step 9. If the consistency of D is checked by examining a cache entry for D, the overall result will be "E is consistent". Obviously, this is erroneous. The reason is that the caching principle described above does not consider the dependency on the satisfiability of C. Therefore, a dependency tracking mechanism for cache entries is implemented in RACE. Once the system detects the inconsistency of a concept (or constraint system) on which a cached pseudo model is dependent, the corresponding cache entries are (recursively) removed.

Caching of a pseudo model for a specific concept (model caching) is a strategy which is implemented in the FaCT system and the DLP system [9]. To the best of our knowledge, only the DLP system offers an additional caching technique. If, during a certain tableaux proof for the consistency of a concept term, a some-constraint (e.g. $\exists\,\mathsf{R}\,.\,\mathsf{C}$ interacts with all-constraints $\forall\,\mathsf{R}\,.\,\mathsf{C}$ and $\forall\,\mathsf{R}\,.\,\mathsf{D}$), then another strategy is to cache the result of checking the consisteny of the subproblem $\{\mathsf{i}_{\mathsf{new}}:\mathsf{C}, \mathsf{i}_{\mathsf{new}}:\mathsf{D}, \mathsf{i}_{\mathsf{new}}:\mathsf{E}\}$. Again, the dependency tracking mechanism implemented in RACE ensures correct behavior of the so-called "subtableaux caching" strategy in the case of blocking. To the best of our knowledge, details about the techniques employed in DLP have not been published yet.

If there exists no cache entry for a subproblem such as $\{\mathsf{i}_{\mathsf{new}}:\mathsf{C}, \mathsf{i}_{\mathsf{new}}:\mathsf{D}, \mathsf{i}_{\mathsf{new}}:\mathsf{E}\}$, then it might be possible check out whether the models of C, D and E can be merged (see [7] for an introduction to model merging). However, for the TANCS benchmarks this technique was disabled because the overhead involved in this test caused the runtimes to increase.

4 Optimizations for ABox Reasoning

Unfortunately, the TANCS benchmarks only consider concept consistency problems (possibly w.r.t. axioms). The RACE architecture has been developed also for ABox reasoning with individuals. Thus, there is some overhead involved if only concept consistency tests are computed with RACE. As the runtime results indicate, the overhead costs of pure consistency reasoning with an ABox reasoner are minimal. However, ABox reasoning itself requires additional optimization techniques in order to ensure adequate average-case performance. For instance, for computing the direct types (most-specific atomic concepts) of an individual a during ABox realization, multiple consistency problems for ABoxes $\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{A}_\mathsf{i}\}$ with different atomic concepts A_i have to be solved. In order to optimize ABox realization (see [4] for other ABox optimization techniques) we use a transformation technique for tree-like ABoxes, called *contraction* w.r.t. an individual a. The idea is to maximize the effect of caching. To speed up the tests we transform $\mathcal{A}$ in such a way that acyclic, tree-like "role paths" between individuals are represented by an appropriate existential restriction. The corresponding concept and role assertions "representing" the role paths are deleted from the ABox. The following contraction rule is applied to $\mathcal{A}$ as often as possible. The final ABox is called $\mathcal{A}'$. Then, the consistency of $\mathcal{A}' \cup \{\mathsf{a}:\neg\mathsf{A}_\mathsf{i}\}$ is checked. Obviously, $\mathcal{A}$ (without $\mathsf{a}:\neg\mathsf{A}_\mathsf{i}$ being added) must be tested for consistency without contraction beforehand.

RC Contraction Rule for $\mathcal{ALCNH}_{R^+}$.
Premise: $\exists(\mathsf{i},\mathsf{j}):\mathsf{R} \in \mathcal{A}, \mathsf{j}:\mathsf{C}_1 \in \mathcal{A}, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n} \in \mathcal{A}$:
$[\mathsf{j} \neq \mathsf{a}, (\neg\exists\,(\mathsf{j},\mathsf{k}):\mathsf{R}' \in \mathcal{A}),$
$(\neg\exists\,(\mathsf{i},\mathsf{j}):\mathsf{R}'' \in \mathcal{A} : \mathsf{R}'' \neq \mathsf{R}), (\neg\exists\,(\mathsf{l},\mathsf{j}):\mathsf{R}''' \in \mathcal{A} : \mathsf{l} \neq \mathsf{i})$
$(\neg\exists\,(\mathsf{i},\mathsf{o}):\mathsf{S} \in \mathcal{A} : \mathsf{o} \neq \mathsf{j}, \mathsf{R}^\uparrow \cap \mathsf{S}^\uparrow \neq \emptyset),$
$(\neg\exists\,\mathsf{j}:\mathsf{C}_{\mathsf{n}+1} \in \mathcal{A} : \forall\,\mathsf{i} \in 1..\mathsf{n} : \mathsf{C}_{\mathsf{n}+1} \neq \mathsf{C}_\mathsf{i})]$
Consequence: $\mathcal{A} := (\mathcal{A} \setminus \{(\mathsf{i},\mathsf{j}):\mathsf{R}, \mathsf{j}:\mathsf{C}_1, \ldots, \mathsf{j}:\mathsf{C}_\mathsf{n}\}) \cup \{\mathsf{i}:\exists\,\mathsf{R}\,.\,\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_\mathsf{n}\}$

Empirical tests with automatically generated ABox problems indicate that the role path contraction technique is very effective for tree-like ABoxes. The contraction technique can give a speed gain of about one order of magnitude (see [4] for a set of benchmarks and evaluation results).

References

1. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, 2(4):109–138, 1994.
2. D. Drollinger. Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken (in German). Document D-93-21, Deutsches Forschungszentrum für Künstliche Intelligence, Germany, 1993.
3. J.W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, Computer and Information Science, 1995.
4. V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In Lambrix et al. [11], pages 115–119.
5. V. Haarslev and R. Möller. Expressive abox reasoning with number restrictions, role hierachies, and transitively closed roles. In A.G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proc. of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, 2000.
6. V. Haarslev, R. Möller, and A.-Y. Turhan. RACE User's guide and reference manual version 1.1. Technical Report FBI-HH-M-289/99, University of Hamburg, Computer Science Department, October 1999.
7. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
8. I. Horrocks. Using an expressive description logic: FaCT or fiction? In A.G. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*, pages 636–647, June 1998.
9. I. Horrocks and P. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.
10. I. Horrocks and P.F. Patel-Schneider. FaCT and DLP: Automated reasoning with analytic tableaux and related methods. In *Proceedings International Conference Tableaux'98*, pages 27–30, 1998.
11. P. Lambrix et al., editor. *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, June 1999.
12. R.M. Stallman and G.J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135–196, 1977.

Benchmark Analysis with FaCT

Ian Horrocks

University of Manchester
Manchester, UK
horrocks@cs.man.ac.uk

Abstract. FaCT (**Fa**st **C**lassification of **T**erminologies) is a Description Logic (DL) classifier that can also be used for modal logic satisfiability testing. The FaCT system includes two reasoners, one for the logic $\mathcal{SHF}$ and the other for the logic $\mathcal{SHIQ}$, both of which use optimised implementations of sound and complete tableaux algorithms. FaCT's most interesting features are its expressive logic (in particular the $\mathcal{SHIQ}$ reasoner), its optimised tableaux implementation (which has now become the standard for DL systems), and its CORBA based client-server architecture.

1 The FaCT System

The logics implemented in FaCT are both based on $\mathcal{ALC}_{R^+}$, an extension of $\mathcal{ALC}$ to include transitive roles [13]. For compactness, this logic has be called $\mathcal{S}$ (due to its relationship with the proposition multi-modal logic $\mathbf{S4}_{(\mathbf{m})}$ [14]). $\mathcal{SHF}$ extends $\mathcal{S}$ with a hierarchy of roles and functional roles (attributes), while $\mathcal{SHIQ}$ adds inverse roles and fully qualified number restrictions.

The $\mathcal{SHIQ}$ reasoner is of particular interest, both form a theoretical and a practical viewpoint. Adding inverse roles to $\mathcal{SHF}$ (to give $\mathcal{SHIF}$) already leads to the loss of the finite model property, and this has necessitated the development of a more sophisticated *double dynamic* blocking strategy that allows the algorithm to find finite representations of infinite models while still guaranteeing termination [7]. Moreover, when $\mathcal{SHIF}$ is generalised to $\mathcal{SHIQ}$, it is necessary to restrict the use of transitive roles in number restrictions in order to maintain decidability [8]. $\mathcal{SHIQ}$ is also of great practical interest as it is powerful enough to encode the logic DLR, and can thus be used for reasoning about conceptual data models, e.g., Extended Entity-Relationship (EER) schemas [2,4].

2 Implementation

FaCT is implemented in Common Lisp, and has been run successfully with several commercial and free lisps, including Allegro, Liquid (formerly Lucid), Lispworks and GNU. Binaries (executable code) are now available (in addition to the source code) for Linux and Windows systems, allowing FaCT to used without a locally available Lisp.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 62–66, 2000.

In order to make the FaCT system usable in realistic applications, a wide range of optimisation techniques are used in the implementation of the satisfiability testing algorithms. These include axiom absorption, lexical normalisation, semantic branching search, simplification, dependency directed backtracking, heuristic guided search and caching [6]. The use of these (and other) optimisation techniques has now become standard in tableaux-based DL implementations [10,5].

The current implementation of $\mathcal{SHIQ}$ is a relatively naive modification of the $\mathcal{SHF}$ reasoner: it does not use the more efficient form of double blocking described in [8], it does not include any special optimisations to deal with inverse roles (or take advantage of their absence), and some optimisations that would require modification in the presence of inverse roles are instead simply disabled. As a result, performance with $\mathcal{SHIQ}$ is significantly worse than with $\mathcal{SHF}$, even w.r.t. $\mathcal{SHF}$ problems. These issues are being addressed in a new implementation of the $\mathcal{SHIQ}$ reasoner.

Work is also underway on the development of Abox reasoning for the FaCT system: an $\mathcal{SHF}$ Abox has recently been released [15] and a full $\mathcal{SHIQ}$ Abox is being developed [9].

3 Special Features

In addition to the standard KRSS functional interface [11], FaCT can also be configured as a classification and reasoning server using the Object Management Group's Common Object Request Broker Architecture (CORBA) [1]. This approach has several advantages: it facilitates the use of FaCT by non-Lisp client applications; the API is defined using CORBA's Interface Definition Language (IDL), which can be mapped to various target languages; a mechanism is provided for applications to communicate with the DL system, either locally or remotely; and server components can be added/substituted without client applications even being aware of the change. This has allowed, for example, the successful use of FaCT's reasoning services in a (Java based) prototype EER schema integration tool developed as part of the DWQ project [3].

4 Performance Analysis

FaCT's $\mathcal{SHIQ}$ reasoner (version 2.13.14) was used with those problems involving inverse roles; in all other cases the $\mathcal{SHF}$ reasoner (version 2.13.3) was used. The tests were run on two machines, one with a 450MHz Pentium III and 128Mb of RAM, the other with a 433MHz Celeron and 256Mb of RAM. In both cases Allegro CL Enterprise Edition 5.0 was used with Red Hat Linux. For the purposes of these tests the difference in performance between the two machines is small enough to be ignored.

As far as FaCT's performance is concerned, the current implementation is beginning to show its age: the system has been used as a testbed for new algorithms and optimisation techniques, and after nearly five years of "evolution"

a major overhaul is long overdue. This is particularly true of the $\mathcal{SHIQ}$ reasoner.[1] It is therefore unlikely that FaCT's performance will be competitive with that of younger and leaner systems whose designs reflect the experience gained with the FaCT system. Moreover, FaCT's optimisations are specifically aimed at improving the system's performance when classifying realistic knowledge bases (KBs), and perform less well with the kinds of randomly generated data used in these tests. In particular, the partial model caching technique used by FaCT relies for its effectiveness on the fact that, in realistic KBs, there are typically large numbers of different roles (modalities). In contrast, most of the TANCS test data uses only a single role. Moreover, the data used in the TANCS test is very susceptible to optimisation by caching the satisfiability status of sets of formulae [6]. This can be seen in Table 1, which shows the results of running the "final" set of TANCS QBF tests using both FaCT's standard $\mathcal{SHF}$ reasoner and a modified version (denoted FaCT†) that includes satisfiability status caching instead of partial model caching. The results for satisfiable and unsatisfiable tests are separated and in each case the number of instances solved (i), median time (m) and worst case time (w) is given. Times are in seconds, with a timeout of 1,000s. There were a total of 64 instances in each test, and all unsolved instances were the result of timeout rather than memory failure.

Table 1. Results of "final" TANCS tests for FaCT and FaCT†

	FaCT						FaCT†					
	SAT			UNSAT			SAT			UNSAT		
Test	i	m	w	i	m	w	i	m	w	i	m	w
C45-V8-D4	4	606.57	717.39	23	106.43	813.78	32	5.26	11.64	32	1.07	10.58
C45-V8-D5	0	–	–	2	547.02	774.59	32	139.97	721.68	32	20.52	259.09
C55-V8-D6	0	–	–	3	595.26	631.98	28	89.04	328.41	36	13.85	215.70

It is interesting to compare these results with the the times taken to classify a large realistic KB (the GALEN medical terminology KB [12]). In this case satisfiability status caching is actually less effective than partial model caching: FaCT takes ≈41s to classify the KB, whereas FaCT† takes ≈50s.

The standard $\mathcal{SHF}$ reasoner was also tested on the Periodic Satisfiability (Global PSpace) reference problems. With the standard (CNF) encoding, these were all very easy (most problems were solved in times too short to by accurately measured), but with the K encoding the PSat problems became *much* harder, and few were solved within the timeout.[2] This effect is much less pronounced with the QBF problems, probably because absorption is only relevant with global axioms. A more detailed analysis was performed using harder PSat problems

[1] As mentioned above, these (and other) issues are being addressed in a new implementation.

[2] This causes of this effect are being investigated—it is probably related to the absorption optimisation.

with the standard encoding (12 variables, depth 1, 24–192 clauses, 30 instances per data point, 600s timeout), and the results are shown in Figure 1 (left).

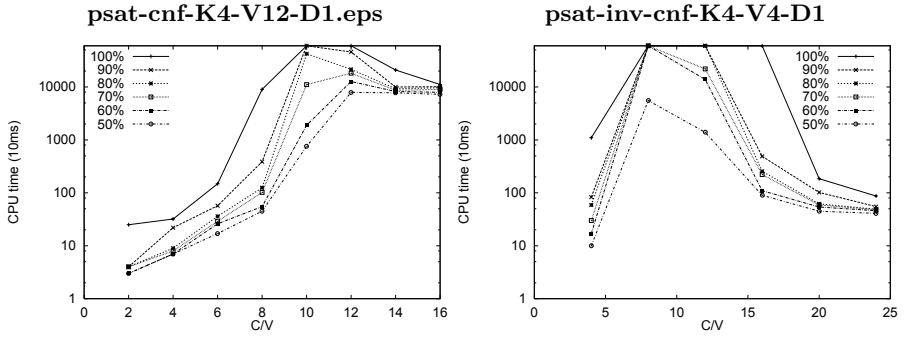


Fig. 1. Percentile satisfiability times for two problem sets

The $\mathcal{SHIQ}$ reasoner was tested using the Modal QBF (Modal PSpace) and Periodic Satisfiability (Global PSpace) problems with inverse. The performance of the reasoner in these tests was quite poor, with most tests ending in either timeout or memory failure. The results of those tests where at least one instance was solved are given in Table 2, using the same format as Table 1, but with the addition of the number of tests resulting in a timeout (T) or memory failure (M). The reason for the poor performance is probably the lack of satisfiability status caching which, as demonstrated above, is a crucially important optimisation with this kind of test data. A more detailed analysis was performed using easier PSat problems with the standard encoding (4 variables, depth 1, 16–96 clauses, 30 instances per data point, 600s timeout), and the results are shown in Figure 1 (right).

5 Availability

FaCT and iFaCT are available (under the GNU general public license) via the WWW at `http://www.cs.man.ac.uk/~horrocks`.

References

1. S. Bechhofer, I. Horrocks, P. F. Patel-Schneider, and S. Tessaris. A proposal for a description logic interface. In *Proc. of DL'99*, pages 33–36, 1999.
2. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proc. of DEXA-98*, pages 192–197, 1998.
3. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Use of the data reconciliation tool at telecom italia. DWQ deliverable D.4.3, Foundations of Data Warehouse Quality (DWQ), 1999.

Table 2. Results of tests using FaCT's $\mathcal{SHIQ}$ reasoner

	SAT			UNSAT				
Test	i	m	w	i	m	w	T	M
p-qbf-inv-cnfSSS-K4-C10-V4-D4	2	11.19	32.70	0	–	–	0	6
p-qbf-inv-cnfSSS-K4-C20-V4-D4	1	36.43	36.43	2	70.37	155.62	0	5
p-qbf-inv-cnfSSS-K4-C30-V4-D4	1	112.48	112.48	1	275.13	275.13	0	6
p-qbf-inv-cnfSSS-K4-C40-V4-D4	0	–	–	5	176.97	376.15	0	3
p-qbf-inv-cnfSSS-K4-C50-V4-D4	0	–	–	8	11.23	61.44	0	0
p-qbf-inv-cnfSSS-K4-C20-V4-D6	0	–	–	1	1.58	1.58	0	7
p-qbf-inv-cnfSSS-K4-C40-V4-D6	0	–	–	1	50.72	50.72	0	7
p-qbf-inv-cnfSSS-K4-C50-V4-D6	0	–	–	6	123.88	341.35	0	2
p-psat-inv-cnf-K4-C20-V4-D1	8	1.59	45.61	0	–	–	0	0
p-psat-inv-cnf-K4-C30-V4-D1	3	11.58	58.59	0	–	–	0	5
p-psat-inv-cnf-K4-C40-V4-D1	4	64.32	106.05	1	2.56	2.56	3	0
p-psat-inv-cnf-K4-C50-V4-D1	5	1.81	17.98	1	1.79	1.79	2	0
p-psat-inv-cnf-K4-C20-V8-D1	7	0.05	39.06	0	–	–	0	1
p-psat-inv-cnf-K4-C30-V8-D1	5	14.03	108.67	0	–	–	0	3
p-psat-inv-cnf-K4-C20-V8-D2	2	0.04	0.14	0	–	–	2	4

4. E. Franconi and G. Ng. The i•com tool for intelligent conceptual modelling. Submitted to VLDB-2000.
5. V. Haarslev, R. Möller, and A.-Y. Turhan. Implementing an $\mathcal{ALCRP}(\mathcal{D})$ abox reasoner – progress report. In *Proc. of DL'98*, pages 82–86, 1998.
6. I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
7. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
8. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180. Springer-Verlag, 1999.
9. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of CADE-2000*, 2000. To appear.
10. P. F. Patel-Schneider. DLP system description. In *Proc. of DL'98*, pages 87–89, 1998.
11. P. F. Patel-Schneider and B. Swartout. Description logic specification from the KRSS effort, June 1993.
12. A. L. Rector, W A Nowlan, and A Glowinski. Goals for concept representation in the GALEN project. In *Proc. of SCAMC'93*, pages 414–418, 1993.
13. U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*, pages 333–345, 1996.
14. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, 1991.
15. S. Tessaris and G. Gough. Abox reasoning with transitive roles and axioms. In *Proc. of DL'99*, 1999.

MSPASS: Modal Reasoning by Translation and First-Order Resolution

Ullrich Hustadt[1] and Renate A. Schmidt[2]

[1] Centre for Agent Research and Development, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom, U.Hustadt@doc.mmu.ac.uk
[2] Department of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom, schmidt@cs.man.ac.uk

Abstract. MSPASS is an extension of the first-order theorem prover SPASS, which can be used as a modal logic theorem prover, a theorem prover for description logics and a theorem prover for the relational calculus.

1 MSPASS

MSPASS [17] is an enhancement of the first-order theorem prover SPASS with a translator of modal formulae, formulae of description logics, and formulae of the relational calculus. SPASS [19,20] is one of the fastest and most sophisticated theorem provers for first-order logic with equality, and it's performance compares well with special purpose theorem provers for modal logics, description logics and first-order logic [7,11,18].

The input language of SPASS was extended to accept as input also modal, relational and description logic formulae. Modal formulae and description logic formulae are built from a vocabulary of propositional symbols of two disjoint types, namely, propositional (Boolean or concept) and relational (role). The repertoire of logical constructs includes:

- the standard Boolean operators on both propositional and relational formulae: true, false, not, and, or, implies (subsumed by), implied (subsumes), equiv,
- multi-modal modal operators, permitting complex relational parameters: dia and box (synonyms are some and all, the existential and universal role restriction operations of description logics), as well as domain and range,
- the relational operators: comp (composition), sum (relative sum), conv (converse), id (identity), div (diversity), and
- test (test), domrestr (domain restriction) and ranrestr (range restriction).

MSPASS supports modal axiom formulae which are true in every possible worlds (generalised terminological axioms for both concepts and roles). In addition, it is possible to specify additional frame properties, or any other first-order restrictions on the translated formulae.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 67–71, 2000.

2 Architecture and Algorithm

Reasoning in MSPASS is performed in three stages: (i) translation of a given set of modal and relational formulae into a set of first-order formulae, (ii) transformation into clausal form, and (iii) saturation-based resolution.

In the current implementation of MSPASS the available translation methods include:

- the standard relational translation method,
- the functional translation method [1,5,14],
- the optimised functional translation method [15] including a variation defined in terms of n-ary predicates [7], as well as
- the semi-functional translation method [12].

The implementation of the relational translation method is most general and applies to the language described in Section 1. Some restrictions apply to the other methods. The functional translation, optimised functional and semi-functional translation methods are available only for multi-modal $\mathrm{K}_{(m)}$, and extensions with serial modalities.

Part of MSPASS is a fast converter of first-order formulae into clausal form with special features such as optimised and strong Skolemisation, and an improved implementation of renaming [13].

The inference engine of MSPASS is an implementation of a saturation-based resolution and superposition calculus with simplification [2]. In particular,

- it uses ordered resolution, and ordered superposition with selection,
- it supports splitting and branch condensing (splitting amounts to case analysis while branch condensing resembles branch pruning or backjumping),
- it has an extensive set of reduction and simplification rules, and
- it supports dynamic sort theories by additional inference and reduction rules.

Ordered inference, splitting, and condensing are of particular importance concerning the performance for satisfiable formulae, and for randomly generated formulae unit propagation and branch condensing are important as well.

Using certain flag settings MSPASS is known to provide decision procedures for a range of logics. More specifically:

- For the relational translation of $\mathrm{K}_{(m)}(\cup, \cap, ^{-}, ^{\smile})$ [3] and the description logic $\mathcal{ALB}$ [9]. $\mathrm{K}_{(m)}(\cup, \cap, ^{-}, ^{\smile})$ is the multi-modal logic defined over relations closed under union, intersection, complementation and converse. It includes logics such as the basic tense logic K_t, Humberstone's logic of inaccessibility, Boolean modal logic, as well as fragments of Tarski's relational calculus.
- For the relational translation of $\mathrm{K}_{(m)}(\cup, \cap, ^{-}, ^{\smile})$ and $\mathcal{ALB}$ where the relations satisfy additional first-order restrictions including reflexivity, irreflexivity, seriality, symmetry, density, relational inclusion, etc [3,9].
- For the optimised functional and semi-functional translation of multi-modal $\mathrm{K}_{(m)}$ in which modalities may be serial [6,16].
- For the guarded fragment and other solvable fragments [4,8].

3 Implementation

MSPASS is implemented in ANSI C and differs from SPASS only in the extended input language and the translation routines.

The inference loop in MSPASS is controlled by two sets of clauses: The set of worked-off clauses and the set of usable clauses. At the beginning all input clauses are usable. The theorem prover starts by choosing, according to some heuristic, a clause from the usable clause set and moving it to the worked-off clause set. Then all inference rules are applied to the chosen clause and clauses from the worked-off clause set. The derived clauses are subjected to the reduction rules, for example, subsumption deletion and clause reduction, and non-redundant clauses are added to the usable clause set. If the empty clause is derived or the set of usable clauses is empty, the theorem prover stops, having derived a contradiction or a satisfiable clause set. Otherwise, the next clause is chosen from the usable clause set, moved to the worked-off clause set, and the loop is repeated. The heuristic used by MSPASS chooses a clauses with the minimal number of symbols.

Among the inference rules, splitting has priority. If splitting is applicable to a clause, a heuristic is used to determine which subclauses are generated and in which order they are considered. The implementation of splitting used in this version of MSPASS does not split Horn clauses, and the heuristic used will simply branch on the first (in the order literals are stored internally) positive literal of the chosen clause. These heuristics are very different to those employed by other provers for modal and description logics, and most probably the heuristics of MSPASS are not adequate for the classes of formulae under consideration here.

As most first-order theorem provers, MSPASS uses an indexing data structure to store clauses with the intention of speeding up access to clauses with particular characteristics. To keep the indexing data structure small, structure sharing for terms and atoms is used. That means every occurrence of any subterm of an atom in the clause set exists exactly once and is shared by all superterms containing this subterm. Further implementation details can be found in [20].

4 Special Features

As mentioned in previous sections MSPASS is a reasoner for a large class of modal logics and description logics. Although it does not provide a decision procedure for all the modal logics one may be interested in, for example, PDL or graded modal logic are exceptions, an attractive feature is the possibility to specify arbitrary first-order restrictions. This allows for its use as a flexible tool for the investigation of combinations of interacting non-classical logics or description logics, which have not been been studied in depth before, and in particular, which have not been anticipated by the implementors.

In this context it is useful that, on termination, MSPASS does not only produce a 'yes'/'no' answer, but it also outputs a proof or a saturated set of clauses, if the input problem is unsatisfiable or satisfiable. A finite saturated set of clauses provides a characterisation of a class of models for the input problem.

5 Performance

The tables below present the performance results for MSPASS on the TANCS-2000 comparison problems. The entries in each column have the form t, s, f, where t is the geometric mean of the runtime of s (successfully solved) problems, in 10ms, and f is the number of problems not solved within the time-limit of 7200 CPU seconds. The tests were performed under Sun Solaris 2.6 on a cluster of PCs equipped with 300 MHz Pentium II processors and 256 MB main memory plus 700 MB swap space.

The modal QBF problems in the modal PSPACE division were tested with the optimised functional translation in terms of n-ary predicates (flag settings `-EMLTranslation=2 -EMLFuncNary=1`^{f}), while the converse PDL problems in the global EXPTIME division (the subset without star) and the problems for periodic satisfiability in the global PSPACE division were tested with the relational translation (flag setting `-EMLTranslation=0`^{r}). In all tests ordered resolution with selection of negative literals was used, flag setting `-Ordering=0` with `-Select=1`[1] for the periodic satisfiability problems and `-Select=2`[2] for all other problems. The latter results in an ordered hyperresolution like behaviour.

The combination of the relational translation with selection-based resolution or hyperresolution is in general not a decision procedure on the converse PDL and periodic satisfiability problems. In contrast, ordered resolution without selection is a decision procedure for these problems [9]. However, the outcome of the tests performed with ordered resolution was poor. This difference in performance can also be observed for other classes of randomly generated modal formulae [10].

Modal QBF^{f,2}	C10	C20	C30	C40	C50
cnfSSS V4 D4	91 8 -	143 8 -	208 8 -	259 8 -	312 8 -
cnfSSS V4 D6	279 8 -	433 8 -	586 8 -	711 8 -	852 8 -
cnfSSS V8 D4	927 8 -	1516 8 -	1896 8 -	2277 8 -	2627 8 -
cnfSSS V8 D6	3403 8 -	5513 8 -	7017 8 -	8694 8 -	9786 8 -
cnfSSS V16 D4	14820 8 -	22852 8 -	31006 8 -	34324 8 -	41265 8 -
cnfSSS V16 D6	57350 8 -	85530 8 -	107630 8 -	120808 8 -	144932 8 -
cnfLadn V4 D4	3539 8 -	5042 8 -	6034 8 -	6809 8 -	7779 8 -
cnfLadn V4 D6	20708 8 -	37015 8 -	43184 8 -	45131 8 -	47822 8 -
cnfLadn V8 D4	140059 8 -	272815 8 -	354670 8 -	398255 8 -	436614 1 -
cnfLadn V8 D6	612866 3 5	8	8	8	8
cnf V4 D4	5145 4 -	12368 4 -	13575 4 -		
cnf V4 D6	58845 4 -	121394 3 1			
cnf V8 D4	168582 4 -	414258 3 1	4		

Converse PDL (no $*$)^{r,2}	C10	C20	C30	C40	C50
cnfSSS V4 D4	657 8 -	2161 8 -	5292 8 -	10512 8 -	14007 8 -
cnfSSS V4 D6	1325 8 -	4802 8 -	12972 8 -	22660 8 -	30536 8 -
cnfSSS V8 D4	2988 8 -	11802 8 -	33512 8 -	55789 8 -	66524 8 -
cnfSSS V8 D6	6890 8 -	27203 8 -	80986 8 -	105825 8 -	149799 8 -
cnfSSS V16 D4	19096 8 -	61311 8 -	160694 8 -	259507 8 -	334452 8 -
cnfSSS V16 D6	42794 8 -	138737 8 -	344882 8 -	551530 8 -	669153 7 1

$PSAT^{r,1}$	C20	C30	C40	C50
cnf V4 D1	9 8 -	15 8 -	24 8 -	31 8 -
cnf V4 D2	131 7 1	3082 6 2	6559 2 6	6582 7 1
cnf V8 D1	8 8 -	18 8 -	66 8 -	184 8 -
cnf V8 D2	46 5 2	163144 2 6	8	8
inv V4 D1	114 8 -	373 8 -	275 8 -	171 8 -
inv V4 D2	8	8	8	8
inv V8 D1	51 8 -	5713 8 -	199490 6 2	8
inv V8 D2	8	8	8	8

Reference problems	
$QBF\text{-}cnf^{f,2}$ C20 V2 D2	44 64 -
$PSAT\text{-}cnf^{r,1}$ C32 V4 D1	16 64 -
$PSAT\text{-}inv^{r,1}$ C32 V4 D1	386 64 -

References

1. Y. Auffray and P. Enjalbert. Modal theorem proving: An equational viewpoint. *J. Logic Computat.*, 2(3):247–297, 1992.
2. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Logic Computat.*, 4(3):217–247, 1994.
3. H. De Nivelle, R. A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic J. IGPL*, 2000. To appear.
4. H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. LICS'99*, pp. 295–303. IEEE Computer Soc., 1999.
5. A. Herzig. *Raisonnement automatique en logique modale et algorithmes d'unification.* PhD thesis, Univ. Paul-Sabatier, Toulouse, 1989.
6. U. Hustadt. *Resolution-Based Decision Procedures for Subclasses of First-Order Logic.* PhD thesis, Univ. d. Saarlandes, Saarbrücken, Germany, 1999.
7. U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *J. Appl. Non-Classical Logics*, 9(4):479–522, 1999.
8. U. Hustadt and R. A. Schmidt. Maslov's class K revisited. In *Proc. CADE-16*, vol. 1632 of *LNAI*, pp. 172–186. Springer, 1999.
9. U. Hustadt and R. A. Schmidt. Issues of decidability for description logics in the framework of resolution. In *Automated Deduction in Classical and Non-Classical Logics*, vol. 1761 of *LNAI*, pp. 192–206. Springer, 2000.
10. U. Hustadt and R. A. Schmidt. Using resolution for testing modal satisfiability and building models. To appear in *J. Automated Reasoning*, 2000.
11. U. Hustadt, R. A. Schmidt, and C. Weidenbach. Optimised functional translation and resolution. In *Proc. TABLEAUX'98*, *LNAI* 1397, pp. 36–37. Springer, 1998.
12. A. Nonnengart. First-order modal logic theorem proving and functional simulation. In *Proc. IJCAI'93*, pp. 80–85. Morgan Kaufmann, 1993.
13. A. Nonnengart, G. Rock, and C. Weidenbach. On generating small clause normal forms. In *Proc. CADE-15*, vol. 1421 of *LNAI*, pp. 397–411. Springer, 1998.
14. H. J. Ohlbach. Semantics based translation methods for modal logics. *J. Logic Computat.*, 1(5):691–746, 1991.
15. H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. *J. Logic Computat.*, 7(5):581–603, 1997.
16. R. A. Schmidt. Decidability by resolution for propositional modal logics. *J. Automated Reasoning*, 22(4):379–396, 1999.
17. R. A. Schmidt. MSPASS, 1999. `http://www.cs.man.ac.uk/~schmidt/mspass/`.
18. G. Sutcliffe and C. B. Suttner. The CADE-14 ATP system competition. *J. Automated Reasoning*, 21(1):99–134, 1998.
19. C. Weidenbach. SPASS, 1999. `http://spass.mpi-sb.mpg.de`.
20. C. Weidenbach. SPASS: Combining superposition, sorts and splitting. In *Handbook of Automated Reasoning.* Elsevier, 2000. To appear.

TANCS-2000 Results for DLP

Peter F. Patel-Schneider

Bell Labs Research, Murray Hill, NJ, U.S.A.
pfps@research.bell-labs.com

Abstract. DLP is an experimental description logic system that can also be used as a satisfiability checker for propositional modal logics. DLP incorporates a number of effective optimizations that, taken together, produce a very fast reasoner. DLP was run on several problem categories in the TANCS-2000 comparison with very gratifying results.

1 Introduction

DLP [7] is a description logic system that contains a sound and complete reasoner for expressive description logics. Due to the equivalences between expressive description logics and propositional modal logics, DLP can be used as a satisfiability checker for propositional modal logics, including $\mathbf{K_{(m)}}$, $\mathbf{KT_{(m)}}$, $\mathbf{K4_{(m)}}$, $\mathbf{S4_{(m)}}$, and **PDL**. DLP has performed very well in recent comparisons of modal provers [5,6]. DLP has undergone several major revisions. The most-recent version of DLP, version 4.1, includes some major improvements over previous versions, including a simple version of dynamic backtracking [2].

DLP is an experimental system, designed to investigate various optimization techniques for description logic systems, including many of the optimizations pioneered in FaCT [4]. DLP is available for research purposes from *http://www.bell-labs.com/user/pfps/dlp*. DLP contains a highly-optimized satisfiability checker for a superset of Propositional Dynamic Logic (**PDL**), and includes a simple interface for the direct checking of the satisfiability of formulae in **PDL** as well as the modal logics $\mathbf{K_{(m)}}$, $\mathbf{KT_{(m)}}$, $\mathbf{K4_{(m)}}$, and $\mathbf{S4_{(m)}}$. However, DLP does not currently handle converse modalities, so it is unable to handle many of the problem categories.

2 Architecture and Algorithm

At the heart of the DLP system is a highly-optimized tableaux satisfiability engine. DLP first performs a lexical normalization phase, uniquely storing subformulae; eliminates repeated conjuncts and disjuncts; replaces local tautologies and contradictions with true and false; and performs several other normalization steps. It then attempts to construct a model of the normalized formulae.

DLP deals with non-determinism in the model construction algorithm by performing a semantic branching search, as in the Davis-Putnam-Logemann-Loveland procedure (DPLL), instead of the syntactic branching search used by

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 72–76, 2000.

Problem	N	D	L	Sat	Unsat	TO	Time	(sat)
qbf-cnf-K4	2	2	20	10	54	0	0.041	0.016
qbf-modS4-K4	2	2	20	0	64	0	0.921	0.008
psat-cnf-K4	4	1	32	63	0	1	0.014	0.008

Fig. 1. Reference Problems Results

most earlier tableaux-based provers [3]. DLP deterministically expands disjunctions that present only one expansion possibility and detects a clash when a disjunction has no expansion possibilities.

DLP performs a form of dependency-directed backtracking called *backjumping*, backtracking to the most-recent choice point that participates in a clash instead of just one choice point. To support backjumping, DLP keeps associated with each formula the set of choice points that gave rise to that formula.

DLP caches the satisfiability status of all modal nodes that it encounters, and uses this status when a node with the same formula is seen again. DLP uses a combination of heuristics to determine the next disjunct on which to branch: it tries to maximize backjumping by first selecting disjunctions that do not depend on recent choice points, and it tries to maximize deterministic expansion by using the MOMS heuristic [1] to select a disjunct from amongst these disjunctions. DLP defers modal processing until all propositional processing is complete at a node, again using a backjumping maximization heuristic to determine the order in which modal successors are explored.

To handle transitive modality constructs in **PDL**, DLP checks for loops in the model it is constructing. If a loop is detected, it must be classified as either as a loop that leads to satisfiability or a loop that is unsatisfiable.

DLP has recently been upgraded to include a number of new optimizations. It now caches dependency information to improve backjumping performance on cached nodes. It also caches some expensive low-level computations and eliminates some computations on large formulae. These two improvements speed up performance on large but easy-to-solve formulae. DLP used to completely generate assignments for the current node before investigating any modal successors, but now has an option to investigate modal successors whenever a choice point is encountered and to store and reuse the results of this early investigation. DLP incorporates a simple variant of dynamic backtracking [2]. When backtracking, this technique remembers the jumped-over choices that do not depend on the removed choice so that they do not have to be reinvestigated.

3 Implementation

DLP is implemented in Standard ML of New Jersey, and uses many of the features of the standard libraries of this language. DLP is a mostly-functional program in that the core of the engine has no side-effects. In fact, the only side effects in the satisfiability engine involve the unique storage of sub-formulae and node caching.

The unique storage of sub-formula and node caching are handled in DLP by a formula cache. When a formula is encountered, it is looked up in the cache.

Problem	D	L	Vars = 4				Vars = 8				Vars = 16			
			Sat	Uns	TO	Time	Sat	Uns	TO	Time	Sat	Uns	TO	Time
qbf-cnfSSS-k4	4	10	8	0	0	0.03	8	0	0	0.03	8	0	0	0.05
qbf-cnfSSS-k4	4	20	6	2	0	0.04	8	0	0	0.11	8	0	0	0.19
qbf-cnfSSS-k4	4	30	2	6	0	0.08	8	0	0	0.27	8	0	0	1.42
qbf-cnfSSS-k4	4	40	1	7	0	0.07	2	6	0	0.54	8	0	0	3.81
qbf-cnfSSS-k4	4	50	0	8	0	0.07	0	8	0	1.44	8	0	0	25.31
qbf-cnfSSS-k4	6	10	8	0	0	0.02	8	0	0	0.04	8	0	0	0.08
qbf-cnfSSS-k4	6	20	8	0	0	0.08	8	0	0	0.19	8	0	0	0.47
qbf-cnfSSS-k4	6	30	7	1	0	0.13	8	0	0	1.06	8	0	0	2.46
qbf-cnfSSS-k4	6	40	2	6	0	0.13	7	1	0	3.96	8	0	0	12.55
qbf-cnfSSS-k4	6	50	0	8	0	0.14	6	2	0	7.80	8	0	0	59.57
qbf-cnfLadn-k4	4	10	8	0	0	0.70	8	0	0	83.97	3	0	5	757.99
qbf-cnfLadn-k4	4	20	8	0	0	1.64	5	0	3	460.41	0	0	8	
qbf-cnfLadn-k4	4	30	2	6	0	1.94	2	0	6	806.31	0	0	8	
qbf-cnfLadn-k4	4	40	0	8	0	0.79	1	1	6	377.39	0	0	8	
qbf-cnfLadn-k4	4	50	1	7	0	1.06	0	5	3	392.81	0	0	8	
qbf-cnfLadn-k4	6	10	8	0	0	6.10	6	0	2	211.28	2	0	6	777.06
qbf-cnfLadn-k4	6	20	8	0	0	20.60	0	0	8		0	8	8	
qbf-cnfLadn-k4	6	30	7	1	0	40.45	0	0	8		0	8	8	
qbf-cnfLadn-k4	6	40	1	7	0	21.98	0	0	8		0	8	8	
qbf-cnfLadn-k4	6	50	0	8	0	3.25	0	0	8		0	8	8	
qbf-cnf-k4	4	10	4	0	0	2.53	4	0	0	113.01	0	0	4	
qbf-cnf-k4	4	20	3	1	0	4.61	1	0	3	632.84	0	0	4	
qbf-cnf-k4	4	30	0	4	0	6.17	0	0	4		0	0	4	
qbf-cnf-k4	6	10	4	0	0	31.98	0	0	4		0	0	4	
qbf-cnf-k4	6	20	4	0	0	100.77	0	0	4					

Fig. 2. Comparison Problems Results—Modal QBF

If the formula is in the cache, it is reused; if not, a new formula is created and added to the cache. Each formula has a satisfiability status; when a new node is created, the formulae for the node are conjoined and this formula is looked up in the formula cache; when a node's status is determined, the satisfiability status of its formula is updated.

Full **PDL** loop checking can be replaced in DLP by a simpler (and much less costly) loop checking mechanism for transitive modalities. An optimization that is valid for transitive modalities but not for transitive closure can also be enabled. These changes turn DLP into a satisfiability checker for a multi-modal logic where some or all of the modalities may be transitive. A standard embedding can also be used to allow DLP to reason with reflexive modalities. DLP is therefore able to handle many modal logics, including $\mathbf{K_{(m)}}$, $\mathbf{KT_{(m)}}$, $\mathbf{K4_{(m)}}$, and $\mathbf{S4_{(m)}}$, and also allows global axioms.

DLP has many options, including turning off all the above non-heuristic optimizations and varying the heuristic optimizations. DLP is also a complete description logic system. It has an interface that can be used to define a collection of concepts and roles. DLP automatically computes the subsumption hierarchy of these concepts and provides facilities for querying this hierarchy.

Problem	D	L	Vars = 4				Vars = 8			
			Sat	Uns	TO	Time	Sat	Uns	TO	Time
psat-cnf-k4	1	20	8	0	0	0.00	8	0	0	0.01
psat-cnf-k4	1	30	8	0	0	0.01	8	0	0	0.02
psat-cnf-k4	1	40	8	0	0	0.02	8	0	0	0.04
psat-cnf-k4	1	50	7	1	0	0.03	6	0	2	0.05
psat-cnf-k4	2	20	7	0	1	0.03	6	0	2	0.06
psat-cnf-k4	2	30	6	0	2	2.17	5	0	3	19.81
psat-cnf-k4	2	40	8	0	0	0.17	0	0	8	
psat-cnf-k4	2	50	4	3	1	1.01	4	0	4	3.75

Fig. 3. Comparison Problems Results—Periodic Satisfiability

4 Performance Analysis

DLP was only tested on the problems that used logics $\mathbf{K_{(m)}}$ and $\mathbf{S4_{(m)}}$, possibly including global axioms. Testing was done on a 400Mhz SPARC Ultra 4 with 512MB of memory, with a timeout of 1000 seconds. A special parser was written for DLP to input the problems. The version of DLP used in the tests employs the simpler transitive modality loop checking, has all optimizations, except the more-recent optional optimizations, enabled, and uses the backjumping maximization and MOMS heuristics as described above. There are two times reported for each reference problem class, both only for those problems that were solved. The first times are the average times for an entire run, including inputing the file and normalizing the resulting formula. The second times are for just the core satisfiability checker. For comparison problems, only the larger time is reported.

The reference problems were almost all trivial for DLP. DLP solved all the qbf-modS4 problems with no splits and the solution time for these problems was completely dominated by the input and normalization time. The single hard reference problem was one of the psat-cnf problems, which timed out.

The results for the Modal QBF problems are given in Figure 2. The SSS encoding was particularly easy for DLP, with most of the problems taking under 1 second, and none timing out. The Ladner encoding was harder—almost all the formulae with 16 variables timed out and all but two at depth 6 with 8 variables timed out. The original encoding is even tougher—for most of the formulae with 8 or 16 variables DLP times out before it does any case splitting, having examined many thousands of modal nodes. The results for the periodic satisfiability problems are given in Figure 3. Most of these formulae were very easy to solve, taking under 0.1 second. Contrastingly, some of the formulae at depth 2 and a few at depth 1 exceeded the timeout.

The overall performance of DLP on these classes of formulae is impossible to determine from the comparison problems as they have too few formulae and too few data points. The results of two more-systematic tests, varying some of the parameters in smaller steps, and having many more formulae at each data point, are given in Figure 4, which gives percentile CPU times for the satisfiability portion of the processing, ignoring the I/O and normalization times. Most of the formulae are in these tests easy for DLP, at least for depth 1—the 90th

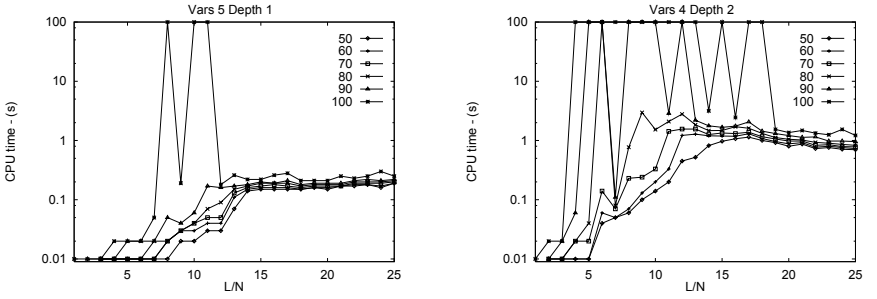


Fig. 4. Results for Systematic Periodic Satisfiability Testing

Problem Set	Satisfiable	Average Time
K4-C45-V8-D4	32	1.097
K4-C45-V8-D5	32	13.676
K4-C55-V8-D6	28	9.029

Fig. 5. Final Results

percentile solution time is everywhere less than 0.25 seconds. However, a very few are much harder, timing out at 100 seconds. For depth 2, the situation is similar, with the vast majority of tests taking around 2 seconds or less and a small number timing out, but with extremely few in the middle.

DLP was able to solve all of the problems in the final set of tests. The longest time for any of the problems was 43.07 seconds. The average solution times for the three sets of problems are given in Figure 5.

References

1. J. W. Freeman. Hard random 3-SAT problems and the Davis-Putnam procedure. *Artificial Intelligence*, 81:183–198, 1996.
2. M. L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.
3. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K. In M. McRobbie and J. Slaney, ed., *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 583–597. Springer-Verlag, 1996.
4. I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, ed., *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98)*, pages 636–647. Morgan Kaufmann Publishers, San Francisco, California, June 1998.
5. I. Horrocks and P. F. Patel-Schneider. FaCT and DLP. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, LNAI 1397, pages 27–30. Springer-Verlag, 1998.
6. I. Horrocks and P. F. Patel-Schneider. Optimising propositional modal satisfiability for description logic subsumption. In J. Calmet, ed., *International Conference AISC'98*, LNAI 1476, pages 234–246. Springer-Verlag, 1998.
7. Peter F. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, ed., *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 87–89, 1998.

Evaluating *SAT on TANCS 2000 Benchmarks

Armando Tacchella

DIST, Università di Genova, Viale Causa, 13 – 16145 Genova, Italy
tac@dist.unige.it

Abstract. In this paper we evaluate *SAT performance on TANCS 2000 benchmarks for the modal logic K. After a brief overview of *SAT implementation, we detail which *SAT options are more relevant in the context of our analysis. Then we present the experimental results: collected data include the CPU time, the number of consistency checks, and the allocated memory, to give a complete picture of *SAT behaviour on the selected benchmarks.

1 Algorithms and implementation

*SAT is a platform for the development of decision procedures for modal logics. Currently, *SAT features decision procedures for the normal modal logic K(m) and for the classical modal logic E(m) (see [1] for a presentation of these logics). In the spirit of [2], *SAT core algorithm can be described by two mutually recursive procedures:

- a SAT checker (see [3]) which *generates* assignments that satisfy the input formula, and
- a routine that *tests* the consistency of each assignment by checking the satisfiability of simpler modal formulas.

For lack of space, we do not elaborate further on *SAT algorithms: more details can be found in [4].

*SAT is implemented in C and should run flawlessly on most Unix platforms equipped with GNU development tools (`gcc` compiler, `make` and `m4` utilities). Libraries, code, and tools needed by *SAT are either publicly available or distributed with the code itself. For an in-depth discussion of *SAT modules, architecture and implementation please refer to [5]. *SAT distribution package, as well as most of the available relevant literature for this paper, can be found at:

`http://www.mrg.dist.unige.it/~tac/StarSAT.html`

2 Running *SAT

*SAT comes with many configurable options that the user selects either at compile-time (using compiler directives) or at run-time (using command line

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 77–81, 2000.

Test (reference)	S/U/T	Cpu	Cons	Mem
C20-V2-D2	10/54/0	0.06	32	638
C20-V2-D2	0/64/0	0.16	157	896
C45-V8-D4	31/32/1	83.42	44418	3815
C45-V8-D5	10/28/26	186.56	88346	4689
C55-V8-D6	0/23/41	172.58	66443	6290

Fig. 1. Reference problems

Test (cnf)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	4/0/0	102.05	26663	2720
C20-V4-D4	3/1/0	321.15	75967	2792
C30-V4-D4	0/4/0	18.08	3953	2802
C10-V8-D4	0/0/4	–	109023	–
C20-V8-D4	0/0/4	–	66377	–
C30-V8-D4	0/0/4	–	55121	–
C10-V16-D4	0/0/4	–	27740	–
C10-V4-D6	1/0/3	981.05	154892	4548
C20-V4-D6	0/0/4	–	152006	–
C10-V8-D6	0/0/4	–	56106	–
C20-V8-D6	0/0/4	–	39312	–

Test (modK)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	0/4/0	30.20	20398	3414
C20-V4-D4	0/4/0	28.99	12796	3515
C30-V4-D4	0/4/0	48.49	28928	3745
C10-V8-D4	0/3/1	704.37	128334	9137
C20-V8-D4	0/0/4	–	262289	–
C30-V8-D4	0/1/3	727.93	96175	13017
C10-V4-D6	0/3/1	64.98	17100	5735
C20-V4-D6	0/1/3	295.58	213999	6388

Fig. 2. Benchmarks using optimized Halpern translation (`hard`)

switches). Most of these options are thoroughly described in *SAT manual [6], to which we point for further details. The results presented in this paper are obtained running *SAT with its default configuration, except for the timeout value that was increased to 1200 seconds of CPU time instead of 1000. Since the timeout includes the time required to build the internal data structures, and the samples of some problems are pretty big, our choice for the timeout value prevents preprocessing overhead to reduce raw search time.

Besides the very basic algorithm and data structures, *SAT currently features the following defaults:

- an optimized internal CNF conversion based on renaming ([7]), and a propositional search control strategy that efficiently takes care of added propositional labels;
- a modal lookahead optimization (called “early pruning” in [2]) that prevents modal inconsistencies from hiding in the propositional search;
- a combination of a caching structure and retrieval algorithms that provides a good balance among fast access time, small memory allocation and global speedup. *SAT caching mechanisms are described in greater detail in [8].

Test (cnfSSS)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	8/0/0	0.08	246	1156
C20-V4-D4	6/2/0	0.48	945	1362
C30-V4-D4	2/6/0	0.66	1036	1434
C40-V4-D4	1/7/0	0.96	1268	1492
C50-V4-D4	0/8/0	0.66	726	1683
C10-V8-D4	8/0/0	0.25	643	1784
C20-V8-D4	8/0/0	3.28	4037	2584
C30-V8-D4	8/0/0	42.72	30507	3409
C40-V8-D4	2/6/0	58.20	32698	3736
C50-V8-D4	0/8/0	75.71	37776	4551
C10-V16-D4	8/0/0	0.46	906	3334
C20-V16-D4	8/0/0	6.57	6684	4565
C30-V16-D4	8/0/0	83.77	43580	5942
C40-V16-D4	2/0/6	604.25	234336	8195
C50-V16-D4	0/0/8	–	340587	–
C10-V4-D6	8/0/0	0.16	410	1455
C20-V4-D6	8/0/0	1.78	2437	1796
C30-V4-D6	7/1/0	7.46	7435	2088
C40-V4-D6	2/6/0	5.83	5396	2492
C50-V4-D6	0/8/0	4.66	3755	2502
C10-V8-D6	8/0/0	0.29	679	2346
C20-V8-D6	8/0/0	4.81	4808	3488
C30-V8-D6	8/0/0	90.65	52982	4262
C40-V8-D6	4/1/3	396.80	163496	5776
C50-V8-D6	0/0/8	–	393254	–
C10-V16-D6	8/0/0	0.73	1316	4370
C20-V16-D6	8/0/0	8.93	6701	5847
C30-V16-D6	8/0/0	176.68	72254	7918
C40-V16-D6	4/0/4	561.58	168941	10445
C50-V16-D6	0/0/8	–	248497	–

Test (modKSSS)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	8/0/0	0.36	750	1501
C20-V4-D4	4/4/0	1.64	2231	1736
C30-V4-D4	3/5/0	2.83	2871	1830
C40-V4-D4	1/7/0	4.43	3461	1879
C50-V4-D4	0/8/0	4.41	3163	2451
C10-V8-D4	8/0/0	0.77	1302	2219
C20-V8-D4	8/0/0	12.77	10773	3230
C30-V8-D4	8/0/0	92.74	49964	4035
C40-V8-D4	2/5/1	517.84	203594	4384
C50-V8-D4	0/3/5	197.68	71422	5582
C10-V16-D4	8/0/0	1.29	1840	3850
C20-V16-D4	8/0/0	75.98	45052	5095
C30-V16-D4	6/0/2	703.53	244983	7068
C40-V16-D4	0/0/8	–	307407	–
C50-V16-D4	0/0/8	–	261662	–
C10-V4-D6	8/0/0	0.50	973	1691
C20-V4-D6	8/0/0	4.86	5279	2102
C30-V4-D6	5/3/0	22.85	16895	2449
C40-V4-D6	1/7/0	33.29	18658	3307
C50-V4-D6	0/8/0	46.07	21509	3353
C10-V8-D6	8/0/0	0.78	1258	2666
C20-V8-D6	8/0/0	22.59	16482	4139
C30-V8-D6	5/0/3	681.74	296324	5988
C40-V8-D6	0/0/8	–	380903	–
C50-V8-D6	0/0/8	–	309856	–
C10-V16-D6	8/0/0	1.75	2064	4850
C20-V16-D6	8/0/0	54.70	25887	6679
C30-V16-D6	5/0/3	502.92	152995	9720
C40-V16-D6	0/0/8	–	253395	–
C50-V16-D6	0/0/8	–	180169	–

Fig. 3. Benchmarks using Schmidt-Schauss-Smolka translation (`easy`)

3 Experiments

Here we present the evaluation of *SAT decision procedure for the modal logic K.[1] We run TANCS 2000 benchmarks, category `MODAL PSPACE Division`, subcategory `Problems for Modal QBF`. The three subclasses `hard`, `easy`, and `medium/hard` are considered, and for each one we run the benchmark problems that can be found on the TANCS 2000 web site, or a subset thereof. For every problem, we also consider the "modalized" version.

We arranged our results in seven tables showed in Figure 1 through Figure 4. In particular:

- the table in Figure 1 collects the results on various reference problems. The top row shows the results that we obtain on 64 samples for the problem `C10-V2-D2` in the `hard` category while the center row shows the result that we obtain on 64 "modalized" samples of the same problem. The entries at the bottom collect the results about three benchmarks in the `easy` category that are suggested as additional reference by the organizers.

[1] All the tests run on a `Pentium II 350 Mhz` equipped with `256 Mb` of main memory and `128 Mb` of swap space, operating system `Linux Suse 6.2` (kernel version `2.2.10`) and compiler `gcc 2.91.66`.

- Figure 2 (left) collects the results that we obtain on 11 benchmarks problems in the `hard` class. For a subset of such problems (8 in number), the table in Figure 2 (right) shows the result that we obtain on the "modalized" versions. In both variants, the number of samples per problem is 4.
- Figure 3 collects the results that we obtain on 30 benchmark problems in the `easy` class. The table on the left shows the results for the "plain" benchmarks, while the table on the right shows the results for the "modalized" variants. The number of samples per problem is 8 for both variants.
- Figure 4 (left) collects the results that we obtain on 19 benchmarks problems in the `medium/hard` class, while Figure 4 (right) shows the results that we obtained on 23 "modalized" versions. For both variants, the number of samples per problem is 8.

Each table consists of five columns. The first column, named **Test**, lists the names of the benchmark problems in each selection. Problems are identified using the official TANCS2000 generator syntax: we write Cx-Vy-Dz to denote a modal formula translated from a QBF having x clauses, alternation depth z and y variables, at most, per each alternation. Each formula has a fixed number of 4 literals per clause. The other columns show the following data:

S/U/T is the number of (S)atisfiable/(U)nsatisfiable/(T)imeout samples according to *SAT; we do not provide a column for memory outs since none of the runs exceeds the physical memory available on our machine;

Cpu is the average running time of *SAT in seconds, including preprocessing and internal CNF conversions; a dash in this column appears when all the samples of the problem exceed the time limit;

Cons is the average number of consistency checks, i.e. the average number of times that *SAT opens up a new modal branch in the search;

Mem is the maximum dynamic memory (in Kbytes) allocated by *SAT down any search path; a dash in this column appears when all the samples of the problem exceed the time limit.

Notice that "average number" here, has a different meaning depending on whether *SAT ends up in a timeout condition or not. If there is at least one sample on which *SAT does not exceed the time limit, the values for **Cpu**, **Cons** and **Mem** are the geometric average on *all* the samples that *do not exceed the time limit*. We decided this setting because, when a sample ends up in a timeout condition, **Cpu** and **Mem** measurements are either not significant (**Cpu** could be way bigger than the time limit) or not reliable (this happens for **Mem** in our implementation of timeouts). If *SAT exceeds the lime limit on all the samples, **Cpu** and **Mem** values are void. A different policy is used for **Cons** values: when all the samples exceed the time limit, **Cons** value is the geometric average on all the samples. In this case, **Cons** denotes the number of consistency checks performed, on the average, up to the time limit.

References

1. B. F. Chellas. *Modal Logic – an Introduction*. Cambridge University Press, 1980.

Test (cnfLadn)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	8/0/0	404.29	314270	2546
C20-V4-D4	2/0/6	851.98	491633	2675
C30-V4-D4	0/4/4	208.54	98368	2797
C40-V4-D4	0/8/0	133.40	55410	2903
C50-V4-D4	0/6/2	57.65	20764	3415
C10-V8-D4	0/0/8	–	460257	–
C20-V8-D4	0/0/8	–	301734	–
C30-V8-D4	0/0/8	–	190460	–
C40-V8-D4	0/0/8	–	179803	–
C10-V16-D4	0/0/8	–	274821	–
C20-V16-D4	0/0/8	–	128255	–
C10-V4-D6	0/0/8	–	637938	–
C20-V4-D6	0/0/8	–	419442	–
C30-V4-D6	0/0/8	–	339490	–
C40-V4-D6	0/0/8	–	313328	–
C50-V4-D6	0/0/8	–	272332	–
C10-V8-D6	0/0/8	–	366596	–
C20-V8-D6	0/0/8	–	178572	–
C10-V16-D6	0/0/8	–	245182	–

Test (modKLadn)	S/U/T	Cpu	Cons	Mem
C10-V4-D4	0/8/0	2.73	2708	3007
C20-V4-D4	0/8/0	6.89	4805	3399
C30-V4-D4	0/8/0	9.08	6383	3429
C40-V4-D4	0/8/0	16.87	11012	3501
C50-V4-D4	0/8/0	20.87	14330	4027
C10-V8-D4	0/8/0	25.18	11868	6264
C20-V8-D4	0/8/0	502.22	151215	9587
C30-V8-D4	0/6/2	570.79	159546	10852
C40-V8-D4	0/3/5	655.53	164708	11309
C50-V8-D4	0/2/6	934.99	199231	12379
C10-V16-D4	0/0/8	–	497959	–
C20-V16-D4	0/0/8	–	150031	–
C10-V4-D6	0/8/0	2.18	1474	4129
C20-V4-D6	0/8/0	7.66	4534	5441
C30-V4-D6	0/8/0	14.82	7395	5628
C40-V4-D6	0/8/0	21.95	10432	6506
C50-V4-D6	0/8/0	33.71	15513	6667
C10-V8-D6	0/8/0	53.73	20948	9938
C20-V8-D6	0/8/0	172.09	38982	16164
C30-V8-D6	0/7/1	447.32	86268	20550
C40-V8-D6	0/2/6	885.67	115207	22524
C50-V8-D6	0/2/6	1113.81	208242	24986
C10-V16-D6	0/3/5	375.51	95405	24267

Fig. 4. Benchmarks using Ladner translation (`medium`)

2. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. CADE-96*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag.
3. H. Zhang. SATO: An efficient propositional prover. In William McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, July13–17 1997. Springer.
4. E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT-Based Decision Procedures for Classical Modal Logics. *Journal of Automated Reasoning*, 2000. To appear.
5. Armando Tacchella. *SAT system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.
6. Armando Tacchella. *SAT User's Manual, 1999.
7. D.A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
8. Enrico Giunchiglia and Armando Tacchella. A Subset-matching Size-bounded Cache for Satisfiability in Modal Logics. In *Proc. of International Conference on Theorem Proving with Analytic Tableaux and Related Methods (Tableaux 2000)*, *LNAI*, Berlin, July 2000. Springer Verlag.

A Labelled Tableau Calculus for Nonmonotonic (Cumulative) Consequence Relations*

Alberto Artosi[1], Guido Governatori[2], and Antonino Rotolo[3]

[1] Department of Philosophy, University of Bologna,
via Zamboni 38, I-40126 Bologna, Italy. E-mail: artosi@cirfid.unibo.it
[2] School of Computing and Information Technology, Griffith University,
Nathan, QLD 4111, Australia. E-mail: guido@cit.gu.edu.au
[3] CIRSFID, University of Bologna,
Via Galliera 3, I-40121 Bologna, Italy. E-mail: rotolo@cirfid.unibo.it

Abstract. In this paper we present a labelled proof method for computing nonmonotonic consequence relations in a conditional logic setting. The method is based on the usual possible world semantics for conditional logic. The label formalism KEM, introduced to account for the semantics of normal modal logics, is easily adapted to the semantics of conditional logic by simply indexing labels with formulas. The inference rules are provided by the propositional system KE^+ —a tableau-like analytic proof system devised to be used both as a refutation and a direct method of proof— enlarged with suitable elimination rules for the conditional connective. The resulting algorithmic framework is able to compute cumulative consequence relations in so far as they can be expressed as conditional implications.

1 Introduction

Recently, a number of proposals have been put forward to find a unifying approach to a plethora of different nonmonotonic formalisms, and even to unify such seemingly distant areas as conditional logic, nonmonotonic inference, belief revision and update. We refer, in particular, to Shoham's [20] general semantic framework for nonmonotonic logics, Kraus, Lehman and Magidor's [16] approach to nonmonotonic consequence relations, and Katsuno and Satoh's [15] "unified" semantic view. All these approaches are based on a preference (ordering) semantics and exploit the strong semantic connections between nonmonotonic inference and conditional logic. In this paper we shall take a different view of what a suitable "unifying" framework looks like. In our view such a framework must pay greater attention to the computational aspects and to proof-theoretical formulations. This view finds strong justification both in the aim of comparing and combining different logics, such as the logic of nonmonotonic inference and conditional logic, and in the potential applications of nonmonotonic inference in the AI field. Accordingly, our purpose in this paper will be to provide a methodology for the proof theoretical treatment of nonmonotonic inference and conditional logic (henceforth CL). We shall outline the fundamentals of a tableau proof system construction aimed to compute nonmonotonic

* Due to space limitations, theorems are provided without proofs. The full version of the paper is available at http://www.cit.gu.edu.au/~guido/papers/tab2000.pdf.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 82–97, 2000.

consequence relations in a (monotonic) CL whose "flat" (i.e., unnested) fragment is shown to correspond to Kraus, Lehmann and Magidor's [16] basic system C for cumulative relations. We shall give this construction a special presentation as an algorithmic proof system which uses a labelling discipline, in the wake of Gabbay's [11] *Labelled Deductive Systems* (LDS), to generate and check models. This closely reflects Gabbay's view of what a unifying framework for presenting and comparing logics comprises.

A detailed discussion of the merits of LDS as a unifying framework is beyond the scope of this paper. However, a key feature of LDS is worth mentioning. LDS are in general very sensitive to the various features of different logics so that differently motivated and formulated logics can very often be combined in a simple and natural way provided we have a suitable LDS formulation for them (see e.g. [12,3]). As is well-known, several attempts to establish close semantic connections between nonmonotonic consequence relations and (monotonic) modal and conditional logics, notably by Boutilier [4] and Katsuno and Satoh [15], rely on Kripke structures very close to Kraus, Lehmann and Magidor's "preferential" models. In particular, Boutilier [4] has shown on this basis that Kraus, Lehmann and Magidor's [16,18] stronger consequence relation systems $\mathbf{P}$ and $\mathbf{R}$ and Degrande's [7] logic N closely correspond to the flat parts of modal CLs definitionally equivalent to the standard modal systems $\mathbf{S4}$ and $\mathbf{S4.3}$. In LDS the usual modal semantics is incorporated in the syntactic label construction and only minor variations are needed to pass from a logic to another [1,3,12]. So, once an automated LDS for $\mathbf{S4}$ and $\mathbf{S4.3}$ is available, a wide range of logics admit computational treatment. However, if we wish automated LDS for CLs on their own, or we are interested in a less restricted fragment of the conditional language, only slight natural changes in the modal LDS are needed to yield the appropriate semantics: in the LDS to be presented in this paper only a simple indexing of labels with formulas.

The approach we propose in this paper can be motivated also from another perspective. Fariñas del Cerro *et al.* [9] have recently emphasized the need for a more computational treatment of nonmonotonic inference. The method they propose for this task consists of reducing computation to a validity test in a (monotonic) CL. This is viewed as a first step towards the "effective computation ... of nonmonotonic inference relations via automated deduction method" in conditional logic. Unfortunately, CL is not particularly well suited for this task. Indeed, its inferential structure has not been sufficiently explored to provide reliable automated deduction methods for effectively computing the inferences sanctioned by nonmonotonic inference relations. We know only two attempts in this direction: Groeneboer and Delgrande's [13] and Lamarre's [17] tableau-based theorem provers for some normal CLs. In both approaches a conditional formula is checked for validity by attempting to construct a model for its negation. What we undertake in this paper can be viewed as a further step in the same direction, as in our approach nonmonotonic consequence relations can be effectively computed by a countermodel validity test for the corresponding class of conditional formulas.

We shall proceed in the following way. First we briefly rehearse Kraus, Lehmann and Magidor's [16] sequent system $\mathbf{C}$ for cumulative relations. Then we introduce Lewis-type semantic structures akin to the kind of models used to characterize $\mathbf{C}$. Such structures will allow us to establish a correspondence between $\mathbf{C}$ and the flat fragment of a suitable extension $\mathbf{CU}$ of Chellas' [5] basic normal system $\mathbf{CK}$. At this point, we shall be able

to show how cumulative relations can be effectively computed by an LDS provided by a tableau-like proof system together with a label formalism adequate to represent the intended semantics. The system is presented in two steps. First, the labelling (formalism + label unification) scheme introduced in [1] to account for the semantics of normal modal logics is adapted to represent Lewis-type semantic structures for **CU**. Then suitable tableau inference and label propagation rules are introduced which provide a sound and complete proof system for the flat fragment of **CU**. These rules are implemented on a classical propositional system designed to be used both as a refutation and a direct method of proof. Finally, we provide some remarks on computational issues and related works.

2 Nonmonotonic Consequence Relations and Conditional Logic

The study of nonmonotonic consequence relations has been undertaken by Gabbay [10] who proposed three minimal conditions a (binary) consequence relation $\mid\!\sim$ on a language L should satisfy to represent a nonmonotonic logic. More recently, Kraus, Lehmann and Magidor [16] have investigated the proof-theoretic and semantic properties of a number of increasingly stronger families of nonmonotonic consequence relations. In particular, they have provided the following sequent system **C** to define the (weakest) class of cumulative consequence relations, that closely corresponds to that satisfying Gabbay's minimal conditions (we assume that $\vdash$ and $\mid\!\sim$ are defined on the language L of classical propositional logic).

$$A \mid\!\sim A \qquad \text{(Reflexivity)}$$

$$\frac{\vdash B \rightarrow C \quad A \mid\!\sim B}{A \mid\!\sim C}\text{Right Weakening} \qquad \frac{\vdash A \equiv B \quad A \mid\!\sim C}{B \mid\!\sim C}\text{Left Logical Equivalence}$$

$$\frac{A \mid\!\sim B \quad A \mid\!\sim C}{A \wedge B \mid\!\sim C}\text{Cautious Monotonicity} \qquad \frac{A \wedge B \mid\!\sim C \quad A \mid\!\sim B}{A \mid\!\sim C}\text{Cut}$$

Notice that the following

$$\frac{A \mid\!\sim B \quad A \mid\!\sim C}{A \mid\!\sim B \wedge C}\ \text{And} \qquad \frac{A \mid\!\sim B \quad B \mid\!\sim A}{A \mid\!\sim C \iff B \mid\!\sim C}\ \text{CSO}$$

are derived rules of C. A sequent $A \mid\!\sim B$, $A, B \in L$ (intended reading: B is a plausible consequence of A), is called a *conditional assertion*. The (proof-theoretic) notion of cumulative entailment is defined for such assertions. Let Γ be a set of conditional assertions. A conditional assertion $A \mid\!\sim B$ is said to be *cumulatively entailed* by Γ iff $A \mid\!\sim B$ is derived from Γ using the rules of **C**.

Let $L_>$ be the language obtained by adding the conditional connective $>$ to L. The set of (well-formed) formulas of $L_>$ is defined in the usual way. Formulas of $L_>$ are interpreted in terms of Lewis-type semantic structures akin to the kind of models used by Kraus, Lehmann and Magidor [16] to characterize **C**.

More precisely, it is enough to introduce some constraints (see definition) 2 on the basic selection function model presented in definition 1.

Definition 1. *A* selection function *(SF)* model *is a triple* $M = \langle W, f, \upsilon \rangle$ *where*

1. W *is a nonempty set (of possible worlds);*
2. f *is a selection function which picks out a subset* $f(A, u)$ *of* W *for each* u *in* W *and* $A \in L_{>}$*;*
3. υ *is a valuation assigning to each* u *in* W *and* $A \in L_{>}$ *an element from the set* $\{T, F\}$.

Truth of a formula A at a world u in a model M, $M \models_u A$, is defined as usual with the conditional case given by

$$M \models_u A > B \text{ iff } f(A, u) \subseteq \|B\| \tag{1}$$

where $\|B\|$ denotes the set of B-worlds, i.e., $\|B\| = \{w \in W : \upsilon(B, w) = T\}$. A formula A is valid ($\models_{SF}$) just when $M \models_u A$ for all worlds in all SF models.

Definition 2. *A* selection function cumulative model *(SFC) is an SF model* $M = \langle W, f, \upsilon \rangle$ *satisfying the following conditions:*

1. $f(A, u) \subseteq \|A\|$ *(Reflexivity)*
2. *If* $||A|| = ||B||$, *then* $f(A, u) = f(B, u)$ *(Left Logical Equivalence)*
3. *If* $f(A, u) \subseteq \|B\|$, *then* $f(A \wedge B, u) \subseteq f(A, u)$ *(Cut)*
4. *If* $f(A, u) \subseteq \|B\|$, *then* $f(A, u) \subseteq f(A \wedge B, u)$ *(Cautious Monotonicity).*

Notice that from 3 and 4 we obtain

$$f(A, u) \subseteq ||B|| \Rightarrow f(A \wedge B, u) = f(A, u) \tag{2}$$

It is not hard to see that the class of SFC models fits exactly the conditional logic —call it $\mathbf{CU}$— containing classical propositional logic and the following axioms

1. $A > A$ (ID)
2. $(A > B) \wedge (A \wedge B > C) \supset (A > C)$ (RT)
3. $(A > B) \wedge (A > C) \supset (A \wedge B > C)$ (CM)

and closed under the usual inference rules $RCEA$ and RCK. Notice that ID, RT, CM, $RCEA$, and RCK correspond, respectively, to Reflexivity, Cut, Cautious Monotonicity, Left Logical Equivalence and Right Weakening. Of course, $\mathbf{CU}$ is nothing but Chellas' [5] conditional logic $\mathbf{CK} + ID$ augmented with RT and CM. A standard Henkin-style construction proves the completeness of $\mathbf{CU}$ with respect to the class of SFC models.

Theorem 1. $\models_{SFC} A$ *iff* $\vdash_{\mathbf{CU}} A$.

Whether $\mathbf{CU}$ is interesting in its own rights is an issue which need not detain us here. What matters is that we can establish a mapping between C and $\mathbf{CU}$ similar to the well-established correspondences between [16]'s stronger systems $\mathbf{P}$ and $\mathbf{R}$ of preferential and rational relations and the flat fragments of well-known conditional logics.

Let $\mid\!\sim_{\mathbf{S}}$ denote the consequence relation $\mathbf{S}$ and let $\mathbf{K}^{-}$ denote the conditional logic $\mathbf{K}$ restricted to the formulas of the form $A > B$ where $A, B \in L$. A consequence relation $\mid\!\sim$ is *defined by* an SF model M if the following condition is satisfied: $A \mid\!\sim B$ iff $M \models A > B$. A consequence relation system $\mathbf{S}$ is said to *correspond to* a conditional logic K if the following condition is satisfied : $A \mid\!\sim_{\mathbf{L}} B$ iff $\vdash_{\mathbf{K}^{-}} A > B$.

Theorem 2. *The consequence relation system* **C** *corresponds to the conditional logic* **CU**.

The theorem follows from showing that the axioms and rules of **CU** are straightforward translations of the rules of **C** and that $A \mid\sim_{\mathbf{C}} B$ is the consequence relation defined by an *SFC* model. From this it follows as a corollary that a consequence relation $\mid\sim$ is cumulative iff it is defined by some *SFC* model. The same holds for the notion of cumulative entailment. For a set Γ of conditional assertions let us denote by Γ' the set containing the $\mathbf{CU}^{-}$ translations of the conditional assertions in Γ (i.e., $A > B \in \Gamma'$ for each $A \mid\sim B \in \Gamma$). The following corollaries follow immediately from Theorem 2 (see Corollaries 3.26, 3.27 and 3.28 of [16] for comparison).

Corollary 1. *Let* Γ *be a set of conditional assertions and* $A \mid\sim B$ *a conditional assertion. The following conditions are equivalent. In case they hold we shall say that* Γ *cumulatively entails* $A \mid\sim B$.

1. $A > B$ *is derived from* Γ' *using the axioms and the rules of* **CU**.
2. $A > B$ *is satisfied by all SCF models which satisfy* Γ'.

Corollary 2. *A set of conditional assertions* Γ *cumulatively entails* $A \mid\sim B$ *iff* $\vdash_{\mathbf{CU}} \bigwedge \Gamma' \rightarrow (A > B)$.

We conclude that the system **C** may be viewed itself as a restricted CL of the standard (normal) type provided the relation symbol $\mid\sim$ is interpreted as a $>$-type conditional connective. With this background we shall be able, in the upcoming sections, to provide an algorithmic framework for computing cumulative consequence relations in so far as they can be expressed as conditional implications.

3 *KEM* for Nonmonotonic Consequence Relations

KEM [1] is an algorithmic modal proof system which, in the spirit of Gabbay's [11] LDS, brings semantics into proof theory using (syntactic) labels in order to simulate models in the proof language. In this section we show how it can be extended, with little modification, to handle **C**. In what follows $\mathcal{L}$ will denote the sublanguage of $L_{>}$ including L and all the boolean combinations of formulas of the form $A > B$ where $A, B \in L$.

3.1 Label Formalism

The format of the labels has been designed to cover general possible world semantics. In passing from Kripke models for modal logics to *SF* models the format of the labels is left unchanged. The only modification is that atomic labels are now indexed by formulas.

Let $\Phi_C = \{w_1, w_2, \dots\}$ and $\Phi_V = \{W_1, W_2, \dots\}$ be two arbitrary sets of *atomic labels*, respectively constants and variables. A *label* in the sense of [1] is an element of the set of labels $\Im$ defined as follows:

Definition 3. $\Im = \bigcup_{1 \leq p} \Im_p$ *where* $\Im_p$ *is:*

$\Im_1 = \Phi_C \cup \Phi_V$
$\Im_2 = \Im_1 \times \Phi_C$
$\Im_{n+1} = \Im_1 \times \Im_n,\ n > 1\,.$

Thus, a label is any $i \in \Im$ such that either i is an atomic label or $i = (k', k)$ where (i) k' is atomic and (ii) $k \in \Phi_C$ or $k = (m', m)$ where (m', m) is a label, i.e., i is generated as a "structured" sequence of atomic labels. We define the length of a label i, $l(i)$, to be the number of atomic labels in i. From now on we shall use $i, j, k, \ldots$ to denote arbitrary labels. For a label $i = (j, k)$, we shall call j the *head* and k the *body* of i, and denote them by $h(i)$ and $b(i)$ respectively; $h^n(i)$ will denote the head of the sub-label of i whose length is n. We shall call a label i *restricted* if its head is a (possibly indexed) constant, otherwise we shall call it *unrestricted*. Let us stipulate that if $i \in \Phi_C \cup \Phi_V$ and $Y \in \mathcal{L}$ then $i^Y \in \Im_1$. We shall call a label i^Y a *formula-indexed label*, and Y the *label formula* of i. For a label i we shall use i^Y to indicate that the label formula of $h(i)$ is Y. In general, when we speak of the label formula of structured label, we mean the label formula of the head of the label.

The notion of a formula-indexed label is meant to capture the intended semantics. An atomic label indexed with a formula Y (such as, e.g., w_1^A or W_1^A) is meant to represent either a Y-world or a set of Y-worlds (equivalently, any Y-world) in some SF model. A label of the form (k'^Y, k) is "structurally" designed to convey information about the worlds in it. For example, (W_1^A, w_1) can be viewed as representing (any world in) the set of those A-worlds that are minimal with respect to w_1 under some ordering relation $\prec$. The label (w_1^A, w_1) represents an A-world in such a set. In this perspective a *labelled signed formula* (*LS-formula*) [1] is a pair X, i where X is a signed formula (i.e., a formula of $\mathcal{L}$ prefixed with a "T" or "F") and i is a label. Intuitively, an LS-formula, e.g. $TC, (W_1^{A \vee B}, w_1)$, means that C is true at all the (minimal) $A \vee B$-worlds.

As we have seen formulas can occur in LS-formulas either as the declarative part or as label formulas; moreover formulas in both parts can and must be used to draw inferences. To deal with this fact we say that SA *occurs* in X, i^Y ($SA : X, i^Y$). More precisely:

$$SA : X, i^Y \iff \begin{cases} X = SA \text{ or} \\ Y = A \text{ and } S = T \end{cases}$$

where $S \in \{T, F\}$, $A, Y \in \mathcal{L}$, X is a signed formula, and $i \in \Im$. That means that either SA is labelled with i, or i is indexed with A. For example, in the expression $SA : X, i^Y$, where $X = FB \to C$ and $i^Y = (W_1^{B \wedge C}, w_1)$, SA stands both for $FB \to C$, and $B \wedge C$, since these are the formulas occurring in X, i^Y.

In what follows we assume familiarity with Smullyan [21] uniform notation for signed formulas.

3.2 Label Unifications

The key feature of KEM approach is that in the course of proof search labels are manipulated in a way closely related to the semantics of modal operators and "matched"

using a specialized unification algorithm. That two labels i and k are unifiable means, intuitively, that the set of worlds they "denote" have a non-null intersection. In this section we define a special notion of unification for $\mathbf{C}$($\sigma_{\mathbf{C}}$-unification) which is meant to "simulate" the conditions on f in SFC-models. We shall proceed by first defining the unification for unindexed labels, and then by extending it to formula-indexed labels.

First of all we introduce a label substitution $\rho : \Im \mapsto \Im$ thus defined:

$$\rho(i) = \begin{cases} i & i \in \Phi_C \\ j \in \Im & i \in \Phi_V \\ (\rho(h(i)), \rho(b(i))) & i \in \Im_n, n > 1 \end{cases}$$

For two labels i and j, and a substitution ρ, if ρ is a unifier of i and j then we shall say that i, j are σ-*unifiable*. We shall use $(i, j)\sigma$ to denote both that i and j are σ-unifiable and the result of their unification. In particular

$$\forall i, j, k \in \Im, (i, j)\sigma = k \text{ iff } \exists \rho : \rho(i) = \rho(j) \text{ and } \rho(i) = k, \text{ and}$$
$$\text{for each } n \text{ at least one of } h^n(i) \text{ or } h^n(j) \text{ is in } \Phi_C.$$

According to the above condition the labels $(w_3, (W_1, w_1))$ and $(W_2, (w_2, w_1))$ σ-unify on $(w_3, (w_2, w_1))$. On the other hand the labels $(w_2, (W_1, w_1))$ and $(W_2, (W_1, w_1))$ do not σ-unify because both h^2s are not in Φ_C.

The definition of the unification for indexed labels is more complicated since we have to take into account label formulas. As said before, the conditions on label formulas should mimic the semantics of SFC-models, but we have to devise them in a syntactic way. In particular, to check that two sets of worlds denoted by different indexed labels overlap, we have to determine a specific mechanism for comparing distinct label formulas. From a proof-theoretical point of view, such a comparison is concerned with the definition of a criterion for composing different structures of formulas. However, it is well-known that cumulative logics do not allow unrestricted compositions of proofs (see, e.g., [6]). In other words, they avoid to substitute an antecedent by another antecedent by transitivity (via cut). The following definitions establish the basic (restricted) conditions for such a substitution. In particular, they say when two formulas are equivalent with respect to $\mid\sim$ ($\mid\sim$-equivalent).

Definition 4.

- *If A is of type α, then $\{\alpha_1, \alpha_2\}$* c-fulfils *A;*
- *if A is of type β, then $\{\beta_1\}$* c-fulfils *A, and $\{\beta_2\}$* c-fulfils *A;*
- *if $\{A_1, \ldots, A_n\}$* c-fulfils *A, and $\{A_{1_1}, \ldots, A_{1_m}\}, \ldots, \{A_{n_1}, \ldots, A_{n_m}\}$* c-fulfil *respectively $A_1, \ldots, A_n$, then $\{A_{1_1}, \ldots, A_{1_m}, \ldots, A_{n_1}, \ldots, A_{n_m}\}$* c-fulfils *$A$.*

It is easy to see that whenever a set of formulas c-fulfils a formula A the conjunction of the formulas in the set propositionally entails A.

Definition 5. *We say that A* forces *B, iff 1) either $A = B$ or A is of type α and $B = \alpha_i$; or 2) there exists a formula C such that A forces C and C forces B.*

Obviously, the notion of "forcing" is meant to determine the subformulas of a formula A that are propositionally entailed from A itself.

Let $\mathcal{B}$ be any set of LS-formulas. (In the course of proof search, $\mathcal{B}$ will be the set of LS-formulas occurring in a branch of a proof tree).

Definition 6. A supports B *in a branch* $\mathcal{B}$ *iff*

1. $A \equiv B$; *or*
2. $\{B_1, \ldots, B_n\}$ *c-fulfils* B, *and for each* k, $1 \leq k \leq n$, B_k , $(W^A_{i_k}, w_1) \in \mathcal{B}$; *or*
3. *there is a set of formulas* $\mathcal{A} = \{Z_1, \ldots, Z_n\}$ *such that,* $1 \leq i \leq n$, $Z_i, (W^A_i, w_1) \in \mathcal{B}$, A *forces* Z_i, *and* $\mathcal{A}$ *c-fulfils* B.

We are now ready to say when two formulas, A and B, are $\mid\!\sim$-equivalent in $\mathcal{B}$ ($A \approx_{\mathcal{B}} B$).

Definition 7. $A \approx_{\mathcal{B}} B$ *iff*

1. A *and* B *support each other; or*
2. *if* $A \approx_{\mathcal{B}} C$ *and* $B \approx_{\mathcal{B}} C$, *then* $A \approx_{\mathcal{B}} B$

If $A \in \mathcal{B}$, with $A_{\approx_{\mathcal{B}}}$ we shall denote the set of formulas $\{B_1, \ldots, B_n\}$ such that, $1 \leq i \leq n$, $B_i \in \mathcal{B}$ and $B_i \approx_{\mathcal{B}} A$. It is obvious that $A_{\approx_{\mathcal{B}}}$ is an equivalence class, thus we abuse the notation and we use $A_{\approx_{\mathcal{B}}}$ to denote a formula in such a class.

Two formulas A and B are obviously equivalent with respect to $\mid\!\sim$, if they are classically equivalent. Otherwise, through the notion of *support* (see definition 6), we have basically the following cases: (i) the set of truth-value assignments which correspond to the consequences of A satisfies B; (ii) the set of consequence relations of A propositionally entails B. So, according to definition 7, A and B are equivalent with respect to $\mid\!\sim$ in $\mathcal{B}$ if (a) the above sets are equal, or (b) such sets are equal to another set. This means that they prove each other.

At this point we are ready to introduce the notion of unification for indexed labels to be used in the calculus. Briefly, two labels unify wrt a set of LS-formulas if the unindexed labels unify and the label formulas satisfy conditions corresponding to clauses 1–4 of the semantic evaluation. In the next definition we provide such conditions.

Definition 8. *Let* i^X *and* j^Y *be two indexed labels, and let* $\mathcal{B}$ *be a set of* LS*-formulas. Then*

$$(i^Y, j^X)\sigma^{\mathcal{B}}_{\mathbf{C}} = (i, j)\sigma$$

where 1) $Y \not\equiv \bot$ *if* $h(i) \in \Phi_V$; *2)* $X \not\equiv \bot$ *if* $h(j) \in \Phi_V$, *and one of the following conditions is satisfied*

a) $Y \approx_{\mathcal{B}} X$;
b) $Y \equiv \top$ *and* $h(i) \in \Phi_V$, *or* $X \equiv \top$ *and* $h(j) \in \Phi_V$;
c) i) X *is of type* α, $Y \approx_{\mathcal{B}} \alpha_i$ *for* $i = 1, 2$, *and* $Z, (W^{Y_{\approx_{\mathcal{B}}}}, w_1) \in \mathcal{B}$ *such that* Z*'s c-fulfil* α_{3-i} , *or*
ii) Y *is of type* α, $X \approx_{\mathcal{B}} \alpha_i$ *for* $i = 1, 2$, *and* $Z, (W^{X_{\approx_{\mathcal{B}}}}, w_1) \in \mathcal{B}$ *such that* Z*'s c-fulfil* α_{3-i}.

According to 1) and 2) no label unifies with an unrestricted label whose label formula is unsatisfiable. Intuitively, this excludes that two propositionally indexed sets of worlds have a null intersection, which would be possible with an unrestricted label indexed with a contradiction: since $f(Y, u) = \emptyset$ if $Y \equiv \bot$, so the "denotation" or the label is empty. Indeed $||\bot|| = \emptyset$, and, by reflexivity, for each $A \in L_>$ and $u \in W$, $f(A, u) \subseteq ||A||$, then $f(\bot, u) = \emptyset$.

Clause a) corresponds to Left Logical Equivalence and CSO: both establish when two formulas are equivalent with respect to $\mid\sim$; but logically and non-monotonically equivalent formulas have the same selection function sets.

According to b), $(W_2^{A\rightarrow A}, w_1)$ and (w_3^C, w_1) unify, as W_2 is a variable indexed with a tautology. In practice a unrestricted label indexed with a tautology unifies with any restricted label.

Clause c) is meant to characterize cumulativity. Cumulativity is a restricted version of Left Weakening. Accordingly, we have to see whether a conjunction is a weakening of one conjunct and the other conjunct is derivable in each minimal world with respect to the former component. This is achieved thanks to the notion of c-fulfillment. Such a notion is nothing else that the condition a set of formulas must satisfy to (propositionally) entail the formula which is "fulfilled". Notice that the notion of c-fulfillment is also strictly related to Right Weakening. As an example, consider the following labels: $i = (w_2^{A\wedge(C\rightarrow(B\wedge D))}, w_1)$, $j = (W_1^A, w_1)$, and the following LS-formulas: $\mathcal{A}_1 = TB, (W_2^A, w_1)$, $\mathcal{A}_2 = TD, (W_3^A, w_1)$. Here $(i, j)\sigma_{\mathbf{C}}^{\mathcal{B}}$, where $\mathcal{B}$ contains $\mathcal{A}_1$ and $\mathcal{A}_2$. Notice that $A \wedge (C \rightarrow (B \wedge D))$ is of type α, and A is α_1. Moreover $\{B, D\}$ c-fulfils $B \wedge D$ which, in turn, c-fulfils $C \rightarrow (B \wedge D)$, i.e. α_2 .Thus $\mathcal{B}$ contains a set of LS-formulas whose labels are appropriate, and whose declarative units c-fulfil α_2.

Although the above conditions seem to be very cumbersome, as we shall see in section 5, they can be easily detected by the LS-formulas occurring in a proof tree, and closely correspond to the semantic conditions of SFC-models.

3.3 Inference Rules

The heart of the proof system for **C** is constituted by the following rules which are designed to work both as inference rules (to make deductions from both the declarative and the labelled part of LS-formulas), and as ways of manipulating labels during proofs. In what follows we write $(i, j)\sigma_{\mathbf{C}}^{\mathcal{B}}$ to denote both that i and j are $\sigma_{\mathbf{C}}^{\mathcal{B}}$-unifiable and the result of their $\sigma_{\mathbf{C}}^{\mathcal{B}}$-unification, and X^C to denote the conjugate of X (i.e., $X^C = FA$ (TA) if $X = TA$ (FA)).

$$\alpha\frac{\alpha : X, k^Y}{\alpha_i, k^Y}[i = 1, 2] \qquad \beta\frac{\beta : X, k^Y}{\beta_i, (k, j)\sigma_{\mathbf{C}}^{\mathcal{B}}}\frac{\beta_{3-i}^C : X', j^{Y'}}{}[i = 1, 2]$$

These are exactly the α and β rules of the original KEM method [1] in a slightly modified version: the formulas the rule is applied to are either the declarative parts or the label formulas. The α rules are just the usual linear branch-expansion rules of the tableau methods, whereas the β rules correspond to such common natural inference patterns as modus ponens, modus tollens, disjunctive syllogism, etc.

$$T \mid\sim \frac{TA \mid\sim B : X, i^Y}{TB, (W_n^A, i^Y)}[W_n^A \text{ new}] \qquad F \mid\sim \frac{FA \mid\sim B : i^Y}{FB, (w_n^A, i^Y)}[w_n^A \text{ new}]$$

The rules $T\mid\sim$ and $F\mid\sim$ closely reflect the semantical evaluation rule 1.They are motivated by the general idea (see [5]) that $>$ can be regarded as a necessity operator on A (i.e.,

$[A >]B$), from which it follows that whenever $A > B$ is true at a world u, B should be true at all the worlds in $f(A, u)$ (A-worlds); and whenever $A > B$ is false at u, there should be some A-world where B is false.

$$PB\ \frac{}{X, i \qquad X^C, i}[i \text{ unrestricted}] \qquad PNC\ \frac{\begin{matrix} X : Y, i^{Y'} \\ X^C : Z, k^{Z'} \end{matrix}}{\times}[(i,k)\sigma_C^{\mathcal{B}}]$$

PB (the "Principle of Bivalence") is exactly the "cut" rule of the original method (it can be thought of as the semantic counterpart of the cut rule of the sequent calculus). PNC (the "Principle of Non-Contradiction") is the modified version of the familiar branch-closure rule of the tableau method. As it stands, it allows closure ("$\times$") to follow from two formulas which are contradictory "in the same world", represented by two $\sigma_{\mathbf{C}}^{\mathcal{B}}$-complementary LS-formulas, i.e., two LS-formulas $X, i^{Y'}$, $X^C, k^{Z'}$ whose labels are $\sigma_{\mathbf{C}}^{\mathcal{B}}$-unifiable (such as, e.g, $TC, (W_1^{A \vee B}, w_1)$ and $FC, (w_3^{A \vee B}, w_1)$). Notice that, in contrast with the usual normal modal setting, in the present setting it may occur a contradiction of the form $FA, (w_2^A, w_1)$, since this LS-formula states that there exists an A-world where A is false.

In the following section the above set of rules will be proved to be sound and complete for $\mathbf{C}$. Notice that the format of the rules prevents labels from having a length greater than two. This is obviously due to the fact that $\mathbf{C}$ corresponds to $\mathbf{CU}^-$ (in the context of $\mathbf{C}$ the nesting of $\mid\!\sim$ is meaningless).

4 Soundness and Completeness

In this section we prove soundness and completeness theorems for KEM. We shall proceed as usual by first proving that the rules for $\mathbf{C}$ are derivable in KEM, and then that the rules of KEM are sound with respect to the semantics for $\mathbf{C}$. Let us first define the following functions which map labels into elements of SF cumulative models.

Let g be a function from $\Im$ to $\wp(W)$ thus defined:

$$g(i^X) = \begin{cases} \{w_i\} \subseteq f(X, g(h(i))) & \text{if } h(i^X) \in \Phi_C \\ \{w_i \in W : w_i \in f(X, g(h(i)))\} & \text{if } h(i^X) \in \Phi_V \\ \{w_i\} & \text{if } i \in \Phi_C \\ W & \text{if } i^X \in \Phi_V \end{cases}$$

Let r be a function from $\Im$ to f thus defined:

$$r(i^X) = \begin{cases} \emptyset & \text{if } l(i) = 1 \\ f(X, g(i^X) & \text{if } l(i) = n > 1 \end{cases}$$

Let m be a function from LS-formulas to v thus defined:

$$m(SA : i^X) =_{def} v(A, w_j) = S$$

for all $w_j \in g(i^X)$.

Lemma 1. *Let $\mathcal{B}$ be a set of LS-formulas and let i, j be labels in $\mathcal{B}$. If $(i^X, j^Y)\sigma^{\mathcal{B}}_{\mathbf{C}}$, then $g(i^X) \cap g(j^Y) \neq \emptyset$.*

This lemma, proved by induction of the length of labels, states that whenever two labels unify, their denotations have a non-null intersection (the result of their unification).

Lemma 2. *Let $\mathcal{B}$ be a set of LS-formulas and let i, j be labels in $\mathcal{B}$. If $m(SA, i)$, and $(i, j)\sigma^{\mathcal{B}}_{\mathbf{C}}$, then $m(SA, (i, j)\sigma^{\mathcal{B}}_{\mathbf{C}})$.*

According to the previous lemma if a formula has a given evaluation in a world denoted by a label, and this label unifies with another label, then the value of the formula remains unchanged in the worlds corresponding to the unification of the labels. This fact allows us to verify the correctness of the rule in a standard semantic setting, whence the following lemma.

Lemma 3. *If $\vdash_{KEM} A$, then $\models_{SFC} A$*

where, a If $\vdash_{\mathbf{CU}} A$ then $\vdash_{KEM} A$ s usual with tableau methods, $\vdash_{KEM} A$ means that the KEM-tree starting with FA, w_1 is closed.

Lemma 4. *Let Γ be a set of conditional assertions, and A be a conditional assertion. If Γ cumulatively entails A, then $\vdash_{KEM} \bigwedge \Gamma \rightarrow A$.*

Proof. We show that the inference rules and the axioms of **C** are derivable in KEM. D'Agostino and Mondadori [8] have shown that KE is sound and complete for classical propositional logic and enjoys the property of transitivity of deductions. We provide KEM-proofs for Reflexivity, Left Logical Equivalence, Right Weakening, Cautious Monotonicity and Cut.

1. $FA \mid\sim A$	w_1
2. FA	(w_2^A, w_1)
3. $\times$	(w_2^A, w_1)

Notice that closure follows from having two complementary formulas FA and A both labelled with (w_2^A, w_1).

1. $TA \mid\sim C$	w_1
2. $FB \mid\sim C$	w_1
3. TC	(W_1^A, w_1)
4. FC	(w_2^B, w_1)
5. $\times$	(w_2^B, w_1)

Here closure is obtained from $TC, (W_1^A, w_1)$ and $FC, (w_2^B, w_1)$. The labels $\sigma^{\mathcal{B}}_{C}$-unify due to the equivalence of the label formulas: by hypothesis A and B are equivalent.

1. $TA \mid\sim B$	w_1
2. $FA \mid\sim C$	w_1
3. TB	(W_1^A, w_1)
4. FC	(w_2^A, w_1)

5. $TB \rightarrow C$	(w_2^A, w_1)	6. $FB \rightarrow C$	(w_2^A, w_1)
7. TC	(w_2^A, w_1)	$\times$	
8. $\times$	(w_2^A, w_1)		

Notice that we have applied PB to $B \to C$ with respect to (w_2^A, w_1). The right branch is closed since, by hypothesis, we have already a KEM proof for $B \to C$.

Finally we present side by side the KEM proofs of Cautious Monotonicity and Cut.

1. $TA \mid\sim B$	w_1	1. $TA \wedge B \mid\sim C$	w_1
2. $TA \mid\sim C$	w_1	2. $TA \mid\sim B$	w_1
3. $FA \wedge B \mid\sim C$	w_1	3. $FA \mid\sim C$	w_1
4. TB	(W_1^A, w_1)	4. TC	$(W_1^{A\wedge B}, w_1)$
5. TC	(W_2^A, w_1)	5. TB	(W_2^A, w_1)
6. FC	$(w_2^{A\wedge B}, w_1)$	6. FC	(w_2^A, w_1)
7. $\times$	$(w_2^{A\wedge B}, w_1)$	7. $\times$	(w_2^A, w_1)

In both proofs the labels unify according to condition c) of Definition 8.

From Theorem 1, Lemmas 4 and 3 we obtain

Theorem 3. $\vdash_{KEM} A$ *iff* $\models_{SFC} A$.

and from Theorem 3 and Corollary 2

Corollary 3. *Let Γ be a set of conditional assertions. Γ cumulatively entails $A \mid\sim B$ iff* $\vdash_{KEM} \bigwedge \Gamma \to (A \mid\sim B)$

5 Proof Search with KE^+

The unification presented in section 3.2 compels us to check (label) formulas either for validity or for logical equivalence. This can be a very expensive task whose accomplishment may require us to open an auxiliary proof tree whenever we have to check either condition (see [2] for details). Fortunately, the main tree provides all the information we need so that we only have to make them available by appealing to a suitable proof method. One such method is provided by the classical system KE^+. KE^+ is based on D'Agostino and Mondadori [8]'s KE, a tableau-like proof system which employs a mixture of tableau, natural deduction and structural rules (in practice, the α-, β-, PB and PNC rules of section 3.3 restricted to the propositional part). KE^+ uses the same rules but adopts a different proof search procedure which makes it completely analytical and able to detect whether 1) a formula is either a tautology, or a contradiction, or only a satisfiable one; and 2) a sub-formula of the formula to be proved is a tautology, and to use this fact in the proof of the initial formula. The KE^+ based method consists in verifying whether the truth of the conjugate of an immediate sub-formula of a β-formula implies the truth of the other immediate sub-formula. If it is so, then we have enough information to conclude that the formula is provable. This result follows from the fact that the branch beginning with $\beta_i^C (i = 1, 2)$ contains no pair of complementary formulas leading to closure. This in turn is proved by seeing whether a formula occurs twice in a branch, and that those occurrences "depend on" appropriate formulas. This last notion is captured by the following definition.

Definition 9. *Each formula* depends on *itself. A formula B* depends on *A either if it is obtained by an application of the α-rule or it is obtained by an application of the*

KE rules on formulas depending on A. A formula C depends on A, B *if it is obtained by an application of a β-rule with A, B as its premises. The formulas obtained by an application of PB* depend on *the formula PB is applied to. If C* depends on A, B *then C* depends on A *and C* depends on B.

We go now to the proof search, but first we need some terminology. us define

An α-formula is *analysed* in a branch when both α_1 and α_2 are in the branch. A β-formula is *analysed* in a branch when either 1) if β_1^C is in the branch also β_2 is in the branch, or 2) if β_2^C is in the branch also β_1 is the branch. A β formula will be said *fulfilled i*n a branch if: 1) either β_1 or β_2 occurs in the branch provided they depend on β, or 2) either β_1 or β_2 is obtained from applying PB on β.

A branch is *E-completed* if all the formulas occurring in it are analysed. A branch is *completed* if it is E-completed and all the β-formulas occurring in it are fulfilled. A branch is *closed* if it ends with an application of PNC. A tree is *closed* if all its branches are closed.

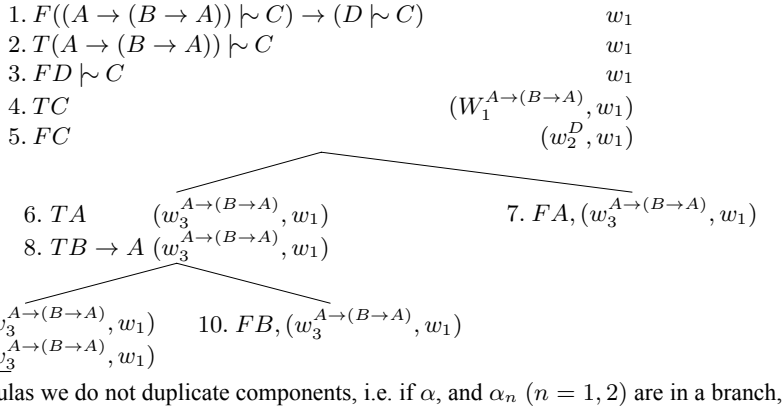
A branch obtained by applying PB to a β-formula with β_i^C as its root is a β^C*-branch*. Each branch generated by an application of PB to a formula occurring in a β^C-branch is a β^C-branch. A branch generated by an application of PB which is not a β^C-branch is a β-branch. A branch is a $\top$-branch if it contains only formulas which are equivalent to $\top$ and the formulas depending on them.

The proof search procedure starts with the formula to be proved; then 1) it selects a β^C-branch ϕ which is not yet completed and which is the β^C-branch with respect to the greatest number of formulas; 2) if ϕ is not E-completed, it expands ϕ by means of the α- and β-rules until it becomes E-completed;[1] 3) if the resulting branch is neither completed nor closed then it selects a β-formula which is not yet fulfilled in the branch — if possible a β-formula resulting from step 2 — then it applies PB with β_1, β_1^C (or, equivalently β_2, β_2^C), and then it returns to step 1; otherwise it returns to step 1.

Theorem 4. *For a formula A, $A \equiv \top$ if either:*

1. *in one of the β^C-branches it generates there is an LS-formula which appears twice, and one occurrence depends on $\beta_i^C, i \in \{1, 2\}$, and the other depends on β, or*
2. *each β^C-branch is closed and the β-branches are $\top$-branches, or*
3. *each β^C-branch is a $\top$-branch.*

We provide an illustration of the above notion by proving $((A \to (B \to A)) \mid\sim C) \to (D \mid\sim C)$.



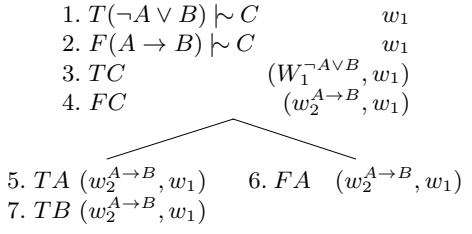
[1] For α-formulas we do not duplicate components, i.e. if α, and α_n $(n = 1, 2)$ are in a branch, then we add to the branch only α_{3-n}.

Here we have to see whether the labels in 4 and 5 unify. However the label formulas of W_1 is of type β and it is not yet analysed in the tree. So we apply PB. Notice that the left branch is a β^C-branch w.r.t. the label formula. We then apply a β rule, and we obtain another β formula. According to the proof search we have to apply again PB and then we have another application of a β rule. At this point we have two occurrences of TA with the right dependencies. So the label formula $A \to (B \to A)$ is $\top$, and the label of 4 and 5 unify, thus closing the tree.

Definition 10. *Let v be a function which maps each formula A into a set of (atomic) formulas in such a way that 1) if A is atomic, then $v(A) = \{A\}$; 2) if A is of type α, then $v(A) = v(\alpha_1) \cup v(\alpha_2)$; 3) if A is of type β, then $v(A) = v(\beta_n^C) \cup v(\beta_{3-n})$ or $v(A) = v(\beta_n)$, $n = 1, 2$. A set S of (atomic) formulas is said to* v-fulfils *a formula A iff $S = v(A)$.*

Corollary 4. *Two formulas A, B are equivalent iff each set of (atomic) formulas which v-fulfils A v-fulfils B.*

The following proof is provided as an illustration of the use of the above notions.



Obviously $\{TA, TB\}$ and $\{FA\}$ v-fulfil both $\neg A \vee B$ and $A \to B$. Accordingly, $(W_1^{\neg A \vee B}, w_1)$ and $(w_2^{A \to B}, w_1)$ σ_C^B-unify, thus closing the tree.

Remark 1. It is worth noting that Theorem 4 provides also completeness of KE^+ for classical propositional logic. This is enough for the tautology test required by Definition 2. It is not necessary to extend it to the whole **C**, since the label formulas are always classical. The same holds for the equivalence test and Corollary 4.

6 Comparison with Other Works

Groeneboer and Delgrande [13] present a method for constructing Kripke models for CLs which generalizes Hughes and Cresswell's [14] classical method of semantic tableau diagrams for the modal logic **S4.3** to Delgrande's [7] conditional logic **N**. This extension is made possible by the correspondence between **S4.3** and **N**. However, as Boutilier [4] has shown, **N** fails to validate the rule of Cautious Monotonicity, and thus it lies outside the scope of Gabbay's [10] minimal conditions for nonmonotonic consequence relations. Lamarre [17] takes a more direct approach by relying on Lewis' [19] system of spheres models. However, his method does not cover **CU**. Moreover, as proof systems for CL, the systems just mentioned can be said to suffer of all well-known computational drawbacks of the tableau method.

As far as the computational complexity of the method we have proposed is concerned, it lies in the same class as Lehmann's algorithm [18]. In fact it easy to prove that the complexity of σ is linear. However this is not the case with $\sigma^{\mathcal{B}}_{\mathbf{C}}$ in so far as it requires either tautology or equivalence or entailment tests on label formulas. Thus its absolute complexity is exponential. Nevertheless, when $\sigma^{\mathcal{B}}_{\mathbf{C}}$ is used in a KEM-proof its complexity weight does not cause any harm to the complexity of the proof since, as we have seen, the tests are performed in the proof itself. Therefore the complexity of $\sigma^{\mathcal{B}}_{\mathbf{C}}$ with respect to KEM-proofs turn out to be the same of σ. From this it follows that the complexity of the KEM-proofs for $\mathbf{C}$ is just the complexity of the propositional modulo (see [8] for a discussion). This is a well known result (see [18]), but we believe that the present approach offers some advantages over Lehmann's [18] algorithm in that it is deterministic and works for cumulative logics in general and not only for those cases where they coincide with the preferential ones.

Although their primary aim is not automated deduction, Crocco and Fariñas [6] present a sequent system for $\mathbf{CU}$ which turns out to be very similar to ours. In their approach the cut rule is replaced by more restricted rules for identifying formulas in deduction. Deductive contexts and restrictions on the transitivity of the deduction relation are represented at the level of auxiliary sequents, i.e., sequents involving a non-transitive deduction relation. Accordingly, structural and logical operations are performed both on this level and on the level of the principal (transitive) relation. The deductive context is fixed by a prefixing rule in the antecedents of auxiliary sequents. Augmentation and reduction rules in such antecedents allow us to identify those deductive contexts which are identical or compatible with other contexts, thus providing criteria for substituting conditional antecedents by conditional antecedents. In the present approach conditional antecedents are fixed by the inference rules at the "auxiliary" level of label formulas, whereas the notion of compatible contexts—or of criteria for antecedent identification—is captured by the label unification rule. Structural and logical operations are performed both at the "principal" level of labelled formulas and at the "auxiliary" level of label formulas, the only deduction relation involved being the transitive one. Thus our approach can be said to perform what Crocco and Fariñas call an "extra-logical" control on the composition of proofs in the sense that the restrictions on the transitivity of the deduction relation are represented at the "auxiliary" level of our labelling scheme. This can be seen as an advantage of our method over Crocco and Fariñas's as it allows to treat a wide range of CLs by providing different constraints, closely related to the appropriate semantic conditions, on the respective unifications (see [2]). Moreover, it do so without banishing the cut rule, thus avoiding the problems arising from defining connectives in the absence of such a rule.

Acknowledgments. This work was partially supported by the Australia Research Council under Large Grant No. A49803544.

References

1. Alberto Artosi, Paola Benassi, Guido Governatori, Antonino Rotolo. Labelled Proofs for Quantified Modal Logic. In J.J. Alferes, L. M. Pereira, and E. Orlowska (eds.) *Logics in Artificial Intelligence*, 70–86, Springer-Verlag, Berlin, 1996.

2. Alberto Artosi and Guido Governatori. A Tableau Methodology for Deontic Conditional Logic. In *Deon'98. 4^{th} International Workshop on Deontic Logic in Computer Science*, Bologna, January 8–10, 1998: 75–91.
`http://arXiv.org/abs/cs.LO/0003050`.
3. Alberto Artosi, Guido Governatori, and Giovanni Sartor. Towards a Computational Treatment of Deontic Defeasibility. In M.A. Brown and J. Carmo (eds.), *Deontic Logic, Agency and Normative Systems*, Springer-Verlag, Berlin, 1996: 27–46.
4. Craig Boutilier. Conditional Logics of Normality: a Modal Approach. *Artificial Intelligence*, 68: 87–154, 1994.
5. Brian Chellas. Basic Conditional Logic. *Journal of Philosophical Logic*, 4: 133–153, 1975.
6. G. Crocco and L. Fariñas del Cerro. Structure, Consequence Relation and Logic. In D. Gabbay (ed.), *What is a Logical System*, Oxford UP, Oxford, 1994. 375–393, 1994.
7. James P. Delgrande. A First-Order Conditional Logic for Prototypical Properties. *Artificial Intelligence*, 33:105–139, 1987.
8. Marcello D'Agostino and Marco Mondadori. The Taming of the Cut. *Journal of Logic and Computation*, 4: 285–319, 1994.
9. L. Fariñas del Cerro, A. Herzig, and J. Lange. From Ordering-Based Nonmonotonic Reasoning to Conditional Logics. *Artificial Intelligence*, 66: 375–393, 1994.
10. Dov. M. Gabbay. Theoretical Foundations for Nonmonotonic Reasoning in Expert Systems. In K. R. Apt, ed., *Proc of the NATO Advanced Study Institute on Logics and Concurrent Systems*, 439–457, 1985, Berlin, Springer-Verlag, 1985.
11. Dov M. Gabbay. *Labelled Deductive Systems*. Oxford University Press, Oxford, 1996.
12. Dov M. Gabbay and Guido Governatori. Dealing with Label Dependent Deontic Modalities. In P. McNamara and H. Prakken (eds.), *Norms, Logics and Information Systems*, IOS Press, Amsterdam, 1998: 311–330.
13. Chris Groeneboer and James P. Delgrande. Tableau-Based Theorem Proving in Normal Conditional Logics. In *AAAI'88*, volume i, 171–176, 1988.
14. G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, 1968.
15. Hirofumi Katsuno and Ken Satoh. A Unified View of Consequence Relation, Belief Revision, and Conditional Logic. In G. Crocco, L. Fariñas del Cerro, A. Herzig (eds.) *Conditionals: From Philosophy to Computer Science*, Oxford University Press, 33–66, 1995.
16. Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic Reasoning, Preferential Models and Cumulative Logics. *Artificial Intelligence*, 44: 167–207, 1990.
17. Philippe Lamarre. A promenade from monotonicity to non-monotonocity following a theorem prover. In *Principles of Knowledge Representation and Reasoning (KR'92)*, 572–580, San Mateo (Ca), 1992, Morgan Kaufman Publishers.
18. Daniel Lehmann. What Does a Conditional Base Entail? In R. J. Brachman, H. J. Levesque, and R. Reiter (eds), *Proceedings of Knowledge Representation and Reasoning, (KR'89)*, Morgan Kaufmann Publishers, S. Mateo (Ca), 1989: 212–222.
19. David Lewis. *Counterfactuals*. Basil Blackwell, Oxford, 1986.
20. Yoav Shoham. A Semantical Approach to Nonmonotonic Logics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* Morgan Kaufmann Publ., Los Altos, CA, 1987: 388–392. Reprinted in M. L. Ginsberg (ed.), *Reading in Nonmonotonic Reasoning*, Morgan Kaufmann Publ., Los Altos, CA, 1987: 227–250.
21. Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, Berlin, 1968.

A Tableau System for Gödel-Dummett Logic Based on a Hypersequent Calculus

Arnon Avron

Computer Science department
Tel Aviv University, Tel Aviv 69978, Israel
aa@math.tau.ac.il

Abstract. We present a terminating contraction-free calculus GLC^* for the propositional fragment of Gödel-Dummett Logic LC. GLC^* uses hypersequents, and unlike other Gentzen-type calculi for LC, all its rules have at most two premises. These rules are all invertible. Hence it can be used as a basis for a deterministic tableau system for LC. This tableau system is presented in the last section.

1 A Review of *LC* and *GLC*

In [7] Gödel introduced a sequence $\{G_n\}$ of n-valued logics, as well as an infinite-valued matrix G_ω in which all the G_ns can be embedded. He used these matrices to show some important properties of intuitionistic logic. The logic of G_ω was later axiomatized by Dummett in [5] and is known since then as Dummett's LC. It probably is the most important intermediate logic, one that turns up in several places, like the provability logic of Heyting's Arithmetics ([10]), relevance logic ([4]), and recently fuzzy logic([8]).

Semantically LC corresponds to linearly ordered Kripke structures. It also corresponds of course to G_ω, which is the matrix $\langle N \cup \{t\}, \leq, \rightarrow, \neg, \vee, \wedge\rangle$ where $\leq$ is the usual order on N extended by a maximal element t, the interpretation of the propositional constant $\bot$ is the number 0, $a \rightarrow b$ is t if $a \leq b$ and b otherwise, $\neg a$ is simply $a \rightarrow 0$, and $\wedge$ and $\vee$ are, respectively, the min and max operations on $\langle N \cup \{t\}, \leq\rangle$.[1] The consequence relation $\vdash_{LC}$ is defined as follows: $A_1, \ldots, A_n \vdash_{LC} B$ iff $\min\{v(A_1), \ldots, v(A_n)\} \leq v(B)$ for every valuation v in G_ω.[2] This is equivalent ([2], p.236) to taking t as the only designated element and defining: $A_1, \ldots, A_n \vdash_{LC} B$ iff for every v in G_ω either $v(B) = t$ or $v(A_i) \neq t$ for some $1 \leq i \leq n$.

A Hilbert-type axiomatization for LC can be obtained from intuitionistic logic by adding to it the axiom $(A \rightarrow B) \vee (B \rightarrow A)$ ([5]).

A cut-free Gentzen-type formulation for LC was first given by Sonobe in [9]. His approach has been improved in [1] and [6], where terminating, contraction-free (and cut-free) versions have been presented. All those systems have, however,

[1] For the application as a fuzzy logic it is more useful (see [8]) to use instead of $N \cup \{t\}$ the real interval [0,1], with 1 playing the role of t. This makes a difference only when we consider inferences from infinite theories.

[2] As usual, in case $n = 0$ the "minimal element" is taken to be t.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 98–111, 2000.

the serious drawback of using a rule with arbitrary number of premises, all of which contain formulae which are essential for the inference (strictly speaking, this is not really a single rule, but an infinite set of rules). A cut-free formulation of *LC* which does not have this drawback, and unlike other formulations has exactly the same *logical* rules as the standard formulation of Intuitionistic Logic, has been given in [2]. This formulation, which we now review, uses single-conclusion hypersequents rather than ordinary sequents.

Definition 1. *A (single-conclusion) hypersequent is a structure of the form:*

$$\Gamma_1 \Rightarrow A_1 \mid \Gamma_2 \Rightarrow A_2 \mid \cdots \mid \Gamma_n \Rightarrow A_n$$

where $\Gamma_i \Rightarrow A_i$ is an ordinary single-conclusion sequent (Note that we do not allow components with an empty conclusion, although it is easy to add such if so one desires).

We use G, H as variables for (possibly empty) hypersequents, S for sequents.

Definition 2. *The System GLC.*

Axioms

$$A \Rightarrow A \qquad\qquad \bot \Rightarrow A$$

Standard External Structural Rules

$$\frac{G}{G|H} \qquad \frac{G|S|S|H}{G|S|H} \qquad \frac{G|S_1|S_2|H}{G|S_2|S_1|H}$$

(External weakening, contraction and permutation, respectively).

Standard Internal Structural Rules

$$\frac{G|\Gamma_1, A, B, \Gamma_2 \Rightarrow D|H}{G|\Gamma_1, B, A, \Gamma_2 \Rightarrow D|H}$$

$$\frac{G|\Gamma, A, A \Rightarrow D|H}{G|\Gamma, A \Rightarrow D|H}$$

$$\frac{G|\Gamma \Rightarrow D|H}{G|\Gamma, A \Rightarrow D|H}$$

$$\frac{G_1|\Gamma_1 \Rightarrow A|H_1 \qquad G_2|A, \Gamma_2 \Rightarrow D|H_2}{G_1|G_2|\Gamma_1, \Gamma_2 \Rightarrow D|H_1|H_2}$$

(Internal permutation, contraction, weakening, and cut, respectively).

Special Structural Rules

$$\frac{G|\Gamma_1, \Gamma_2 \Rightarrow D|H}{G|\Gamma_1 \Rightarrow D|\Gamma_2 \Rightarrow D|H}$$

$$\frac{G_1|\Gamma_1 \Rightarrow A_1|H_1 \qquad G_2|\Gamma_2 \Rightarrow A_2|H_2}{G_1|G_2|\Gamma_1 \Rightarrow A_2|\Gamma_2 \Rightarrow A_1|H_1|H_2}$$

(Splitting (S) and Communication (Com), respectively).

Logical Rules

$$\frac{G_1|\Gamma_1 \Rightarrow A|H_1 \qquad G_2|B,\Gamma_2 \Rightarrow D|H_2}{G_1|G_2|A \to B,\Gamma_1,\Gamma_2 \Rightarrow D|H_1|H_2} \qquad \frac{G|\Gamma,A \Rightarrow B|H}{G|\Gamma \Rightarrow A \to B|H}$$

(and similar hypersequential versions of the intuitionistic rules for $\vee$ and $\wedge$).

Definition 3. *The interpretation of a standard* sequent $A_1, A_2, \ldots, A_n \Rightarrow B$ *is* $A_1 \to (A_2 \to \cdots \to (A_n \to B)\cdots)$. *The interpretation of a hypersequent* $\Gamma_1 \Rightarrow A_1|\cdots|\Gamma_n \Rightarrow A_n$ *is* $\varphi_{\Gamma_1 \Rightarrow A_1} \vee \cdots \vee \varphi_{\Gamma_n \Rightarrow A_n}$, *where for each* i $\varphi_{\Gamma_i \Rightarrow A_i}$ *is the interpretation of* $\Gamma_i \Rightarrow A_i$.

Theorem 1. *The cut elimination theorem obtains for GLC.*

Theorem 2. *A hypersequent is provable in GLC iff its interpretation is valid in* G_ω *(iff it is provable in Dummett's system LC).*

The proofs of both theorems can be found in [2].

GLC, despite its elegance, is not very convenient for proof search. The main reason is that some of its rules are not invertible. In the next section we present a calculus which is based on *GLC*, and which does not have this defect. This calculus, like those of [1] and [6], is terminating, cut-free, and contraction-free. Hence it can be used as a basis for more efficient tableau system for *LC*. This tableau system is presented in section 3.

2 The System *GLC**

The following theorem provides the key to a cut-free version of *GLC* in which all the logical rules are invertible.

Theorem 3.

$(\Rightarrow \wedge)$ $\quad \vdash_{GLC} G|\Gamma \Rightarrow \varphi \wedge \psi$ *iff* $\vdash_{GLC} G|\Gamma \Rightarrow \varphi$ *and* $\vdash_{GLC} G|\Gamma \Rightarrow \psi$

$(\wedge \Rightarrow)$ $\quad \vdash_{GLC} G|\Gamma,\varphi \wedge \psi \Rightarrow \theta$ *iff* $\vdash_{GLC} G|\Gamma,\varphi,\psi \Rightarrow \theta$

$(\Rightarrow \vee)$ $\quad \vdash_{GLC} G|\Gamma \Rightarrow \varphi \vee \psi$ *iff* $\vdash_{GLC} G|\Gamma \Rightarrow \varphi|\Gamma \Rightarrow \psi$

$(\vee \Rightarrow)$ $\quad \vdash_{GLC} G|\Gamma,\varphi \vee \psi \Rightarrow \theta$ *iff* $\vdash_{GLC} G|\Gamma,\varphi \Rightarrow \theta$ *and* $\vdash_{GLC} G|\Gamma,\psi \Rightarrow \theta$

$(\Rightarrow \to)$ $\quad \vdash_{GLC} G|\Gamma \Rightarrow \varphi \to \psi$ *iff* $\vdash_{GLC} G|\Gamma,\varphi \Rightarrow \psi$

$(\to \wedge)$ $\quad \vdash_{GLC} G|\Gamma,\varphi \to \psi_1 \wedge \psi_2 \Rightarrow \theta$ *iff* $\vdash_{GLC} G|\Gamma,\varphi \to \psi_1,\varphi \to \psi_2 \Rightarrow \theta$

$(\wedge\rightarrow)$ $\vdash_{GLC} G|\Gamma, \varphi_1 \wedge \varphi_2 \rightarrow \psi \Rightarrow \theta$ *iff*

$\vdash_{GLC} G|\Gamma, \varphi_1 \rightarrow \psi \Rightarrow \theta$ and $\vdash_{GLC} G|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta$

$(\rightarrow\vee)$ $\vdash_{GLC} G|\Gamma, \varphi \rightarrow \psi_1 \vee \psi_2 \Rightarrow \theta$ *iff*

$\vdash_{GLC} G|\Gamma, \varphi \rightarrow \psi_1 \Rightarrow \theta$ and $\vdash_{GLC} G|\Gamma, \varphi \rightarrow \psi_2 \Rightarrow \theta$

$(\vee\rightarrow)$ $\vdash_{GLC} G|\Gamma, \varphi_1 \vee \varphi_2 \rightarrow \psi \Rightarrow \theta$ *iff* $\vdash_{GLC} G|\Gamma, \varphi_1 \rightarrow \psi, \varphi_2 \rightarrow \psi \Rightarrow \theta$

$(\rightarrow_{(}\rightarrow)$ $\vdash_{GLC} G|\Gamma, \varphi \rightarrow (\psi_1 \rightarrow \psi_2) \Rightarrow \theta$ *iff*

$\vdash_{GLC} G|\Gamma, \varphi \rightarrow \psi_2 \Rightarrow \theta$ and $\vdash_{GLC} G|\Gamma, \psi_1 \rightarrow \psi_2 \Rightarrow \theta$

$(\rightarrow_{)}\rightarrow)$ $\vdash_{GLC} G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta$ *iff*

$\vdash_{GLC} G|\Gamma, \psi \Rightarrow \theta$ and $\vdash_{GLC} G|\varphi_1 \Rightarrow \varphi_2|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta$

$(\rightarrow\Rightarrow)$ $\vdash_{GLC} G|\Gamma, p \rightarrow q \Rightarrow r$ *iff*

$\vdash_{GLC} G|\Gamma \Rightarrow r|p \rightarrow q \Rightarrow p$ and $\vdash_{GLC} G|\Gamma, q \Rightarrow r$

Proof. We provide formal derivations in GLC for $(\rightarrow_{)}\rightarrow)$ and $(\rightarrow\Rightarrow)$ (the other cases are very easy). These derivations employ cuts, and sequents which are intuitionistically valid (like $(\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \varphi_2 \rightarrow \psi$ in the first derivation) are used in them as if they were axioms (i.e.: their easy proofs are omitted).

– The 'if' part of $(\rightarrow_{)}\rightarrow)$:

$$\dfrac{\dfrac{\dfrac{G|\varphi_1 \Rightarrow \varphi_2|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta}{G| \Rightarrow \varphi_1 \rightarrow \varphi_2|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta} \qquad G|\Gamma, \psi \Rightarrow \theta}{G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta} \qquad (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \varphi_2 \rightarrow \psi}{G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta}$$
$$G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta$$

– The "only if" part of $(\rightarrow_{)}\rightarrow)$:

(i) $$\dfrac{\psi \Rightarrow (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \qquad G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta}{G|\Gamma, \psi \Rightarrow \theta}$$

(ii) (S) $$\dfrac{\dfrac{\dfrac{\dfrac{\varphi_1, \varphi_1 \rightarrow \varphi_2 \Rightarrow \varphi_2}{\varphi_1 \Rightarrow \varphi_2|\varphi_1 \rightarrow \varphi_2 \Rightarrow \varphi_2} \qquad \psi \Rightarrow \psi}{\varphi_1 \Rightarrow \varphi_2|\varphi_1 \rightarrow \varphi_2, \varphi_2 \rightarrow \psi \Rightarrow \psi}}{\varphi_1 \Rightarrow \varphi_2|\varphi_2 \rightarrow \psi \Rightarrow (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi} \qquad G|\Gamma, (\varphi_1 \rightarrow \varphi_2) \rightarrow \psi \Rightarrow \theta}{G|\varphi_1 \Rightarrow \varphi_2|\Gamma, \varphi_2 \rightarrow \psi \Rightarrow \theta}$$

– The "if" part of $(\rightarrow\Rightarrow)$:

$$\dfrac{\dfrac{G|\Gamma \Rightarrow r|p \rightarrow q \Rightarrow p \qquad G|\Gamma, q \Rightarrow r}{G|\Gamma \Rightarrow r|\Gamma, p \rightarrow q, p \rightarrow q \Rightarrow r}}{\dfrac{G|\Gamma, p \rightarrow q \Rightarrow r|\Gamma, p \rightarrow q \Rightarrow r}{G|\Gamma, p \rightarrow q \Rightarrow r}}$$

– The "only if" part of $(\rightarrow\Rightarrow)$:

(i) $$\dfrac{q \Rightarrow p \rightarrow q \qquad G|\Gamma, p \rightarrow q \Rightarrow r}{G|\Gamma, q \Rightarrow r}$$

(ii) $$\dfrac{\dfrac{\dfrac{\dfrac{p, p \rightarrow q \Rightarrow q}{p \Rightarrow q|p \rightarrow q \Rightarrow q} \qquad p \Rightarrow p}{p \Rightarrow q|p \Rightarrow q|p \rightarrow q \Rightarrow p}}{p \Rightarrow q|p \rightarrow q \Rightarrow p}}{\dfrac{\Rightarrow p \rightarrow q|p \rightarrow q \Rightarrow p \qquad G|\Gamma, p \rightarrow q \Rightarrow r}{G|\Gamma \Rightarrow r|p \rightarrow q \Rightarrow p}}$$

Remarks

1. The last derivation starts with an application of (S) followed by an application of (Com). It ends with a Cut.
2. Because of the soundness of GLC, our proof shows that Theorem 3 also obtains if instead of provability in GLC we talk about validity in G_ω. It is not difficult to prove this directly, and then use the soundness *and* completeness of GLC to derive Theorem 3. Our proof here will allow us, however, to provide later a new proof of the completeness of GLC.
3. Although $(\rightarrow\Rightarrow)$ is valid for all formulas p, q and r, we shall use it only in case p, q and r are atomic (the notation we use was chosen accordingly). In fact, we'll use it only if in addition $p \neq q$, $p \neq \bot$ and $p \neq r$. In the other cases we can use instead the following obvious reductions:

Lemma 1.

$\vdash_{GLC} G|\Gamma, \bot \rightarrow \psi \Rightarrow \theta$ iff $\vdash_{GLC} G|\Gamma \Rightarrow \theta$
$\vdash_{GLC} G|\Gamma, \psi \rightarrow \psi \Rightarrow \theta$ iff $\vdash_{GLC} G|\Gamma \Rightarrow \theta$
$\vdash_{GLC} G|\Gamma, p \rightarrow q \Rightarrow p$ iff $\vdash_{GLC} G|\Gamma \Rightarrow p|p \rightarrow q \Rightarrow p$

Definition 4. *A* basic *hypersequent is a hypersequent every component of which has one of the following two forms:*

1. $\Gamma \Rightarrow p$, *where* p *and all elements of* Γ *are atomic.*
2. $p \rightarrow q \Rightarrow p$, *where* p *and* q *are atomic, and* $p \neq q$, $p \neq \bot$.

Theorem 4. *For every hypersequent* G *we can effectively find a set* $\mathcal{B}$ *of basic hypersequents, so that* $\vdash_{GLC} G$ *iff* $\vdash_{GLC} H$ *for every* $H \in \mathcal{B}$.

Proof. This follows easily from Theorem 3 and Lemma 1.

To get an effective decision procedure for GLC, we have to determine which basic hypersequents are provable in it. This is equivalent to determining which basic hypersequents are valid in G_ω. We next describe an algorithm for deciding this question.

Algorithm BH Let G be a basic hypersequent. Let $p_1, \ldots, p_n$ be the atomic propositional variables which occur in G, and let $x_1, \ldots, x_n$ be n corresponding variables for natural numbers. Construct a set C_G of constraints on $x_1, \ldots, x_n$ as follows:

- $(x_i > 0) \in C_G$ whenever $\Gamma \Rightarrow \bot$ is a component of G and $p_i \in \Gamma$
- $(x_i < 0) \in C_G$ whenever $\Gamma \Rightarrow p_i$ is a component of G and $\bot \in \Gamma$
- $(x_i \leq 0) \in C_G$ whenever $p_i \to \bot \Rightarrow p_i$ is a component of G
- $(x_j < x_i) \in C_G$ whenever $\Gamma \Rightarrow p_j$ is a component of G and $p_i \in \Gamma$
- $(x_j \leq x_i) \in C_G$ whenever $p_j \to p_i \Rightarrow p_j$ is a component of G.

Determine now whether there exist $y_1, \ldots, y_{k+1} \in \{x_1, \ldots, x_n\}$ $(k \geq 0)$ such that $y_1, \ldots, y_k$ are all distinct, $(y_k > y_{k+1}) \in C_G$ (where we take y_0 to be 0 in case $k = 0$), and one of the following holds:

- y_1 is identical with y_{k+1} and for every $1 \leq i \leq k-1$, $(y_i \geq y_{i+1}) \in C_G$ or $(y_i > y_{i+1}) \in C_G$.
- $(y_i \geq y_{i+1}) \in C_G$ for all $0 \leq i \leq k-1$, where again we take y_0 to be 0.

If such a sequence exists then G is valid. Otherwise it is not.

Note. The last step (after constructing C_G) is simply a description of how we determine whether or not a given set of constraints of the forms $x_i \geq x_j, x_i > x_j$, $x_i > 0$, $x_i < 0$, $x_i \leq 0$ is satisfiable in the natural numbers (If C_G is satisfiable then G is not valid, otherwise it is valid).

To prove the correctness of Algorithm BH, we need the next definition and theorem.

Definition 5. *A valuation v in G_ω is called simple if $v(p) \in N$ (i.e.: $v(p) \neq t$ for every atomic variable p).*

Theorem 5. *A sentence φ, or a hypersequent G, is valid in G_ω iff there is no simple valuation that refutes it.*

Proof. Since G is valid iff its interpretation φ_G is valid, it suffices to prove the theorem for a sentence φ. So assume that φ is not valid, and let v be a valuation in G_ω which refutes it (i.e. $v(\varphi) \neq t$). Let n be a natural number such that $v(p) \in \{0, 1, \ldots, n, t\}$ for every atomic variable p of φ. Define:

$$v'(p) = \begin{cases} v(p) & v(p) \neq t \\ n+1 & v(p) = t \end{cases}$$

It is easy to prove by structural induction, that if ψ is a subformula of φ, then $v'(\psi) \in \{n+1, t\}$ in case $v(\psi) = t$, and $v'(\psi) = v(\psi)$ otherwise. In particular $v'(\varphi) = v(\varphi) \neq t$, and so v' is a simple valuation which refutes φ.

Theorem 6. *Algorithm BH is correct.*

Proof. This easily follows from Theorem 5 and the following three facts:

1. If Γ consists of atomic formulas then a simple valuation v refutes $\Gamma \Rightarrow p$ if $v(p) < v(q)$ for all $q \in \Gamma$ (in particular: a simple v refutes $q \Rightarrow p$ iff $v(p) < v(q)$).
2. If p, q are atomic and $p \neq \bot$ then a simple valuation v is a refutation of $p \to q \Rightarrow p$ iff $v(p) \leq v(q)$.

3. A hypersequent is refutable by v iff v refutes all its components.

Definition 6. *A generalized axiom is a basic hypersequent of the form* $\varphi_1 \Rightarrow \psi_1|\varphi_2 \Rightarrow \psi_2|\ldots|\varphi_n \Rightarrow \psi_n$ *such that the set of constraints which corresponds to it is a minimal unsatisfiable set.*

Note. If we ignore the order of the components in a hypersequent (which we can do), then a basic sequent is a generalized axiom iff it has one of the following two forms:

1. $$p_1 \prec p_2|p_2 \prec p_3|\cdots|p_{n-1} \prec p_n|p_n \Rightarrow p_1$$
   where $n \geq 1$, $p_1, \ldots, p_n$ are n distinct propositional variables, and for all $1 \leq i \leq n-1$, "$p_i \prec p_{i+1}$" is either $p_i \Rightarrow p_{i+1}$ or $(p_{i+1} \to p_i) \Rightarrow p_{i+1}$.
2. $$(p_1 \to \bot) \Rightarrow p_1|(p_2 \to p_1) \Rightarrow p_2|\cdots|(p_{n-1} \to p_{n-2}) \Rightarrow p_{n-1}|p_{n-1} \Rightarrow p_n$$
   where $n \geq 1$, and $p_1, \ldots, p_n$ are n distinct propositional variables (in the case $n = 1$ we take p_0 to be $\bot$).

It is easy to see that a simple v can refute a hypersequent of the first type iff $v(p_1) < v(p_1)$, and of the second type iff $v(p_n) < 0$. Hence such v cannot exist.

Here is the list (up to order of components and names of variables) of all the generalized axioms which use at most 3 different variables:

$$\bot \Rightarrow \bot$$
$$p \Rightarrow p \qquad \bot \Rightarrow p$$
$$p \Rightarrow q|q \Rightarrow p$$
$$(p \to q) \Rightarrow p|p \Rightarrow q$$
$$(p \to \bot) \Rightarrow p|p \Rightarrow q$$
$$p \Rightarrow q|q \Rightarrow r|r \Rightarrow p$$
$$p \Rightarrow q|(r \to q) \Rightarrow r|r \Rightarrow p$$
$$q \to p \Rightarrow q|q \Rightarrow r|r \Rightarrow p$$
$$(q \to p) \Rightarrow q|(r \to q) \Rightarrow r|r \Rightarrow p$$
$$(q \to \bot) \Rightarrow q|(r \to q) \Rightarrow r|r \Rightarrow p$$

We are now ready to introduce our new calculus of hypersequents for LC:

The System GLC^*

Axioms: Every basic sequent which can be derived from some generalized axiom using (internal and external) weakenings and permutations.
Rules: The rules that correspond to the "if" parts of Theorem 3.

Theorem 7. *A sequent* G *is valid iff* $\vdash_{GLC^*} G$.

Proof. It can easily be shown, using theorem 6 (and the description of algorithm BH) that a basic sequent is valid iff it can be derived from some generalized axiom using weakenings and permutations (both internal and external). Hence the theorem follows from Theorem 3 and Lemma 1.

Corollary 1.

1. *The cut rule is admissible in GLC^*.*
2. *A formula φ is valid in LC iff $\Rightarrow \varphi$ has a proof in GLC^*.*

Note. As hinted above, Theorem 7 can be used to give a new proof of the completeness of GLC (with cuts). Because of Theorem 3, all we need to do is to show that every generalized axiom is derivable in GLC. This is easy (using an induction on the number of propositional variables involved). As an example, we show how $(p_2 \to p_1) \Rightarrow p_2 | (p_3 \to p_2) \Rightarrow p_3 | p_3 \Rightarrow p_1$ can be derived from $p_2 \to p_1 \Rightarrow p_2 | p_2 \Rightarrow p_1$:

$$\text{(S)}\ \dfrac{\dfrac{p_3 \Rightarrow p_3 \qquad\qquad p_2 \to p_1 \Rightarrow p_2 | p_2 \Rightarrow p_1}{p_2 \to p_1 \Rightarrow p_2 | p_3, p_3 \to p_2 \Rightarrow p_1}}{}$$

$$\text{(Com)}\ \dfrac{\dfrac{p_2 \to p_1 \Rightarrow p_2 | p_3 \Rightarrow p_1 | p_3 \to p_2 \Rightarrow p_1 \qquad\qquad p_3 \Rightarrow p_3}{p_2 \to p_1 \Rightarrow p_2 | p_3 \Rightarrow p_1 | p_3 \to p_2 \Rightarrow p_3 | p_3 \Rightarrow p_1}}{p_2 \to p_1 \Rightarrow p_2 | p_3 \to p_2 \Rightarrow p_3 | p_3 \Rightarrow p_1}$$

3 A Tableau System for *LC* Based on *GLC**

In order to develop a tableau system for searching for a proof in GLC^*, we represent a hypersequent G by a set S_G of signed formulas with links between them. An occurrence of $T\varphi$ in S_G means that φ occurs on the l.h.s. of at least one component of G, while an occurrence of $F\varphi$ in S_G means that φ is the r.h.s of at least one component of G. A link from $T\varphi$ to $F\psi$ means that φ occurs on the l.h.s. of a component of G in which ψ is the r.h.s. Thus every signed formula in S_G of the form $T\varphi$ is linked to at least one signed formula of the form $F\varphi$.

In our tableau system, the set of signed formulas on some branch, to which no reduction rule has yet been applied on that branch, represents (together with the links between its elements) a hypersequent that we need to prove. A branch is in principle closed if it contains a set of signed formulas (and links) which represents a substitution instance of an axiom. In practice we may apply this test only when no reduction rule may be applied on that branch (the branch represents in such a case a basic sequent, and algorithm BH can easily be applied then). Before this stage is reached we will close a branch only in some simple cases (for example: when the branch contains a formula of the form $T\bot$, or a pair $T\varphi, F\varphi$ with a link between these two formulas).

As usual, the expansion rules of our system replace some signed formula on a branch by other (usually simpler) ones. In addition they also change the set of links on that branch.

We give now the list of these expansion rules. We then describe the changes in links that each rule causes. Using the results of the previous section it is easy to show that the resulting tableau system is sound and complete.

$(F\wedge)$ $\quad F\varphi\wedge\psi \;\Big/\; F\varphi \quad F\psi$ $\qquad$ $(T\wedge)$ $\quad T\varphi\wedge\psi \;\big|\; T\varphi, T\psi$

$(F\vee)$ $\quad F\varphi\vee\psi \;\big|\; F\varphi, F\psi$ $\qquad$ $(T\vee)$ $\quad T\varphi\vee\psi \;\Big/\; T\varphi \quad T\psi$

$(F\to)$ $\quad F\varphi\to\psi \;\big|\; T\varphi, F\psi$ $\qquad$ $(T\to)$ $\quad Tp\to q \;\Big/\; Tp\to q, Fp \quad Tq$

$(T\to\wedge)$ $\quad T\varphi\to\psi\wedge\theta \;\big|\; T\varphi\to\psi, T\varphi\to\theta$ $\qquad$ $(T\wedge\to)$ $\quad T\psi\wedge\theta\to\varphi \;\Big/\; T\psi\to\varphi \quad T\theta\to\varphi$

$(T\to\vee)$ $\quad T\varphi\to\psi\vee\theta \;\Big/\; T\varphi\to\psi \quad T\varphi\to\theta$ $\qquad$ $(T\vee\to)$ $\quad T\psi\vee\theta\to\varphi \;\big|\; T\psi\to\varphi, T\theta\to\varphi$

$(T\to_(\to)$ $\quad T\psi\to(\theta\to\varphi) \;\Big/\; T\psi\to\varphi \quad T\theta\to\varphi$ $\qquad$ $(T\to_)\to)$ $\quad T(\psi\to\theta)\to\varphi \;\Big/\; T\varphi \quad T\psi, F\theta, T\theta\to\varphi$

Note. The rule $(T\to)$ should be applied only if p, q are atomic, $p\neq q$, $p\neq\bot$, and there is a formula different from Fp to which $Tp\to q$ is linked on that branch. Moreover: in practice no expansion rule should be applied to signed formulas of the forms $T\bot\to\psi$ and $T\psi\to\psi$, and such formulas should simply be ignored.

The effects of the various rules on the set of links on a given branch are as follows:

$(F\wedge)$: Every formula that was linked to $F\varphi\wedge\psi$ before the application of the rule should be linked to the new $F\varphi$ and to the new $F\psi$ after it.

Similar rules apply in all the other cases, except:

($F \to$): Every formula that was linked to $F\varphi \to \psi$ before the application of the rule should be linked to the new $F\psi$ after it. In addition a link should be added between the new $T\varphi$ and the new $F\psi$.

($T \to$): The new $Tp \to q$ should be linked to the new Fp. The new Tq should be linked to all the formulas other than Fp to which $Tp \to q$ was linked before the application of the rule.

($T(\to) \to$): The new $T\psi$ should be linked to the new $F\theta$. The new $T\varphi$ and the new $T\theta \to \varphi$ should be linked to the formulas to which $T(\psi \to \theta) \to \varphi$ was linked before the application of the rule.

As an example, we describe the stages which are involved in the construction of a tableau for $((p \to q) \to r) \to ((q \to p) \to r) \to r$. These stages are depicted in the picture below, which contains several graphs representing the states reached after these stages. In these graphs links are represented by curved arrows. Each stage is obtained from the previous one by applying an expansion rule to some formula. That formula, together with the associated links, are deleted from the graph, and so they do not appear in the graph which represents the stage which is reached after the application of that rule. If a branch is closed at some stage, the formulas on it which do not belong to any other open branch are also deleted. In more detail:

- In stage 1 the construction of the tableau begins in the usual manner (with a single node).
- In stages 2 and 3 the rule ($F \to$) is applied, still in the usual manner. $F((p \to q) \to r) \to ((q \to p) \to r) \to r$ and later $F((q \to p) \to r) \to r$ are deleted. The resulting single node of the graph represents the ordinary sequent

$$(p \to q) \to r,\ (q \to p) \to r \Rightarrow r$$

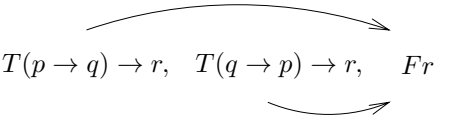
  and so there are links between its T-formulas and its single F-formula.
- In stage 4 ($T \to_{)} \to$) is applied to $T(p \to q) \to r$. This formula and its link are deleted therefore from the original node. Two new nodes, each belonging to a different branch are created. The new node on the left branch contains just Tr, which is linked to Fr in the parent node. Since this link represents an axiom, this branch is immediately closed (and deleted). The resulting graph has two nodes (the original one and the new node on the right branch) with formulas and links as shown in the picture. The two nodes together represent the hypersequent

$$q \to r,\ (q \to p) \to r \Rightarrow r \mid p \Rightarrow q$$

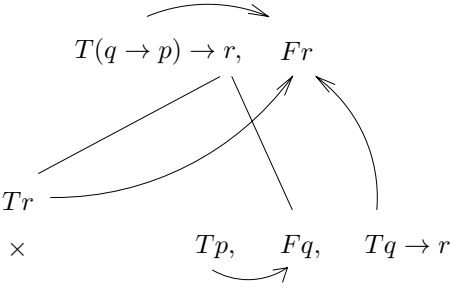
  (while the deleted right branch represents $r,\ (q \to p) \to r \Rightarrow r$).
- In stage 5 ($T \to_{)} \to$) is applied again, this time to $T(q \to p) \to r$ at the root. Again this formula and its link are deleted from the original node, two new branches are created, and this time both are immediately closed (the right branch is closed because it contains the links $Tp - Fq$ and $Tq - Fp$, which together represent the generalized axiom $p \Rightarrow q | q \Rightarrow p$).

$$F\big((p \to q) \to r\big) \to \big((q \to p) \to r\big) \to r$$

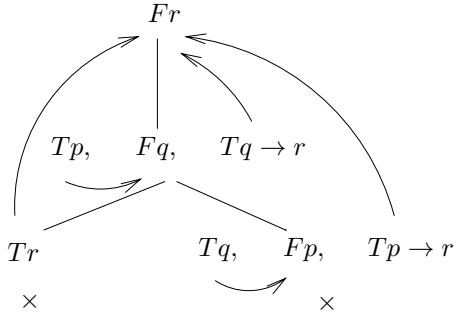
Stage 1



Stage 2 + 3



Stage 4



Stage 5

As a second example, we take the formula $\phi = \neg\neg(\neg p \vee p)$. This formula is an abbreviation of $(((p \to \bot) \vee p) \to \bot) \to \bot$. So we start with $F\phi$, and

apply $(F \to)$ followed by $(T\vee \to)$. We obtain a single node which contains $T(p \to \bot) \to \bot$, $Tp \to \bot$ and $F\bot$ (with the obvious links). It represents the sequent $(p \to \bot) \to \bot,\ p \to \bot \Rightarrow \bot$. We next apply $(\to_)\to)$ to $T(p \to \bot) \to \bot$. We get two branches. The left one contains $T\bot$ and so it is immediately closed. In the other one $T(p \to \bot) \to \bot$ is replaced by Tp, $F\bot$ (with a link between them) and $T\bot \to \bot$ (which is linked to $F\bot$ in the parent node). Finally, we apply $(T \to)$ to $Tp \to \bot$ (which is in the root). Again we get two branches. The right one contains $T\bot$ (and so it is closed). In the other branch Fp and a new $Tp \to \bot$ are added, with a link between them. This branch contains the links $Tp - F\bot$ and $Tp \to \bot - Fp$, which together represent the generalized axiom $p \to \bot \Rightarrow p \mid p \Rightarrow \bot$. Hence this branch is also closed.

What is interesting about the last example is that the formula ϕ which is proved there is intutionistically valid. It has therefore an intuitionistic cut-free proof which uses only sequents. In contrast, the proof which we just have found (which is contraction free) employs proper hypersequents!

4 Conclusion and Comparison with Other Works

We have introduced a new cut-free calculus GLC^* of hypersequents for LC. In this calculus the greatest advantage of GLC over all other known systems for LC is lost: GLC has exactly the same *logical* rules as intuitionistic logic, and the differences between the two logics is (according to GLC) only with respect to the *structural rules*. This is no longer true for GLC^*, which employs several logical rules which are not intuitionistically valid. Another nice property of GLC which is partially lost in GLC^* is the pure subformula property. In GLC^* we have only a certain analytic substitute, which is less pure and elegant. For *understanding* LC, and for reasoning about it, GLC is therefore better than GLC^*. On the other hand GLC^* is much more suitable than GLC for proof search and for developing a practical tableau version. This is due to the fact that unlike GLC, GLC^* is terminating, contraction-free, and all its rules are invertible.

It is interesting to compare GLC^* with two other systems that have recently (and independently) been introduced for the same goal.

- The system $G4 - LC$ of [6] (which is an improvement of the system in [1]) shares with GLC^* its advantages (of being cut-free, terminating, contraction-free, and with only invertible rules). Like in GLC^*, this is achieved by giving up the pure subformula property, and replacing it with the same analytic subformula property that GLC^* has.
  $G4 - LC$ has two main advantages over GLC^*: it uses ordinary sequents (rather than hypersequents), and its axioms are much simpler. On the other hand its principal rule (the one that allows the inference of the characteristic axiom of LC) does not have a fixed number of premises (so from a certain point of view it is an infinite collection of rules, not a single one). Moreover: the corresponding tableau rule requires analysing several formulas *simultaneously*, so it has a global character. In GLC^*, in contrast, all the rules are

strictly local, and the tableau system analyses, as usual, just one formula at a time. Another important shortcoming of $G4-LC$ is that its principal rule may be applied only if its conclusion cannot possibly be obtained by one of the other rules of the system (the rule will still be sound if this side condition is removed, but then it will not be invertible any more). In GLC^*, in contrast, all the rules are pure, with no side conditions[3], and they are completely independent of each other (this emphasizes again its purely local nature). We hope that this advantage will make it easier to extend GLC^* to the first-order case (something which is somewhat problematic for $G4-LC$. See [6]). This, however, should be checked in the next stage of this research.

– The system $\mathbf{R}G_\infty$ of [3] is similar to GLC^* in that it employs hypersequents rather than ordinary sequents (unlike GLC^* it allows only single-premise components, but because of the splitting rule this limitation can also be forced on GLC^*, without losing or gaining anything significant). Its main advantage over GLC^* is that it has the pure subformula property. This, however, is achieved only at the cost of using *two* types of components: $A \leq B$ and $A < B$ (with the obvious semantical interpretations in G_ω). Because of theorem 5, on the *atomic* level these two types of components correspond to the two types of components that fully split, irreducible hypersequents have in GLC^*: $p \Rightarrow q$ and $q \to p \Rightarrow q$. Indeed, $p \Rightarrow q$ is true in a valuation v iff $v(p) \leq v(q)$, while $q \to p \Rightarrow q$ is true in a *simple* valuation v iff $v(p) < v(q)$ [4]. In GLC^*, however, this separation is made only at the end of a reduction, and it is fully expressed *within* the language of LC. In $\mathbf{R}G_\infty$, in contrast, it is an essential part of the logical rules, and it is forced by introducing special symbols in the metalanguage of hypersequents. As a result, the hypersequents of $\mathbf{R}G_\infty$ cannot (as far as we can see) in general be translated into sentences of the language of LC, since $A < B$ does not have a general translation (hence the equivalence of $\mathbf{R}G_\infty$ with the standard Hilbert-type formulation of LC can be proved only via the semantics). This again is in a sharp contrast with GLC^*.

  We add that (properly interpreted as above) the axioms of $\mathbf{R}G_\infty$ and of GLC^* are practically the same. So is the number of their logical rules (both systems need four rules for each connective. It seems to me, however, that here $\mathbf{R}G_\infty$ provides a better understanding *why* do we need here four rules rather than the usual two). Also the complexity (but not the form) of the rules of the two systems seems to be similar.

[3] This is true even for the rule $(\to\Rightarrow)$, although its use in the corresponding tableau system does have a side condition.

[4] There is in fact a third type of component that fully split, irreducible hypersequents may have in GLC^*: $\Rightarrow p$. By Theorem 5 such components are irrelevant to the validity of hypersequents and can in fact be omitted (Algorithm BH ignores them anyway). They correspond to components of the form $1 \leq p$ in [3] (where 1 is used for our t).

References

1. A. Avellone, M. Ferrari, P. Miglioli. Duplication-free Tableaux Calculi Together with Cut-free and Contraction-free Sequent Calculi for the Interpolable Propositional Intermediate Logics. *Logic J. IGPL*, 7:447–480, 1999.
2. A. Avron. Using Hypersequents in Proof Systems for Non-classical Logics. *Annals of Mathematics and Artificial Intelligence*, 4:225–248, 1991.
3. M. Baaz, G.C. Fermüller. Analytic Calculi for Projective Logics. In *Proc. of TABLEAUX'99*, LNCS 1617, pp. 36–50, Springer-Verlag, Berlin, 1999.
4. J.M Dunn, R.K. Meyer. Algebraic Completeness Results for Dummett's LC and its extensions. *Z. math. Logik und Grundlagen der Mathematik*, 17:225–230, 1971.
5. M. Dummett. A Propositional Calculus with a Denumerable matrix. *Journal of Symbolic Logic*, 24:96–107, 1959.
6. R. Dyckhoff. A Deterministic Terminating Sequent Calculus for Gödel-Dummett Logic.*Logic J. IGPL*, 7:319–326, 1999.
7. K. Gödel. Zum intuitionistischen Aussagenkalkül, *Ergeb. Math. Koll.* 4:40, 1933.
8. P. Hajek. *Metamathematics of Fuzzy Logic*, Kluwer Academic Publishers, 1998.
9. O. Sonobe. A Gentzen-type Formulation of Some Intermediate Propositional Logics. *Journal of Tsuda College*, 7:7–14, 1975.
10. A. Visser. On the Completeness Principle: A study of provability in Heyting's arithmetic. *Annals of Mathematical Logic*, 22:263–295, 1982.

An Analytic Calculus for Quantified Propositional Gödel Logic*

Matthias Baaz[1], Christian Fermüller[1], and Helmut Veith[1,2]

[1] Technische Universität Wien, Karlsplatz 13, A-1040 Austria
[baaz, chrisf]@logic.at, veith@dbai.tuwien.ac.at
[2] Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract. We define a hypersequent calculus for Gödel logic enhanced with (fuzzy) quantifiers over propositional variables. We prove soundness, completeness and cut-elimination for this calculus and provide a detailed investigation of the so-called Takeuti-Titani rule which expresses density of the ordering of truth values. Since this rule is critical from the point of view of proof search we characterize a fragment of the logic for which it can be eliminated.

1 Introduction

Gödel logic—also called Dummett's *LC* or Gödel-Dummett logic, since Dummett [11] presented the first axiomatization matching Gödel's matrix characterization—arguably is one of the most interesting non-classical logics. One reason for this is that it naturally turns up in a number of different fields in logic and computer science. Already in the 1930's Gödel [15] used it to shed light on aspects of intuitionistic logic; later Dunn and Meyer [12] pointed out its relevance for relevance logic; Visser [21] employed it in investigations of the provability logic of Heyting arithmetic; and eventually it was recognized as one of the most useful species of a "fuzzy logic" (see [16]). In our context it is most important that, in contrast to other fuzzy logics, convincing analytic proof systems have been presented for propositional Gödel logic, here called GL. In particular, we will build on Avron's elegant hypersequent calculus [1]. (Alternative analytic systems can be found, e.g., in [4,18,13].)

In this paper we investigate *quantified* propositional Gödel logic QGL, i.e., GL extended by existential and universal quantifiers over propositional variables. As is well known from classical logic, propositional quantifiers make it possible to express complicated properties in a natural and succinct way. Although from a semantic point of view quantified Boolean formulas cannot express *more* Boolean functions than ordinary Boolean formulas, statements about satisfiability and validity of formulas are easily expressible within the logic itself once such

* This work was partially supported by the Austrian Science Fund Projects NZ29-INF and P10282-MAT, and the Max Kade Foundation.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 112–126, 2000.

quantifiers are available[1]. This fact can, e.g., be used to provide efficient proof search methods for a number of non-monotonic reasoning formalisms [14].

For Gödel logic the increase in expressive power is, e.g., witnessed by the fact that statements about the topological structure of the set of truth values (taken as infinite subsets of the real interval $[0, 1]$) can be expressed using propositional quantifiers. Indeed, while only *one* unique infinite valued Gödel logic exists at the quantifier free level, *uncountably many* different quantified propositional Gödel logics exist if arbitrary infinite subsets over $[0, 1]$, that are closed under infima and suprema, are allowed as sets of truth values. (This was proved in [9].) Remarkably enough however, QGL—i.e., quantified propositional Gödel logic over the (full) real interval $[0, 1]$—is not only (recursively) axiomatizable but even decidable [8].

It has been shown [7] that the interpolation property for many-valued logics is closely related to elimination of fuzzy quantifiers. Indeed, classical logic and Gödel logic over $[0, 1]$ enjoy quantifier elimination, which yields an immediate proof of the uniform interpolation property for Gödel logic.

There is yet another reason that renders the investigation of QGL very interesting, namely the clarification of the role of the Takeuti-Titani rule

$$\frac{G \supset [C \vee (A \supset p) \vee (p \supset B)]}{G \supset [C \vee (A \supset B)]} \text{ Takeuti-Titani}$$

which was used in [20] to axiomatize first-order Gödel logic (called "intuitionistic fuzzy logic" by Takeuti and Titani). Takano [19] has shown later that this rule is in fact redundant in the calculus for first-order Gödel logic by referring to arguments already present in A. Horn's [17]. However, an instance of the rule turned out to be *essential* to obtain a complete (Hilbert-style) axiomatization of QGL [9]. Here we will use it to formulate a cut-free hypersequent system for QGL.

The paper is structured as follows. In Section 2 we provide the necessary background on Gödel logic with propositional quantifiers. In particular we recall from [9] the corresponding Hilbert-style system QGL. In Section 3 we define the hypersequent calculus $\mathcal{H}$QGL. Section 4 contains completeness and soundness proofs for $\mathcal{H}$QGL (relative to QGL). In Section 5 we prove the eliminability of cuts from $\mathcal{H}$QGL. Section 6 is devoted to the investigation of the role of the Takeuti-Titani rule. In particular, we prove that this rule can be eliminated from proofs of formulas without strong quantifier occurrences.

2 Gödel Logic

Syntax. We work in the language of propositional logic containing a countably infinite set $Var = \{a, p, q, \dots\}$ of (propositional) variables, the constants $\bot, \top$, as well as the connectives $\wedge, \vee$, and $\supset$. Propositional constants, variables and truth constants are considered atomic formulas. Uppercase letters will serve as

[1] Note that quantifier elimination for QBF exponentially increases formula size unless the polynomial hierarchy collapses.

meta-variables for formulas. If $A(p)$ denotes a formula, then $A(X)$ denotes the formula with all occurrences of the variable p replaced by the formula X.

We use the abbreviation $\neg A$ for $A \supset \bot$.

Semantics. The most important form of Gödel logic—the one also used here—is defined over the real unit interval $[0, 1]$. (In a more general framework, the truth values are taken from a set W such that $\{0, 1\} \subseteq W \subseteq [0, 1]$.)

Let v be a function from *Var* to W, i.e., a variable valuation. Then v is extended to formulas as follows:

$$v(\bot) = 0 \qquad v(A \vee B) = \max(v(A), v(B))$$
$$v(\top) = 1 \qquad v(A \supset B) = \begin{cases} 1 & \text{if } v(A) \leq v(B) \\ v(B) & \text{otherwise} \end{cases}$$
$$v(A \wedge B) = \min(v(A), v(B))$$

A formula A is a tautology, if for all v, $v(A) = 1$.

Equivalent semantics, which stress the close relationship with intuitionistic logic, are provided by linearly ordered Kripke structures [11] and linearly ordered Heyting algebras [17].

Propositional Quantification. In *classical* propositional logic we define $\exists pA(p)$ by $A(\bot) \vee A(\top)$ and $\forall pA(p)$ by $A(\bot) \wedge A(\top)$. In other words, propositional quantification is semantically defined by the supremum and infimum, respectively, of truth functions (with respect to the usual ordering "$0 < 1$" over the classical truth values $\{0, 1\}$).

This correspondence can be extended to Gödel logic by using *fuzzy quantifiers*. Syntactically, this means that we allow formulas $\forall p\phi$ and $\exists p\phi$ in the language. Free and bound occurrences of variables are defined in the usual way.

Given two valuations v_1, v_2, we say that $v_1 \sim_p v_2$ if v_1 and v_2 agree on all variables except possibly p. Then we define the semantics of fuzzy quantifiers as follows:

$$v(\exists pA) = \sup\{w(A) \mid w \sim_p v\} \qquad v(\forall pA) = \inf\{w(A) \mid w \sim_p v\}$$

When we consider fuzzy quantifiers, W has to be closed under taking infima and suprema. If W is the unit interval, this closure property trivially holds. Note that for propositional Gödel logic without quantifiers, the tautologies coincide for all infinite W. However it was shown in [9] that there is an uncountable number of different quantified Gödel logics.

Hilbert Style Calculi. As a basis for Gödel logic we use the following version of *intuitionistic propositional calculus* IPC:

I1	$A \supset (B \supset A)$	I6	$B \supset (A \vee B)$
I2	$(A \wedge B) \supset A$	I7	$(A \wedge \neg A) \supset B$
I3	$(A \wedge B) \supset B$	I8	$(A \supset \neg A) \supset \neg A$
I4	$A \supset (B \supset (A \wedge B))$	I9	$\bot \supset A$
I5	$A \supset (A \vee B)$	I10	$A \supset \top$

I11 $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$
I12 $((A \supset C) \wedge (B \supset C)) \supset ((A \vee B) \supset C)$

The system GL for propositional Gödel logic is obtained by adding to IPC the following axiom which expresses the linearity of the ordering of truth values or linear Kripke models, respectively:
(*Linearity*) $(A \supset B) \vee (B \supset A)$

Theorem 1 ([3,16,11]). *The system* GL *is sound and complete for propositional Gödel logic (over* $[0,1]$*).*

Turning to quantified propositional logic, we add to IPC the following two axioms:

Implies-$\exists$: $A(X) \supset \exists p A(p)$
$\forall$-*Implies*: $[\forall p A(p)] \supset A(X)$

and the rules:

$$\frac{Z(a) \supset Y^{(a)}}{[\exists p Z(p)] \supset Y^{(a)}}\ \mathrm{R}\exists \qquad \frac{Y^{(a)} \supset Z(a)}{Y^{(a)} \supset \forall p Z(p)}\ \mathrm{R}\forall$$

Here and below, for any formula X, $X^{(a)}$ denotes that a does not occur free in X, i.e., a is a (propositional) *eigenvariable*.

Remark 1. Implies-$\exists$, $\forall$-*Implies* as well as R$\exists$ and R$\forall$ are already sound for intuitionistic logic with propositional quantifiers.

The system of quantified Gödel logic QGL is obtained by taking all above-mentioned axioms and rules plus the following two axioms:

$\vee$-*Shift*: $[\forall p(A^{(p)} \vee B)] \supset [A^{(p)} \vee \forall p B]$
Density: $[\forall p((A^{(p)} \supset p) \vee (p \supset B^{(p)}))] \supset (A^{(p)} \supset B^{(p)})$

We recall the following result from [9]:

Theorem 2. *The calculus* QGL *is sound and complete for quantified propositional Gödel logic.*

It has been proved in [9] that instances of the quantifier axioms, where the formula X is quantifier free, suffice for the completeness of the calculus. We use this restricted version of the axioms here.

3 A Hypersequent Calculus for QGL

Hypersequent calculi are a simple and natural generalization of Gentzen's (ordinary) sequent calculi. They have been demonstrated to be of great use in enhancing the realm of logics for which elegant analytic proof systems can be defined. (See, e.g., [1,2,5,6,10].)

Definition 1. *A* hypersequent *is a structure of the form*

$$\Gamma_1 \Rightarrow \Delta_1 \mid \Gamma_2 \Rightarrow \Delta_2 \mid \cdots \mid \Gamma_n \Rightarrow \Delta_n$$

where each $\Gamma_i \Rightarrow \Delta_i$ *is a sequent called* component *of the hypersequent.*

The intended interpretation of the symbol "|" is disjunctive (at the meta-level). We consider the left and right hand sides of sequents as *multisets* of formulas and hypersequents as *multisets* of sequents. In the following, we use single-conclusioned (intuitionistic) versions of hypersequents, i.e., the right hand side of each component (i.e., of each sequent) contains at most one formula.

Like in ordinary sequent calculi, rules of a hypersequent calculus are divided into two groups: *logical rules* and *structural rules*. The logical rules are essentially the same as those in sequent calculi, the only difference being the presence of dummy contexts $\mathcal{H}$, called *side hypersequents* which are used as meta-variables for (possibly empty) hypersequents. The structural rules are divided into *internal* and *external rules*.

3.1 The Calculus $\mathcal{H}$QGL

Axioms: $\bot \Rightarrow, \quad \Rightarrow \top, \quad A \Rightarrow A$, for any formula A.

In the following rules, Δ is either a single formula or empty.

Internal structural rules:

$$\frac{\mathcal{H} \mid \Gamma \Rightarrow \Delta}{\mathcal{H} \mid A, \Gamma \Rightarrow \Delta}\ (iw \Rightarrow) \qquad \frac{\mathcal{H} \mid \Gamma \Rightarrow}{\mathcal{H} \mid \Gamma \Rightarrow A}\ (\Rightarrow iw) \qquad \frac{\mathcal{H} \mid A, A, \Gamma \Rightarrow \Delta}{\mathcal{H} \mid A, \Gamma \Rightarrow \Delta}\ (ic \Rightarrow)$$

External structural rules:

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

Logical rules:

$$\frac{\mathcal{H} \mid A_1, \Gamma \Rightarrow \Delta \quad \mathcal{H} \mid A_2, \Gamma \Rightarrow \Delta}{\mathcal{H} \mid A_1 \vee A_2, \Gamma \Rightarrow \Delta}\ (\vee \Rightarrow) \qquad \frac{\mathcal{H} \mid \Gamma \Rightarrow A_i}{\mathcal{H} \mid \Gamma \Rightarrow A_1 \vee A_2}\ (\Rightarrow \vee_i)\, i \in \{1,2\}$$

$$\frac{\mathcal{H} \mid A_i, \Gamma \Rightarrow \Delta}{\mathcal{H} \mid A_1 \wedge A_2, \Gamma \Rightarrow \Delta}\ (\wedge \Rightarrow_i)\, i \in \{1,2\} \qquad \frac{\mathcal{H} \mid \Gamma \Rightarrow A \quad \mathcal{H} \mid \Gamma \Rightarrow B}{\mathcal{H} \mid \Gamma \Rightarrow A \wedge B}\ (\Rightarrow \wedge)$$

$$\frac{\mathcal{H} \mid \Gamma \Rightarrow A \quad \mathcal{H} \mid B, \Gamma \Rightarrow \Delta}{\mathcal{H} \mid A \supset B, \Gamma \Rightarrow \Delta}\ (\supset \Rightarrow) \qquad \frac{\mathcal{H} \mid A, \Gamma \Rightarrow B}{\mathcal{H} \mid \Gamma \Rightarrow A \supset B}\ (\Rightarrow \supset)$$

In the following rules for introducing propositional quantifiers, formula X is required to be *quantifier free* and the propositional variable a is subject to the usual eigenvariable condition; i.e., it must not occur freely in the lower hypersequent.

$$\frac{\mathcal{H} \mid A(X), \Gamma \Rightarrow \Delta}{\mathcal{H} \mid (\forall p)A(p), \Gamma \Rightarrow \Delta}\ (\forall \Rightarrow) \qquad \frac{\mathcal{H} \mid \Gamma \Rightarrow A(a)}{\mathcal{H} \mid \Gamma \Rightarrow (\forall p)A(p)}\ (\Rightarrow \forall)$$

$$\frac{\mathcal{H} \mid A(a), \Gamma \Rightarrow \Delta}{\mathcal{H} \mid (\exists p)A(p), \Gamma \Rightarrow \Delta}\ (\exists \Rightarrow) \qquad \frac{\mathcal{H} \mid \Gamma \Rightarrow A(X)}{\mathcal{H} \mid \Gamma \Rightarrow (\exists p)A(p)}\ (\Rightarrow \exists)$$

Cut:

$$\frac{\mathcal{H} \mid \Gamma \Rightarrow A \quad \mathcal{H} \mid A, \Gamma \Rightarrow B}{\mathcal{H} \mid \Gamma \Rightarrow B}\ (cut)$$

Communication:

$$\frac{\mathcal{H} \mid \Gamma_1, \Gamma_1' \Rightarrow A_1 \quad \mathcal{H} \mid \Gamma_2, \Gamma_2' \Rightarrow A_2}{\mathcal{H} \mid \Gamma_1', \Gamma_2' \Rightarrow A_1 \mid \Gamma_1, \Gamma_2 \Rightarrow A_2} \; (comm)$$

Density:

$$\frac{\mathcal{H} \mid \Pi \Rightarrow a \mid a, \Gamma \Rightarrow C}{\mathcal{H} \mid \Pi, \Gamma \Rightarrow C} \; (tt)$$

In the density rule (tt) the propositional variable a is subject to the eigenvariable condition, i.e., it must not occur freely in the lower hypersequent.

Remarks on the communication rule. The significance of the communication rule stems from the fact that it allows to derive the linearity axiom, as can be seen in the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \Rightarrow A \quad B \Rightarrow B}{A \Rightarrow B \mid B \Rightarrow A}\,(comm)}{A \Rightarrow B \mid \Rightarrow B \supset A}\,(\Rightarrow\supset)}{\Rightarrow A \supset B \mid \Rightarrow B \supset A}\,(\Rightarrow\supset)}{\Rightarrow A \supset B \vee B \supset A \mid \Rightarrow B \supset A}\,(\Rightarrow \vee)}{\Rightarrow A \supset B \vee B \supset A \mid \Rightarrow A \supset B \vee B \supset A}\,(\Rightarrow \vee)}{\Rightarrow A \supset B \vee B \supset A}\,(ec)$$

A number of variants of the communication rule exist. In [1]

$$\frac{\mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow A_1 \quad \mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow A_2}{\mathcal{H} \mid \Gamma_1 \Rightarrow A_1 \mid \Gamma_2 \Rightarrow A_2} \; (comm')$$

was used. In our context also the combination of the following two rules

$$\frac{\mathcal{H}_1 \mid \Gamma_1 \Rightarrow A_1 \quad \mathcal{H}_2 \mid \Gamma_2 \Rightarrow A_2}{\mathcal{H}_1 \mid \mathcal{H}_2 \mid \Gamma_1 \Rightarrow A_1 \mid \Gamma_2 \Rightarrow A_2} \; (comm'_m) \qquad \frac{\mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow A}{\mathcal{H} \mid \Gamma_1 \Rightarrow A \mid \Gamma_2 \Rightarrow A} \; (S_I)$$

is equivalent to the use of $(comm)$ as remarked in [2].

Remarks on the density rule. A variant of the density rule (tt), as described in the Introduction, was first considered by Takeuti and Titani in the context of intuitionistic fuzzy logic [20]. Note that the system $\mathcal{H}\mathsf{QGL}$—at first sight—looks similar to first order intuitionistic fuzzy logic [20]. In $\mathcal{H}\mathsf{QGL}$ however, the rule (tt) can not be eliminated, since it expresses density of truth values; semantically, density does not follow from the other axioms. We shall investigate the status of the density rule more carefully in Section 6 below.

4 Soundness and Completeness

We show the soundness and completeness of $\mathcal{H}\mathsf{QGL}$ relative to the Hilbert style axiomatization QGL of quantified Gödel logic.

To facilitate argumentation we will make use of the following lemma:

Lemma 1. *If* $\mathcal{H}\mathsf{QGL} \vdash U, \Pi \Rightarrow U'$ *and* $\mathcal{H}\mathsf{QGL} \vdash V, \Gamma \Rightarrow V'$ *then also:* $\mathcal{H}\mathsf{QGL} \vdash U \vee V, \Pi \Rightarrow U' \mid U \vee V, \Gamma \Rightarrow V'$.

Proof.

$$\dfrac{V, \Gamma \Rightarrow V' \qquad \dfrac{\dfrac{U, \Pi \Rightarrow U' \qquad V, \Pi \Rightarrow U' \mid U, \Gamma \Rightarrow V'}{}(comm)}{U, \Pi \Rightarrow U' \mid U, \Gamma \Rightarrow V'}(ew), (\vee \Rightarrow)}{U \vee V, \Pi \Rightarrow U' \mid U \vee V, \Gamma \Rightarrow V'}(ew), (\vee \Rightarrow)$$

□

Theorem 3. $\mathcal{H}\mathsf{QGL}$ *is complete for quantified propositional Gödel logic.*

Proof. We use the completeness of QGL and thus have to show that $\Rightarrow A$ is derivable in $\mathcal{H}\mathsf{QGL}$ for every formula A that is derivable in QGL.

It is straightforward to check this for every axiom A of IPC. The *linearity* axiom is derivable by use of the communication rule as was demonstrated in Section 3 above.

By Lemma 1 we obtain:

$$\mathcal{H}\mathsf{QGL} \vdash A^{(p)} \vee B(p) \Rightarrow B(p) \mid A^{(p)} \vee B(p) \Rightarrow A$$

From this the axiom $\vee$-*Shift* is derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A^{(p)} \vee B(p) \Rightarrow B(p) \mid A^{(p)} \vee B(p) \Rightarrow A^{(p)}}{}2\times(\forall \Rightarrow)}{\forall p(A^{(p)} \vee B(p)) \Rightarrow B(p) \mid \forall p(A^{(p)} \vee B(p)) \Rightarrow A^{(p)}}(\Rightarrow \forall)}{\forall p(A^{(p)} \vee B(p)) \Rightarrow \forall p B(p) \mid \forall p(A^{(p)} \vee B(p)) \Rightarrow A^{(p)}}2\times(\Rightarrow \vee)}{\forall p(A^{(p)} \vee B(p)) \Rightarrow A^{(p)} \vee \forall p B(p) \mid \forall p(A^{(p)} \vee B(p)) \Rightarrow A^{(p)} \vee \forall p B(p)}(ec)}{\forall p(A^{(p)} \vee B(p)) \Rightarrow A^{(p)} \vee \forall p B(p)}(\Rightarrow \supset)}{\Rightarrow [\forall p(A^{(p)} \vee B(p))] \supset [A^{(p)} \vee \forall p B(p)]}$$

For the *density* axiom the density rule (tt) has to be used. Again we apply Lemma 1 and obtain:

$$\mathcal{H}\mathsf{QGL} \vdash (A^{(p)} \supset p) \vee (p \supset B^{(p)}), A^{(p)} \Rightarrow p \mid (A^{(p)} \supset p) \vee (p \supset B^{(p)}), p \Rightarrow B^{(p)}$$

From this we proceed as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{(A^{(p)} \supset p) \vee (p \supset B^{(p)}), A^{(p)} \Rightarrow p \mid (A^{(p)} \supset p) \vee (p \supset B^{(p)}), p \Rightarrow B^{(p)}}{}2\times(\forall \Rightarrow)}{\forall p(A^{(p)} \supset p) \vee (p \supset B^{(p)}), A^{(p)} \Rightarrow p \mid \forall p(A^{(p)} \supset p) \vee (p \supset B^{(p)}), p \Rightarrow B^{(p)}}(tt)}{\forall p(A^{(p)} \supset p) \vee (p \supset B^{(p)}), A^{(p)} \Rightarrow B^{(p)}}(\Rightarrow \supset)}{\forall p(A^{(p)} \supset p) \vee (p \supset B^{(p)}) \Rightarrow A^{(p)} \supset B^{(p)}}(\Rightarrow \supset)}{\Rightarrow \forall p(A^{(p)} \supset p) \vee (p \supset B^{(p)}) \supset (A^{(p)} \supset B^{(p)})}$$

Concerning the two quantifier axioms $A(X) \supset \exists p A(p)$ and $\forall p A(p) \supset A(X)$ we use the fact (noted above) that instances where A is *quantifier-free* suffice for

the completeness of QGL. Such instances are easily derivable using ($\Rightarrow \exists$) and ($\forall \Rightarrow$), respectively.

Modus Ponens corresponds to the derivability of $A, A \supset B \Rightarrow B$ and the *cut rule*.

Finally, we observe that the QGL-rules R$\forall$ and R$\exists$ correspond to a combination of the cut rule and $\mathcal{H}$QGL-rules ($\Rightarrow \forall$) and ($\exists \Rightarrow$), respectively. We show this for R$\forall$:

$$\dfrac{\dfrac{\dfrac{\Rightarrow Y \supset Z(a)}{Y \Rightarrow Y \supset Z(a)}\,(iw \Rightarrow) \qquad \dfrac{Y \Rightarrow Y \qquad Z(a) \Rightarrow Z(a)}{Y \supset Z(a), Y \Rightarrow Z(a)}\,(\supset\Rightarrow)}{Y \Rightarrow Z(a)}\,(cut)}{\dfrac{Y \Rightarrow \forall p Z(p)}{\Rightarrow Y \supset \forall p Z(p)}\,(\Rightarrow\supset)}\,(\Rightarrow \forall)$$

□

In order to prove soundness we have to translate hypersequents into corresponding formulas.

Definition 2. *The* generic interpretation $\mathfrak{I}(\mathcal{H})$ *of a hypersequent* $\mathcal{H}$ *is defined as follows:*

$\mathfrak{I}(\Rightarrow B) := B$
$\mathfrak{I}(A_1, \ldots, A_n \Rightarrow B) := (A_1 \supset \ldots \supset (A_n \supset B) \ldots)$ *and*
$\mathfrak{I}(A_1, \ldots, A_n \Rightarrow) := (A_1 \supset \ldots \supset (A_n \supset \bot) \ldots)$
$\mathfrak{I}(S_1 \mid \ldots \mid S_n) := \forall \boldsymbol{p}[\mathfrak{I}(S_1) \vee \ldots \vee \mathfrak{I}(S_n)]$, *where* $\boldsymbol{p}$ *is the vector of all free variables occurring in the sequents* $S_1, \ldots, S_n$.

Theorem 4. $\mathcal{H}$QGL *is sound for quantified propositional Gödel logic.*

Proof. Since the axioms of $\mathcal{H}$QGL are trivially sound it suffices to show that for every rule

$$\frac{\mathcal{H}}{\mathcal{H}^*} \quad \text{and} \quad \frac{\mathcal{H} \quad \mathcal{H}'}{\mathcal{H}^*}$$

their translations $\mathfrak{I}(\mathcal{H}) \supset \mathfrak{I}(\mathcal{H}^*)$ and $\mathfrak{I}(\mathcal{H}) \supset (\mathfrak{I}(\mathcal{H}') \supset \mathfrak{I}(\mathcal{H}^*))$, respectively, are derivable in QGL.

First observe that the derivability of $A \supset B$ implies the derivability of $(A \vee C) \supset (B \vee C)$. Therefore we can disregard side sequents in proving the soundness of hypersequent rules. Moreover, if

$$\mathsf{QGL} \vdash (A_1 \supset (\ldots \supset (A_n \supset B) \ldots))$$

then, for any permutation π, also

$$\mathsf{QGL} \vdash (A_{\pi(1)} \supset (\ldots \supset (A_{\pi(n)} \supset B) \ldots)).$$

Moreover,

$$\mathsf{QGL} \vdash (B \supset C) \quad \text{implies} \quad \mathsf{QGL} \vdash (A \supset B) \supset (A \supset C).$$

Consequently, we can also disregard side formulas in all rules.

Given these preparations it is straightforward to check that the formulas corresponding to translations of $\mathcal{H}\mathsf{QGL}$-rules are derivable in QGL. For example, the translation of the density rule (tt) is

$$[\forall \boldsymbol{p} \forall a(B \vee (C \supset a) \vee (a \supset D))] \supset [\forall \boldsymbol{p}(B \vee (C \supset D))]$$

where $\boldsymbol{p}$ is the vector all free variables in $B \vee (C \supset D)$ and a does not occur in $\boldsymbol{p}$. This formula is easily derived from the density axiom of QGL. □

We have already remarked that we obtain *different* quantified propositional Gödel logics if we consider arbitrary infinite subsets of $[0, 1]$, that contain 0 and 1 and are closed under taking infima and suprema, as sets of truth values. Since only the density axiom (or equivalently: the density rule) is not sound under all of these interpretations we can draw the following interesting conclusion about the *intersection* of all Gödel logics from Theorem 4.

Corollary 1. $\mathcal{H}\mathsf{QGL}$ *without the density rule is sound for* all *quantified propositional Gödel logics (and therefore also in their intersection).*

5 Cut-Elimination

Obviously, the question whether $\mathcal{H}\mathsf{QGL}$ remains complete without the cut rule is of central interest. (Observe that the proof of Theorem 3 does not give any hint in that direction.) Cut-elimination for Avron's hypersequent calculus $\mathcal{G}\mathsf{LC}$ for quantifier free propositional Gödel logics was demonstrated in [1]. Extending this result to $\mathcal{H}\mathsf{QGL}$ is non-trivial because of the fact that arbitrarily complex formulas are replaced by propositional variables when introducing weak quantifier occurrences.

Theorem 5. $\mathcal{H}\mathsf{QGL}$ *enjoys cut-elimination, i.e., any derivation of a hypersequent* $\mathcal{G}$ *in* $\mathcal{H}\mathsf{QGL}$ *can be transformed into a derivation of* $\mathcal{G}$ *that does not use the cut rule.*

Proof. Except for the quantifier rules and the density rule, $\mathcal{H}\mathsf{QGL}$ is like Avron's system $\mathcal{G}\mathsf{LC}$ presented in [1] and in [2]. Since it is impossible to present a fully self-contained cut-elimination proof for $\mathcal{H}\mathsf{QGL}$ within the available space we focus on the single argument that is different from the cut-elimination proof for $\mathcal{G}\mathsf{LC}$ (see [1]).

In order to see that reducing cuts on *quantified* formulas does not spoil the termination of the cut-elimination procedure of [1] we have to introduce the number of quantifier occurrences in a formula A—qfo(A)—as an additional parameter. More exactly, let π be a derivation with a single application of the cut rule as last inference. We show by induction along the lexicographical ordering of triples (q, c, r) of non-negative integers that π can be stepwise transformed into a cut-free derivation of the same end hypersequent. If A is the cut-formula

to be considered, then q will be qfo(A), c the complexity of A, (i.e., the total number of logical symbols occurring in A), and r the "rank" of the derivation, i.e., the sum of the heights of the derivation of the premisses of the cut.

As usual, the rank r is decreased until the cut-formula can be assumed to have been introduced immediately before the application of the cut rule. Eliminating cuts with axioms is also exactly as usual. Since all structural rules are present they obviously pose no problem in reducing a cut. The cases for logical rules introducing propositional connectives, too, are—modulo side hypersequents—identical to those for LJ (i.e., the complexity c of cut-formula is decreased by cut-reduction). The fact that the cut rule permutes with the communication rule (thereby decreasing the rank r) is (easy to see and) known from [1]. The density rule (tt) straightforwardly commutes with the cut rule modulo structural rules. Since it does not introduce any formula, no additional reduction rule is needed.

The only remaining cases are those with quantified cut-formulas.

$$\dfrac{\dfrac{\dfrac{\pi_1}{\mathcal{H} \mid \Gamma \Rightarrow A(a)}}{\mathcal{H} \mid \Gamma \Rightarrow (\forall p)A(p)}\ (\Rightarrow \forall) \qquad \dfrac{\dfrac{\pi_2}{\mathcal{H} \mid A(X), \Gamma \Rightarrow B}}{\mathcal{H} \mid (\forall p)A(p), \Gamma \Rightarrow B}\ (\forall \Rightarrow)}{\mathcal{H} \mid \Gamma \Rightarrow B}\ (cut)$$

is reduced to

$$\dfrac{\dfrac{\pi_1[X/a]}{\mathcal{H} \mid \Gamma \Rightarrow A(X)} \qquad \dfrac{\pi_2}{\mathcal{H} \mid A(X), \Gamma \Rightarrow B}}{\mathcal{H} \mid \Gamma \Rightarrow B}\ (cut)$$

where $\pi_1[X/a]$ denotes π_1 after substituting all occurrences of a by X. The number q of occurrences of quantifiers in the cut-formula is decreased, since X is quantifier free. Therefore the reduced derivation is strictly smaller with respect to our induction ordering (even if c is increased). The case for existential quantification is analogous.

It follows that the iterated application of the indicated reduction procedures is guaranteed to terminate and yields a cut-free proof. □

6 The Role of the Density Rule

Obviously the density rule is needed to derive instances of the density axiom in $\mathcal{H}$QGL. On the other hand it renders tableau style proof search in $\mathcal{H}$QGL problematic. Therefore it is useful to know for which fragments of quantified propositional Gödel logic the rule (tt) (or a variant thereof) is actually needed to find a proof. To this aim we show (in Theorem 6 below) that the density rule is in fact redundant for the existential fragment of QGL (see Definition 4).

We first need to replace rule (tt) by the following variant of it:

$$\dfrac{\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\ (tt^*)$$

where a must not occur freely in the lower sequent. ($a, ..., a$ indicates one or more occurrences of a.)

(tt^*) incorporates external and internal contractions and therefore commutes with most other rules of $\mathcal{H}\mathsf{QGL}$. (See proof of Theorem 6, below.)

Proposition 1. *Rules* (tt) *and* (tt^*) *are equivalent in* $\mathcal{H}\mathsf{QGL}$. *More exactly, any derivation using applications of* (tt^*) *can be transformed into one using applications of* (tt) *(and the other rules of* $\mathcal{H}\mathsf{QGL}$*) only, and vice versa.*

Proof. The second direction is trivial since (tt) is just an instance of (tt^*).

The first direction, i.e., the derivation of the conclusion of (tt^*) from its premiss using only the rules of $\mathcal{H}\mathsf{QGL}$ is as follows.

We first apply internal contraction to the premiss hypersequent

$$\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m$$

to obtain the hypersequent

$$\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, \Gamma_m \Rightarrow C_m$$

By weakening the left hand sides of the first n exhibited components of the hypersequent from Π_i into $\Pi_1, \ldots, \Pi_n$ these components become identical. Therefore, by external contraction, we obtain:

$$\mathcal{H} \mid \Pi_1, \ldots, \Pi_n \Rightarrow a \mid a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, \Gamma_m \Rightarrow C_m$$

Applying rules ($\wedge \Rightarrow$) and ($\Rightarrow \supset$) to the m rightmost components leads to

$$\mathcal{H} \mid \Pi_1, \ldots, \Pi_n \Rightarrow a \mid a \Rightarrow \left(\bigwedge \Gamma_1\right) \supset C_1 \mid \ldots \mid a \Rightarrow \left(\bigwedge \Gamma_m\right) \supset C_m$$

where $\bigwedge\{B_1, \ldots B_k\}$ abbreviates $B_1 \wedge \ldots \wedge B_k$. We then combine the right hand sides of the last m components using ($\Rightarrow \vee$) and (ec) to derive

$$\mathcal{H} \mid \Pi_1, \ldots, \Pi_n \Rightarrow a \mid a \Rightarrow \bigvee_{i=1}^{i=m} \left(\left(\bigwedge \Gamma_i\right) \supset C_i\right)$$

At this point (tt) can be applied to obtain

$$\mathcal{H} \mid \Pi_1, \ldots, \Pi_n \Rightarrow \bigvee_{i=1}^{i=m} \left(\left(\bigwedge \Gamma_i\right) \supset C_i\right) \tag{1}$$

It now remains to take apart the right hand side into the relevant sub-formulas. By Lemma 2 (below) we have a proof of

$$\bigvee_{i=1}^{i=m} \left(\left(\bigwedge \Gamma_i\right) \supset C_i\right) \Rightarrow \left(\bigwedge \Gamma_1\right) \supset C_1 \mid \ldots \mid \bigvee_{i=1}^{i=m} \left(\left(\bigwedge \Gamma_i\right) \supset C_i\right) \Rightarrow \left(\bigwedge \Gamma_m\right) \supset C_m \tag{2}$$

After augmenting (2) with the corresponding side hypersequents and side formulas by weakenings, we can repeatedly apply the cut rule to eliminate all occurrences of $\bigvee_{i=1}^{i=m} \left(\left(\bigwedge \Gamma_i\right) \supset C_i\right)$ in (1) and (2). This leads to a derivation of

$$\mathcal{H} \mid \Pi_1, \ldots, \Pi_n \Rightarrow \left(\bigwedge \Gamma_1\right) \supset C_1 \mid \ldots \mid \Pi_1, \ldots, \Pi_n \Rightarrow \left(\bigwedge \Gamma_m\right) \supset C_m \tag{3}$$

To break up $\left(\bigwedge \Gamma_i\right) \supset C_i$ into sub-formulas we use cuts with easily provable

hypersequents of the form $(B_1 \wedge \ldots \wedge B_r) \supset A, B_1, \ldots B_r \Rightarrow A$. More exactly, we derive (from axioms) for each i $(1 \leq i \leq m)$ the hypersequent

$$(\bigwedge \Gamma_i) \supset C_i, \Gamma_i \Rightarrow C_i,$$

augment it, through external weakenings, by appropriate side hypersequents, and cut upon the formula $\left(\bigwedge \Gamma_i\right) \supset C_i)$ with the right hand side of the corresponding component of (3). These m cuts result, modulo applications of external weakenings and contractions, in the hypersequent

$$\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m$$

which is the conclusion of rule (tt^*). □

Lemma 2. *Every hypersequent $\mathcal{H}$ of the form*

$$B_1 \vee \ldots \vee B_r \Rightarrow B_1 \mid \ldots \mid B_1 \vee \ldots \vee B_r \Rightarrow B_r$$

is provable in $\mathcal{H}$QGL.

Proof. Space limits do not permit us to present a derivation. However, by the completeness proof of Section 4 it suffices to argue semantically and show that the formula $\mathfrak{I}(\mathcal{H}) = \bigvee_{1 \leq i \leq r}(B_1 \vee \ldots \vee B_r) \supset B_i$ is *valid* in QGL. Since $v(B_1 \vee \ldots \vee B_r) = \max_{1 \leq i \leq r}(v(B_i))$ and $v(A \supset B) = 1$ if $v(A) \leq v(B)$ (see Section 2) we know that, for every variable valuation v, at least one disjunct of $\mathfrak{I}(\mathcal{H})$ must take value 1. Therefore, $\mathfrak{I}(\mathcal{H})$ always evaluates to 1; i.e., it is valid. □

Definition 3. *A quantifier occurrence "$\exists x$" is* weak *(*strong*) in a formula F if*

- *$F \equiv \exists x G$ $(\neg \exists x G)$, or*
- *$F \equiv G \circ H$ for $\circ \in \{\vee, \wedge\}$ and the occurrence of "$\exists x$" is weak (strong) in either G or H, or*
- *$F \equiv G \supset H$ and the occurrence of "$\exists x$" is weak (strong) in H or strong (weak) in G.*

The definition for "$\forall x$" is completely dual. I.e., the occurrence of "$\forall x$" is strong *in $\forall x G$ and* weak *in $\neg \forall x G$, etc.*

Definition 4. *We call the set of valid formulas of quantified propositional Gödel logic in which all quantifier occurrences are weak the* existential fragment *of* QGL. *The calculus* $\mathcal{H}$QGL *without the rules $(\Rightarrow \forall)$ and $(\exists \Rightarrow)$ is denoted by* $\mathcal{H}$QGLm.

Corollary 2. *Cut-free* $\mathcal{H}$QGLm *is sound and complete for the existential fragment of* QGL.

Proof. Follows directly from the proofs of Theorems 3, 4, and 5. □

Theorem 6. *Every* $\mathcal{H}$QGLm*-proof π of a hypersequent $\mathcal{G}$ (in the language of* QGL*) can be transformed into a cut-free proof π' of $\mathcal{G}$ in which no application of the density rule occurs.*

Proof. By Proposition 1 we can transform π into a proof of $\mathcal{G}$ containing applications of (tt^*) instead of (tt). By Theorem 5 we can assume this proof to be cut-free. Without loss of generality, we only have to show how to eliminate a single application of (tt^*) which appears as last inference. We show that the application of (tt^*) can either be moved upwards (i.e., closer to the axioms) or simply be replaced by weakenings and contractions. Since the premiss of (tt^*) has to contain at least *two* components it cannot be an axiom. Therefore the redundancy of (tt^*) follows by induction on the height of the proof above the application of (tt^*).

We distinguish cases according to the type of rule applied immediately before (tt^*). Due to space limitations, we only present a few exemplary sub-cases. completely

Internal structural rules: We present the case where $(\Rightarrow iw)$ is used to complete an exhibited component of the premiss of (tt^*):

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi_1 \Rightarrow \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}\,(\Rightarrow iw)}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)$$

is transformed into

$$\dfrac{\dfrac{\dfrac{\mathcal{H} \mid \Pi_1 \Rightarrow \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{\mathcal{H} \mid \Pi_1 \Rightarrow \mid \Pi_2, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_2, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(\Rightarrow iw)'s}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(ec)$$

If the occurrence of a introduced by weakening is a necessary condition for the application of rule (tt^*), then the application of (tt^*) can be replaced by weakenings and external contractions. E.g., if π ends in

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid \Gamma \Rightarrow C}{\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, \Gamma \Rightarrow C}\,(iw \Rightarrow)}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma \Rightarrow C}\,(tt^*)$$

then, since there are no left hand side occurrences of a in the first hypersequent, the remaining occurrences of a must have been introduced by weakenings. By dropping these weakenings we can transform π into a proof ending with

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi_1 \Rightarrow \mid \ldots \mid \Pi_n \Rightarrow \mid \Gamma \Rightarrow C}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma \Rightarrow C \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma \Rightarrow C}\,(iw \Rightarrow)'s, (\Rightarrow iw)'s}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma \Rightarrow C}\,(ec)'s$$

If $(iw \Rightarrow)$, $(\Rightarrow iw)$ do not introduce an occurrence of a, then the permutation with (tt^*) is trivial. The cases for $(ic \Rightarrow)$ are straightforward, too.

External structural rules: Like for $(\Rightarrow iw)$, above, the application of (tt^*) is replaced by weakenings and contractions if an essential component of the premiss of (tt^*) arises from external weakening.

Otherwise we can permute (tt^*) with (ew). E.g.,

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{\mathcal{H} \mid \Pi_1 \Rightarrow a \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}\,(ew)}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)$$

is transformed into

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi_2 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{\mathcal{H} \mid \Pi_2, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_2, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)}{\mathcal{H} \mid \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(iw \Rightarrow)'s$$

Similarly, external contractions turn into internal contractions by moving (tt^*) upwards.

Logical rules: We present the transformation for the case where an application of rule $(\forall \Rightarrow)$ is situated above an application of (tt^*):

$$\dfrac{\dfrac{A(X), \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{(\forall p)A(p), \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}\,(\forall \Rightarrow)}{(\forall p)A(p), \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid (\forall p)A(p), \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)$$

is transformed into

$$\dfrac{\dfrac{A(X), \Pi_1 \Rightarrow a \mid \ldots \mid \Pi_n \Rightarrow a \mid a, ..., a, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid a, ..., a, \Gamma_m \Rightarrow C_m}{A(X), \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid A(X), \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,(tt^*)}{(\forall p)A(p), \Pi_1, ..., \Pi_n, \Gamma_1 \Rightarrow C_1 \mid \ldots \mid (\forall p)A(p), \Pi_1, ..., \Pi_n, \Gamma_m \Rightarrow C_m}\,n \times (\forall \Rightarrow)$$

Permuting (tt^*) with other logical inferences of $\mathcal{H}\mathsf{QGLm}$ is straight forward, too.

Communication: Consider, e.g., the following typical case:

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi, a, ..., a \Rightarrow D \mid \Gamma_1, \Gamma_1' \Rightarrow a \qquad \mathcal{H} \mid \Pi, a, ..., a \Rightarrow D \mid \Gamma_2, \Gamma_2', a \Rightarrow C}{\mathcal{H} \mid \Pi, a, ..., a \Rightarrow D \mid \Gamma_1', \Gamma_2' \Rightarrow a \mid \Gamma_1, \Gamma_2, a \Rightarrow C}\,(comm)}{\mathcal{H} \mid \Gamma_1', \Gamma_2', \Gamma_1, \Gamma_2 \Rightarrow C \mid \Gamma_1', \Gamma_2', \Pi \Rightarrow D}\,(tt^*)$$

is transformed into

$$\dfrac{\dfrac{\mathcal{H} \mid \Pi, a, ..., a \Rightarrow D \mid \Gamma_1, \Gamma_1' \Rightarrow a}{\mathcal{H} \mid \Pi \Rightarrow D \mid \Gamma_1, \Gamma_1' \Rightarrow}\,(tt^*)}{\mathcal{H} \mid \Gamma_1', \Gamma_2', \Gamma_1, \Gamma_2 \Rightarrow C \mid \Gamma_1', \Gamma_2', \Pi \Rightarrow D}\,(iw)'s$$

If an application of the communication rule which has at least one premise is of the form $\mathcal{H} \mid \Gamma, a \Rightarrow a$ is followed by an application of (tt^*), then it is replaced by subsequent weakenings. E.g., we replace

$$\dfrac{\dfrac{\mathcal{H} \mid \Gamma_1, a \Rightarrow a \qquad \mathcal{H} \mid \Gamma_2, \Gamma_2' \Rightarrow \Delta}{\mathcal{H} \mid \Gamma_1, \Gamma_2 \Rightarrow a \mid \Gamma_2', a \Rightarrow \Delta}\,(comm)}{\mathcal{H} \mid \Gamma_1, \Gamma_2, \Gamma_2' \Rightarrow \Delta}\,(tt^*) \qquad \text{by:} \qquad \dfrac{\mathcal{H} \mid \Gamma_2, \Gamma_2' \Rightarrow \Delta}{\mathcal{H} \mid \Gamma_1, \Gamma_2, \Gamma_2' \Rightarrow \Delta}\,(iw \Rightarrow)'s$$

□

Remark 2. Permuting (tt^*) with $(\Rightarrow \forall)$ or $(\exists \Rightarrow)$ is unsound because the eigenvariable condition may be violated.

Theorem 6 and Corollaries 2 and 1 jointly imply the following interesting fact:

Corollary 3. *The existential fragment of* QGL *coincides with the existential fragment of the intersection of* all *quantified propositional Gödel logics.*

References

1. Avron, A.: Hypersequents, Logical Consequence and Intermediate Logics for Concurrency. *Annals of Mathematics and Artificial Intelligence* Vol.4, 1991, 225–248.
2. Avron, A.: The Method of Hypersequents in Proof Theory of Propositional Non-Classical Logics. In *Logic: From Foundations to Applications. European Logic Colloquium, Keele, UK, July 20–29, 1993*. Oxford, Clarendon Press, 1996, 1–32.
3. Baaz, M.: Infinite-Valued Gödel Logics with 0-1-Projections and Relativizations. In *Proceedings of Gödel 96 - Kurt Gödel's Legacy*, LNML 6, Springer, 23–33.
4. Baaz, M., Fermüller, C.: Analytic Calculi for Projective Logics. *Automated Reasoning with Analytic Tableaux and Related Methods*, TABLEAUX'99, LNAI 1617, Springer, 1999, 36–50.
5. Baaz, M., Ciabattoni, A., Fermüller, C., Veith, H.: Proof Theory of Fuzzy Logics: Urquhart's C and Related Logics, Brim, Lubos (ed.) et al., Proc. *Mathematical Foundations of Computer Science*, 23rd International Symposium, MFCS '98. LNCS 1450, Springer, 1998, 203–212.
6. Baaz, M., Ciabattoni, A., Fermüller, C., Veith, H.: On the Undecidability of Some Sub-classical First-order Logics. Proc. *Foundations of Software Technology and Theoretical Computer Science, 19th FST&TCS*. LNCS 1738, Springer, 1999, 258–268.
7. Baaz, M., Veith, H.: Interpolation in Fuzzy Logic. *Archive for Mathematical Logic*, to appear.
8. Baaz, M., Veith, H.: Quantifier Elimination in Fuzzy Logic. *Computer Science Logic, 12th International Workshop, CSL'98*, LNCS 1584, 1999, 399–414.
9. Baaz, M., Veith, H.: An Axiomatization of Quantified Propositional Gödel Logic Using the Takeuti-Titani Rule. Proc. Logic Colloquium 1998, LNML, Springer, to appear.
10. Ciabattoni, A.: *Proof-theoretic Techniques in Many-valued Logics.* Doctoral dissertation, University of Milano, 1999.
11. Dummett, M.: A propositional calculus with denumerable matrix. *J. Symbolic Logic*, 24(1959), 97–106.
12. Dunn, J.M., Meyer, R.K.: Algebraic Completeness Results for Dummett's *LC* and its Extensions. *Z. math. Logik und Grundlagen der Mathematik*, 17(1971), 225–230.
13. Dyckhoff, R.: A Deterministic Terminating Sequent Calculus for Gödel-Dummett Logic. *Logic Journal of the IGPL*, 7(1999), 319–326.
14. Egly, U., Eiter, T., Tompits, H., Woltran, S.: Quantified Boolean logic as framework for non-monotonic reasoning. Unpublished manuscript, Vienna, 1999.
15. Gödel, K.: Zum intuitionistischen Aussagenkalkül. *Anz. Akad. Wiss. Wien*, 69(1932), 65–66.
16. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, 1998.
17. Horn, A.: Logic with truth values in a linearly ordered Heyting algebra. *J. Symbolic Logic*, 27(1962), 159-170.
18. Sonobe, O.: A Gentzen-type Formulation of Some Intermediate Propositional Logics. *Journal of Tsuda College*, 7(1975), 7–14.
19. Takano, M.: Another proof of the strong completeness of the intuitionistic fuzzy logic. *Tsukuba J. Math.* 11(1984), 851–866.
20. Takeuti, G., Titani, T.: Intuitionistic fuzzy logic and intuitionistic fuzzy set theory. *J. Symbolic Logic*, 49(1984), 851–866.
21. Visser, A.: On the Completeness Principle: A Study of Provability in Heyting's Arithmetic. *Annals of Math. Logic*, 22(1982), 263–295.

A Tableau Method for Inconsistency-Adaptive Logics*

Diderik Batens and Joke Meheus**

Centre for Logic and Philosophy of Science
Universiteit Gent, Belgium
Diderik.Batens@rug.ac.be Joke.Meheus@rug.ac.be

Abstract. We present a tableau method for inconsistency-adaptive logics and illustrate it in terms of the two best studied systems. The method is new in that adaptive logics require a more complex structure of the tableaus and of some rules and conditions. As there is no positive test for derivability in inconsistency-adaptive logics, the tableau method is important for providing criteria for derivability.

1 Aim of This Paper

Inconsistency-adaptive logics, surveyed in [8], interpret a set of premises 'as consistently as possible', and hence define a consequence set that is (except for a border case) much richer than that defined by paraconsistent logics.[1] Semantically, they do so by selecting a set of the paraconsistent models (see [2] and [6]). However, they also have a dynamic proof theory—see [1], [3] (the oldest paper), and [6]—that we characterize below.

Consider a theory T that was intended as consistent and has Classical Logic (**CL**) as its underlying logic. If T turns out to be inconsistent, we shall want to replace it by some consistent improvement T'. To this end, we cannot start from T itself, as it is trivial. Replacing **CL** by some (monotonic) paraconsistent logic will give us a theory that is too poor, for example in that Disjunctive Syllogism[2] and other inferential rules are invalid. What we want to improve on is T in its full richness, except for the pernicious consequences of its inconsistency—see [15] for a case study. So, we want to *locate* the inconsistencies, to restrict applications of **CL**-rules whenever inconsistencies are involved, but to apply these rules elsewhere.[3] Similarly, if a database turns out to be inconsistent, but

* The research for this paper was financed by the Fund for Scientific Research – Flanders, and indirectly by the Flemish Minister responsible for Science and Technology (contract BIL98/37). We are indebted to the three referees that reviewed this paper.

** Joke Meheus is a Postdoctoral Fellow of the Fund for Scientific Research – Flanders.

[1] If the premises are consistent, all now existing inconsistency-adaptive logics deliver the same consequence set as classical logic.

[2] Disjunctive Syllogism is invalid in nearly all (monotonic) paraconsistent logics. An exception is the system **AN** from [16], which invalidates Addition.

[3] While monotonic paraconsistent logics invalidate some rules of inference of **CL**, inconsistency-adaptive logics invalidate specific applications of such rules.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 127–142, 2000.

we nevertheless have to rely on it, we shall not give up the **CL**-rules, but restrict them locally, where an inconsistency turns up. The adaptive logics studied in the present paper rely solely on logical considerations. They enable one to analyze the theory or database in its present state, to act on it (if we must), and to devise experiments and other ways to gain more relevant information.[4] They do not in themselves provide a consistent replacement—that will usually require new information and always non-logical preferences.

The phrase "as consistent as possible" is not unambiguous. Inconsistency-adaptive logics may be defined from a variety of paraconsistent logics by a variety of adaptive strategies (that specify the ambiguous phrase)—more variation is possible as appears from [8], that also describes other sorts of adaptive logics.

The derivability relation of adaptive logics is not only undecidable, there is no positive test for it.[5] This is why a *dynamic* proof theory was devised for them. In a dynamic proof, a line added at some stage may be marked (as not any more derivable) at a later stage in view of the understanding of the premises offered by the proof at that stage. At a still later stage, the line may be non-marked again. Apart from derivability at a stage, *final derivability*, which is sound and complete with respect to the semantics, may be defined (in terms of extensions of the proof). So, it is central to specify criteria that warrant that a formula derived at a stage is finally derived. Some such criteria are provided by the dynamic semantics of [5], but more are provided by tableau methods. Apart from its direct interest, the importance of the present paper lies there.

It was shown in [9] that all flat Rescher–Manor consequence relations (Free, Strong, Argued, C-Based, and Weak consequences—see [11]) are characterized by an adaptive logic (studied here or belonging to the same family). Similar results are forthcoming about prioritized such consequence relations (as described in [12]). It also was shown (in [4] and [13]) that several popular 'non-monotonic logics' are characterized by an adaptive logic combined with preferences.

We present tableau methods for two inconsistency-adaptive logics that are defined from the paraconsistent logic **P** (see Sect. 2) by the Minimal Abnormality Strategy and the Reliability Strategy respectively; they will be named $\mathbf{P}^m$ and $\mathbf{P}^r$.[6] We start with a tableau method for **P** and stepwise transform it to a method for the adaptive $\mathbf{P}^m$. We also present the variant for $\mathbf{P}^r$. The tableau methods may be modified to suit various other systems.

---

[4] So, the aim of adaptive logics is drastically different from the non-monotonic systems discussed, for example, in [14].

[5] Where A is a formula and Γ a set of formulas, there is no Turing Machine that, started with a couple $\langle \Gamma, A \rangle$, halts with a positive answer after finitely many steps whenever A follows from Γ. This should not be counted as a weakness of adaptive logics. To the contrary, there is no positive test for important common inference relations, and adaptive logics provide a dynamic proof theory for some of them.

[6] In other papers, the three systems are called **CLuN** (**CL** allowing for gl*u*ts with respect to *n*egation), **ACLuN2**, and **ACLuN1** respectively (pronounce clun, aclun-one and aclun-two); the referees complained about these names. The names of the authors were left unchanged.

As all $\mathbf{P}^r$-models, $\mathbf{P}^m$-models, and **CL**-models (if any) of some set of formulas Γ are **P**-models of Γ, the unqualified "model" will always mean "**P**-model".

2 The Paraconsistent P and the Inconsistency-Adaptive $\mathbf{P}^m$

Negation in **CL** is semantically characterized by (i) $v_M(\sim A) = 1$ if $v_M(A) = 0$ (negation-completeness) and (ii) $v_M(\sim A) = 0$ if $v_M(A) = 1$ (consistency). **P** is obtained by simply dropping (ii). So, in a sense, it is the most straightforward paraconsistent logic obtained from **CL**. **P** is an extremely weak paraconsistent logic,[7] that maximally isolates inconsistencies. For example, some **P**-models verify both $p \wedge q$ and $\sim(p \wedge q)$, but falsify any other inconsistency. As we shall see, this has specific advantages for the inconsistency-adaptive logics defined from **P**. Syntactically, **P** is full positive **CL** plus the axiom schema $A \vee \sim A$.[8]

Let $\mathcal{L}$ be the usual predicative language schema, with $\mathcal{S}$, $\mathcal{C}$, $\mathcal{V}$, $\mathcal{P}^r$, $\mathcal{F}$ and $\mathcal{W}$ respectively the sets of sentential letters, individual constants, individual variables, predicate letters of rank r, formulas, and wffs (closed formulas). To simplify the semantic handling of quantifiers, we extend $\mathcal{L}$ to the pseudo-language schema $\mathcal{L}^+$ by introducing a set of pseudo-constants, $\mathcal{O}$, that has at least the cardinality of the largest model one wants to consider. Let $\mathcal{W}^+$ denote the set of wffs of $\mathcal{L}^+$ (in which $\mathcal{C} \cup \mathcal{O}$ plays the role played by $\mathcal{C}$ in $\mathcal{L}$) and let ${}^\sim\mathcal{W}^+ = \{\sim A \mid \sim A \in \mathcal{W}^+\}$.

A model $M = \langle D, v\rangle$, in which D is a set and v an assignment function, is an interpretation of $\mathcal{W}^+$, and hence of $\mathcal{W}$, which is what we are interested in. The assignment function v is defined by:

C1.1 $v : \mathcal{S} \longrightarrow \{0, 1\}$
C1.2 $v : \mathcal{C} \cup \mathcal{O} \longrightarrow D$ (where $D = \{v(\alpha) \mid \alpha \in \mathcal{C} \cup \mathcal{O}\}$)
C1.3 $v : \mathcal{P}^r \longrightarrow \wp(D^r)$ (the power set of the r-th Cartesian product of D)
C1.4 $v : {}^\sim\mathcal{W}^+ \longrightarrow \{0, 1\}$

The valuation function, $v_M : \mathcal{W}^+ \longrightarrow \{0, 1\}$, determined by M is defined by:

C2.1 where $A \in \mathcal{S}$, $v_M(A) = v(A)$
C2.2 $v_M(\pi^r \alpha_1 \ldots \alpha_r) = 1$ iff $\langle v(\alpha_1), \ldots, v(\alpha_r)\rangle \in v(\pi^r)$
C2.3 $v_M(\alpha = \beta) = 1$ iff $v(\alpha) = v(\beta)$
C2.4 $v_M(\sim A) = 1$ iff $v_M(A) = 0$ or $v(\sim A) = 1$
C2.5 $v_M(A \supset B) = 1$ iff $v_M(A) = 0$ or $v_M(B) = 1$
C2.6 $v_M(A \wedge B) = 1$ iff $v_M(A) = 1$ and $v_M(B) = 1$

[7] De Morgan properties are invalid, and Replacement of Identicals is only valid outside the scope of a negation.

[8] If the language contains $\bot$, characterized by the schema $\bot \supset A$, classical negation is definable (by $\neg A =_{df} A \supset \bot$). Then, **P** may also be seen as **CL** extended with a weak paraconsistent negation $\sim$ (which is intended for formalizing negation). In the semantics below, $v_M(\bot) = 0$ is then added to C2.1; the additions to the tableau methods below are obvious.

C2.7 $v_M(A \vee B) = 1$ iff $v_M(A) = 1$ or $v_M(B) = 1$
C2.8 $v_M(A \equiv B) = 1$ iff $v_M(A) = v_M(B)$
C2.9 $v_M((\forall\alpha)A(\alpha)) = 1$ iff $v_M(A(\beta)) = 1$ for all $\beta \in \mathcal{C} \cup \mathcal{O}$
C2.10 $v_M((\exists\alpha)A(\alpha)) = 1$ iff $v_M(A(\beta)) = 1$ for at least one $\beta \in \mathcal{C} \cup \mathcal{O}$

A is true in M (M verifies A) iff $v_M(A) = 1$; $\Gamma \models A$ iff all models of Γ (the models that verify all members of Γ) verify A; A is valid iff it is verified by all models. For the Soundness and Completeness proofs, we refer to [6].

To characterize $\mathbf{P}^m$, we first define the 'abnormal part' of a model. Let $\exists A$ denote A preceded (in some specified order) by an existential quantifier over any variable free in A.

Definition 1. $Ab(M) =_{df} \{A \mid A \in \mathcal{F}; v_M(\exists(A \wedge \sim A)) = 1\}$.

Definition 2. *M is a* minimally abnormal *model of Γ iff (i) M is a model of Γ, and (ii) there is no model M' of Γ such that $Ab(M') \subset Ab(M)$.*

Definition 3. *$\Gamma \models_{\mathbf{P}^m} A$ iff A is true in all minimally abnormal models of Γ.*

Thus, $\mathbf{P}^m$ is obtained by eliminating all models of Γ that are more inconsistent than required: those for which some other model of Γ verifies (set theoretically) less inconsistencies. In [7], it was proved that, if M is not a minimally abnormal model of Γ, then Γ has a minimally abnormal model M' such that $Ab(M') \subset Ab(M)$ (Strong Reassurance). For the dynamic proof theory, soundness and completeness, and further metatheory, we refer to [6].

Inconsistency-adaptive logics may be defined from many paraconsistent logics. The fact that **P** does not spread inconsistencies causes the adaptive logics defined from it to have a richer consequence set. For example, all minimally abnormal models of $\{\sim(p \wedge q), p \wedge q, \sim p \vee r\}$ falsify $\sim p$ and verify r, whence r is an $\mathbf{P}^m$-consequence of the set.

3 Tableau Method for P

We shall follow [19], but use a slightly different metalanguage and introduce some further terminology. A tableau is started as a list containing each premise (if any) preceded by T and the conclusion preceded by F (*Starting Rule*). The Starting Rule brings a tableau at stage 1; applying a rule to a branch of a tableau at stage s brings the tableau at stage $s + 1$. Where a new arbitrary constant is introduced in [19], we introduce a new pseudo-constant (member of $\mathcal{O}$).[9] The set of constants and pseudo-constants that occur in branch θ at stage s will be denoted by θ_s^{con}. We shall refer to the rules by a T or F followed by a logical constant—for example, $T\equiv$ refers to the left rule in the next paragraph. Finally, a branch or tableau will be said to be *open* iff it is complete and not-closed.

The rules for the propositional fragment are as in [19], except that (i) there is *no rule* for $T\sim A$, and (ii) we need explicit rules for equivalence:[10]

[9] This clearly separates the constants from the dummy constants in the tableau (and will greatly simplify the metatheory of the adaptive logics).

[10] Equivalence cannot be defined in **P**. For example, $\sim(A \equiv B) \equiv \sim((A \supset B) \wedge (B \supset A))$ is not **P**-valid.

$$\frac{T(A \equiv B)}{\begin{array}{l}T(A \supset B)\\ T(B \supset A)\end{array}} \qquad \frac{F(A \equiv B)}{F(A \supset B) \mid F(B \supset A)}$$

The rules for the quantifiers are as follows:

$$\frac{T(\forall\alpha)A}{TA^\alpha_\beta} \qquad \frac{F(\exists\alpha)A}{FA^\alpha_\beta} \qquad \text{provided } \beta \in \theta^{con}_s$$

$$\frac{F(\forall\alpha)A}{FA^\alpha_\beta} \qquad \frac{T(\exists\alpha)A}{TA^\alpha_\beta} \qquad \text{provided } \beta \in \mathcal{O} - \theta^{con}_s$$

If $\theta^{con}_s = \emptyset$ and θ contains formulas of the form $T(\forall\alpha)A$ or $F(\exists\alpha)A$, then $T\alpha = \alpha$ is added to θ for some $\alpha \in \mathcal{O}$. (If you want a name for this, call it NED, as it depends on the supposition that the domain is non-empty.)

Finally, here are the rules for identity:

$$\frac{F\alpha = \alpha}{T\alpha = \alpha} \qquad \frac{T\alpha = \beta}{TA^\gamma_\alpha \equiv A^\gamma_\beta} \qquad \begin{array}{l}\text{provided } A \text{ is primitive, } TA \text{ or } FA \text{ occurs}\\ \text{in the branch, and } \gamma \in \mathcal{C} \cup \mathcal{O} \text{ occurs in } A\end{array}$$

Figure 1 contains simple applications of Modus Ponens, Disjunctive Syllogism, and Modus Tollens—the first is valid in **P**, the latter two are not.

$$\begin{array}{l} Tp \\ Tp \supset q \\ Fq \\ Fp \mid Tq \\ \times \mid \times \end{array} \qquad \begin{array}{l} Tp \\ T{\sim}p \vee q \\ Fq \\ T{\sim}p \mid Tq \\ \quad\;\; \mid \times \end{array} \qquad \begin{array}{l} Tp \\ T{\sim}q \supset {\sim}p \\ Fq \\ F{\sim}q \mid T{\sim}p \\ Tq \mid \\ \times \mid \end{array}$$

Fig. 1. Modus Ponens, Disjunctive Syllogism, and Modus Tollens

Let $f : \mathcal{O} \longrightarrow \mathcal{C} \cup \mathcal{O}$ be a partial function and let A^f be the result of replacing any $\alpha \in \mathcal{O}$ that occurs in A by $f(\alpha)$. We shall say that a model M *agrees* with a branch θ iff, for some f, $v_M(A^f) = 1$ iff $TA \in \theta$, and $v_M(A^f) = 0$ iff $FA \in \theta$.

In preparation of the tableau methods for the adaptive logics, we phrase this in words. A branch θ imposes *requirements* on any $\alpha \in \theta^{con}_s$ by requiring that some members of $\mathcal{W}^+$ in which α occurs are true and others false. To see the resulting requirements on the models that agree with θ, it is sufficient to realize that the pseudo-constants that occur in θ may be replaced (in some systematic way characterized by f) by arbitrary constants or pseudo-constants. The proof of the following theorem is left to the reader.[11]

[11] Suppose that $A_1, \ldots, A_n \not\models B$; then some model M agrees with the sole branch of the tableau at stage 1—select the empty f at this point. Next show that, if M agrees at stage s with a branch that has X as its end point, then it agrees at stage $s + 1$ with some branch passing through X—extend f as pseudo-constants are introduced in the branch.

Theorem 1. *If the tableau for* $A_1, \ldots, A_n \models B$ *is closed, then* $A_1, \ldots, A_n \models B$.

For each open branch θ we define a set of *characteristic* models. Let s be the last stage of the tableau. For each $\alpha \in \theta_s^{con}$, we define an equivalence class $[\alpha]$ fulfilling the following conditions: (i) $\alpha \in [\alpha]$, and (ii) $[\alpha] = [\beta]$ iff $T\alpha = \beta \in \theta$. Moreover, for each $\alpha \in \theta_s^{con}$, we define an equivalence class $[[\alpha]]$ by the following conditions: (i) $\alpha \in [[\alpha]]$ and (ii) $[[\alpha]] = [[\beta]]$ iff $T\alpha = \beta \in \theta$ and, for each formula A and $\gamma \in \mathcal{V}$, $TA^{\gamma}_{\alpha} \in \theta$ iff $TA^{\gamma}_{\beta} \in \theta$ and $FA^{\gamma}_{\alpha} \in \theta$ iff $FA^{\gamma}_{\beta} \in \theta$.[12] Both kinds of equivalence classes are obviously well-defined.[13] Let $g : \mathcal{C} \cup \mathcal{O} \longrightarrow \theta_s^{con}$ be a function such that $g(\alpha) = \alpha$ for all $\alpha \in \theta_s^{con}$.

In terms of these equivalence classes, we define the characteristic model $M = \langle D, v\rangle$ of θ by choosing a function g and stipulating:

CM1 $D = \{[\alpha] \mid \alpha \in \theta_s^{con}\}$.
CM2 For all $C \in \mathcal{S}$, $v(C) = 1$ iff $TC \in \theta$.
CM3 For all $\alpha \in \mathcal{C} \cup \mathcal{O}$, $v(\alpha) = [g(\alpha)]$.
CM4 For all $\pi^r \in \mathcal{P}^r$, $v(\pi^r) = \{\langle[\alpha_1], \ldots, [\alpha_r]\rangle \mid T\pi^r\beta_1 \ldots \beta_r \in \theta$ for some $\beta_1 \in [\alpha_1], \ldots, \beta_r \in [\alpha_r]\}$.
CM5 For all $\sim C(\alpha_1, \ldots, \alpha_n) \in {}^{\sim}\mathcal{W}^{+}$ (where $n \geq 0$), $v(\sim C(\alpha_1, \ldots, \alpha_n)) = 1$ iff, for some $\beta_1, \ldots, \beta_n \in \theta_s^{con}$, $T\sim C(\beta_1, \ldots, \beta_n) \in \theta$, $FC(\beta_1, \ldots, \beta_n) \notin \theta$, and $g(\alpha_1) \in [[\beta_1]]$, ..., and $g(\alpha_n) \in [[\beta_n]]$.

It is helpful to check that, if M and M' are characteristic models of θ, and $v_M(A) \neq v_{M'}(A)$, then A contains some $\alpha \in \mathcal{C} \cup \mathcal{O} - \theta_s^{con}$. It follows that, for the propositional fragment of **P**, an open branch has a unique characteristic model.

Lemma 1. *If* θ *is open,* $T\pi^r\alpha_1 \ldots \alpha_r \in \theta$ *and* $F\pi^r\beta_1 \ldots \beta_r \in \theta$, *then* $[\alpha_1] \neq [\beta_1]$ *or* ... *or* $[\alpha_n] \neq [\beta_n]$.

Proof. Suppose that the antecedent is true and that $[\alpha_1] = [\beta_1]$ and ... and $[\alpha_n] = [\beta_n]$. So, $T\alpha_1 = \beta_1, \ldots, T\alpha_n = \beta_n \in \theta$ by the definition of $[\alpha]$. But then θ cannot be open in view of rules $T{=}$ and $T{\equiv}$. □

Lemma 2. *A characteristic model* M *of an open branch* θ *agrees with* θ.

Proof. Let $M = \langle D, v\rangle$ be a characteristic model of some open θ. Consider

(†) If $TC \in \theta$ then $v_M(C) = 1$; if $FC \in \theta$ then $v_M(C) = 0$.

[12] The condition is obviously fulfilled if γ is not free in A or if other variables are free in A.

[13] The only thing worth mentioning is that, if $\alpha, \beta \in \theta_s^{con}$ and $[\alpha] = [\beta]$, then there are $\gamma_1, \ldots, \gamma_n \in \theta_s^{con}$ such that $T\alpha = \gamma_1, T\gamma_1 = \gamma_2, \ldots, T\gamma_n = \beta \in \theta$. But then $T{=}$, $T{\equiv}$, and $T{\supset}$ warrant that $T\alpha = \beta \in \theta$. In the construction below in the text, $[\alpha] = [\beta]$ warrants that $v(\alpha) = v(\beta)$, whereas $[[\alpha]] = [[\beta]]$ moreover warrants that $v_M(A^{\gamma}_{\alpha}) = v_M(A^{\gamma}_{\beta})$ for all A and for all $\gamma \in \mathcal{V}$.

We first show that (†) holds for primitive formulas. The proof is obvious for members of $\mathcal{S}$.

If $T\alpha = \beta \in \theta$, then $v(\alpha) = [\alpha] = [\beta] = v(\beta)$ and hence $v_M(\alpha = \beta) = 1$. If $F\alpha = \beta \in \theta$, then $T\alpha = \beta \notin \theta$ (as θ is open), whence $[\alpha] \neq [\beta]$ in view of the definition of $[\alpha]$.

If $T\pi^r\alpha_1 \dots \alpha_r \in \theta$, then $v_M(\pi^r\alpha_1 \dots \alpha_r) = 1$ in view of CM4. If $F\pi^r\alpha_1 \dots \alpha_r \in \theta$, then $v_M(\pi^r\alpha_1 \dots \alpha_r) = 0$ in view of CM4 and Lemma 1.

We show that (†) holds for all formulas by an induction on their complexity. We consider only two cases, the proof of the others being analogous or straightforward.

Case 1. If $T{\sim}A \in \theta$, then $FA \in \theta$ or $FA \notin \theta$. So, $v_M({\sim}A) = 1$ in view of C2.4 and either the induction hypothesis or CM5.

Case 6. $T(\forall\alpha)C \in \theta$. In view of rule $T\forall$, $TC^\alpha_\beta \in \theta$ for all $\beta \in \theta^{con}_s$. By the induction hypothesis, $v_M(C^\alpha_\beta) = 1$ for all $\beta \in \theta^{con}_s$. But then $v_M(C^\alpha_\gamma) = 1$ for all $\gamma \in \mathcal{C} \cup \mathcal{O}$ in view of the definition of g, and CM3–5. Hence $v_M((\forall\alpha)C) = 1$ in view of C2.9. □

Theorem 2. *If the tableau for $A_1, \dots, A_n \models B$ is open, then $A_1, \dots, A_n \not\models B$.*

Proof. Suppose that the antecedent is true, that θ is an open branch of the tableau, and that M is a characteristic model of θ. In view of Lemma 2, M agrees with θ and hence verifies $A_1, \dots, A_n$ and falsifies B. □

As any tableau for the propositional case is completed after finitely many steps:

Theorem 3. *The propositional fragment of* **P** *is decidable.*

4 Tableau Method for $\mathbf{P}^m$

To decide whether $A_1, \dots, A_n \models_{\mathbf{P}^m} B$, we need to identify, in the **P**-tableau, the branches that correspond to $\mathbf{P}^m$-models of $A_1, \dots, A_n$. The **P**-tableau method is unfit for this. Step by step, we shall introduce modifications to overcome this problem. If $T{\sim}A$ occurs on a branch, any model that agrees with it may be inconsistent, for example if A is **P**-valid, but the present method fails to reveal this. This is repaired by the rule

$$\frac{T{\sim}A}{TA \mid FA}$$

which is obviously redundant in the tableau method for **P**. The rule is illustrated in the left and middle tableau in Fig. 2. Next, we need to keep track of the formulas *that stem from the premises*. This will be done by labeling them—we shall prefix them with a dot, as in Fig. 2.

Before we comment on the examples, let us make things precise. Applying the Starting Rule, we *label* the formulas of the form TA (the premises). Next, when a rule is applied to a labeled formula, the result of the application is also

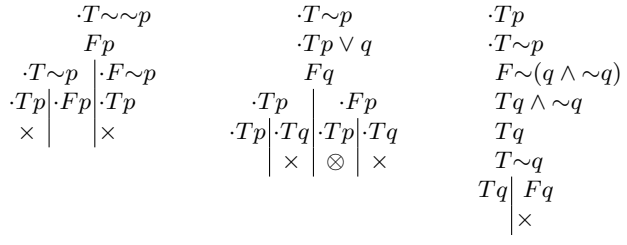


Fig. 2. Three tableaus illustrating $T\sim$ and labeling

labeled.[14] Let $\overline{\theta}$ be the set of labeled formulas in θ—thus $TA \in \overline{\theta}$ denotes that TA occurs labeled in θ and $TA \in \theta$ that it occurs in θ, whether labeled or not. A set $\overline{\theta}$ may itself be *closed*, *complete*, or *open* (the definitions are obvious). Moreover, we define a *complete tableau* as one in which each branch θ is closed or complete and each $\overline{\theta}$ is closed or complete. We *postpone* the definitions of closed and open *tableaus* (and provisionally stick to an intuitive approach).

Without the modifications, the leftmost tableau in Fig. 2 would just consist of the two top nodes. Applying rule $T\sim$, we obtain three branches (we shall always number branches from left to right). The set θ_2 is open; θ_1 and θ_3 are closed. All three interpretations of the premises, $\overline{\theta_1}$–$\overline{\theta_3}$, are open, but only $\overline{\theta_3}$ is consistent. As all branches that correspond to a minimally inconsistent (in this case consistent) interpretation of the premises are closed, $\sim\sim p \models_{\mathbf{P}^m} p$. The tableau is easily modified to show that $\sim\sim p, \sim p \not\models_{\mathbf{P}^m} p$.

In the second tableau, the $\otimes$ indicates that $\overline{\theta_3}$ is an incoherent interpretation of the premises—no model of the premises corresponds to it. $\overline{\theta_4}$ represents the only consistent interpretation of the premises. As it closes, $\sim p, p \vee q \models_{\mathbf{P}^m} q$. The tableau is easily modified to show that $p, \sim p, p \vee q \not\models_{\mathbf{P}^m} q$.

The third tableau reveals that any model falsifying the conclusion (see θ_1) is more inconsistent than some models of the premises (see $\overline{\theta_1}$ and $\overline{\theta_2}$). Hence, $p, \sim p \models_{\mathbf{P}^m} \sim(q \wedge \sim q)$. The tableau is easily modified to show that $q, \sim q \not\models_{\mathbf{P}^m} \sim(q \wedge \sim q)$. Needless to say, we still have to make the method precise and to prove its adequacy.

Even with these modifications, the branches do not provide a correct indication of the abnormal part of the models that correspond to them. If a is the only member of $\mathcal{C} \cup \mathcal{O}$ that occurs in a branch, then the formulas $TPa \wedge \sim Pa$ and $T(\forall x)(Px \wedge \sim Px)$ have exactly the same effect. However, some models that agree with a branch in which the first formula occurs, falsify $Pb \wedge \sim Pb$, whereas all models that agree with a branch in which the latter formula occurs, verify $Pb \wedge \sim Pb$. This difficulty is solved by the *finishing procedure*.

[14] Consider an application of $T\forall$ to $\cdot T(\forall x)Px$, resulting in $\cdot TPa$. Even if a occurs only in the conclusion, it is justified that TPa is labeled: any model of the premises verifies Pa. Similarly if the result involves a pseudo-constant. Applications of $F\exists$ and $T=$ are justified in the same way.

To simplify the notation, let us say that, in a completed tableau, a branch θ is in stage S_0 if $\overline{\theta}$ is open—the (earlier introduced) stages $1, 2, \ldots$ refer to the tableau—and let us suppose that $o_1, o_2, \ldots \in \mathcal{O}$ do *not* occur in any branch at stage S_0. If we add $To_1 = o_1$ to branch θ at stage S_0,[15] the rules $T\forall$ and $F\exists$ will lead to o_1-instances of quantified formulas. While we again complete the tableau, θ may be extended into several branches. The resulting branch or branches will be said to be at stage S_1 and will be called *children* of θ. Adding $To_2 = o_2$ to such a child θ' will make θ' the parent of one or more children which are at stage S_2, etc. From now on, o_i, o_j, o_1, ... will always refer to pseudo-constants added during the finishing procedure.

Definition 4. *Where θ is at stage S_i and is a child of θ', o_i is* redundant *in θ iff there is an o_j $(1 \leq j < i)$ such that, for all A, (i) $TA^x_{o_i} \in \overline{\theta}$ iff $TA^x_{o_j} \in \overline{\theta'}$, and (ii) $FA^x_{o_i} \in \overline{\theta}$ iff $FA^x_{o_j} \in \overline{\theta'}$.*

The effects of the addition of $To_i = o_i$ on non-labeled formulas are immaterial to determine whether o_i is redundant in the branch.

Definition 5. *A branch θ at stage S_i is* finished *iff o_i is redundant in θ.*

Definition 6. *A* tableau *is* finished *iff, for each branch θ, θ is finished or $\overline{\theta}$ is closed.*

As any branch contains at most a finite number of quantifiers:

Theorem 4. *Any completed tableau can be finished.*

A simple example involving the finishing procedure is listed in Fig. 3. We did not apply the rule $T\sim$ to formulas of the form $\cdot T{\sim}P\alpha$ (in the left subbranch, θ_{i1}, $\cdot TP\alpha$ would be duplicated, for the right one, $\overline{\theta_{i2}}$ is closed).

The relevant properties of the o_i in the descendants of θ are only determined by formulas of the forms $T(\forall\alpha)A$ and $F(\exists\alpha)A$ that occur in θ. In this sense, the descendants of θ reveal the *requirements* that θ imposes on the constants and pseudo-constants in models that agree with θ. Some θ and $\overline{\theta}$ specify alternative requirements. For example, if $T(\forall x)(Px \supset Qx), T(\forall x)(Rx \supset Sx) \in \overline{\theta}$, there are four alternative requirements (neither P nor R, Q and not R, ...). At least one of these will be fulfilled by the o_i in each descendant of θ. In some descendants of θ, henceforth called the *richest* descendants, *each* alternative requirement is fulfilled by one or other o_i. This is obvious in view of the finishing procedure, and we shall rely upon it below.

Our last step consists in defining the *marked* branches—those that no minimally abnormal model of the premises strongly agrees with—see below for strongly agreeing.

Let $A^\star$ be the set of all $B \in \mathcal{F}$ for which $\exists B$ is obtained by existential generalization from $A \in \mathcal{W}^+$. Remark that members of $\mathcal{C}$ that occur in A occur in some members of $A^\star$ and not in others, and that $A \in A^\star$ if $A \in \mathcal{W}$.[16]

[15] It is immaterial whether $To_1 = o_1$ is labeled or not.

[16] Each member of $A^\star$ is obtained by systematically replacing all pseudo-constants in A (if any), and possibly some constants in A, by variables.

$$
\begin{array}{l}
\cdot T(\forall x)(Px \wedge (\sim Px \vee Qx)) \\
\ F(\forall x)(Px \wedge Qx) \\
\ FPo \wedge Qo \\
\cdot TPo \wedge (\sim Po \vee Qo) \\
\cdot TPo \\
\cdot T\sim Po \vee Qo
\end{array}
$$

$$
\begin{array}{cccc}
\cdot T\sim Po & & \cdot TQo & \\
FPo & FQo & FPo & FQo \\
\times & & \times & \times
\end{array}
$$

Each branch is extended as follows by the finishing procedure:

$$
\begin{array}{l}
To_1 = o_1 \\
\cdot TPo_1 \wedge (\sim Po_1 \vee Qo_1) \\
\cdot TPo_1 \\
\cdot T\sim Po_1 \vee Qo_1
\end{array}
$$

$$
\begin{array}{lll}
\cdot T\sim Po_1 & & \cdot TQo_1 \\
\ To_2 = o_2 & & \ To_2 = o_2 \\
\cdot TPo_2 \wedge (\sim Po_2 \vee Qo_2) & & \cdot TPo_2 \wedge (\sim Po_2 \vee Qo_2) \\
\cdot TPo_2 & & \cdot TPo_2 \\
\cdot T\sim Po_2 \vee Qo_2 & & \cdot T\sim Po_2 \vee Qo_2
\end{array}
$$

$$
\begin{array}{llll}
\cdot T\sim Po_2 & \cdot TQo_2 & \cdot T\sim Po_2 & \cdot TQo_2 \\
 & \ To_3 = o_3 & \ To_3 = o_3 & \\
 & \cdot TPo_3 \wedge (\sim Po_3 \vee Qo_3) & \cdot TPo_3 \wedge (\sim Po_3 \vee Qo_3) & \\
 & \cdot TPo_3 & \cdot TPo_3 & \\
 & \cdot T\sim Po_3 \vee Qo_3 & \cdot T\sim Po_3 \vee Qo_3 & \\
 & \cdot T\sim Po_3 \quad \cdot TQo_3 & \cdot T\sim Po_3 \quad \cdot TQo_3 &
\end{array}
$$

Fig. 3. The tableau for $(\forall x)(Px \wedge (\sim Px \vee Qx)) \models (\forall x)(Px \wedge Qx)$.

Definition 7. $Ab(\theta) =_{df} \{B \mid B \in A^{\star}; TA, T\sim A \in \theta\}$; $Ab(\overline{\theta}) =_{df} \{B \mid B \in A^{\star}; TA, T\sim A \in \overline{\theta}\}$.

These infinite sets may be represented by the finite sets of the formulas that behave inconsistently in θ and $\overline{\theta}$ respectively. The same holds for the sets that characterize the 'abnormal behaviour' of the $\alpha \in \theta_s^{con}$ in θ and in $\overline{\theta}$:

Definition 8. $Ab^{\alpha}(\theta) = \{A \mid A \in Ab(\theta); \alpha \textit{ occurs in } A\}$; $Ab^{\alpha}(\overline{\theta}) = \{A \mid A \in Ab(\overline{\theta}); \alpha \textit{ occurs in } A\}$.

Given a finished tableau (at stage s), we define:

Definition 9. *An open* branch θ *is* marked *iff some* θ' *(for which* $\overline{\theta'}$ *is open) is such that (i)* $Ab(\overline{\theta'}) \subset Ab(\theta)$ *and, for all* $o_i \in \theta_s^{con}$, *there is an* $o_j \in {\theta'}_s^{con}$ *such that* $Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{o_i}(\theta)$, *or (ii)* $Ab(\overline{\theta'}) = Ab(\theta)$ *and, for all* $o_i \in \theta_s^{con}$, *there is an* $o_j \in {\theta'}_s^{con}$ *such that* $Ab^{o_j}(\overline{\theta'}) \subset Ab^{o_i}(\theta)$.

The definition should be taken literally: θ is also marked if it is more abnormal than $\overline{\theta}$. In the example in Fig. 3, the finishing procedure leads to 24 branches. Of these, only $Ab(\overline{\theta_{24}})$ is empty, and hence all open branches are marked. An

example that fully illustrates Definition 9 would require several pages. We shall, however, explain the definition below.

Definition 10. *A* $\mathbf{P}^m$*-tableau is* closed *iff all its branches are either closed or marked. A* $\mathbf{P}^m$*-tableau is* open *iff it is finished and not closed.*

This ends the description of the tableau method. Given the complexity of the models, the method is quite simple—even extremely simple at the propositional level, for which it forms a decision method.

To prove that the method is adequate, we only need one more definition and one small change. We shall say that a model M *strongly agrees* with a branch θ of a finished tableau (at stage s) iff M agrees with θ and each constant ($\alpha \in \mathcal{C}$) that does not occur in θ fulfills in M the requirements imposed on some $o_i \in \theta_s^{con}$. Formally: for all $\alpha \in \mathcal{C} - \theta_s^{con}$, there is a f—see Sect. 3—such that $f(o_i) = \alpha$ for some o_i in θ, $v_M(A^f) = 1$ for all $TA \in \theta$, and $v_M(A^f) = 0$ for all $FA \in \theta$. That a model M *strongly agrees* with $\overline{\theta}$ is defined similarly. The change is that we impose a similar requirement on *characteristic models*: for all $\alpha \in \mathcal{C} - \theta_s^{con}$, $g(\alpha) = o_i$, for some $o_i \in \theta_s^{con}$, and that we extend the notion of a characteristic model to the sets $\overline{\theta}$—the required changes are obvious.

At this point, the importance of the finishing procedure appears. The pseudo-constants o_i that occur in θ indicate the requirements θ and $\overline{\theta}$ impose, in the models that correspond to them, on constants that do not occur in θ. Let $Ab^{\alpha}(M) = \{A \mid A \in Ab(M); \alpha \text{ occurs in } A\}$. We may consider $Ab(M)$ as consisting of two parts, $Ab^{\dagger}(M)$, containing the members of $Ab(M)$ in which occur only constants that also occur in the premises and conclusion,[17] and $Ab(M) - Ab^{\dagger}(M)$, containing the other members of $Ab(M)$, viz. those in which some other constant occurs. In other words, $Ab(M) - Ab^{\dagger}(M)$ is the union, for all $\alpha \in \mathcal{C} - \theta_s^{con}$, of $Ab^{\alpha}(M)$. Where $Ab^{\dagger}(M)$ is adequately measured by $Ab(\theta)$, $Ab(M) - Ab^{\dagger}(M)$ depends on the o_i in θ, viz. on the $Ab^{o_i}(\theta)$; similarly for $\overline{\theta}$ and $Ab^{o_i}(\overline{\theta})$.

Consider three branches, θ, θ', and θ'', of a tableau and let the labeled inconsistent o_i-formulas in them be as follows:

$\overline{\theta}$: $Po_1,\ Qo_1,\ Po_2,\ Qo_2$
$\overline{\theta'}$: $Po_1,\ Po_2,\ Qo_2,\ Po_3,\ Qo_3$
$\overline{\theta''}$: $Po_1,\ Qo_2,\ Ro_3,\ Ro_4$

Suppose that $Ab(\overline{\theta}) \subset Ab(\theta'')$ and $Ab(\overline{\theta'}) \subset Ab(\theta'')$. Branch θ'' will not be marked in view of $\overline{\theta}$ or $\overline{\theta'}$, because no $o_i \in \theta_s^{con}$ is such that $Ab^{o_i}(\overline{\theta}) \subseteq Ab^{o_3}(\theta'')$; similarly for the $o_i \in \theta'^{con}_s$. The reason for this is quite simple. In a model M'' that strongly agrees with $\overline{\theta''}$, some constant, say c, may be such that $Ab^c(M'') = \{Rc\}$. However, in any model M that strongly agrees with $\overline{\theta}$, $Ab^c(M)$ will be $\{Pc, Qc\}$. So, $Ab(M) \not\subseteq Ab(M'')$. Also, if $Ab(\overline{\theta}) = Ab(\theta')$, branch θ' is not marked in view of $\overline{\theta}$ because no $o_i \in \theta_s^{con}$ is such that $Ab^{o_i}(\overline{\theta}) \subset Ab^{o_1}(\theta')$. The effect of this becomes apparent in the proof of Theorem 8.

In view of the finishing procedure (and the fact that each θ has a *richest* descendant), the proof of the following lemma is obvious:

[17] So, if $A \in Ab(M)$ and *no* constant occurs in A, then $A \in Ab^{\dagger}(M)$.

Lemma 3. *A model M agrees with some completed branch θ (at stage S_0) iff M strongly agrees with some finished descendant of θ. A model M agrees with some completed $\overline{\theta}$ iff θ has a descendant θ' such that M strongly agrees with $\overline{\theta'}$.*

Lemma 4. *Any model M of $A_1, \ldots, A_n$ strongly agrees with some open $\overline{\theta}$ in the tableau for $A_1, \ldots, A_n \models B$. Any model M of $A_1, \ldots, A_n$ that falsifies B strongly agrees with some open branch θ in the tableau for $A_1, \ldots, A_n \models B$.*

Proof. At stage 1 of the tableau, any model M of $A_1, \ldots, A_n$ agrees with a non-closed $\overline{\theta}$, viz. the sole one. By the standard inductive proof, this is extended to any subsequent stage relying on the correctness of the tableau rules with respect to the **P**-semantics and on the **P**-validity of $o_i = o_i$.

Let M agree with an open $\overline{\theta}$ at stage S_0. In view of Lemma 3, θ has a descendant θ' such that M strongly agrees with $\overline{\theta'}$.

Similarly for a model of $A_1, \ldots, A_n$ that falsifies B. □

Lemma 5. *If M strongly agrees with some finished open θ (respectively $\overline{\theta}$) of a tableau (at stage s), then*

(i) for all $\alpha \in \theta_s^{con}$, $Ab^{\alpha}(M) \supseteq Ab^{\alpha}(\theta)$ (respectively $Ab^{\alpha}(M) \supseteq Ab^{\alpha}(\overline{\theta})$),
(ii) for all $\alpha \in \mathcal{C} - \theta_s^{con}$, there is an $o_i \in \theta_s^{con}$ such that $Ab^{\alpha}(M) \supseteq Ab^{o_i}(\theta)$ (respectively $Ab^{\alpha}(M) \supseteq Ab^{o_i}(\overline{\theta})$),
(iii) $Ab^{\dagger}(M) \supseteq Ab(\theta)$ (respectively $Ab^{\dagger}(M) \supseteq Ab(\overline{\theta})$).

Proof. Statements (i)–(ii) follow directly from the fact that M strongly agrees with θ (respectively $\overline{\theta}$). Statement (iii) is a direct consequence of (i). □

Lemma 6. *In a finished tableau (at stage s), any characteristic model M of some open θ (respectively $\overline{\theta}$), is such that*

(i) for all $\alpha \in \theta_s^{con}$, $Ab^{\alpha}(M) = Ab^{\alpha}(\theta)$ (respectively $Ab^{\alpha}(M) = Ab^{\alpha}(\overline{\theta})$),
(ii) for all $\alpha \in \mathcal{C} - \theta_s^{con}$, there is an $o_i \in \theta_s^{con}$ such that $Ab^{\alpha}(M) = Ab^{o_i}(\theta)$ (respectively $Ab^{\alpha}(M) = Ab^{o_i}(\overline{\theta})$),
(iii) for all $\alpha \in \mathcal{O} - \theta_s^{con}$, there is a $\beta \in \theta_s^{con}$ such that $Ab^{\alpha}(M) = Ab^{\beta}(\theta)$ (respectively $Ab^{\alpha}(M) = Ab^{\beta}(\overline{\theta})$),
(iv) $Ab^{\dagger}(M) = Ab(\theta)$ (respectively $Ab^{\dagger}(M) = Ab(\overline{\theta})$).

Proof. Statements (i)–(iii) are obvious in view of the definition of a characteristic model. Statement (iv) is a direct consequence of (i). □

Lemma 7. *If a branch θ of a finished tableau (at stage s) is marked, then no minimally abnormal model M of the premises strongly agrees with θ.*

Proof. Let θ be marked in view of $\overline{\theta'}$ but open, and suppose that M is a minimally abnormal model of the premises that strongly agrees with θ.

Case 1. $Ab(\overline{\theta'}) \subset Ab(\theta)$ and, for all $o_i \in \theta_s^{con}$, there is an $o_j \in {\theta'}_s^{con}$ such that $Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{o_i}(\theta)$. By Lemma 6, $Ab^{\dagger}(M') = Ab(\overline{\theta'})$ for any characteristic model M' of $\overline{\theta'}$. By Lemma 5, $Ab(\theta) \subseteq Ab^{\dagger}(M)$ for any model M that agrees with θ. Hence, $Ab^{\dagger}(M') \subset Ab^{\dagger}(M)$ for some characteristic model M' of $\overline{\theta'}$.

It remains to be shown that, for some characteristic model M' of $\overline{\theta'}$, $Ab(M') - Ab^{\dagger}(M') \subseteq Ab(M) - Ab^{\dagger}(M)$, which means that $Ab^{\alpha}(M') \subseteq Ab^{\alpha}(M)$ for all $\alpha \in \mathcal{C} - \theta_s^{con}$. We construct such a model by selecting a suitable function g. Consider some $\alpha \in \mathcal{C} - \theta_s^{con}$. By Lemma 5, $Ab^{o_i}(\theta) \subseteq Ab^{\alpha}(M)$ for some $o_i \in \theta_s^{con}$. We pick such o_i. As θ is marked, there is an $o_j \in {\theta'}_s^{con}$ such that $Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{o_i}(\theta)$. As M' is a characteristic model of $\overline{\theta'}$, $Ab^{o_j}(M') = Ab^{o_j}(\overline{\theta'})$ by Lemma 6. So, by setting $g(\alpha) = o_j$, we make sure that $Ab^{\alpha}(M') = Ab^{o_j}(M') = Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{o_i}(\theta) \subseteq Ab^{\alpha}(M)$.

Case 2. $Ab(\overline{\theta'}) = Ab(\theta)$ and, for all $o_i \in \theta_s^{con}$, there is an $o_j \in {\theta'}_s^{con}$ such that $Ab^{o_j}(\overline{\theta'}) \subset Ab^{o_i}(\theta)$. The proof proceeds exactly as in case 1, except that the chosen g now warrants that, for any $\alpha \in \mathcal{C} - \theta_s^{con}$, $Ab^{\alpha}(M') \subset Ab^{\alpha}(M)$ as desired. □

Theorem 5. *If the* $\mathbf{P}^m$*-tableau for* $A_1, \ldots, A_n \models B$ *is closed, then* $A_1, \ldots, A_n \models_{\mathbf{P}^m} B$.

Proof. Suppose that some minimally abnormal model M of the premises falsifies the conclusion. In view of Lemmas 4 and 7, M strongly agrees with some non-marked open branch θ in the finished tableau. Hence, the tableau is not closed. □

Lemma 8. *If a branch* θ *of a finished tableau (at stage* s*) is open and not marked, then some minimally abnormal model of the premises strongly agrees with* θ.

Proof. Suppose that the antecedent is true and the conclusion false. Consider a characteristic model M of θ in which, for each $o_i \in \theta_s^{con}$, $o_i = g(\alpha)$ for some $\alpha \in \mathcal{C} - \theta_s^{con}$. As M is not a minimally abnormal model of the premises, it follows by Strong Reassurance (proved as Theorem 1 of [7]), that $Ab(M') \subset Ab(M)$ for some *minimally abnormal* model M' of the premises. But then $Ab^{\dagger}(M') \subseteq Ab^{\dagger}(M)$. In view of Lemma 4, M' strongly agrees with some open $\overline{\theta'}$. By Lemma 5, $Ab(\overline{\theta'}) \subseteq Ab^{\dagger}(M')$. By Lemma 6, $Ab^{\dagger}(M) = Ab(\theta)$. Hence, $Ab(\overline{\theta'}) \subseteq Ab(\theta)$.

Case 1. $Ab(\overline{\theta'}) \subset Ab(\theta)$. Consider an arbitrary $o_i \in \theta_s^{con}$ and a $\beta \in \mathcal{C} - \theta_s^{con}$ such that $g(\beta) = o_i$. As M is a characteristic model of θ, $Ab^{\beta}(M) = Ab^{o_i}(\theta)$. By Lemma 5, $Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{\beta}(M')$ for some $o_j \in {\theta'}_s^{con}$. But $Ab(M') \subset Ab(M)$ entails that $Ab^{\beta}(M') \subseteq Ab^{\beta}(M)$. It follows that $Ab^{o_j}(\overline{\theta'}) \subseteq Ab^{o_i}(\theta)$. As this holds for any $o_i \in \theta_s^{con}$, θ is marked, which contradicts the supposition.

Case 2. $Ab(\overline{\theta'}) = Ab(\theta)$. The proof proceeds literally as for case 1, except for the obvious replacements of "$\subset$" by "$=$" and of "$\subseteq$" by "$\subset$". □

This gives us at once:

Theorem 6. *If the* $\mathbf{P}^m$*-tableau for* $A_1, \ldots, A_n \models B$ *is open, then* $A_1, \ldots, A_n \not\models_{\mathbf{P}^m} B$.

Notwithstanding Theorems 5 and 6, there is no positive test for $\mathbf{P}^m$-semantic consequence. If $A_1, \ldots, A_n \models_{\mathbf{P}^m} B$ but $A_1, \ldots, A_n \not\models_{\mathbf{P}} B$, the construction might never be complete (nor a fortiori finished) and hence non-closed branches

might not be marked. As any finite premise set leads to a finite completed tableau in the propositional case, the $\mathbf{P}^m$-tableau method forms a decision method for the propositional fragment.

5 $\mathbf{P}^r$ and Its Tableau Method

$\mathbf{P}^r$ is obtained from $\mathbf{P}$ by the Reliability Strategy—see [3] for the propositional fragment of $\mathbf{P}^r$ and [6] for a full study of the predicate logic. We first briefly characterize $\mathbf{P}^r$, and then offer some motivation. Let

$$DEK(A_1, \ldots, A_n) = \exists(A_1 \wedge \sim A_1) \vee \ldots \vee \exists(A_n \wedge \sim A_n).$$

$DEK(A_1, \ldots, A_n)$ is a *minimal DEK-consequence* of Γ iff it is a $\mathbf{P}$-consequence of Γ, and no result of dropping some disjunct from $DEK(A_1, \ldots, A_n)$ is. $U(\Gamma)$, the set of formulas that is unreliable with respect to Γ, is defined by:

Definition 11. $U(\Gamma) = \{A \mid$ *for some* $B_1, \ldots, B_n$, $DEK(A, B_1, \ldots, B_n)$ *is a minimal DEK-consequence of* $\Gamma\}$.

Definition 12. *A model* M *of* Γ *is* reliable *iff* $Ab(M) \subseteq U(\Gamma)$.

Definition 13. $\Gamma \models_{\mathbf{P}^r} A$ *iff all reliable models of* Γ *verify* A.

If $DEK(A_1, \ldots, A_n)$ is a *minimal DEK-consequence* of Γ, the latter informs you that one of the A_i behaves inconsistently, but does not inform you which one. So, it is sensible not to rely on the consistent behaviour of any of the A_i. This is exactly what the Reliability Strategy comes to. Moreover, it is attractive (and much simpler than the Minimal Abnormality Strategy) from a computational point of view.

The Reliability Strategy defines a slightly poorer consequence set than the Minimal Abnormality Strategy. The basic difference is best seen from a propositional example: r is a $\mathbf{P}^m$-consequence, but not a $\mathbf{P}^r$-consequence of $\{(p \wedge \sim p) \vee (q \wedge \sim q), (p \wedge \sim p) \vee r, (q \wedge \sim q) \vee r\}$. In this case, $\mathbf{P}^m$ rules out models that verify *both* $(p \wedge \sim p)$ and $(q \wedge \sim q)$, whereas $\mathbf{P}^r$ does not. Apart from such cases, both strategies lead to the same consequences.

The tableau method for $\mathbf{P}^r$ differs only from that for $\mathbf{P}^m$ in the definition of marked branches. As no room for metatheoretic proofs is left, we mention that it is easy to prove, in view of Sect. 4 and [6, § 5–6], that the definition given below is correct. We shall always consider the finished tableau (at stage s) for some $\Gamma \models A$, where Γ is finite.

Let us say that a branch θ verifies $A \wedge \sim A$ iff $TA, T{\sim}A \in \theta$, and that θ verifies $(A_1 \wedge \sim A_1) \vee \ldots \vee (A_n \wedge \sim A_n)$ iff it verifies some of the $A_i \wedge \sim A_i$. It is easy enough to identify the (finitely many) disjunctions of contradictions that are verified by all $\overline{\theta}$, and to sift out the minimal ones. $U^{\ddagger}(\Gamma)$ is the smallest set fulfilling: (i) if $(A \wedge \sim A) \vee \ldots$ is such a minimal disjunction, $A \in U^{\ddagger}(\Gamma)$, and (ii) where Σ is the set of pseudo-constants that occur in the tableau, if $A \in U^{\ddagger}(\Gamma)$, h is a bijection of Σ on Σ, and B is obtained by replacing in A any $\alpha \in \mathcal{O}$ by $h(\alpha)$, then $B \in U^{\ddagger}(\Gamma)$. Finally, let $Ab^{\ddagger}(\theta) = \{A \mid TA, T{\sim}A \in \theta\}$.

Definition 14. *A branch θ of a finished tableau for $\Gamma \models A$, is* marked *iff it is open and $Ab^{\ddagger}(\theta) \not\subseteq U^{\ddagger}(\Gamma)$.*

Open and *closed* tableaus are defined as for $\mathbf{P}^m$, this time with reference to Definition 14. It is provable that:

Theorem 7. *If the $\mathbf{P}^r$-tableau for $\Gamma \models A$ is closed, then $\Gamma \models_{\mathbf{P}^r} A$. If the $\mathbf{P}^r$-tableau for $\Gamma \models A$ is open, then $\Gamma \not\models_{\mathbf{P}^r} A$.*

As might be expected, the tableau method for $\mathbf{P}^r$ is much simpler than that for $\mathbf{P}^m$, both because $U^{\ddagger}(\Gamma)$ and $Ab^{\ddagger}(\theta)$ are *finite* sets of pseudo-wffs, and because marking is much simpler.

6 Some Remarks in Conclusion

For the propositional fragments, the methods reduce to something very simple. One constructs the tableau and makes sure any non-closed $\overline{\theta}$ is completed (finishing does not apply). $Ab(\theta) = \{A \mid TA, T{\sim}A \in \theta\}$; similarly for $Ab(\overline{\theta})$. For $\mathbf{P}^m$, θ is marked iff $Ab(\overline{\theta_i}) \subset Ab(\theta)$ for some $\overline{\theta_i}$. For $\mathbf{P}^r$, $U\{A_1, \ldots, A_n\}$ is directly read off from the non-closed $\overline{\theta_i}$, and θ is marked iff $Ab(\theta) \not\subseteq U\{A_1, \ldots, A_n\}$. Both methods are decision procedures for derivability at the propositional level.

As explained in Sect. 1, the special importance of the present results derives from the absence of a positive test for derivability in $\mathbf{P}^r$ and $\mathbf{P}^m$. The tableau methods nevertheless enable one to decide that some conclusions are consequences of a set of premises. Given the relation between tableau methods and proofs, the former allow one to formulate *criteria* for final derivability in terms of proofs.

Several simplifications and shortcuts are possible. For example, the methods may be turned into unsigned tableau methods by relying on the defined classical negation: TA and FA are then respectively replaced by A and $\neg A$. The tableau methods may also be made more efficient by applying criteria that allow one to mark branches before they are finished or even completed (or by marking the labeled subsets of a branch before they are finished or completed) and hence stop applying rules to them. Of special interest is dynamic marking. This procedure introduces provisional marks in view of the present stage of the tableau. These marks must be updated at each stage, but 'guide' the construction (a closed/open tableau is reached more quickly). All this will be presented in a separate paper.

As these are the first published results on tableau methods for adaptive logics, there are many open problems concerning other adaptive logics—see, e.g., [18], [20], [16], and [13]—including ampliative logics.[18]

[18] Inconsistency-adaptive logics and other corrective adaptive logics deliver consequence sets that are subsets of the **CL**-consequence set if the latter is trivial. Ampliative adaptive logics (compatibility, abduction, ...) deliver consequence sets that are supersets of the **CL**-consequence set, but still are not trivial (and are determined by some normality suppositions).

References

1. D. Batens. Dynamic dialectical logics as a tool to deal with and partly eliminate unexpected inconsistencies. In J. Hintikka and F. Vandamme, editors, *The Logic of Discovery and the Logic of Discourse*, pages 263–271. Plenum Press, New York, 1985.
2. D. Batens. Dialectical dynamics within formal logics. *Logique et Analyse*, 114:161–173, 1986.
3. D. Batens. Dynamic dialectical logics. In G. Priest, R. Routley, and J. Norman, editors, *Paraconsistent Logic. Essays on the Inconsistent*, pages 187–217. Philosophia Verlag, München, 1989.
4. D. Batens. Inconsistency-adaptive logics and the foundation of non-monotonic logics. *Logique et Analyse*, 145:57–94, 1994. Appeared 1996.
5. D. Batens. Blocks. The clue to dynamic aspects of logic. *Logique et Analyse*, 150–152:285–328, 1995. Appeared 1997.
6. D. Batens. Inconsistency-adaptive logics. In Orłowska [17], pages 445–472.
7. D. Batens. Minimally abnormal models in some adaptive logics. *Synthese*, In print.
8. D. Batens. A survey of inconsistency-adaptive logics. In Batens et al. [10], pages 59–83.
9. D. Batens. Towards the unification of inconsistency handling mechanisms. *Logic and Logical Philosophy*, in print.
10. D. Batens, C. Mortensen, G. Priest, and J. P. Van Bendegem, editors. *Frontiers of Paraconsistent Logic*. King's College Publications, Research Studies Press, Baldock, UK, in print.
11. S. Benferhat, D. Dubois, and H. Prade. Some syntactic approaches to the handling of inconsistent knowledge bases: A comparative study. Part 1: The flat case. *Studia Logica*, 58:17–45, 1997.
12. S. Benferhat, D. Dubois, and H. Prade. Some syntactic approaches to the handling of inconsistent knowledge bases: A comparative study. Part 2: The prioritized case. In Orłowska [17], pages 473–511.
13. K. De Clercq. Two new strategies for inconsistency-adaptive logics. *Logic and Logical Philosophy*, in print.
14. S. Kraus, D. Lehman, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
15. J. Meheus. Adaptive logic in scientific discovery: the case of Clausius. *Logique et Analyse*, 143–144:359–389, 1993. Appeared 1996.
16. J. Meheus. An extremely rich paraconsistent logic and the adaptive logic based on it. In Batens et al. [10], pages 197–209.
17. E. Orłowska, editor. *Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa*. Physica Verlag (Springer), Heidelberg, New York, 1999.
18. G. Priest. Minimally inconsistent **LP**. *Studia Logica*, 50:321–331, 1991.
19. R. M. Smullyan. *First Order Logic*. Dover, New York, 1995. Original edition: Springer, 1968.
20. G. Vanackere. Ambiguity-adaptive logic. *Logique et Analyse*, 159:261–280, 1997. Appeared 1999.

A Tableau Calculus for Integrating First-Order and Elementary Set Theory Reasoning*

Domenico Cantone[1] and Calogero G. Zarba[1,2]

[1] Department of Mathematics and Computer Science, University of Catania,
Viale A. Doria 6, I-95125, Catania, Italy
[2] Department of Computer Science, Stanford University,
Gates Building, Stanford, CA 94305, USA

e-mail: cantone@dmi.unict.it, zarba@theory.stanford.edu

Abstract. In this paper we describe an approach to integrate first-order reasoning with stratified set theory reasoning, resulting into a technique which allows to lift decision procedures for ground theories to $\exists\forall$-sentences and combine them also with elementary set constructs, such as union, intersection, finite enumerations, membership, inclusion, equality, and monotonicity predicates over function maps.
A tableau calculus, both sound and complete, is provided. The calculus is proven complete also in presence of suitable restrictions which enforce termination, thus yielding decision procedures in certain cases.

1 Introduction

Most of the work in computable set theory has concentrated on the decision problem for fragments of Zermelo-Fraenkel set theory, in presence of foundation or anti-foundation axiom (cf. [4] and [5]).

Some exceptions are [10], [6], [3], and [2], where decision procedures have been provided for fragments of stratified set theory, and precisely for (1) a quantifier-free language **2LS** (two-level syllogistic) with individual constants (ranging over some domain **D**) and set constants (ranging over the collection of subsets of **D**), (2) **2LS** in presence of a topological closure operator, (3) two **2LS**-like languages over the set of real numbers and the set of natural numbers, and (4) a three-level syllogistic with *unionset* and *powerset* operators, respectively.

It has been observed that decision procedures in the context of stratified sets are usually at least one order of magnitude more efficient than the analogous ones for pure sets. That is because, essentially, in pure set theory everything is treated as a set: for instance, to prove the simple arithmetic identity $2+1=3$ in a pure set-theoretic context, one has to show that $\{\emptyset,\{\emptyset\}\}+\{\emptyset\}=\{\emptyset,\{\emptyset\},\{\emptyset,\{\emptyset\}\}\}$ (where natural numbers are represented *à la* von Neumann and the addition among natural numbers is defined appropriately in set-theoretic terms), which

* This work has been partially supported by M.U.R.S.T. Project "Tecniche speciali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di programmi".

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 143–159, 2000.

needs certainly more machinery and effort than a simple binary computation within the microprocessor of a computer or a pocket calculator.

For this reason, it is important to use decision procedures for sets only when set reasoning is the main point. This can be done effectively in a stratified setting, by designing decision procedures in a 2-level environment, where the bottom level consists of urelements, and the upper level consists of sets of urelements. Then the urelements can be treated with "ad hoc" methods, if required. For instance, in proving a theorem like

$$\{4x : x \in \mathbb{Z}\} \subseteq \{2x + 2 : x \in \mathbb{Z}\}$$

one mixes set reasoning together with Presburger arithmetic, and aims to treat the set-theoretic part and the arithmetic part efficiently using specialized methods for each part.

Typical situations in which the ability to reasoning effectively with sets of urelements of a given nature (such as integers, reals, lists, etc.) is particularly important arise in the mechanization of proofs of verification conditions which come up during the proof of correctness of programs. In such cases, it is much more convenient to rely on proof procedures for stratified sets and domain specific decision procedures.

In this paper we address the problem of integrating simple set theory reasoning with first-order reasoning. For practical reasons, we focus our attention on stratified set theory and consider only those elementary set operators and monotonicity predicates that can be handled quite efficiently. Our approach results into a technique which allows to lift decision procedures for ground theories to $\exists\forall$-sentences and combine them also with elementary set constructs.[1]

More specifically, given a ground-decidable first-order theory $\mathcal{T}$ for a language $\mathcal{L}$, namely a theory for which there is a decision procedure for the satisfiability problem for conjunctions of ground $\mathcal{T}$-literals, we define $\mathbf{2LS}(\mathcal{L})$ as the language resulting by extending $\mathcal{L}$ with (a) set constants, (b) set functional symbols, (c) standard boolean set operators (union, intersection, set difference, and finite enumerations), (d) the predicates membership, set equality, and set inclusion, and (e) the predicates $inc(f)$ and $dec(f)$ over set functional symbols, where the intended meaning is, respectively, "f is increasing" and "f is decreasing" (with respect to set inclusion).

A tableau calculus, both sound and complete, is provided for the $\mathcal{T}$-satisfiability problem of $\mathbf{2LS}(\mathcal{L})$-sentences. Our calculus is largely based on the calculi presented in [7,8], though the latter have been developed to deal with fragments of (non-stratified) Zermelo-Fraenkel set theory and do not address the problem of integrating first-order reasoning with set theory reasoning.

For restricted $\exists\forall$-sentences of $\mathbf{2LS}(\mathcal{L})$, the calculus is proven complete also in presence of suitable restrictions which enforce termination, thus yielding a decision procedure.[2]

[1] We recall that an $\exists\forall$-sentence is a closed formula in which no essentially existential quantifier is within the scope of an essentially universal quantifier.

[2] A restricted $\mathbf{2LS}(\mathcal{L})$-formula is one in which each quantified variable has to range over a specific set term.

Furthermore, it is shown that when the underlying theory $\mathcal{T}$ enjoys certain properties, our decision result extends naturally to general $\exists\forall$-sentences of $\mathbf{2LS}(\mathcal{L})$, not necessarily restricted.

Notice that the satisfiability problem for pure first-order $\exists\forall$-sentences with equality has been first solved by Bernays and Schönfinkel (cf. [9]).

A more general approach to theory reasoning has been first described in [12] (in the context of resolution) and, more recently, in [1] (in the context of semantic tableaux). In particular, [1] proposes a general incremental approach to apply theory reasoning to the framework of free-variable semantic tableaux. Though our approach is more restricted in scope, in some favorable cases it yields decidability results.

2 First-Order Theories

We closely follow the notation and terminology of [11], with the difference that the word *structure* is used in place of *model* (for us a model of a formula φ is just a structure satisfying φ). Throughout the paper, $\mathcal{L}$ will denote a first-order language with equality and function symbols.

Definition 2.1 (First-order theories). *A* THEORY $\mathcal{T}$ *for a first-order language* $\mathcal{L}$ *is any (not necessarily finite) collection of sentences of* $\mathcal{L}$.

Given a theory $\mathcal{T}$*, by* $ground(\mathcal{T})$ *we denote the collection of ground terms occurring in* $\mathcal{T}$.

Example 2.1. Notice that if $\mathcal{T}$ is finite, then $ground(\mathcal{T})$ must be finite too. Thus, for instance, for any finite axiomatization $\mathcal{T}$ of Presburger arithmetic, $ground(\mathcal{T})$ is finite and therefore does not contain the natural numbers. To further clarify the above definition, let $\mathcal{T} = \{(\forall x)(\forall y)f(x) = f(y), \quad f(a) = f(f(a))\}$; then we have $ground(\mathcal{T}) = \{a, f(a), f(f(a))\}$.

Definition 2.2 (Structures and models). *A* STRUCTURE *for a language* $\mathcal{L}$ *is a pair* $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$*, where* $\mathbf{D}$ *is a non-empty domain and* $\mathbf{I}$ *is an interpretation of the symbols of* $\mathcal{L}$ *over* $\mathbf{D}$.

A structure $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ *is said to be a* MODEL *for a theory* $\mathcal{T}$ *if* $\varphi^{\mathbf{I}} = \mathbf{t}$*, for each sentence* φ *in* $\mathcal{T}$.

Definition 2.3. *Given a theory* $\mathcal{T}$*, a sentence* φ *is said to be*

- $\mathcal{T}$-VALID*, if it is true in every model of* $\mathcal{T}$*;*
- $\mathcal{T}$-SATISFIABLE*, if it is true in some model of* $\mathcal{T}$*;*
- $\mathcal{T}$-UNSATISFIABLE*, if it is not* $\mathcal{T}$*-satisfiable.*

The notions of $\mathcal{T}$-validity, $\mathcal{T}$-satisfiability, and $\mathcal{T}$-unsatisfiability extend naturally to collections of sentences.

Definition 2.4 (Ground-decidability). *A theory* $\mathcal{T}$ *is* GROUND-DECIDABLE *if there exists an algorithm which, given as input a finite collection* $\mathcal{C}$ *of ground literals, always terminates returning* `yes` *if* $\mathcal{C}$ *is* $\mathcal{T}$*-satisfiable and* `no` *otherwise.*

Example 2.2. Examples of ground–decidable theories are: the theory of equality, Presburger arithmetic $\langle \mathbb{N}, 0, 1, +, < \rangle$, Tarski's theory of reals $\langle \mathbb{R}, 0, 1, +, \times, < \rangle$, the theory of abelian groups, the theory of linear orderings, etc. .

Next we define the notion of *canonical model property*, expressing the existence of models which are canonical with respect to suitable collections of ground terms. For theories $\mathcal{T}$ enjoying the canonical model property and such that $ground(\mathcal{T})$ is finite, we will be able to solve the $\mathcal{T}$-satisfiability problem for slightly larger classes of sentences.

Definition 2.5 (Canonical model property). *A theory $\mathcal{T}$ has the* CANONICAL MODEL PROPERTY *if for every $\mathcal{T}$-satisfiable collection $\mathcal{C}$ of ground literals there exists a model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ of $\mathcal{T} \cup \mathcal{C}$ over a domain $\mathbf{D}$ such that $\mathbf{D} = \{s^{\mathbf{I}} : s \in ground(\mathcal{T}) \cup ground(\mathcal{C})\}$.*

We have immediately

Theorem 2.1. *Let $\mathcal{T}$ be a theory such that $ground(\mathcal{T})$ is finite. If $\mathcal{T}$ has the canonical model property, then it is ground-decidable.*

Example 2.3. Notice that if $\mathcal{T}$ is a finite axiomatization of Presburger arithmetic, then $\mathcal{T}$ does not have the canonical model property, though it is ground-decidable.

Nontrivial examples of theories enjoying the canonical model property are those whose axioms (with the only exception of the equality axioms, if present) are expressible by means of $\forall$-sentences, defined as follows:

Definition 2.6. *A first-order* $\forall$-SENTENCE *φ is a sentence in which*

- *no essentially existential quantifier occurs, and*
- *if $f(u)$ is an individual term in φ, then u is ground.*

The following result, which is easily seen to be a variant of Bernays-Schönfinkel result [9], holds:

Theorem 2.2. *Let $\mathcal{T}$ be a theory for the first-order language $\mathcal{L}$, whose axioms (with the only exception of the equality axioms, if present) are $\forall$-sentences. Then $\mathcal{T}$ has the canonical model property.*

Proof (Sketch). Let $\mathcal{C}$ be a $\mathcal{T}$-satisfiable collection of ground-literals of $\mathcal{L}$ and let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a model for $\mathcal{T}$ which satisfies $\mathcal{C}$. Let $T_{\mathcal{C} \cup \mathcal{T}}$ be the collection of ground terms occurring in $\mathcal{C}$ or in $\mathcal{T}$ and consider the set $\mathbf{D}' =_{Def} \{t^{\mathbf{I}} : t \in T_{\mathcal{C} \cup \mathcal{T}}\}$. Also, let $\mathbf{d_0}$ be a fixed element in $\mathbf{D}'$.

For each k-ary function symbol f in $\mathcal{L}$ and $b_1, \ldots, b_k \in \mathbf{D}'$, let us put

$$f^{\mathbf{I}'}(b_1, \ldots, b_k) =_{Def} \begin{cases} f^{\mathbf{I}}(b_1, \ldots, b_k) & \text{if } f^{\mathbf{I}}(b_1, \ldots, b_k) \in \mathbf{D}' \\ \mathbf{d_0} & \text{otherwise.} \end{cases}$$

Likewise, for each k-ary predicate symbol P in $\mathcal{L}$ and $b_1, \ldots, b_k \in \mathbf{D}'$, let us put

$$P^{\mathbf{I}'}(b_1, \ldots, b_k) =_{Def} P^{\mathbf{I}}(b_1, \ldots, b_k)\,.$$

Then it can easily be seen that the structure $\mathbf{M}' = \langle \mathbf{D}', \mathbf{I}' \rangle$ $\mathcal{T}$-satisfies $\mathcal{C}$. Moreover, $\mathbf{D}' = \{t^{\mathbf{I}'} : t \in T_{\mathcal{C} \cup \mathcal{T}}\}$. □

3 Basic 2-Level Set Theoretic Extension

Let $\mathcal{L}$ be a first-order language with equality and let $\mathcal{T}$ be a theory for $\mathcal{L}$. We now define a language $\mathbf{2LS}(\mathcal{L})$ which extends $\mathcal{L}$ with basic 2-level set-theoretic constructs. It turns out that the satisfiability problem for the language **2LS** studied in [10] is a particular instance of the $\mathcal{T}$-satisfiability problem for ground sentences of $\mathbf{2LS}(\mathcal{L})$, where $\mathcal{T}$ is the theory of equality and $\mathcal{L}$ does not contain any functional or relational symbol in its signature.

Note 3.1. In the rest of the paper we shall refer to formulae of a first-order language $\mathcal{L}$ (not involving any set-theoretic symbol) as *first-order formulae*. Terms of a language $\mathbf{2LS}(\mathcal{L})$ which do not involve any set-theoretic symbol will be referred to as *individual terms*; otherwise they will be called *set terms*. Finally, by a *set formula* we shall mean any formula of $\mathbf{2LS}(\mathcal{L})$.

3.1 Syntax

Let $\mathcal{T}$ be a theory for the first-order language $\mathcal{L}$. The vocabulary of $\mathbf{2LS}(\mathcal{L})$ extends that of $\mathcal{L}$ with:

- a denumerable infinity of uninterpreted set constants $s_1, s_2, \ldots$;
- a denumerable infinity of uninterpreted set functional symbols $f_1, f_2, \ldots$;
- the interpreted constant $\emptyset$ (empty set);
- the set operators $\sqcup$ (union), $\sqcap$ (intersection), $-$ (set difference), and $\langle \cdot, \ldots, \cdot \rangle$ (finite enumeration);
- the set-theoretic relational symbols $\sqsubseteq$ (membership), $\approx$ (set equality), and $\sqsubseteq$ (set inclusion);
- the unary predicates *inc* and *dec*.

Definition 3.1 (Set terms). SET TERMS *of* $\mathbf{2LS}(\mathcal{L})$ *are recursively defined as follows:*

- *each set constant is a set term;*
- $\emptyset$ *is a set term;*
- *if* s *and* t *are set terms, so are* $s \sqcup t$, $s \sqcap t$ *and* $s - t$;
- $\langle u_1, \ldots, u_n \rangle$ *is a set term, provided that* $u_1, \ldots, u_n$ *are individual terms;*
- $f(t)$ *is a set term, provided that* f *is a set functional symbol and* t *is a set term.*[3]

Definition 3.2 (Set formulae). SET FORMULAE *of* $\mathbf{2LS}(\mathcal{L})$ *are recursively defined as follows:*

- *each first-order formula is a set formula;*
- $s \approx t$ *and* $s \sqsubseteq t$ *are set formulae, provided that* s *and* t *are set terms;*

[3] For the sake of simplicity, we shall limit our treatment to unary set functional symbols. It is not difficult to generalize what follows to the general case of k-ary set functional symbols.

- $u \sqsubset t$ *is a set formula, provided that* u *is an individual term and* t *is a set term;*
- *if* f *and* g *are set functional symbols, then* $inc(f)$ *and* $dec(f)$ *are set formulae;*
- *if* φ *is a set formula, so is* $\neg\varphi$*;*
- *if* φ *and* ψ *are set formulae and* $\circ$ *is a binary propositional connective, then* $\varphi \circ \psi$ *is a set formula;*
- *if* x *is an individual variable and* φ *is a set formula, then both* $(\forall x)\varphi$ *and* $(\exists x)\varphi$ *are set formulae.*

It is convenient to adopt Smullyan's unifying notation. Accordingly, we divide the formulae of the language into four categories: conjunctive, disjunctive, universal, and existential formulae (called α-, β-, γ-, and δ-formulae, respectively).

Given a δ-formula δ, the notation $\delta_0(x)$ will be used to denote the formula φ or $\neg\varphi$, according to whether δ has the form $(\exists x)\varphi$ or $\neg(\forall x)\varphi$. Likewise, for any γ-formula γ, $\gamma_0(x)$ will denote the formula φ or $\neg\varphi$, according to whether γ has the form $(\forall x)\varphi$ or $\neg(\exists x)\varphi$, respectively.

By defining the complement operator $\complement$ as

$$\complement(X) = \begin{cases} Z & \text{if } X = \neg Z \\ \neg X & \text{otherwise,} \end{cases}$$

to each α- and β-formula, we associate two components, α_1, α_2 and β_1, β_2, respectively, in the following way:

α	α_1	α_2
$X \wedge Y$	X	Y
$\neg(X \vee Y)$	$\complement(X)$	$\complement(Y)$
$\neg(X \to Y)$	X	$\complement(Y)$

β	β_1	β_2
$X \vee Y$	X	Y
$\neg(X \wedge Y)$	$\complement(X)$	$\complement(Y)$
$(X \to Y)$	$\complement(X)$	Y

The following equivalences are immediate to check:

$$\models \alpha \leftrightarrow \alpha_1 \wedge \alpha_2 \qquad \models \beta \leftrightarrow \beta_1 \vee \beta_2 \qquad \models \gamma \leftrightarrow (\forall x)\gamma_0(x) \qquad \models \delta \leftrightarrow (\exists x)\delta_0(x)\,.$$

We shall be interested in $\exists\forall$-set formulae and restricted $\exists\forall$-set formulae. Such notions are defined as follows:

Definition 3.3. *An* $\exists\forall$-FORMULA φ *in the language* $\mathbf{2LS}(\mathcal{L})$ *is a set formula in which*

- *no essentially existential quantifier falls within the scope of some essentially universal quantifier;*
- *if* $f(u)$ *is an individual term in* φ*, then* u *is ground;*
- *if the set term* $\langle u_1, \ldots, u_n\rangle$ *occurs in* φ*, then the individual terms* $u_1, \ldots u_n$ *are ground.*

Definition 3.4. *A set formula* φ *is said to be* RESTRICTED *if each quantified subformula of* φ *is of type* $(\exists x)(x \sqsubset t \wedge \psi)$*, or* $(\forall x)(x \sqsubset t \to \psi)$*.*

3.2 Semantics

The intended semantics for $\mathbf{2LS}(\mathcal{L})$ is the most natural one and is given in the following definition.

Definition 3.5. *A* SET STRUCTURE $\mathbf{M}$ *for a language* $\mathbf{2LS}(\mathcal{L})$ *is a pair* $\langle \mathbf{D}, \mathbf{I} \rangle$ *such that*

- *the pair* $\langle \mathbf{D}, \mathbf{I}' \rangle$*, where* $\mathbf{I}'$ *is the restriction of* $\mathbf{I}$ *to the symbols of* $\mathcal{L}$*, is a first-order structure;*
- $\emptyset^{\mathbf{I}} = \emptyset$*;*
- $s^{\mathbf{I}} \in pow(\mathbf{D})$*, for each set constant* s*;*
- $f^{\mathbf{I}}$ *is a map from* $pow(\mathbf{D})$ *into* $pow(\mathbf{D})$*, for each set functional symbol* f*;*
- *the set operators* $\sqcup$*,* $\sqcap$*, and* $-$ *are interpreted in the standard way as* $\cup$*,* $\cap$*, and* $\setminus$ *over* $pow(\mathbf{D}) \times pow(\mathbf{D})$*, respectively;*
- $\langle \cdot, \ldots, \cdot \rangle$ *is interpreted as* $\{\cdot, \ldots, \cdot\}$ *over* $\mathbf{D}^n$*, for* $n \in \mathbb{N}$*;*
- *the set-theoretic predicate symbols* $\approx$ *(set equality) and* $\sqsubseteq$ *are interpreted respectively as* $=$ *and* $\subseteq$ *over* $pow(\mathbf{D}) \times pow(\mathbf{D})$*;*
- $\sqsubseteq$ *is interpreted as the relationship* $\in$ *over* $\mathbf{D} \times pow(\mathbf{D})$*;*
- $inc(f)^{\mathbf{I}} = \mathbf{t}$ *iff* $u_1 \subseteq u_2$ *implies* $f^{\mathbf{I}}(u_1) \subseteq f^{\mathbf{I}}(u_2)$*, for all* $u_1, u_2 \in pow(\mathbf{D})$ *(similar for* $dec(f)^{\mathbf{I}}$*).*

Given a theory $\mathcal{T}$ over a first-order language $\mathcal{L}$, the notions of $\mathcal{T}$-validity, $\mathcal{T}$-satisfiability, and $\mathcal{T}$-unsatisfiability extend naturally to collections of $\mathbf{2LS}(\mathcal{L})$-sentences.

4 A Ground Tableau Calculus for 2LS($\mathcal{L}$)-sentences

In order to simplify the presentation of our calculus, without loss of generality we will assume implicitly that from now on every first-order theory which we consider embodies the axioms of equality.

Let $\mathcal{T}$ be a theory over a first-order language $\mathcal{L}$. In this section we present a ground tableau calculus for the $\mathcal{T}$- satisfiability problem for $\mathbf{2LS}(\mathcal{L})$-sentences. We shall prove that our calculus is both sound (i.e., if a sentence φ is $\mathcal{T}$-satisfiable, then it has no closed tableau) and complete (if a sentence φ is $\mathcal{T}$-unsatisfiable, then it must have a closed tableau). We will also spot two classes of formulae for which the calculus can be made terminating without losing completeness, thus obtaining a decidability result.

More precisely, we shall exhibit a terminating variant of our calculus which is complete for

- restricted $\exists\forall$-sentences of $\mathbf{2LS}(\mathcal{L})$, provided that $\mathcal{T}$ is ground-decidable;
- (unrestricted) $\exists\forall$-sentences of $\mathbf{2LS}(\mathcal{L})$, provided that $\mathcal{T}$ has the canonical model property and $ground(\mathcal{T})$ is finite.

Standard first-order tableau rules in uniform notation are listed in Table 1. Rules dealing with set constructs are presented in Table 2.

$\mathbf{2LS}(\mathcal{L})$-tableaux can be constructed as follows.

Table 1. First-order tableau rules.

$$\frac{\neg\neg Z}{Z}\ (\neg\neg) \qquad \frac{\alpha}{\alpha_1\\ \alpha_2}\ (\alpha) \qquad \frac{\beta}{\beta_1 \mid \beta_2}\ (\beta) \qquad \frac{\gamma}{\gamma_0(u)}\ (\gamma) \qquad \frac{\delta}{\delta_0(p)}\ (\delta)$$

(u can be any ground term and p must be a new individual constant)

$$0 : \neg((\forall x)(x \leq b \to x < a) \to \langle a, b\rangle \not\approx \langle a\rangle)$$
$$1 : (\forall x)(x \leq b \to x < a)$$
$$2 : \langle a, b\rangle \approx \langle a\rangle$$
$$3 : \langle a, b\rangle \sqsubseteq \langle a\rangle$$
$$4 : \langle a\rangle \sqsubseteq \langle a, b\rangle$$
$$5 : b \sqsubseteq \langle a, b\rangle$$
$$6 : b \sqsubseteq \langle a\rangle$$
$$7 : b \approx a$$
$$8 : b \leq b \to b < a$$

$$\begin{array}{c|c} \neg 9 : \neg(b \leq b) & 10 : b < a \\ \bot & \bot \end{array}$$

Fig. 1. A tableau proof for $(\forall x)(x \leq b \to x < a) \to \langle a, b\rangle \not\approx \langle a\rangle$.

Definition 4.1. *An* INITIAL **2LS**($\mathcal{L}$)-TABLEAU *for a set formula φ is a tree consisting of only one node, with such a node labeled with φ. A* **2LS**($\mathcal{L}$)-TABLEAU *for φ is a tree whose nodes are labeled with set formulae and which is constructed starting from the initial tableau for φ by applying the rules of Tables 1 and 2.*

Definition 4.2. *A (possibly infinite) branch θ of a* **2LS**($\mathcal{L}$)*-tableau is* SATURATED *if no application of any rule in Tables 1 and 2 can add new formulae to θ. A* **2LS**($\mathcal{L}$)*-tableau is* SATURATED *if all its branches are saturated.*

We give next the notion of closure, relativized to a given theory $\mathcal{T}$.

Definition 4.3. *Given a theory $\mathcal{T}$ for a first-order language $\mathcal{L}$, a branch θ of a* **2LS**($\mathcal{L}$)*-tableau is $\mathcal{T}$-*OPEN *if the following four conditions are met:*

- *the collection of first-order literals occurring in θ is $\mathcal{T}$-satisfiable;*
- *no complementary set literals $\ell, \neg\ell$ occur in θ;*
- *no set literal of the form $t \not\approx t$ occurs in θ;*
- *no set literal of the form $u \sqsubseteq \emptyset$ occurs in θ.*

*A branch which is not $\mathcal{T}$-open is said to be $\mathcal{T}$-*CLOSED.

4.1 Two examples

In the first example in Fig. 1, we show a tableau proof of the sentence

$$(\forall x)(x \leq b \to x < a) \to \langle a, b\rangle \not\approx \langle a\rangle$$

Table 2. Set-theoretic tableau rules.

$$\frac{u \sqsubseteq t_1 \sqcup t_2}{u \sqsubseteq t_1 \,|\, u \sqsubseteq t_2} \ (1) \qquad \frac{u \sqsubseteq t_1}{u \sqsubseteq t_1 \sqcup t_2} \ (2) \qquad \frac{u \sqsubseteq t_2}{u \sqsubseteq t_1 \sqcup t_2} \ (3)$$

$$\frac{u \sqsubseteq t_1 \sqcap t_2}{\begin{array}{l} u \sqsubseteq t_1 \\ u \sqsubseteq t_2 \end{array}} \ (4) \qquad \frac{\begin{array}{l} u \sqsubseteq t_1 \\ u \sqsubseteq t_2 \end{array}}{u \sqsubseteq t_1 \sqcap t_2} \ (5) \qquad \frac{u \sqsubseteq t_1 - t_2}{\begin{array}{l} u \sqsubseteq t_1 \\ u \not\sqsubseteq t_2 \end{array}} \ (6)$$

$$\frac{\begin{array}{l} u \sqsubseteq t_1 \\ u \not\sqsubseteq t_2 \end{array}}{u \sqsubseteq t_1 - t_2} \ (7) \qquad \frac{u \sqsubseteq \langle u_1, \ldots, u_n \rangle}{u \approx u_1 \,|\, \ldots \,|\, u \approx u_n} \ (8) \qquad \frac{}{\begin{array}{l} u_1 \sqsubseteq \langle u_1, \ldots, u_n \rangle \\ \vdots \\ u_n \sqsubseteq \langle u_1, \ldots, u_n \rangle \end{array}} \ (9)$$

$$\frac{t_1 \approx t_2}{\begin{array}{l} t_1 \sqsubseteq t_2 \\ t_2 \sqsubseteq t_1 \end{array}} \ (10) \qquad \frac{\begin{array}{l} t_1 \sqsubseteq t_2 \\ u \sqsubseteq t_1 \end{array}}{u \sqsubseteq t_2} \ (11) \qquad \frac{t_1 \not\approx t_2}{t_1 \not\sqsubseteq t_2 \,|\, t_2 \not\sqsubseteq t_1} \ (12)$$

$$\frac{t_1 \not\sqsubseteq t_2}{\begin{array}{l} p \sqsubseteq t_1 \\ p \not\sqsubseteq t_2 \end{array}} \ (13) \qquad \frac{\begin{array}{l} u \sqsubseteq t_1 \\ v \not\sqsubseteq t_1 \end{array}}{u \not\approx v} \ (14) \qquad \frac{t_1 \approx t_2}{f(t_1) \approx f(t_2)} \ (15)$$

$$\frac{\begin{array}{l} inc(f) \\ t_1 \sqsubseteq t_2 \end{array}}{f(t_1) \sqsubseteq f(t_2)} \ (16) \qquad \frac{\neg inc(f)}{\begin{array}{l} s_1 \sqsubseteq s_2 \\ f(s_1) \not\sqsubseteq f(s_2) \end{array}} \ (17) \qquad \frac{\begin{array}{l} dec(f) \\ t_1 \sqsubseteq t_2 \end{array}}{f(t_2) \sqsubseteq f(t_1)} \ (18)$$

$$\frac{\neg dec(f)}{\begin{array}{l} s_1 \sqsubseteq s_2 \\ f(s_2) \not\sqsubseteq f(s_1) \end{array}} \ (19)$$

$$\frac{}{u \sqsubseteq t \,|\, u \not\sqsubseteq t} \ (C_1) \qquad \frac{}{t_1 \approx t_2 \,|\, t_1 \not\approx t_2} \ (C_2) \qquad \frac{}{t_1 \sqsubseteq t_2 \,|\, t_1 \not\sqsubseteq t_2} \ (C_3)$$

Notice that in rule (13), p stands for a new individual constant. Likewise, in rules (17) and (19), the symbols s_1 and s_2 stand for new uninterpreted set constants.

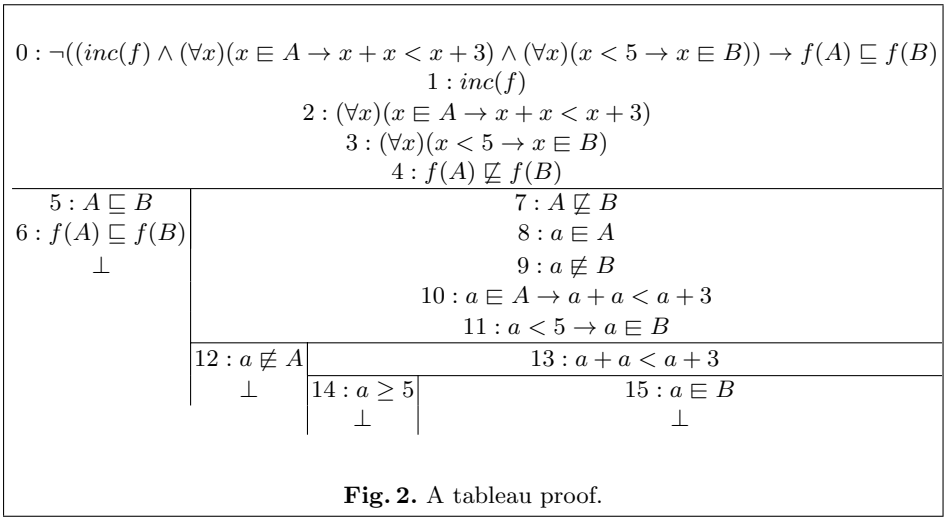


Fig. 2. A tableau proof.

in the theory $\mathcal{T}$ of partial orders, with axioms

$$(\forall x)\neg(x < x)$$
$$(\forall x)(\forall y)(\forall z)((x < y \wedge y < z) \rightarrow x < z)$$
$$(\forall x)(\forall y)((x \leq y) \leftrightarrow (x < y \vee x \approx y))$$

Denoting with φ_i the formula labeling node i in Fig. 1, deductions are justified as follows.

- φ_1, φ_2, are obtained from φ_0 by an application of the α-rule;
- φ_3 and φ_4 are obtained from φ_2 by an application of rule (10);
- φ_5 is obtained by an application of rule (9) (subject to restriction R2; see below);
- φ_6 is obtained from φ_3 and φ_5 by an application of rule (11);
- φ_7 is obtained from φ_6 by an application of rule (8);
- φ_8 is obtained from φ_1 by an application of the γ-rule (subject to restriction R2; see below);
- φ_9 and φ_{10} are obtained form φ_8 by an application of the β-rule;
- the leftmost branch is $\mathcal{T}$-closed since φ_9 is contradictory in the theory of partial orders;
- the rightmost branch is $\mathcal{T}$-closed since literals φ_7 and φ_{10} are contradictory in the theory of partial orders.

As a second example, we exhibit a tableau proof for the sentence $(inc(f) \wedge (\forall x)(x \in A \rightarrow x + x < x + 3) \wedge (\forall x)(x < 5 \rightarrow x \in B)) \rightarrow f(A) \sqsubseteq f(B)$ in Presburger arithmetic (see Figure 2). Though the present example falls outside the scope of the decision procedures which will be derived later, tableau closure has been obtained under the same restrictions R1 and R2 (see below) which force termination and, in general, spoil completeness.

Denoting with ψ_i the formula labeling node i in Figure 2, deductions are justified as follows.

- ψ_1, ψ_2, ψ_3 and ψ_4 are obtained from ψ_0 by propositional rules;
- ψ_5 and ψ_7 are obtained with an application of the cut rule (C_3);
- ψ_6 is obtained from ψ_4 and ψ_5 by means of rule (16); the branch is closed by the complementary literals ψ_4 and ψ_6;
- ψ_8 and ψ_9 are obtained from ψ_7 by using rule (13);
- ψ_{10} and ψ_{11} are obtained by applications of the γ-rule from ψ_2 and ψ_3, respectively;
- an application of the β-rule to ψ_{10} yields ψ_{12} and ψ_{13}. The left branch is closed by the complementary literals ψ_8 and ψ_{12};
- finally, an application of the β-rule to ψ_{11} yields ψ_{14} and ψ_{15}. The left branch is closed because ψ_{14} and ψ_{13} are contradictory literals in Presburger arithmetic. The right branch is closed by ψ_{15} and ψ_9.

4.2 Soundness

Soundness of our tableau calculus follows immediately by inspection of its rules in Tables 1 and 2, and by observing that any $\mathcal{T}$-closed branch is obviously $\mathcal{T}$-unsatisfiable, with respect to structures for $\mathbf{2LS}(\mathcal{L})$. Hence we have the following result:

Theorem 4.1 (Soundness). *If a set sentence of* $\mathbf{2LS}(\mathcal{L})$ *has a* $\mathcal{T}$*-closed tableau then it is* $\mathcal{T}$*-unsatisfiable.*

4.3 Termination

In general, our tableau calculus does not terminate. However, in favorable cases, restrictions on the applicability of some of its rules can be imposed in such a way that in a finite number of steps one can construct a *weakly* saturated tableau which, though not necessarily saturated, carries enough information to let one decide whether it is satitisfiable or not.

For a given theory $\mathcal{T}$ relative to a first-order language $\mathcal{L}$, the restrictions we shall enforce are the following:

R1. rules are to be applied strictly (i.e., no rule can be applied twice on a branch with the same premisses[4]);
R2. rules (9), (15), (16), (18), (C_1), (C_2), (C_3), and (γ) are allowed to add to the branch to which they are applied *only* terms already occurring on the branch itself or belonging to $\{\emptyset\} \cup \mathit{ground}(\mathcal{T})$ (thus, genuine new terms can only be introduced by rules (13), (17), (19), and (δ) only).

Definition 4.4. *Let* $\mathcal{T}$ *be a theory for a first-order language* $\mathcal{L}$ *and let* T *be a* $\mathbf{2LS}(\mathcal{L})$*-tableau constructed by a sequence of applications of rules in Tables 1 and 2, subject to restrictions R1 and R2 above.*

[4] We assume that cut rules (C_1), (C_2), (C_3) have as premisses the literals $u \in t$, $t_1 \approx t_2$, and $t_1 \sqsubseteq t_2$, respectively.

A branch θ of T is said to be WEAKLY $\mathcal{T}$-SATURATED *if no further application of any rule on θ is possible without disrupting either R1 or R2.*

If each branch of T is weakly $\mathcal{T}$-saturated, then T is said to be weakly $\mathcal{T}$-saturated.

Restrictions R1 and R2 are at the base of the following termination property.

Theorem 4.2 (Termination). *Let $\mathcal{T}$ be a theory for a first-order language $\mathcal{L}$, such that ground($\mathcal{T}$) is finite. Then any $\exists\forall$-sentence of $\mathbf{2LS}(\mathcal{L})$ has a finite weakly $\mathcal{T}$-saturated $\mathbf{2LS}(\mathcal{L})$-tableau.*

Proof (Sketch). Let T_φ be a weakly $\mathcal{T}$-saturated $\mathbf{2LS}(\mathcal{L})$-tableau for φ (the existence of T_φ could be shown by exhibiting a suitable fair weakly $\mathcal{T}$-saturation strategy). We need to show that T_φ is finite. Let θ be any branch in T_φ. Since *ground*($\mathcal{T}$) is finite, restrictions R1 and R2 imply that θ can contain only finitely many different individual and set terms. Therefore θ, and more in general every branch in T_φ, must be necessarily finite so that, by König's lemma, T_φ is finite. □

4.4 Completeness

We introduce the notion of *realization*, which will be used next to extend a first-order structure into a set structure.

Definition 4.5. *Let $\mathcal{T}$ be a theory for a first-order language $\mathcal{L}$, let θ be a $\mathcal{T}$-open branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, and let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a first-order structure satisfying all first-order literals occurring in θ. Then the* REALIZATION *$R^{\mathbf{I}}_\theta$ of θ with respect to $\mathbf{M}$ is the map $R^{\mathbf{I}}_\theta : S_\theta \to pow(\mathbf{D})$ defined by*

$$R^{\mathbf{I}}_\theta(t) = \{u^{\mathbf{I}} : u \sqsubseteq t \text{ occurs in } \theta\},$$

where S_θ is the collection of the set terms occurring in θ.

We have the following elementary result.

Lemma 4.1. *Let $\mathcal{T}$ be a theory for a first-order language $\mathcal{L}$, let θ be a $\mathcal{T}$-open and weakly $\mathcal{T}$-saturated branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, and let $R^{\mathbf{I}}_\theta$ be a realization of θ with respect to a first-order structure $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ satisfying all first-order literals occurring in θ. We have:*

(i) if $u \sqsubseteq t$ occurs in θ, then $u^{\mathbf{I}} \in R^{\mathbf{I}}_\theta(t)$;
(ii) if $u \not\sqsubseteq t$ occurs in θ, then $u^{\mathbf{I}} \notin R^{\mathbf{I}}_\theta(t)$;
(iii) if $t_1 \sqsubseteq t_2$ occurs in θ, then $R^{\mathbf{I}}_\theta(t_1) \subseteq R^{\mathbf{I}}_\theta(t_2)$;
(iv) if $t_1 \not\sqsubseteq t_2$ occurs in θ, then $R^{\mathbf{I}}_\theta(t_1) \not\subseteq R^{\mathbf{I}}_\theta(t_2)$;
(v) if $t_1 \approx t_2$ occurs in θ, then $R^{\mathbf{I}}_\theta(t_1) = R^{\mathbf{I}}_\theta(t_2)$;
(vi) if $t_1 \not\approx t_2$ occurs in θ, then $R^{\mathbf{I}}_\theta(t_1) \neq R^{\mathbf{I}}_\theta(t_2)$.

(u stands for an individual term and t_1, t_2, and t stand for set terms.)

Proof. (i) Trivial.

(ii) Let $u \not\sqsubseteq t$ be in θ and suppose, by contradiction, that $u^{\mathbf{I}} \in R^{\mathbf{I}}_{\theta}(t)$. Then $u^{\mathbf{I}} = v^{\mathbf{I}}$, for some individual term v such that the literal $v \sqsubseteq t$ occurs in θ. By saturation with respect to rule (14), the literal $u \not\approx v$ must also be in θ, and since θ is $\mathcal{T}$-open, we can infer the contradiction $u^{\mathbf{I}} \neq v^{\mathbf{I}}$. Thus $u^{\mathbf{I}} \notin R^{\mathbf{I}}_{\theta}(t)$.

(iii) Let $t_1 \sqsubseteq t_2$ be in θ. By saturation with respect to rule (11), for each literal $u \sqsubseteq t_1$ occurring in θ, the corresponding literal $u \sqsubseteq t_2$ must also occur in θ. Thus, plainly, $R^{\mathbf{I}}_{\theta}(t_1) \subseteq R^{\mathbf{I}}_{\theta}(t_2)$.

(iv) Let $t_1 \not\sqsubseteq t_2$ be in θ. By saturation with respect to rule (13), a pair of literals $u \sqsubseteq t_1$ and $u \not\sqsubseteq t_2$ must occur in θ, for some individual term u. Then, by (i) and (ii) above, $u^{\mathbf{I}} \in R^{\mathbf{I}}_{\theta}(t_1) \setminus R^{\mathbf{I}}_{\theta}(t_2)$, and, in turn, $R^{\mathbf{I}}_{\theta}(t_1) \not\subseteq R^{\mathbf{I}}_{\theta}(t_2)$.

(v) Let $t_1 \approx t_2$ be in θ. By saturation with respect to rule (10), the literals $t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq t_1$ must occur in θ too, so that, by (iii) above, $R^{\mathbf{I}}_{\theta}(t_1) \subseteq R^{\mathbf{I}}_{\theta}(t_2)$ and $R^{\mathbf{I}}_{\theta}(t_2) \subseteq R^{\mathbf{I}}_{\theta}(t_1)$. Thus $R^{\mathbf{I}}_{\theta}(t_1) = R^{\mathbf{I}}_{\theta}(t_2)$.

(vi) Let $t_1 \not\approx t_2$ be in θ. By saturation with respect to rule (12), one of the two literals $t_1 \not\sqsubseteq t_2$ and $t_2 \not\sqsubseteq t_1$ must occur in θ. Thus, by (iv) above, in any case we have $R^{\mathbf{I}}_{\theta}(t_1) \neq R^{\mathbf{I}}_{\theta}(t_2)$. □

Given a first-order structure $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ satisfying all first-order literals occurring in a $\mathcal{T}$-open branch θ of a $\mathbf{2LS}(\mathcal{L})$-tableau, where $\mathcal{T}$ is a theory for a first-order language $\mathcal{L}$, we next define how $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ can be extended to a set structure.

Definition 4.6 (Promising structure). *Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a first-order structure satisfying all first-order literals occurring in a $\mathcal{T}$-open branch θ of a $\mathbf{2LS}(\mathcal{L})$-tableau, where $\mathcal{T}$ is a theory for a first-order language $\mathcal{L}$. The* PROMISING STRUCTURE $\mathbf{S}^{\mathbf{M}}_{\theta}$ *of θ relative to $\mathbf{M}$ is the set structure $\langle \mathbf{D}, \mathbf{J} \rangle$ extending $\mathbf{M}$ and such that*

$$s^{\mathbf{J}} = R^{\mathbf{I}}_{\theta}(s)\,, \qquad \text{for each uninterpreted set constant } s,$$

and

$$\begin{aligned} f^{\mathbf{J}}(a) = &\{u^{\mathbf{I}} : u \sqsubseteq f(t) \text{ occurs in } \theta \text{ and } R^{\mathbf{I}}_{\theta}(t) = a, \text{ for some } t\} \cup \\ &\{u^{\mathbf{I}} : u \sqsubseteq f(t),\ inc(f) \text{ occur in } \theta \text{ and } R^{\mathbf{I}}_{\theta}(t) \subseteq a, \text{ for some } t\} \cup \\ &\{u^{\mathbf{I}} : u \sqsubseteq f(t),\ dec(f) \text{ occur in } \theta \text{ and } a \subseteq R^{\mathbf{I}}_{\theta}(t), \text{ for some } t\}\,, \end{aligned}$$

for each set functional symbol f and each set $a \in pow(\mathbf{D})$.

In view of Lemma 4.1, the next two lemmas show that if θ is a $\mathcal{T}$-open and weakly $\mathcal{T}$-saturated branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, whose first-order literals are satisfied by a first-order structure $\mathbf{M}$, then the promising structure of θ relative to $\mathbf{M}$ satisfies also the set-literals in θ.

Lemma 4.2 (Coherence). *Let θ be a $\mathcal{T}$-open and weakly $\mathcal{T}$-saturated branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, where $\mathcal{T}$ is a theory for the first-order language $\mathcal{L}$. Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ be a first-order structure satisfying all first-order literals occurring in θ, and let $R^{\mathbf{I}}_{\theta}$ be the realization of θ relative to $\mathbf{M}$. Also, let $\mathbf{S}^{\mathbf{M}}_{\theta} = \langle \mathbf{D}, \mathbf{J} \rangle$ be*

the promising structure of θ relative to $\mathbf{M}$. Then $R^{\mathbf{I}}_\theta$ is coherent *with $\mathbf{S}^{\mathbf{M}}_\theta$, in the sense that*

$$R^{\mathbf{I}}_\theta(t) = t^{\mathbf{J}}\,, \qquad \text{for each set term } t \text{ occurring in } \theta.$$

Proof. The proof can be done by structural induction on t, where t is a set term occurring in a $\mathcal{T}$-open and saturated branch θ. For brevity, we omit the verification of the base case and, for the inductive step, we concentrate only on the case in which t is of type $f(t_0)$.

Assume first that $a \in R^{\mathbf{I}}_\theta(f(t_0))$. Then $a = u^{\mathbf{I}}_0$, for an individual term u_0 such that $u_0 \sqsubseteq f(t_0)$ occurs in θ. By induction, $R^{\mathbf{I}}_\theta(t_0) = t^{\mathbf{J}}_0$, so that $u^{\mathbf{I}}_0 \in \{u^{\mathbf{I}} : u \sqsubseteq f(t')$ is in θ and $R^{\mathbf{I}}_\theta(t') = t^{\mathbf{J}}_0\}$. Therefore $R^{\mathbf{I}}_\theta(f(t_0)) \subseteq [f(t_0)]^{\mathbf{J}}$.

Conversely, assume that $a \in [f(t_0)]^{\mathbf{J}}$. By definition of $f^{\mathbf{J}}(t^{\mathbf{J}}_0)$, we have that there must exist an individual term u_0 and a set term t' such that $a = u^{\mathbf{I}}_0$, the literal $u_0 \sqsubseteq f(t')$ occurs in θ, and one of the following conditions holds:

(a) $R^{\mathbf{I}}_\theta(t') = t^{\mathbf{J}}_0$;
(b) $inc(f)$ is in θ and $R^{\mathbf{I}}_\theta(t') \subseteq t^{\mathbf{J}}_0$;
(c) $dec(f)$ is in θ and $t^{\mathbf{J}}_0 \subseteq R^{\mathbf{I}}_\theta(t')$.

In case (a), by inductive hypothesis we have $R^{\mathbf{I}}_\theta(t') = R^{\mathbf{I}}_\theta(t_0)$. Hence, by saturation with respect to rule (C_2), either $t' \approx t_0$ or $t' \not\approx t_0$ occurs in θ. The second case is ruled out by Lemma 4.1. Therefore the literal $t' \approx t_0$ must occur in θ. Since both $f(t')$ and $f(t_0)$ occur in θ, by saturation with respect to rule (15), also $f(t') \approx f(t_0)$ occurs in θ. Therefore, again by Lemma 4.1, we have $R^{\mathbf{I}}_\theta(f(t')) = R^{\mathbf{I}}_\theta(f(t_0))$. Finally, since $u_0 \sqsubseteq f(t')$ occurs in θ, we have $a = u^{\mathbf{I}}_0 \in R^{\mathbf{I}}_\theta(f(t_0))$. Hence $[f(t_0)]^{\mathbf{J}} \subseteq R^{\mathbf{I}}_\theta(f(t_0))$.

In case (b), by inductive hypothesis we have $R^{\mathbf{I}}_\theta(t') \subseteq R^{\mathbf{I}}_\theta(t_0)$. By saturation with respect to rule (C_3), either $t' \sqsubseteq t_0$ or $t' \not\sqsubseteq t_0$ occurs in θ. The second case is excluded by Lemma 4.1, so that $t' \sqsubseteq t_0$ must occur in θ. Thus, by saturation with respect to rule (16), $f(t') \sqsubseteq f(t_0)$ occurs in θ and therefore, again by Lemma 4.1, $R^{\mathbf{I}}_\theta(f(t')) \subseteq R^{\mathbf{I}}_\theta(f(t_0))$. Again, since $u^{\mathbf{I}}_0 \in R^{\mathbf{I}}_\theta(f(t'))$, we have $a = u^{\mathbf{I}}_0 \in R^{\mathbf{I}}_\theta(f(t_0))$, and therefore $[f(t_0)]^{\mathbf{J}} \subseteq R^{\mathbf{I}}_\theta(f(t_0))$.

Case (c) is similar to case (b).

In conclusion, in either case we have $[f(t_0)]^{\mathbf{J}} \subseteq R^{\mathbf{I}}_\theta(f(t_0))$, which together with $R^{\mathbf{I}}_\theta(f(t_0)) \subseteq [f(t_0)]^{\mathbf{J}}$ shown earlier yields $R^{\mathbf{I}}_\theta(f(t_0)) = [f(t_0)]^{\mathbf{J}}$. □

Lemma 4.3. *Let θ be a $\mathcal{T}$-open and weakly $\mathcal{T}$-saturated branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, where $\mathcal{T}$ is a theory for the first-order language $\mathcal{L}$. Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{I}\rangle$ be a first-order structure satisfying all first-order literals occurring in θ, and let $R^{\mathbf{I}}_\theta$ be the realization of θ relative to $\mathbf{M}$. Also, let $\mathbf{S}^{\mathbf{M}}_\theta = \langle \mathbf{D}, \mathbf{J}\rangle$ be the promising structure of θ relative to $\mathbf{M}$. We have:*

(i) if $inc(f)$ is in θ then $f^{\mathbf{J}}$ is increasing;
(ii) if $dec(f)$ is in θ then $f^{\mathbf{J}}$ is decreasing;
(iii) if $\neg inc(f)$ is in θ then $f^{\mathbf{J}}$ is not increasing;
(iv) if $\neg dec(f)$ is in θ then $f^{\mathbf{J}}$ is not decreasing.

Proof. Concerning (i), let $inc(f)$ be in θ, let $a \subseteq b$, for $a, b \in pow(\mathbf{D})$, and let $c \in f^{\mathbf{J}}(a)$. We shall prove that $c \in f^{\mathbf{J}}(b)$.

From $c \in f^{\mathbf{J}}(a)$, it follows that there exist an individual term u and a set term t such that $c = u^{\mathbf{I}}$, the literal $u \sqsubseteq f(t)$ occurs in θ, and one of the following conditions holds:

(a) $R^{\mathbf{I}}_{\theta}(t) \subseteq a$;
(b) $dec(f)$ is in θ and $a \subseteq R^{\mathbf{I}}_{\theta}(t)$.

In case (a), since $R^{\mathbf{I}}_{\theta}(t) \subseteq a \subseteq b$ we have plainly $c = u^{\mathbf{I}} \in f^{\mathbf{J}}(b)$.

In case (b), by saturation with respect to rule (C_3) (restricted by R2), either $\emptyset \sqsubseteq t$ or $\emptyset \not\sqsubseteq t$ occurs in θ. The latter case would lead to a closed branch by saturation with respect to rule (13), hence $\emptyset \sqsubseteq t$ must occur in θ. Since we are also assuming that $dec(f)$ occurs in θ, by saturation with respect to rule (18), the literal $f(t) \sqsubseteq f(\emptyset)$ is in θ, so that, by rule (11), the literal $u \sqsubseteq f(\emptyset)$ must also be in θ. But, plainly, $R^{\mathbf{I}}_{\theta}(\emptyset) = \emptyset \subseteq b$, and therefore $c = u^{\mathbf{I}} \in f^{\mathbf{J}}(b)$.

Case (ii) is analogous to case (i).

Finally, cases (iii) and (iv) plainly hold, since θ is saturated with respect to rules (17) and (19), respectively. □

The following lemma contains the main results towards the proof of completeness of our tableau calculus (even of the terminating variant).

Lemma 4.4. *Let $\mathcal{T}$ be a ground-decidable theory for a first-order language $\mathcal{L}$, let θ be a $\mathcal{T}$-open branch of a $\mathbf{2LS}(\mathcal{L})$-tableau, and let $\mathcal{C}_\theta$ be the collection of all first-order literals occurring in θ. Let us also assume that θ is weakly $\mathcal{T}$-saturated, provided that either*

(a) φ is a restricted $\exists\forall$-sentence of $\mathbf{2LS}(\mathcal{L})$, or
(b) φ is an $\exists\forall$-sentence of $\mathbf{2LS}(\mathcal{L})$, $\mathcal{T}$ has the canonical model property, and ground($\mathcal{T}$) is finite;

otherwise let θ be $\mathcal{T}$-saturated.

If $\mathcal{C}_\theta$ is $\mathcal{T}$-satisfiable by a first-order structure, then θ is $\mathcal{T}$-satisfiable by a set structure.

Proof. Let $\mathbf{M} = \langle \mathbf{D}, \mathbf{J} \rangle$ be a first-order model for $\mathcal{T}$ which satisfies $\mathcal{C}_\theta$, and let $\mathbf{S}^{\mathbf{M}}_{\theta} = \langle \mathbf{D}, \mathbf{J} \rangle$ be its corresponding promising structure, relative to θ and $\mathbf{M}$. Plainly, $\mathbf{S}^{\mathbf{M}}_{\theta}$ satisfies $\mathcal{T} \cup \mathcal{C}_\theta$. Moreover, by Lemmas 4.1 and 4.2, $\mathbf{S}^{\mathbf{M}}_{\theta}$ satisfies all set literals in θ, as well. Proceeding by structural induction, it is straightforward to show that $\mathbf{S}^{\mathbf{M}}_{\theta}$ satisfies also all formulae in θ of type α, β, and δ. The case of γ-formulae is more delicate.

So, let γ be in θ. We distinguish the following three cases:

θ is $\mathcal{T}$-saturated

Let γ be in θ and suppose, by contradiction, that $\gamma^{\mathbf{J}} = \mathbf{f}$. Then there exists some $a \in \mathbf{D}$ such that $\gamma(x)^{\mathbf{J},\{x/a\}} = \mathbf{f}$. Since we can suppose, without loss of generality, that $\mathbf{M}$ is a canonical model, there exists an individual term u such that $u^{\mathbf{J}} = a$. By saturation with respect to the γ-rule, the instance $\gamma(u)$ is in θ. Therefore, by inductive hypothesis, $\gamma(u)^{\mathbf{J}} = \mathbf{t}$, contradicting $\gamma(x)^{\mathbf{I},\{x/a\}} = \mathbf{f}$.

θ is weakly $\mathcal{T}$-saturated and condition (a) holds
Let γ be in θ and suppose, by contradiction, that $\gamma^{\mathbf{J}} = \mathbf{f}$. Notice that γ must be equivalent to a sentence of the form $(\forall x)(x \sqsubseteq t \rightarrow \psi)$. Then there exists some $a \in \mathbf{D}$ such that $a \in t^{\mathbf{J}}$ and $\psi(x)^{\mathbf{J},\{x/a\}} = \mathbf{f}$. By Definition 4.5, there exists an individual term u occurring in θ such that $u^{\mathbf{J}} = a$ and the literal $u \sqsubseteq t$ occurs in θ. Thus, by saturation with respect to the γ- and β-rules, the formula $\psi(u)$ must occur in θ. By inductive hypothesis, $\psi(u)^{\mathbf{J}} = \mathbf{t}$, contradicting $\psi(x)^{\mathbf{J},\{x/a\}} = \mathbf{f}$.

θ is weakly $\mathcal{T}$-saturated and condition (b) holds
Let γ be in θ and suppose, by contradiction, that $\gamma^{\mathbf{J}} = \mathbf{f}$. Then there exists some $a \in \mathbf{D}$ such that $\gamma(x)^{\mathbf{J},\{x/a\}} = \mathbf{f}$. Since $\mathcal{T}$ has the canonical model property, we may assume that $\mathbf{D} = \{u^{\mathbf{I}} : u \in \mathit{ground}(\mathcal{T} \cup \mathcal{C}_\theta)\}$. Thus, there exists an individual term $u \in \mathit{ground}(\mathcal{T} \cup \mathcal{C}_\theta)$ such that $u^{\mathbf{J}} = a$. By saturation with respect to γ-rule (even restricted by R2), the instance $\gamma(u)$ is in θ. Therefore, by inductive hypothesis, $\gamma(u)^{\mathbf{J}} = \mathbf{t}$, contradicting $\gamma(x)^{\mathbf{J},\{x/a\}} = \mathbf{f}$. □

The preceding results can be combined and summarized in the following two main theorems.

Theorem 4.3 (Completeness). *Given a theory $\mathcal{T}$ for a first-order language $\mathcal{L}$, the tableau calculus in Tables 1 and 2 is complete for the $\mathcal{T}$-satisfiability problem of* **2LS**$(\mathcal{L})$*-sentences.*

Moreover, if restrictions R1 and R2 are enforced and ground$(\mathcal{T})$ is finite, our tableau calculus remains complete for restricted $\exists\forall$-sentences of **2LS**$(\mathcal{L})$*, provided that $\mathcal{T}$ is ground-decidable, and for (unrestricted) $\exists\forall$-sentences of* **2LS**$(\mathcal{L})$*, provided that $\mathcal{T}$ has the canonical model property.*

Theorem 4.4 (Decidability). *Given a ground-decidable theory $\mathcal{T}$ for a first-order language $\mathcal{L}$ such that ground$(\mathcal{T})$ is finite, the $\mathcal{T}$-satisfiability problem for restricted $\exists\forall$-sentences of* **2LS**$(\mathcal{L})$ *is decidable. If in addition $\mathcal{T}$ has the canonical model property, then the $\mathcal{T}$-satisfiability problem for (unrestricted) $\exists\forall$-sentences of* **2LS**$(\mathcal{L})$ *is also decidable.*

5 Directions for Further Research

The results presented in the preceding sections can be extended by also allowing occurrences of literals of type $f \preceq g$, with f and g functional set variables, whose intended meaning is that $f(s) \subseteq g(s)$, for every set s.

Another possible elementary extension concerns the introduction of set terms ranging over tuples of urelements. In such a context, besides the set-theoretic constructs seen before, it would be possible also to allow the cartesian product.

Concerning future research, we are currently investigating the following further extensions:

- drop from the definition of $\exists\forall$-sentences of the language **2LS**$(\mathcal{L})$ the requirement that each set term of the form $\langle u_1, u_2, \ldots, u_n\rangle$ cannot involve individual variables (we have already some partial results);

– in the case of ground-decidable theories $\mathcal{T}$ for a first-order language $\mathcal{L}$ such that $ground(\mathcal{T})$ is finite, solve the $\mathcal{T}$-satisfiability problem for (unrestricted) $\exists\forall$-sentences.

Finally, we intend to investigate the ideas presented in this paper also from an experimental point of view. To this end, we are currently implementing our calculus in Java. Using the terminology introduced in [1], we intend to explore different policies of interaction between the *foreground reasoner* (namely, the tableau calculus) and the *background reasoner* (namely, the $\mathcal{T}$-reasoner).

Acknowledgments

The authors wish to thank the reviewers for some useful comments.

References

1. B. Beckert and C. Pape. Incremental theory reasoning methods for semantic tableaux. In *Proc. of 5th International Workshop TABLEAUX '96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 93–109. Springer-Verlag, 1996.
2. D. Cantone and V. Cutello. Decision procedures for stratified set-theoretic syllogistics. In *Proceedings of 1993 Int. Symp. on Symbolic and Algebraic Computation, Kiev-Ukraine*, pages 105–110. ACM Press, 1993.
3. D. Cantone, V. Cutello, and J.T. Schwartz. Decision problems for tarski's and presburger's arithmetics extended with sets. In *Proc. of 4th Computer Science Logic Workshop*, volume 533 of *Lecture Notes in Computer Science*, pages 95–109. Springer-Verlag, 1990.
4. D. Cantone and A. Ferro. Techniques of computable set theory with applications to proof verification. *Comm. Pure Appl. Math.*, XLVIII:1–45, 1995.
5. D. Cantone, A. Ferro, and E.G. Omodeo. *Computable set theory*, volume 6 of *International Series of Monographs on Computer Science*. Clarendon Press, 1989.
6. D. Cantone and E.G. Omodeo. Topological syllogistic with continuous and closed functions. *Comm. Pure Appl. Math.*, XLII(8):1175–1188, 1989.
7. D. Cantone and C.G. Zarba. A tableau-based decision procedure for a fragment of set theory involving a restricted form of quantification. In N.V. Murray, editor, *Proc. of Int. Conf. TABLEAUX '99*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 97–112. Springer-Verlag, 1999.
8. D. Cantone and C.G. Zarba. A new fast tableau-based decision procedure for an unquantified fragment of set theory. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, LNCS 1761 (LNAI), pages 127–137. Springer, 2000.
9. B. Dreben and W.D. Goldfarb. *The decision problem. Solvable classes of quantificational formulas*. Addison-Wesley Publ. Comp. Inc., Reading, Massachusetts, 1979.
10. A. Ferro and E.G. Omodeo. An efficient validity test for formulae in extensional two-level syllogistic. *Le Matematiche*, 33:130–137, 1978.
11. M.C. Fitting. *First-Order Logic and Automated Theorem Proving*. Graduate Texts in Computer Science. Springer-Verlag, Berlin, 2nd edition, 1996. 1st ed., 1990.
12. M.E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

Hypertableau and Path-Hypertableau Calculi for Some Families of Intermediate Logics

Agata Ciabattoni and Mauro Ferrari

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico 39, 20135 Milano–Italy
{ciabatto,ferram}@dsi.unimi.it

Abstract. In this paper we investigate the tableau systems corresponding to hypersequent calculi. We call these systems hypertableau calculi. We define hypertableau calculi for some propositional intermediate logics. We then introduce path-hypertableau calculi which are simply defined by imposing additional structure on hypertableaux. Using path-hypertableaux we define analytic calculi for the intermediate logics Bd_k, with $k \geq 1$, which are semantically characterized by Kripke models of depth $\leq k$. These calculi are obtained by adding one more structural rule to the path-hypertableau calculus for Intuitionistic Logic.

1 Introduction

Hypersequent calculi are a simple and natural generalization of Gentzen sequent calculi to *sets* of sequents (see [4] for an overview). Hypersequents allow to formalize logics of a different nature ranging from modal to many-valued logics.

In this paper we are concerned with intermediate logics, that is, logics between Intuitionistic and Classical Logic. In [4,3,9,8] cut-free hypersequent calculi have been defined for:

1. the logics Bw_k, with $k \geq 1$, which are semantically characterized by Kripke models of width $\leq k$;
2. the logics Bc_k, with $k \geq 1$, which are semantically characterized by Kripke models of cardinality $\leq k$;
3. the logics G_{k+1}, with $k \geq 1$, which are semantically characterized by linearly ordered Kripke models of cardinality $\leq k$;
4. LQ logic (also known as Jankov logic [18]).

In particular Bw_1 coincides with *infinite-valued Gödel (Dummett) logic* G_∞, while for $k \geq 2$, G_k is k-valued Gödel logic. Bc_2 is identical with *Sm* logic [7].

In the literature, there do exist sequent or tableau calculi for some of these logics. For instance, in [1] duplication-free tableau calculi for *Sm*, *LQ* and G_∞ have been defined (see also [10] for a deterministic terminating sequent calculus for G_∞). Analytic calculi for finite-valued Gödel logics, based on their many-valued semantics, can be found, e.g., in [19,14,6]. Nevertheless all these calculi are

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 160–174, 2000.

so much tailored to their corresponding logic that they hardly give information on the existing connections with other logics. In particular, they cannot help to define analytic calculi for related logics.

Hypersequent calculi for all the above logics are simply obtained by adding just one structural rule to a common system, namely the hypersequent calculus for Intuitionistic Logic [4,3,9,8]. This structural rule reflects in a natural way the characteristic semantical features of the corresponding logic.

In this paper we introduce the notion of hypertableau[1]. Hypertableau calculi stand to hypersequent calculi as tableau systems stand to sequent calculi.

By dualizing the hypersequent calculi of [4,3,9,8] we define hypertableau calculi for the logics $\mathrm{Bw}_k, \mathrm{Bc}_k, \mathrm{G}_k$ and LQ. Then, by simply generalizing the peculiar rule of LQ logic, we obtain a hypertableau calculus for the family of intermediate logics semantically characterized by rooted posets with at most k final states. Although hypertableaux (hypersequents) turn out to be more expressive than tableaux (sequents), there do exist intermediate logics with a simple Kripke semantics for which such calculi seem to be hardly definable. An important example of a logic for which no hypertableau (hypersequent) systems have been devised so far is the logic Bd_2. This logic is semantically characterized by the class $\mathcal{F}_{d\leq 2}$ of all rooted posets whose depth is at most 2 (see [7]). As is well known, like LQ, Sm and G_∞, Bd_2 is one of the seven interpolable propositional logics [15]. More generally, for each $k > 2$, the intermediate logic Bd_k semantically characterized by Kripke models of depth $\leq k$ does not have any tableau (sequent)-style formalization yet.

In Section 4 we introduce a new hypertableau framework, called path-hypertableaux. The notion of path-hypertableau naturally arises by introducing additional structure on hypertableaux. Using path-hypertableaux we define uniform analytic calculi for the Bd_k logics, with $k \geq 1$. This is done by adding one more structural rule to the path-hypertableau calculus for intuitionisticlogic.

We assume familiarity with intermediate logics and Kripke models. Introductory material can be found, e.g., in [7]. Henceforth we shall denote by Int and Cl the set of valid well-formed formulas (*wffs* for short) of propositional intuitionistic logic and classical logic, respectively.

2 Hypersequent Calculi

Hypersequent calculi [17,2] are a simple and natural generalization of Gentzen calculi. See [4] for an overview.

Definition 1. *A* hypersequent *is an expression of the form*

$$\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_n \vdash \Delta_n,$$

where, for all $i = 1, \ldots n$, $\Gamma_i \vdash \Delta_i$ *is an ordinary sequent.* $\Gamma_i \vdash \Delta_i$ *is called a* component *of the hypersequent.*

[1] The name "hypertableau" was already used in [5] in the context of a different kind of calculus.

The intended meaning of the symbol | is disjunctive.

Like in ordinary sequent calculi, in a hypersequent calculus there are axioms and rules, which are divided into *logical* and *structural rules*. The logical rules are the same as in sequent calculi but for the presence of dummy contexts, denoted by G and G', that are called *side hypersequents*. For instance, in the hypersequent calculus for Intuitionistic Logic, the rules for the $\rightarrow$ connective are:

$$(\rightarrow, l) \quad \frac{G \mid \Gamma \vdash A \quad G' \mid \Gamma, B \vdash C}{G \mid G' \mid \Gamma, A \rightarrow B \vdash C} \qquad (\rightarrow, r) \quad \frac{G \mid \Gamma, A \vdash B}{G \mid \Gamma \vdash A \rightarrow B}$$

The structural rules can either be *internal* or *external*. The internal rules deal with wff's within components. They are the same as in ordinary sequent calculi. The external rules manipulate whole components within a hypersequent. They are external weakening (EW), exchange (EE) and contraction (EC):

$$(EW) \quad \frac{G}{G \mid G'} \qquad (EE) \quad \frac{G \mid G'}{G' \mid G} \qquad (EC) \quad \frac{G \mid G' \mid G'}{G \mid G'}$$

In hypersequent calculi it is possible to define new kind of structural rules which simultaneously act on several components of one or more hypersequents. It is this type of rule which increases the expressive power of hypersequent calculi with respect to ordinary sequent calculi. See [2,3,4,9,8] for some examples of hypersequent calculi ranging from modal to many-valued logics.

3 Hypertableau Calculi

It is well known that tableau calculi can be easily obtained by dualizing sequent calculi (see, e.g., [20,11]). Here we define the tableau systems corresponding to hypersequent calculi. We call these systems hypertableau calculi.

As usual, a *signed formula* (*swff* for short) is an expression of the form $\mathbf{T}X$ or $\mathbf{F}X$ where X is any wff. The meaning of the signs $\mathbf{T}$ and $\mathbf{F}$ is as follows: Given a Kripke model $\underline{K} = \langle P, \leq, v \rangle$ and a swff H, we say that $\alpha \in P$ *realizes* H (in symbols $\alpha \rhd H$) if $H \equiv \mathbf{T}X$ and $\alpha \| \!\!- X$, or $H \equiv \mathbf{F}X$ and $\alpha \| \!\!\not\vdash X$.

A set S of swff's is realized in $\underline{K}$ (in symbols $\alpha \rhd S$) if there exists an element α realizing all the swff's in S. S is *contradictory* if it contains either $\mathbf{T}\bot$ or both $\mathbf{T}X$ and $\mathbf{F}X$ for some wff X[2].

Definition 2. *An* h-set *is an expression of the form*

$$S_1 \mid \ldots \mid S_n$$

where, for all $i = 1, \ldots, n$, S_i *is a set of swff's.* S_i *is called a* component *of the h-set. An* h-configuration *is an expression of the form*

$$\Psi_1 \parallel \ldots \parallel \Psi_m$$

where, for all $i = 1, \ldots, n$, Ψ_i *is an h-set called* component *of the h-configuration.*

[2] This condition can be equivalently reformulated by requiring X to be a propositional variable.

As shown by the following definition the intended meaning of the symbol $\mid$ is conjunctive while the one of the symbol $\parallel$ is disjunctive.

Definition 3. *We say that an* h-set $S_1 \mid \ldots \mid S_n$ *is* realized *in* $\underline{K}$*, if all the sets* S_j*, with* $j = 1, \ldots, n$*, are realized in* $\underline{K}$*.*

An h-configuration $\Psi_1 \parallel \ldots \parallel \Psi_m$ *is* realized *in* $\underline{K}$*, if there exists* $j \in \{1, \ldots, m\}$ *such that* Ψ_j *is realized in* $\underline{K}$*.*

Definition 4. *An h-set is* contradictory *if at least one of the* S_i*, with* $i = 1, \ldots, n$*, is contradictory.*

Hypertableau calculi are defined by dualizing hypersequent calculi as follows: Given a hypersequent $\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_n \vdash \Delta_n$, each component $\Gamma_i \vdash \Delta_i$ translates into the set of swff's $\mathbf{T}(\Gamma_i) \cup \mathbf{F}(\Delta_i)$ where $\mathbf{T}(\Gamma_i) = \{\mathbf{T}A \mid A \in \Gamma_i\}$ and $\mathbf{F}(\Delta_i) = \{\mathbf{F}A \mid A \in \Delta_i\}$. Thus the above hypersequent translates into the h-set

$$\mathbf{T}(\Gamma_1) \cup \mathbf{F}(\Delta_1) \mid \ldots \mid \mathbf{T}(\Gamma_n) \cup \mathbf{F}(\Delta_n).$$

Clearly, the axioms of hypersequent calculi are translated into contradictory h-sets. As for the tableau rules, the hypertableau ones are obtained by simply reversing the corresponding hypersequent rules.

Definition 5. *A* hypertableau *for* $\Psi_1 \parallel \ldots \parallel \Psi_m$ *is a finite sequence of h-configurations obtained by applying the rules of the calculus to* $\Psi_1 \parallel \ldots \parallel \Psi_m$*.*

A hypertableau is said to be closed *if all the h-sets in its final configuration are contradictory.*

In Table 1 one can find the rules of the hypertableau calculus T-Int for Intuitionistic Logic coming from the above translation. The notation S, H with S set of swff's and H swff will denote the set $S \cup \{H\}$.

We remark that, as in the hypersequent calculus for Intuitionistic Logic, in T-Int the external structural rules are redundant.

To simplify the notation, in the above rules we omitted the components of the h-configurations not involved in the derivation. E.g., the schema

$$\frac{\Psi_1 \parallel \ldots \parallel \Psi \mid \Phi \parallel \ldots \parallel \Psi_n}{\Psi_1 \parallel \ldots \parallel \Psi \parallel \ldots \parallel \Psi_n}\,\text{HEW}$$

illustrates the external weakening rule HEW spelt out in more detail, and similarly for the other rules.

Remark 1. In the above calculus the internal structural rules are internalized into logical rules, for the former are playing no semantical rôle.

Remark 2. By reversing the $(\vee, l), (\wedge, r), (\rightarrow, l)$ rules of the hypersequent calculus for Intuitionistic Logic one would obtain the following hypertableau rules:

$$\frac{\Psi \mid \Psi' \mid S, \mathbf{T}(A \vee B)}{\Psi \mid S, \mathbf{T}A \parallel \Psi' \mid S, \mathbf{T}B} \qquad \frac{\Psi \mid \Psi' \mid S, \mathbf{F}(A \wedge B)}{\Psi \mid S, \mathbf{F}A \parallel \Psi' \mid S, \mathbf{F}B} \qquad \frac{\Psi \mid \Psi' \mid S, \mathbf{T}(A \rightarrow B)}{\Psi \mid S, \mathbf{T}(A \rightarrow B), \mathbf{F}A \parallel \Psi' \mid S, \mathbf{T}B}$$

Table 1. Hypertableau calculus T-Int for Intuitionistic Logic

External Structural Rules

$$\frac{\Psi \mid \Phi}{\Psi}\,\text{HEW} \qquad \frac{\Psi \mid S}{\Psi \mid S \mid S}\,\text{HEC} \qquad \frac{\Psi \mid S_1 \mid S_2 \mid \Phi}{\Psi \mid S_2 \mid S_1 \mid \Phi}\,\text{HEE}$$

Logical Rules

$$\frac{\Psi \mid S, \mathbf{T}(A_1 \wedge A_2)}{\Psi \mid S, \mathbf{T}A_i}\,\mathbf{T}\wedge_i \quad \text{for } i = 1, 2 \qquad \frac{\Psi \mid S, \mathbf{F}(A \wedge B)}{\Psi \mid S, \mathbf{F}A \parallel \Psi \mid S, \mathbf{F}B}\,\mathbf{F}\wedge$$

$$\frac{\Psi \mid S, \mathbf{T}(A \vee B)}{\Psi \mid S, \mathbf{T}A \parallel \Psi \mid S, \mathbf{T}B}\,\mathbf{T}\vee \qquad \frac{\Psi \mid S, \mathbf{F}(A_1 \vee A_2)}{\Psi \mid S, \mathbf{F}A_i}\,\mathbf{F}\vee_i \quad \text{for } i = 1, 2$$

$$\frac{\Psi \mid S, \mathbf{T}(A \to B)}{\Psi \mid S, \mathbf{T}(A \to B), \mathbf{F}A \parallel \Psi \mid S, \mathbf{T}B}\,\mathbf{T}\to \qquad \frac{\Psi \mid S, \mathbf{F}(A \to B)}{\Psi \mid S^{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}\to$$

$$S^{\mathbf{T}} = \{\mathbf{T}X \mid \mathbf{T}X \in S\}$$

Nevertheless, these rules introduce non-determinism in proof search. It is easy to see that using the external rules, these rules are interderivable with the rules $\mathbf{T}\vee, \mathbf{F}\wedge$ and $\mathbf{T}\to$ of T-Int, respectively.

Any hypertableau rule coming from the above translation is *correct*, i.e., when its premise is realized, so is the conclusion. This immediately follows from the correctness of the corresponding hypersequent rule. Thus, the proof of the soundness theorem for the hypersequent calculus can be translated into a proof of the soundness theorem for the corresponding hypertableau calculus. Accordingly, one can look at the proof of the completeness theorem given for the hypersequent calculus as a completeness proof for the corresponding hypertableau calculus. Indeed, each proof of a valid hypersequent directly translates into a closed hypertableau.

Theorem 1. *A wff A is intuitionistically valid iff there exists a closed hypertableau for $\{\mathbf{F}A\}$ in* T-Int.

3.1 On Hypertableaux for Intermediate Logics

The hypertableau framework is stronger than that of tableau. Intuitively, the former allows to formalize logics whose properties can be simply expressed in a disjunctive form. This section is devoted to define hypertableau calculi for some families of intermediate logics and to give some insights on the expressive power of hypertableaux (hypersequents).

In [4,3,9,8] cut-free hypersequent calculi have been defined for the intermediate logics Bw_k, Bc_k, G_{k+1} and LQ, whose semantics are given by

Bw_k : the class of finite trees not containing $k+1$ pairwise incomparable nodes (for short, finite trees of *width* $\leq k$);
Bc_k : the class of trees containing at most k nodes;
G_{k+1} : the class of trees of width ≤ 1 and with at most k nodes;
LQ : the class of finite posets with a single final node.

By dualizing these calculi as described in the previous section, one gets sound and complete hypertableau calculi for $\mathrm{Bw}_k, \mathrm{Bc}_k, \mathrm{G}_{k+1}$ and LQ. All these logics share the property that their Kripke models $\underline{K}$ can be described by disjunctively combining "basic" conditions of the form:

$$(a)\ \alpha_i \leq \alpha_j \text{ or } \alpha_j \leq \alpha_i \quad (b)\ \alpha_i = \alpha_j \quad (c)\ \exists \alpha_k \in \underline{K} :\ \alpha_i \leq \alpha_k \text{ and } \alpha_j \leq \alpha_k$$

Indeed, the Kripke models of the Bw_k logic can be characterized as follows: For every $k+1$ elements $\alpha_0, \ldots, \alpha_k$,

$$\bigvee_{i \neq j \in \{0,\ldots,k\}} \alpha_i \leq \alpha_j \quad \text{or} \quad \alpha_j \leq \alpha_i.$$

The Kripke models of the Bc_k logic have the following property: For every $k+1$ elements $\alpha_0, \ldots, \alpha_k$,

$$\bigvee_{i \neq j \in \{0,\ldots,k\}} \alpha_i = \alpha_j.$$

Finally, the Kripke models of the LQ logic satisfy (c).

The above conditions can be formalized in the hypertableau framework. Indeed, consider the following rule, originally defined in [4] in the context of cut-free hypersequent calculi

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{F}A_0 \mid \mathbf{T}(\Gamma_1), \mathbf{F}A_1}{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{T}(\Gamma_1), \mathbf{F}A_1 \parallel \Psi \mid \mathbf{T}(\Gamma_0), \mathbf{T}(\Gamma_1), \mathbf{F}A_0} (\leq)$$

By adding this rule to T-Int one gets a hypertableau calculus for infinite-valued Gödel logic.

Example 1. We display a proof of axiom $(q \to p) \vee (p \to q)$ in the above calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{F}(q \to p) \vee (p \to q)}{\mathbf{F}(q \to p) \vee (p \to q) \mid \mathbf{F}(q \to p) \vee (p \to q)}\,\mathrm{HEC}}{\mathbf{F}(q \to p) \vee (p \to q) \mid \mathbf{F}(p \to q)}\,\mathbf{F}\vee}{\mathbf{F}(q \to p) \vee (p \to q) \mid \mathbf{T}p, \mathbf{F}q}\,\mathbf{F}\to}{\mathbf{T}p, \mathbf{F}q \mid \mathbf{F}(q \to p) \vee (p \to q)}\,\mathrm{HEE}}{\mathbf{T}p, \mathbf{F}q \mid \mathbf{F}(q \to p)}\,\mathbf{F}\vee}{\mathbf{T}p, \mathbf{F}q \mid \mathbf{T}q, \mathbf{F}p}\,\mathbf{F}\to}{\mathbf{T}p, \boxed{\mathbf{T}\mathrm{q}\ ,\ \mathbf{F}\mathrm{q}} \parallel \mathbf{T}q, \boxed{\mathbf{T}\mathrm{p}\ ,\ \mathbf{F}\mathrm{p}}}\,(\leq)$$

As is well known, in every Kripke model $\underline{K}$ of G_∞ for all $\alpha_i, \alpha_j \in \underline{K}$, either we have $\alpha_i \leq \alpha_j$ or $\alpha_j \leq \alpha_i$. Thus a hypertableau calculus for the Bw_k logic is simply defined by adding to T-Int the following generalization of the ($\leq$) rule (see [8])

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{F}A_0 \mid \ldots \mid \mathbf{T}(\Gamma_k), \mathbf{F}A_k}{\ldots \parallel \Psi \mid \mathbf{T}(\Gamma_i), \mathbf{T}(\Gamma_j), \mathbf{F}A_i \parallel \text{ for every } 0 \leq i \neq j \leq k \ldots} (\mathrm{Bw_k})$$

As an example we show a proof of axiom $(p_0 \to p_1 \vee p_2) \vee (p_1 \to p_0 \vee p_2) \vee (p_2 \to p_0 \vee p_1)$ in the above calculus for Bw_2.

Example 2. Let $H = (p_0 \to p_1 \vee p_2) \vee (p_1 \to p_0 \vee p_2) \vee (p_2 \to p_0 \vee p_1)$. The derivation proceeds as follows:

$$\dfrac{\dfrac{\dfrac{\mathbf{F}H}{\mathbf{F}H \mid \mathbf{F}H}\,\mathrm{HEC}}{\mathbf{F}H \mid \mathbf{F}H \mid \mathbf{F}H}\,\mathrm{HEC}}{\begin{array}{c}\vdots \\ \text{by several applications of } \mathbf{F}\vee \text{ and HEE} \\ \mathbf{F}(p_0 \to p_1 \vee p_2) \mid \mathbf{F}(p_1 \to p_0 \vee p_2) \mid \mathbf{F}(p_2 \to p_0 \vee p_1) \\ \vdots \\ \text{by several applications of } \mathbf{F}\to \text{ and HEE} \\ \mathbf{T}p_0, \mathbf{F}(p_1 \vee p_2) \mid \mathbf{T}p_1, \mathbf{F}(p_0 \vee p_2) \mid \mathbf{T}p_2, \mathbf{F}(p_0 \vee p_1)\end{array}}(\mathrm{Bw_2})$$

$$\begin{array}{c}\mathbf{T}p_0, \mathbf{T}p_1, \mathbf{F}(p_1 \vee p_2) \parallel \mathbf{T}p_0, \mathbf{T}p_2, \mathbf{F}(p_1 \vee p_2) \parallel \mathbf{T}p_1, \mathbf{T}p_0, \mathbf{F}(p_0 \vee p_2) \parallel \\ \mathbf{T}p_1, \mathbf{T}p_2, \mathbf{F}(p_0 \vee p_2) \parallel \mathbf{T}p_2, \mathbf{T}p_0, \mathbf{F}(p_0 \vee p_1) \parallel \mathbf{T}p_2, \mathbf{T}p_1, \mathbf{F}(p_0 \vee p_1) \\ \vdots \\ \text{by several applications of } \mathbf{F}\vee \text{ and HEE} \\ \vdots \\ \mathbf{T}p_0, \boxed{\mathbf{T}p_1, \mathbf{F}p_1} \parallel \mathbf{T}p_0, \boxed{\mathbf{T}p_2, \mathbf{F}p_2} \parallel \mathbf{T}p_1, \boxed{\mathbf{T}p_0, \mathbf{F}p_0} \parallel \\ \mathbf{T}p_1, \boxed{\mathbf{T}p_2, \mathbf{F}p_2} \parallel \mathbf{T}p_2, \boxed{\mathbf{T}p_0, \mathbf{F}p_0} \parallel \mathbf{T}p_2, \boxed{\mathbf{T}p_1, \mathbf{F}p_1}\end{array}$$

By extending the hypertableau calculus for Intuitionistic Logic with the following rule

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{F}A_0 \mid \mathbf{T}(\Gamma_1)}{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{T}(\Gamma_1), \mathbf{F}A_0} (=)$$

one gets a calculus for Classical Logic (see [9]). As is well known, in this logic for every two states $\alpha_i, \alpha_j \in \underline{K}$, $\alpha_i = \alpha_j$. A hypertableau calculus for the Bc_k logic is simply defined by adding to T-Int the following generalization of the (=) rule (see [8])

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{F}A_0 \mid \ldots \mid \mathbf{T}(\Gamma_k)}{\ldots \parallel \Psi \mid \mathbf{T}(\Gamma_i), \mathbf{T}(\Gamma_j), \mathbf{F}A_i \parallel \text{ for every } 0 \leq i \leq k-1 \text{ and } i+1 \leq j \leq k \ldots}$$

To obtain a hypertableau calculus for G_{k+1} it suffices to extend T-Int with both the $(\leq)$ rule and the above rule. However, an alternative hypertableau calculus for G_{k+1} can be defined by replacing these two rules with the following one (see [8])

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{F}A_0 \mid \ldots \mid \mathbf{T}(\Gamma_k)}{\ldots \parallel \Psi \mid \mathbf{T}(\Gamma_i), \mathbf{T}(\Gamma_{i+1}), \mathbf{F}A_i \parallel_{\text{ for every } 0 \leq i \leq k-1} \ldots}$$

Finally, the (c) condition exactly corresponds to the following rule defined for LQ logic (see [9])

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0) \mid \mathbf{T}(\Gamma_1)}{\Psi \mid \mathbf{T}(\Gamma_0), \mathbf{T}(\Gamma_1)} (\exists \leq)$$

thus, for each $k \geq 1$, adding to T-Int the following generalization of the $(\exists \leq)$ rule

$$\frac{\Psi \mid \mathbf{T}(\Gamma_0) \mid \ldots \mid \mathbf{T}(\Gamma_k)}{\ldots \parallel \Psi \mid \mathbf{T}(\Gamma_i), \mathbf{T}(\Gamma_j) \parallel_{\text{ for every } 0 \leq i \neq j \leq k} \ldots}$$

one gets a calculus for the logic which is semantically characterized by Kripke models with at most k final states.

However, there do exists intermediate logics characterized by simple semantical conditions that cannot be described only combining the (a)-(c) conditions above. Let us consider, for instance, the Bd_2 logic which is semantically characterized by the class $\mathcal{F}_{d \leq 2}$ of all rooted posets with depth at most 2 (see [7]). To describe its models one needs to express the condition: for all $\alpha_i, \alpha_j, \alpha_k$

$$\alpha_i \leq \alpha_j \leq \alpha_k \implies \alpha_i = \alpha_j \quad \text{or} \quad \alpha_j = \alpha_k.$$

We believe that this condition is hardly formalizable in the hypertableau (hypersequent) framework.

In the next section we will show how to suitably modify hypertableaux in order to define an analytic calculus for this logic.

4 Path-Hypertableau Calculi

In this section we introduce path-hypertableaux. Whereas hypertableaux are based on h-sets with explicit external rules for manipulating the order and the number of their components, the idea behind path-hypertableaux is to consider the components of h-sets as suitable ordered sequences. This eliminates the external exchange rule. Thus path-hypertableaux can be seen as substructural hypertableaux.

Let us call a *path* of a Kripke model $\underline{K}$ any sequence $\underline{\alpha} = \alpha_1, \ldots, \alpha_n \in \underline{K}$ such that $\alpha_1 \leq \ldots \leq \alpha_n$.

Definition 6. *We say that an h-set* $S_1 \mid \ldots \mid S_n$ *is* path-realized *(*p-realized *for short) in a Kripke model* $\underline{K}$*, if there exists a path* $\underline{\alpha} = \alpha_1, \ldots, \alpha_n \in \underline{K}$ *such that* $\alpha_i \rhd S_i$*, for every* $1 \leq i \leq n$*. In this case we say that* the path $\underline{\alpha}$ realizes the h-set $S_1 \mid \ldots \mid S_n$*. An* h-configuration $\Psi_1 \parallel \ldots \parallel \Psi_m$ *is* p-realized *in* $\underline{K}$*, if there exists* $j \in \{1, \ldots, m\}$ *such that* Ψ_j *is p-realized in* $\underline{K}$*.*

A *p-hypertableau* is a finite sequence of h-configurations obtained by applying the rules of the p-hypertableau calculus. The closure condition for a p-hypertableau is the same as that we gave for a hypertableau (see Definition 5).

Intuitively, each component of an h-set describes a particular state of a Kripke model. Thus path-hypertableaux allow to simultaneously explore all the states constituting an ascending chain.

It is immediate to verify that the following proposition holds:

Proposition 1. *If an h-set is contradictory then it is not p-realizable.*

In Table 2 we display the p-hypertableau calculus PT-Int for Intuitionistic Logic.

Table 2. Path-hypertableau calculus PT-Int for Intuitionistic Logic

External Structural Rules

$$\frac{\Psi \mid \Phi}{\Psi}\,\mathrm{HEW_r} \qquad \frac{\Psi \mid \Phi}{\Phi}\,\mathrm{HEW_l} \qquad \frac{\Psi \mid \Phi}{\Psi \mid \Phi \mid \Phi}\,\mathrm{HEC_r} \qquad \frac{\Psi \mid \Phi}{\Psi \mid \Psi \mid \Phi}\,\mathrm{HEC_l}$$

Logical Rules

$$\frac{\Psi \mid S, \mathbf{T}(A_1 \wedge A_2) \mid \Psi'}{\Psi \mid S, \mathbf{T}A_i \mid \Psi'}\,\mathbf{T}\wedge_{\mathrm{i}} \quad \text{for } i = 1, 2 \qquad \frac{\Psi \mid S, \mathbf{F}(A \wedge B) \mid \Psi'}{\Psi \mid S, \mathbf{F}A \mid \Psi' \parallel \Psi \mid S, \mathbf{F}B \mid \Psi'}\,\mathbf{F}\wedge$$

$$\frac{\Psi \mid S, \mathbf{T}(A \vee B) \mid \Psi'}{\Psi \mid S, \mathbf{T}A \mid \Psi' \parallel \Psi, S, \mathbf{T}B \mid \Psi'}\,\mathbf{T}\vee \qquad \frac{\Psi \mid S, \mathbf{F}(A_1 \vee A_2) \mid \Psi'}{\Psi \mid S, \mathbf{F}A_i \mid \Psi'}\,\mathbf{F}\vee_{\mathrm{i}} \quad \text{for } i = 1, 2$$

$$\frac{\Psi \mid S, \mathbf{T}(A \to B) \mid \Psi'}{\Psi \mid S, \mathbf{F}A, \mathbf{T}(A \to B) \mid \Psi' \parallel \Psi \mid S, \mathbf{T}B \mid \Psi'}\,\mathbf{T}\to \qquad \frac{\Psi \mid S, \mathbf{F}(A \to B) \mid \Psi'}{\Psi \mid S^{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B}\,\mathbf{F}\to$$

$$S^{\mathbf{T}} = \{\mathbf{T}X \mid \mathbf{T}X \in S\}$$

Remark 3. In the new interpretation of h-sets the external exchange rule does not hold. This entails the splitting of the external rules of weakening and contraction into left and right rules, according to the part of the h-set they modify.

As usual, the main step of the Soundness Theorem is to prove that the rules of the calculus preserve p-realizability.

Lemma 1. *The rules of* PT-Int *preserve p-realizability.*

Proof. As an example, we shall give the proof for the $\mathbf{F}\rightarrow$ rule. Suppose that $\alpha_1, \ldots, \alpha_n, \beta, \gamma_1, \ldots, \gamma_m$ is a path of a model $\underline{K}$ p-realizing $S_1 \mid \ldots \mid S_n \mid S, \mathbf{F}(A \rightarrow B) \mid S'_1 \mid \ldots \mid S'_m$. Thus, there exists an element δ in $\underline{K}$ such that $\beta \leq \delta$ and $\delta \rhd \mathbf{T}A, \mathbf{F}B$. Hence the path $\alpha_1, \ldots, \alpha_n, \delta$ p-realizes $S_1 \mid \ldots \mid S_n \mid S^{\mathbf{T}}, \mathbf{T}A, \mathbf{F}B$.

Remark 4. The absence of the right context Ψ' in the consequent of the $\mathbf{F}\rightarrow$ rule is essential to preserve p-realizability of $\mathbf{F}\rightarrow$. Indeed the p-realizability of $\mathbf{F}(A \rightarrow B)$ in a state β requires the existence in the model of a state $\delta \geq \beta$ p-realizing $\mathbf{T}A$ and $\mathbf{F}B$. However nothing is known about the relationship between δ and the path p-realizing Ψ'.

Theorem 2 (Soundness). *If there exists a closed p-hypertableau for* $\{\mathbf{F}A\}$ *in* PT-Int*, then* A *is valid in Intuitionistic Logic.*

Proof. Let us suppose, by way of contradiction, that there are a Kripke model $\underline{K} = \langle P, \leq, v\rangle$ and $\alpha \in P$ such that $\alpha \rhd \mathbf{F}A$. By the previous lemma the final h-configuration of the closed p-hypertableau for $\{\mathbf{F}A\}$ is p-realizable. This means that a contradictory h-set is p-realizable, contradicting Proposition 1.

Theorem 3 (Completeness). *If a wff* A *is valid in Intuitionistic Logic, then there exists a closed p-hypertableau for* $\{\mathbf{F}A\}$ *in* PT-Int.

Proof. Straightforward since the logical rules of PT-Int are essentially the same as in ordinary tableau calculi for Intuitionistic Logic (see, e.g., [11]) with in addition the contexts Ψ and Ψ'.

4.1 Logics of Bounded Depth Kripke Models

In the following we define path-hypertableau calculi for the intermediate logics of bounded depth Kripke models Bd_k, with $k \geq 1$. Our calculi are uniform, and are simply obtained by adding a suitable structural rule to PT-Int (see Table 2).

Bd_k is characterized by the class $\mathcal{F}_{d\leq k}$ of rooted posets with depth $\leq k$. In other words, every chain of its models has at most k elements. Thus Bd_1 coincides with Classical Logic. A Hilbert style axiomatization of Bd_k is obtained by extending the axioms of Intuitionistic Logic with the axiom scheme (Bd_k) recursively defined as follows (see [12]):

$$(\mathrm{Bd}_1)\ A_1 \vee \neg A_1$$
$$(\mathrm{Bd}_{i+1})\ A_{i+1} \vee (A_{i+1} \rightarrow (\mathrm{Bd}_i))$$

For $k \geq 1$, the p-hypertableau calculus PT-Bd_k is simply obtained by adding to PT-Int the following structural rule:

$$\frac{\Psi \mid S_0 \mid \ldots \mid S_k \mid \Psi'}{\Psi \mid S_0, S_1 \mid \Psi' \parallel \Psi \mid S_0 \mid S_1, S_2 \mid \Psi' \parallel \ldots \parallel \Psi \mid S_0 \mid \ldots \mid S_{k-2} \mid S_{k-1}, S_k \mid \Psi'}\ (\leq \mathrm{k})$$

Remark 5. The $(\leq k)$ rule resembles the n-Shifting restart rule introduced in [13] in order to define goal-oriented deduction methods for Bd_k.

We prove that PT-Bd_k is sound and complete with respect to the Bd_k logic.

Lemma 2. *The rules of* PT-Bd_k *preserve p-realizability.*

Proof. By Lemma 1 we only have to show that the $(\leq k)$ rule preserves p-realizability over the frames for Bd_k. Indeed, if its premise is realized in a model $\underline{K}$ built on a frame of $\mathcal{F}_{d\leq k}$, then there exists a path $\beta_0 \leq \ldots \leq \beta_p \leq \alpha_0 \leq \ldots \leq \alpha_k \leq \gamma_0 \leq \ldots \leq \gamma_q$ in $\underline{K}$ such that $\underline{\beta} = \beta_0, \ldots, \beta_p$ realizes the h-set Ψ, $\alpha_i \rhd S_i$ for every $i = 0, \ldots, k$, and $\underline{\gamma} = \gamma_0, \ldots, \gamma_q$ realizes the h-set Ψ'. Since any path in $\underline{K}$ has depth at most k, there exists $h \in \{0, \ldots, k\}$ such that $\alpha_h = \alpha_{h+1}$. Thus the sequence $\underline{\beta}, \alpha_0, \ldots, \alpha_h, \underline{\gamma}$ realizes the h-set $\Psi \mid S_0 \mid \ldots \mid S_h, S_{h+1} \mid \Psi'$.

Theorem 4 (Soundness). *If there exists a closed p-hypertableau for* $\{\mathbf{F}A\}$ *in* PT-Bd_k*, then* A *is valid in* Bd_k.

Proof. The proof proceeds as in Theorem 2.

Definition 7. *An h-set* Φ *is* Bd_k-consistent *if it has no closed p-hypertableau in* PT-Bd_k.

The completeness theorem has the following form: *If a wff* A *is valid in every Kripke model of depth* $\leq k$*, then there is a closed p-hypertableau for* $\{\mathbf{F}A\}$ *in* PT-Bd_k. According to the semantical interpretation of the swff's, it suffices to prove that: *If an h-set* S *of swff's is* Bd_k*-consistent then there is a Kripke model* $\underline{K}$ *built on a poset in* $\mathcal{F}_{d\leq k}$ *realizing it.*

Our proof is based on a general method allowing to built up a rooted Kripke model $\underline{K}(S)$ realizing each Bd_k-consistent h-set S (see, e.g., [1,16]). We start by defining the basic notions of *node set* and *successor set.*

Let $\Phi \mid S$ be any Bd_k-consistent h-set with $S = \{A_1, \ldots, A_n\}$. We define the sequence $\{S_i\}_{i\in\omega}$ of sets of swff's as follows:

- $S_0 = S$;
- Let $S_i = \{H_1, \ldots, H_q\}$; then $S_{i+1} = \bigcup_{H_j \in S_i} \mathcal{U}(H_j, i)$

where, setting $S'_j = \bigcup_{k=1}^{j-1} \mathcal{U}(H_k, i) \cup \bigcup_{k=j+1}^{q} H_k$, $\mathcal{U}(H_j, i)$ is defined as follows:

1. If $H_j \equiv \mathbf{T}(A \vee B)$ and $\Phi \mid (S'_j \cup \{\mathbf{T}A\})$ is Bd_k-consistent, then $\mathcal{U}(H_j, i) = \{\mathbf{T}A\}$, otherwise $\mathcal{U}(H_j, i) = \{\mathbf{T}B\}$;
2. If $H_j \equiv \mathbf{F}(A \wedge B)$ and $\Phi \mid (S'_j \cup \{\mathbf{F}A\})$ is Bd_k-consistent, then $\mathcal{U}(H_j, i) = \{\mathbf{F}A\}$, otherwise $\mathcal{U}(H_j, i) = \{\mathbf{F}B\}$;
3. If $H_j \equiv \mathbf{T}(A \rightarrow B)$ and $\Phi \mid (S'_j \cup \{\mathbf{T}B\})$ is Bd_k-consistent, then $\mathcal{U}(H_j, i) = \{\mathbf{T}B\}$, otherwise $\mathcal{U}(H_j, i) = \{\mathbf{F}A, \mathbf{T}(A \rightarrow B)\}$;
4. If $H_j \equiv \mathbf{F}(A \rightarrow B)$, then $\mathcal{U}(H_j, i) = \{\mathbf{F}(A \rightarrow B)\}$.
5. Otherwise, $\mathcal{U}(H_j, i)$ is the set of h-configurations obtained by applying the rule in PT-Bd_k for H_j to the h-configuration $\{H_j\}$.

Since each step in the construction of S_{i+1} corresponds to the application of a rule of PT-Bd_k, it is easy to see (by induction on i) that every $\Phi \mid S_i$ is Bd_k-consistent. The finiteness of each S_i directly follows for S being finite. Thus there exists $k \geq 0$ such that $S_k = S_{k+1}$. We call S_k the *node set of S w.r.t. Φ* and we denote it with $\overline{S}$. We call the set $S^* = \bigcup_{i \geq 0} S_i$ the *saturated set related to $\overline{S}$*.

Given an h-set $\Phi \mid S$, for every $\mathbf{F}(A \to B) \in S$ we call *path-extension of $\Phi \mid S$ (w.r.t. $\mathbf{F}(A \to B)$)* the path $\Phi \mid S \mid (S^{\mathbf{T}} \cup \{\mathbf{T}A, \mathbf{F}B\})$. Moreover, we call $S^{\mathbf{T}} \cup \{\mathbf{T}A, \mathbf{F}B\}$ the *successor set of S (w.r.t. $\mathbf{F}(A \to B)$)*.

Using the rules HEC_r and $\mathbf{F}\to$, one can easily check that every path-extension of a Bd_k-consistent h-set is Bd_k-consistent.

Henceforth, we shall consider trees $\mathcal{T}$ whose nodes are sets of swff's and we shall identify each path $\Gamma_0, \ldots, \Gamma_m$ of these trees with the h-set $\Gamma_0 \mid \ldots \mid \Gamma_m$.

To help the reader, we first give an overview on the construction of $\underline{K}(S)$. Let $\overline{\mathcal{T}}_0$ be the tree only containing a node set of S (w.r.t. the empty h-set) as root. Our aim is to build up a sequence of trees $\overline{\mathcal{T}}_1, \ldots, \overline{\mathcal{T}}_t$ having depth $\leq k$. This shall be done in the following steps:

Step 1: Expansion. For every leaf $\overline{\Delta}$ of depth j in $\overline{\mathcal{T}}_i$, with $j < k+1$, the nodes of depth $j+1$ in $\overline{\mathcal{T}}_{i+1}$ are obtained by adding, as the immediate descendent of $\overline{\Delta}$, the node sets of its successor sets. If $\overline{\mathcal{T}}_{i+1}$ has depth $\leq k$ go to Step 3. Otherwise, go to Step 2.

Step 2: Contraction. The contraction of $\overline{\mathcal{T}}_i$ results in a tree of depth $\leq k$ obtained by deleting some subtrees of $\overline{\mathcal{T}}_i$. This shall be essentially done by using the $(\leq k)$ rule. Go to Step 1.

Step 3: Model construction. $\underline{K}(S)$ is built on the last tree $\overline{\mathcal{T}}_{last}$ of the sequence and for every element α of $\overline{\mathcal{T}}_{last}$, $v(\alpha)$ is defined as $\{p : \mathbf{T}p \in \alpha\}$.

Expansion step

First we define an invariant on the trees $\mathcal{T}$ that we shall consider hereafter

T.1 For each internal node $\overline{\Gamma}$, the path $\overline{\Gamma}_0 \mid \ldots \mid \overline{\Gamma}$ is Bd_k-consistent and each set occurring in it is a node set;
T.2 For every leaf Δ, the path $\overline{\Gamma}_0 \mid \ldots \mid \overline{\Gamma}_m \mid \Delta$ is Bd_k-consistent (possibly, Δ is not a node set);
T.3 The depth of $\mathcal{T}$ is at most $k+1$.

Let $\mathcal{T}$ be a tree satisfying the above conditions. The sequence $\{\mathcal{S}_i\}^{\mathcal{T}}_{i \leq k+1}$ generated by $\mathcal{T}$ is defined as follows:

- $\mathcal{S}_0 = \mathcal{T}$;
- Given $\mathcal{S}_i$ with $i \leq k+1$, let $V_1, \ldots, V_q$ be the leaves of depth $\leq k$. For every $j = 1, \ldots, q$ let $\overline{V}_j$ be the node set with respect to the h-set Φ_j (corresponding to the path from the root of $\mathcal{S}_i$ to the parent of V_j). Let $U^j_1, \ldots, U^j_{l_j}$ be the successor sets of $\overline{V}_j$. For every U^j_g, let $\overline{U}^j_g$ be its node set w.r.t. $\Phi_j \mid \overline{V}_j$. $\mathcal{S}_{i+1}$ is the tree obtained by substituting the node V_j in $\mathcal{S}_i$ with the subtree of depth 2 having $\overline{V}_j$ as root and as leaves the node sets $\overline{U}^j_1, \ldots, \overline{U}^j_{l_j}$.

It is easy to check that $\mathcal{S}_{i+1}$ satisfies the conditions (T.1)-(T.3).

Contraction

We call *cut of the sequence* $\{\mathcal{S}_i\}^{\mathcal{T}}_{i\leq k+1}$ the tree $\overline{\mathcal{T}}$ defined as follows:

- If the depth of $\mathcal{S}_{k+1}$ is $\leq k+1$, then $\overline{\mathcal{T}} = \mathcal{S}_{k+1}$;
- Otherwise, let $\pi_0, \ldots, \pi_r$ be the paths in $\mathcal{S}_{k+1}$ with length $k+1$. Since any $\pi_j = \overline{\Gamma}_0 \mid \ldots \mid \overline{\Gamma}_k$ is Bd_k-consistent, the application of the $(\leq k)$ rule to this h-set yields an h-configuration

$$\overline{\Gamma}_0, \overline{\Gamma}_1 \parallel \overline{\Gamma}_0 \mid \overline{\Gamma}_1, \overline{\Gamma}_2 \parallel \ldots \parallel \overline{\Gamma}_0 \mid \ldots \mid \overline{\Gamma}_{k-2} \mid \overline{\Gamma}_{k-1}, \overline{\Gamma}_k$$

  that contains at least a Bd_k-consistent h-set. Let π^+_j be the first Bd_k-consistent h-set in this h-configuration, i.e., the one corresponding to the shortest path. Consider the paths $\pi^+_0, \ldots, \pi^+_r$ together with the ordering induced on this set from the sub-path relation. Let us denote with $\tilde{\pi}_0, \ldots, \tilde{\pi}_z$ the minimal elements of this ordered set. For every $j = 0, \ldots, z$, if $\tilde{\pi}_j = \overline{\Gamma}^j_0 \mid \ldots \mid \overline{\Gamma}^j_{p-1} \mid \overline{\Gamma}^j_p, \overline{\Gamma}^j_{p+1}$ and H_j is the swff used to build up the successor node $\overline{\Gamma}^j_{p+1}$ of $\overline{\Gamma}^j_p$, we define $\overline{\pi}_j = \overline{\Gamma}^j_0 \mid \ldots \mid \overline{\Gamma}^j_{p-1} \mid \overline{\Gamma}^j_p, \overline{\Gamma}^j_{p+1} \setminus \{H_j\}$. We remark that the Bd_k-consistency of $\tilde{\pi}_j$ immediately implies the Bd_k-consistency of $\overline{\pi}_j$. Let $\overline{\Delta}_j$ be the node set of $(\overline{\Gamma}^j_p \setminus \{H_j\}) \cup \overline{\Gamma}^j_{p+1}$ with respect to $\overline{\Gamma}^j_0 \mid \ldots \mid \overline{\Gamma}^j_{p-1}$. We define $\overline{\mathcal{T}}$ as the tree obtained by replacing the subtree of root $\overline{\Gamma}^j_p$ in $\mathcal{S}_{i+1}$ with the subtree only consisting of the node set $\overline{\Delta}_j$. Moreover, we say that $\overline{\Delta}_j$ is obtained by a *cut on the set* $(\overline{\Gamma}^j_p \setminus \{H_j\}) \cup \overline{\Gamma}^j_{p+1}$.

By the above construction we immediately get that, if $\mathcal{T}$ satisfies conditions (T.1)-(T.3), $\overline{\mathcal{T}}$ satisfies these conditions too. Moreover $\overline{\mathcal{T}}$ has depth $\leq k$.

The sequence of trees

Given a Bd_k-consistent h-set S, we define the sequence of trees $\{\overline{\mathcal{T}}_i\}_{i\in\omega}$ generated by S, as follows:

- $\overline{\mathcal{T}}_1$ is the cut of the sequence $\{\mathcal{S}_i\}^{\{S\}}_{i\leq k+1}$ where $\{S\}$ is the tree only consisting of the root S.
- $\overline{\mathcal{T}}_{i+1}$ is the cut of $\{\mathcal{S}\}^{\overline{\mathcal{T}}_i}_{i\leq k+1}$.

For every tree $\overline{\mathcal{T}}_i$ in the above sequence, we define a function ρ_i associating with each node of $\overline{\mathcal{T}}_i$ a saturated set as follows: For every node $\overline{\Gamma}$ in $\overline{\mathcal{T}}_i$ $(i \geq 1)$:

1. If $\overline{\Gamma}$ is a node of the tree $\mathcal{S}_{k+1}$ in the sequence $\{\mathcal{S}_i\}^{\mathcal{T}_i}_{i\leq k+1}$, then

$$\rho_i(\overline{\Gamma}) = \begin{cases} \text{the saturated set related to } \overline{\Gamma} & \text{if } i = 1 \\ \rho_{i-1}(\overline{\Gamma}) & \text{otherwise} \end{cases}$$

2. If $\overline{\Gamma}$ is not a node of $\mathcal{S}_{k+1}$, then it is obtained by a cut on a set $\Gamma = (\overline{\Gamma}_p \setminus \{H\}) \cup \overline{\Gamma}_{p+1})$, thus

$$\rho_i(\overline{\Gamma}) = \begin{cases} \Gamma_p^* \cup \Gamma_{p+1}^* \cup \Gamma^* & \text{if } i = 1 \\ \rho_{i-1}(\overline{\Gamma}_p) \cup \rho_{i-1}(\overline{\Gamma}_{p+1}) \cup \Gamma^* & \text{otherwise} \end{cases}$$

where Γ_p^*, Γ_{p+1}^* and Γ^* are the saturated sets related to $\overline{\Gamma}_p$, $\overline{\Gamma}_{p+1}$ and $\overline{\Gamma}$, respectively.

An inspection on the construction of the trees $\overline{\mathcal{T}}_i$, easily shows that for any node $\overline{\Gamma}$ of $\overline{\mathcal{T}}_i$, the set $\rho_i(\overline{\Gamma})$ has the usual properties of a saturated set.

Model Construction

Being S a finite set of swff's, one can easily prove that there exists an integer t such that, for every $i \geq t$, $\overline{\mathcal{T}}_i = \overline{\mathcal{T}}_t$. The Kripke model $\underline{K}(S)$ is defined thus:

1. $\langle P, \leq \rangle$ is the poset where P contains all the nodes of $\overline{\mathcal{T}}_t$ and $\leq$ is the reflexive and transitive closure of the immediate descendent relation of $\overline{\mathcal{T}}_t$.
2. For every $\overline{\Gamma} \in P$ and for every propositional variable p, $p \in v(\overline{\Gamma})$ iff $\mathbf{T}p \in \overline{\Gamma}$.

We associate with $\underline{K}(S)$ the function $\rho = \rho_t$. Using the construction of the trees in $\{\overline{\mathcal{T}}_i\}_{i \in \omega}$, it is easy to check that $\underline{K}(S)$ is a Kripke model built on $\mathcal{F}_{d \leq k}$.

Remark 6. The above construction shows that in PT-Bd_k the HEC_l rule is actually redundant.

Now, it is only a matter of technicality to prove the main lemma:

Lemma 3. *For each $\overline{\Gamma} \in \underline{K}(S)$ and swff $H \in \rho(\Gamma)$, one has $\overline{\Gamma} \rhd H$ in $\underline{K}(S)$.*

Proof. By induction on the structure of H. The base cases $H \equiv \mathbf{T}p$ and $H \equiv \mathbf{F}p$ immediately follow by definition of $\underline{K}(S)$. As for the induction step consider the case $H \equiv \mathbf{F}(A \to B)$. Let $\overline{\Gamma}_1, \ldots, \overline{\Gamma}_{t-1}, \overline{\Gamma}_t = \overline{\Gamma}$ be the "history" of $\overline{\Gamma}$ in the sequence $\{\overline{\mathcal{T}}_i\}_{i \leq t}$. That is, for every $j = t-1, \ldots, 1$, if $\overline{\Gamma}_j$ is obtained by a cut on the set $(\overline{\Gamma}_p \setminus \{H\}) \cup \overline{\Gamma}_{p+1}$ then $\overline{\Gamma}_j$ is $\overline{\Gamma}_p$, otherwise $\overline{\Gamma}_j$ is $\overline{\Gamma}_{j+1}$. Let $\overline{\Gamma}_h$ be the first set in this sequence not containing $\mathbf{F}(A \to B)$. Thus there exists a node set $\overline{\Delta}_h$ in $\overline{\mathcal{T}}_h$ such that $\overline{\Delta}_h$ is an immediate descendent of $\overline{\Gamma}_h$ and $\mathbf{T}A, \mathbf{F}B$ belongs to $\rho_h(\overline{\Delta}_h)$. By definition of the sequence $\{\overline{\mathcal{T}}_i\}_{i \leq t}$ and of the model $\underline{K}(S)$ there exists an element $\overline{\Delta} \in P$ such that $\overline{\Gamma} \leq \overline{\Delta}$ and $\{\mathbf{T}A, \mathbf{F}B\} \subseteq \rho(\Delta)$. By induction hypothesis one has $\overline{\Delta} \rhd \mathbf{T}A, \mathbf{F}B$. Therefore, $\overline{\Gamma} \rhd \mathbf{F}(A \to B)$.

This immediately yields the completeness theorem:

Theorem 5 (Completeness). *If a wff A is valid in $\mathcal{F}_{d \leq k}$, then there exists a closed p-hypertableau for $\{\mathbf{F}A\}$ in* PT-Bd_k.

Proof. Let us suppose by way of contradiction that $\{\mathbf{F}A\}$ is Bd_k-consistent. By the above lemma, this implies that $\mathbf{F}A$ is p-realizable in $\underline{K}(S)$, contradicting the assumed validity of A in $\mathcal{F}_{d \leq k}$.

Remark 7. By dualizing PT-Bd_k, with $k \geq 1$, one can easily define path-hypersequent calculi for the Bd_k logics. However, while in path-hypertableau calculi there is an immediate relation between the semantics of these logics and the interpretation of h-sets, this relation is not so obvious in the corresponding path-hypersequent calculi.

References

1. *A. Avellone, M. Ferrari, P. Miglioli.* Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. *Logic Journal of the IGPL*, vol. **7**(4), pp 447–480, 1999.
2. *A. Avron.* A constructive analysis of RM. *J. Symbolic Logic***52**, pp 939–951, 1987.
3. *A. Avron.* Hypersequents, logical consequence and intermediate logics for concurrency. *Annals of Mathematics and Artificial Intelligence***4**, pp 225–248, 1991.
4. *A. Avron.* The method of hypersequents in the proof theory of propositional nonclassical logics. In **Logic: from Foundations to Applications, European Logic Colloquium**, pp 1–32, Oxford University Press, 1996.
5. *P. Baumgartner, U. Furbach, I. Niemelae.* Hyper Tableaux. **Proc. of JELIA-96: Logics in Artificial Intelligence**. LNAI **1126**, pp 1–17, 1996.
6. *M. Baaz, C.G. Fermüller.* Analytic calculi for projective logics. **Proc. of Tableaux'99**. LNAI **1617**, pp 36–51, 1999.
7. *A. Chagrov, M. Zakharyaschev.* **Modal Logic**. Oxford University Press, 1997.
8. *A. Ciabattoni, M. Ferrari.* Hypersequent calculi for some intermediate logics with bounded Kripke models. *Journal of Logic and Computation.* To appear.
9. *A. Ciabattoni, D.M. Gabbay, N. Olivetti.* Cut-free proof systems for logics of weak excluded middle. *Soft Computing* **2**(4), pp 147-156, 1998.
10. *R. Dyckhoff.* A deterministic terminating sequent calculus for Gödel-Dummett logic. *Logic J. of the IGPL* **7**, pp 319–326, 1999.
11. *M.C. Fitting.* **Proof Methods for Modal and Intuitionistic Logics**. Reidel, Dordrecht. 1983.
12. *D.M. Gabbay.* **Semantical Investigations in Heyting's Intuitionistic Logic**. Reidel, Dordrecht. 1983.
13. *D.M. Gabbay, N. Olivetti.* **Goal-Directed Proof Theory**. Kluwer. To appear.
14. *R. Hähnle.* **Automated Deduction in Multiple-valued Logics**. Oxford University Press, 1993.
15. *L. Maksimova.* Craig's theorem in superintuitionistic logics and amalgamable varieties of pseudo-boolean algebras. *Algebra and Logic* **16**, pp 427–455, 1977.
16. *P. Miglioli, U. Moscato, M. Ornaghi.* Avoiding duplications in tableau systems for intuitionistic logic and Kuroda logic. *Logic J. of the IGPL* **5**(1), pp 145–167, 1997.
17. *G. Pottinger.* Uniform, cut-free formulation of T,S_4 and S_5, (abstract). *J. Symbolic Logic* **48**, p 900, 1983.
18. *V. Jankov.* The calculus of the weak "law of excluded middle". *Mathematics of the USSR* **8**, pp. 648–658, 1968.
19. *G. Rousseau.* Sequent in many valued logic, I and II. *Fund. Math.* **60** and **67**, pp 23–33, 1967 & pp 125–131, 1970.
20. *R.M. Smullyan.* **First-Order Logic**. Springer, Berlin. 1968.

Variants of First-Order Modal Logics

Marta Cialdea Mayer[1] and Serenella Cerrito[2]

[1] Università Roma Tre, Dipartimento di Informatica e Automazione
[2] Université de Paris-Sud, L.R.I.

Abstract. In this paper we study proof procedures for some variants of first order modal logics, where domains may be either cumulative or freely varying and terms may be either rigid or non-rigid, local or non-local. We define both *ground* and *free variable* tableau methods, parametric with respect to the variants of the considered logics. The treatment of each variant is equally simple and is based on the annotation of functional symbols by natural numbers, conveying some semantical information on the worlds where they are meant to be interpreted.

1 Introduction

First order modal logics constitute a delicate and difficult subject (see for example [12] for an overview). In principle, their semantics could simply be obtained by extension of the modal propositional semantics: a first order modal structure can be built on the base of a set of first order classical interpretations (the "possible worlds"), connected by a binary relation (the accessibility relation). However, there are a number of different possibilities that can be considered, concerning for example:

The object domains: is there any relation between the universes of the different worlds? They can be required to be the same for all worlds (*constant domains*), they can bear no relation one to the other (*varying domains*), or the object domains can vary, but monotonically, i.e. if w' is accessible from w, then the object domain of w is included in the domain of w' (*cumulative domains*).

The designation of terms: is the denotation of terms the same in all worlds or can it vary? When the answer is positive, then designation is taken to be *rigid*, otherwise it is *non-rigid* (often, mixed approaches are of interest, too, where the interpretations of some symbols are rigid, others are not).

The existence of objects: does the extension of any ground term belong to every object domain or not? If the answer is positive, then we assume terms to be *local*, otherwise terms are *non-local*.

So, several variants of quantified modal logics (QMLs) are possible, just by choosing different combinations of the cases considered above. We call them DDE variants (Domains/Designation/Existence variants) of QML. Obviously, the logic also depends on its propositional kernel, so that we have a four-dimensional

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 175–189, 2000.

space of possible QMLs. As often happens with modal logics, different choices are appropriate for different applications. A discussion can be found in [12] as well as [11].

Some DDE variants can easily be given a proof theoretic characterization, some are harder to treat. The easiest approach is obviously obtained just by adding the principles of classical logic to a propositional modal system. What we obtain then is a logic with cumulative domains, rigid designation and local terms. In fact, such a logic validates the converse of Barcan Formula (**CBF**), that characterizes cumulative domains: $\Box\forall xA \rightarrow \forall x\Box A$. Rigid designation and locality of terms are consequences of the instantiation rule of classical logic $\forall xA \rightarrow A[t/x]$. For example, $\Box p(c)$ follows from the instantiation rule and $p(c) \wedge \forall x(p(x) \rightarrow \Box p(x))$, but it is not a logical consequence of the latter formula alone, if the denotation of c can vary from world to world (the example is from [16]).

Besides the axiomatic characterization, some first order modal logics with cumulative domains, rigid designation and local terms have been given sequent and tableau calculi [9,10,15], natural deduction [5], matrix proof procedures [19], resolution style calculi [1,7], and translation based procedures [3,18].

Now, while translation methods are general enough to treat also other DDE variants of QML (for instance, [3] deals with constant domains logics, where designation may be rigid as well as non rigid, and [18] handles all the DDE variants of QML), the direct methods are less flexible. For instance, it can be proved that constant domain logics cannot be treated by standard tableau systems (cut free sequent calculi). The addition of prefixes labelling tableau nodes solves this problem [9], as matrix methods do [19]: in both kinds of calculi in fact it is possible to analyse more than one possible world at a time, and this allows the proof to "go back and forth" (the same mechanism solves the problem of symmetric logics). In [9] all the variants of QML concerning the object domains of possible worlds are treated by prefix tableau methods and some suggestions on how to treat the varying domains case in calculi without prefixes can be found. A direct approach dealing with both varying domains and non-rigid symbols has a representative in [16], which defines a resolution method for epistemic logics, where terms can be annotated by a "bullet" constructor distinguishing rigid terms from non-rigid ones. A different direction is followed by [11], where the language of modal logic is enriched by means of a predicate abstraction operator, in order to capture differences on the denotation of terms, and a tableau proof procedure is presented for such a logic, with no restriction on the domains of possible worlds.

In general, direct proof methods can treat the less easy variants of QML either by modifications of classical and/or modal rules in an often cumbersome manner (if the axiomatic approach is taken), or else by attaching some kind of semantical information to the proof structures.

The aim of this work is to provide a treatment of some variants of QML, where the explicit intrusion of semantics in the proof structures is kept as simple and minimal as possible. We propose calculi in the tableau style, that allow us to treat all the DDE variants of QML (that are formally defined at the end of this section), with the exclusion of constant domain logics, that we believe

cannot be captured in the same style. Namely, we deal with the combinations cumulative/varying domains, rigid/non rigid designation, local/non local terms, for the analytic logics **K, D, T, K4, S4**. The treatment of each DDE variant is equally simple and is based on the annotation of functional symbols by natural numbers, resulting in a minimal departure from standard tableau methods (i.e. cut-free sequent systems). A similar mechanism is used in [7,8], to obtain Skolem-like normal forms and define modal resolution proof procedures.

We define both *ground* tableau calculi — called Tab_G (Section 2), dealing with closed formulae — and *free variable* tableau calculi — called Tab_{FV} (Section 3). In both cases, the calculi are parametric with respect to the DDE variants of the considered logic. The treatments differ only in the annotation mechanism and a suitable classification of terms according to the annotation of their symbols, that determines, in Tab_G, which applications of the γ-rule are correct and, in Tab_{FV}, which substitutions are allowed. Both classes of calculi are sound and complete. Due to space restrictions, proofs are only sketched (Section 4).

In the rest of this section, we briefly recall the syntax and semantics of QML. A first order modal language L is constituted by logical symbols (propositional connectives, quantifiers, modal operators and a countable set X of individual variables), a non empty set L_P of predicate symbols and a set L_F of functional symbols with an associated arity. Constants are considered as functional symbols with null arity. Terms are built using symbols from L_F and X. The inductive definition of terms and formulae is as usual.

A first order modal interpretation $\mathcal{M}$ of a language L is a tuple $\langle W, w_0, R, D, \delta, \phi, \pi \rangle$ such that:

- W is a non empty set (the set of "possible worlds");
- w_0 is a distinguished element of W;
- R is a binary relation on W (the *accessibility relation*);
- D is a non empty set (the object domain);
- δ is a function assigning to each $w \in W$ a non empty subset of D, the domain of w: $\delta(w) \subseteq D$;
- ϕ represents the interpretation of constants and functional symbols in the language: for every world $w \in W$ and n-ary functional symbol in the language (with $n \geq 0$), $\phi(w, f) \in D^n \to D$.
- π is the interpretation of predicate symbols: if p is a n-ary predicate symbol and $w \in W$, then $\pi(w, p) \subseteq D^n$ is a set of n-tuples of elements in D.

The accessibility relation R of a modal structure can be required to satisfy additional properties, characterizing different logics: we consider seriality (**D**), reflexivity (**T**), transitivity (**K4**), both reflexivity and transitivity (**S4**). The logic **K** makes no additional assumption on R.

The interpretation function ϕ is extended to terms in the usual way, and, by an abuse of notation, $\phi(w, t)$ denotes the interpretation of t in w. If $\mathcal{M} = \langle W, w_0, R, D, \delta, \phi, \pi \rangle$ is an interpretation of the language L, the **language of the model** $\mathcal{M}$, L_D, is obtained from L by addition of a "name" for each $d \in D$, i.e. a new constant $\overline{d}$. The interpretation $\mathcal{M}$ is assumed to interpret names in

such a way that that for every $d \in D$ and $w \in W$, $\phi(w, \overline{d}) = d$. Note that the interpretation of names $\overline{d}$ is always rigid.

Below we take $\neg$, $\wedge$ and $\vee$ as the only primitive propositional connectives. The relation $\models$ between an interpretation $\mathcal{M} = \langle W, w_0, R, D, \delta, \phi, \pi \rangle$, a world $w \in \mathcal{M}$ and a closed formula in L_D is defined inductively as follows:

1. $\mathcal{M}, w \models p(t_1, ..., t_n)$ iff $\langle \phi(w, t_1), ..., \phi(w, t_n) \rangle \in \pi(w, p)$.
2. $\mathcal{M}, w \models \neg A$ iff $M, w \not\models A$.
3. $\mathcal{M}, w \models A \wedge B$ iff $\mathcal{M}, w \models A$ and $\mathcal{M}, w \models B$.
4. $\mathcal{M}, w \models A \vee B$ iff $\mathcal{M}, w \models A$ or $\mathcal{M}, w \models B$.
5. $\mathcal{M}, w \models \forall x A$ iff for all $d \in \delta(w)$, $\mathcal{M}, w \models A[\overline{d}/x]$
6. $\mathcal{M}, w \models \exists x A$ iff there exists $d \in \delta(w)$ such that $\mathcal{M}, w \models A[\overline{d}/x]$
7. $\mathcal{M}, w \models \Box A$ iff for any $w' \in W$ such that $\langle w, w' \rangle \in R$, $\mathcal{M}, w' \models A$
8. $\mathcal{M}, w \models \Diamond A$ iff there is a $w' \in W$ such that $\langle w, w' \rangle \in R$ and $\mathcal{M}, w' \models A$

A closed formula A is true in $\mathcal{M}$ iff $\mathcal{M}, w_0 \models A$, and it is valid iff it is true in any interpretation $\mathcal{M}$.

Let $\mathcal{M} = \langle W, w_0, R, D, \delta, \phi, \pi \rangle$ be a first order modal interpretation. We shall deal with the following DDE variants of QML:

Cumulative domains: for all $w, w' \in W$, if wRw' then $\delta(w) \subseteq \delta(w')$.
Varying domains: no restrictions on $\delta(w)$.
Rigid designation: for all $w, w' \in \mathcal{M}$ and for every functional symbol f, $\phi(w, f) = \phi(w', f)$.
Non-rigid designation: no restrictions on the relation between $\phi(w, f)$ and $\phi(w', f)$, for any $w \neq w'$ and $f \in L_F$.
Local terms: for all $w \in \mathcal{M}$ and for every functional symbol f, if $d_1, ..., d_n \in \delta(w)$, then $\phi(w, f)(d_1, ..., d_n) \in \delta(w)$.
Non-local terms: no restriction on the closure of domains $\delta(w)$ with respect to the application of the designation of functional symbols.

For the sake of simplicity, we treat each variant as a different logic. However, mixed approaches are also possible. For instance, we can establish that constant symbols denote rigidly and functional symbols (of arity greater than 0) do not. In the calculi described in the rest of this work, each symbol would then be treated according to the rules of the corresponding logic.

Note that, if terms are local, the denotation of every ground term t is in every domain $\delta(w)$. However, if domains are allowed to vary without restrictions, if $\mathcal{M}, w \models \exists x \Diamond p(f(x))$, and d is the element of $\delta(w)$ such that $\mathcal{M}, w \models \Diamond p(f(\overline{d}))$, then the denotation of $f(\overline{d})$ is not necessarily in domain of the world w' accessible from w and such that $\mathcal{M}, w' \models p(f(\overline{d}))$, unless $d \in \delta(w')$.

2 Ground Tableaux

In this section we illustrate the mechanism that allows us to deal with some DDE variants of QML. We show how to treat the cases of cumulative and varying

domains, where designation of terms can be either rigid or non-rigid, and terms either local or not. In the following, we assume that a given logic has been fixed, with a propositional basis that is one of the analytic logics **K, D, K4, T**, or **S4**, and a first order variant.

The tableau systems presented in the sequel work with formulae in negation normal form: negation over non-atomic formulae is considered as a defined symbol. In other terms, modal formulae are built out of literals (atoms and negated atoms) by use of $\wedge$, $\vee$, $\Box$, $\Diamond$ and the quantifiers $\forall$ and $\exists$. Negation over non-atomic formulae is defined as usual.

Below, we present the ground calculus Tab_G. Formulae occurring in a ground tableau for a set S of modal formulae are built over an extension L^+ of the language L of S, where constants and functional symbols can be annotated by a numerical superscript, that is called the *level* of the symbol. Symbols with superscripts are called annotated symbols. L^+ extends L with two different infinite sets of constants: a set $\mathcal{A}$ containing a constant for each level $k \in \mathbb{N}$ and a set $\mathcal{C}$, containing an infinite number of constants for each level $k > 0$. Moreover, L^+ contains the annotated versions of the constants and functional symbols in L. Precisely:

Definition 1. *Let L be a modal language, $\mathcal{A} = \{a_0^0, a_1^1, ...\}$ and $\mathcal{C}$ disjoint sets of annotated constants different from any symbol in L, such that $\mathcal{A}$ contains exactly one constant a_k^k for each $k \in \mathbb{N}$, and $\mathcal{C}$ contains a denumerable set of constants $c_{k_1}^k, c_{k_2}^k, ...$ annotated with level k, for each $k \geq 1$. Then the* labelled extension of L *is the language L^+ such that $L_P^+ = L_P$ and L_F^+ is:*

$$L_F \cup \mathcal{C} \cup \mathcal{A} \cup \{f^k \mid f \in L_F \text{ and } k \in \mathbb{N}\}$$

The nodes of a ground tableau for a set of closed formulae in the modal language L are labelled sets of the form $n : S$ where n is a natural number, called the *depth* of the node, and S is a set of closed modal formulae in the language L^+. Similarly to the sequent calculus in [17], in this approach there is no labelling of single formulae, but of whole sets of formulae.

Symbol annotations and node depths are related: let t be the occurrence of a term in a node $n : S$, that we can assume describes a given world w; then the levels of the symbols in t tell us whether t has a denotation in the domain of w, and whether its denotation can be assumed to be the same as the denotation of another occurrence of the same term t (with possibly different annotations).

Next, we define the initial tableau for S and an operation that is used in some propositional expansion rules.

Definition 2.

1. *If S is a set of modal formulae, then the initial tableau for S is the single node $1 : S^*$, where:*
   - **local terms and rigid designation:** *S^* is obtained from S by annotating each functional symbol with level* 0.
   - **local terms and non-rigid designation:** *S^* is obtained from S by annotating each* occurrence *of a functional symbol outside the scope of any modal operator with level* 1.

non local terms: $S^* = S$

2. *The* **base language** *of the tableau rooted at* $1 : S^*$ *is the language* L *of* S. *The* **language of the tableau** *is the labelled extension* L^+ *of* L.
3. *If* S *is a set of modal formulae and* $n \in \mathbb{N}$, *then:*
   local terms: S^n *is obtained from* S *by annotating each non-annotated* occurrence *of a functional symbol outside the scope of any modal operator with level* n;
   non local terms: $S^n = S$.

The main intuition behind the above notions is that symbols annotated with level 0 are always symbols of the original language (and have not been introduced by means of the δ-rule, defined below), their interpretation is the same in every possible world and the domain of every world is closed with respect to the application of the functions they denote. Thus, symbols of level 0 occur only in the calculus for the DDE variant with rigid designation and local terms.

In the case of non-rigid designation and local terms, on the contrary, a symbol of the base language L can occur with different levels in a tableau branch: when occurring with level n, it actually refers to the denotation of the symbol in the world corresponding to the node with depth n in that branch. Symbols occurring outside the scope of a modal operator are initialized with level 1, the depth of the initial node. Symbols of L_F occurring inside a modal operator receive their levels only when, by application of a propositional rule, they come to the surface (by means of operation 3 in Definition 2).

In the case of non local terms, on the contrary, no assumption is made on the denotation of symbols of the base language: they have no level in the initial node of the tableau and never get one.

We consider a simple set of propositional tableau rules (others may be chosen as well), shown in Table 1, where S, S' are sets of modal formulae (in the language of the tableau), $\Box S$ stands for $\{\Box G \mid G \in S\}$, S' is a set of non-boxed modal formulae, comma is set union. The propositional part of the tableau systems we consider consists of the classical rules, and: the rule π_K in systems **K**, **D** and **T**, the rule π_4 in **K4** and **S4**, the rule ν_T in **T** and **S4**, the rule ν_D in **D**.

Adopting a terminology from [13], the rules that do not modify the depth of the node are called *static*, the others are called *dynamic*. So, a non-annotated symbol can obtain a superscript only with the application of dynamic rules.

As can be noted, the depth n of a node $n : S$ in a tableau "counts" the number of dynamic rules applied in the branch before $n : S$. And, in fact, the labelling of nodes is not essential and the depth of a node could be defined as a meta-level concept. For this reason the present approach is not in the style of prefixed tableaux.

The next definition establishes the relation between the depth of a node and levels of symbols, by means of the concept of *domain* of a node, that is needed to formulate the quantifier tableau rules.

Definition 3. *Let* L *be a modal language and* L^+ *its labelled extension.*

1. *A term* t *in* L^+ *is in* **depth** n *iff every symbol in* t *has a level and* t *contains only symbols with:*

Classical rules	
$(\alpha)\quad \dfrac{n : A \wedge B, S}{n : A, B, S}$	$(\beta)\quad \dfrac{n : A \vee B, S}{n : A, S \qquad n : B, S}$
ν-rules	
$(\nu_D)\quad \dfrac{n : \Box S, S'}{n+1 : S^{n+1}}$	$(\nu_T)\quad \dfrac{n : \Box A, S}{n : A^n, \Box A, S}$
π-rules	
$(\pi_K)\quad \dfrac{n : \Diamond A, \Box S, S'}{n+1 : A^{n+1}, S^{n+1}}$	$(\pi_4)\quad \dfrac{n : \Diamond A, \Box S, S'}{n+1 : A^{n+1}, S^{n+1}, \Box S}$

Fig. 1. Propositional expansion rules

varying domains and non-rigid designation: *level n;*
varying domains and rigid designation: *level n and level 0;*
cumulative domains: *level $k \leq n$.*

2. *If $n : S$ is a tableau node, then its domain is:*

$$dom(n : S) = \{a_p^p\} \cup \{t \mid t \text{ occurs in } S \text{ and } t \text{ is in depth } n\},$$

where

varying domains, rigid designation and local terms: *if there is at least a constant in L then $p = 0$, otherwise $p = n$;*
varying domains and either non-rigid designation or non-local terms: *$p = n$;*
cumulative domains: *$p = 1$*

Roughly speaking, the object denoted by a term in $dom(n : S)$ (and occurring in a tableau) can be assumed to exist in the domain of the world "corresponding" to the node $n : S$. Such terms are formed by use of symbols of level n itself, symbols of level 0 in the case of rigid designation (i.e. symbols of the base language L), and symbols having lower level in the case of cumulative domains, e.g. constants denoting objects existing in the domain of some previous world, therefore existing also in the considered world (in the ground version of the calculus we are presenting in this section, the only functional symbols with non null arity are those of the base language L).

Note that an annotated term may be in more than one depth. Moreover, if $t \in dom(n : S)$, then t is a ground term, it does not contain any non-annotated symbol and every subterm t' of t is in $dom(n : S)$.

The quantifiers expansion rules in Tab_G are the following:

$(\gamma)\quad \dfrac{n : \forall xA, S}{n : A[t/x], \forall xA, S}$	$(\delta)\quad \dfrac{n : \exists xA, S}{n : A[c^n/x], S}$

In the γ-rule, t is any ground term in $dom(n : \forall xA, S)$. In the δ-rule, c is a constant in $\mathcal{C}$ that does not occur in $\{A\} \cup S$. Note that c needs not be new in the whole tableau: the local restriction on the δ-rule corresponds exactly to the *eigenvariable* condition in sequent calculi.

Since $dom(n : \forall xA, S)$ is never empty, even if $n : \forall xA, S$ does not contain any ground term, the γ-rule can always be applied at least once to such a node. The constants of the set $\mathcal{A}$ (see Definition 1) in fact are used to take into account the fact that the object domains are never empty, in much the same way as an additional constant is used to form Herbrand domains in classical logic when there are no ground terms. The difference, here, is that we may need to have an infinite number of such constants, one for each depth. If domains are cumulative, then one is enough: it denotes any object in the initial world w_0, that exists in any other world accessible from w_0. Also if domains can vary but the base language L contains at least a constant c and terms are assumed to be local, one additional constant is enough: in fact, if terms of L are local, c denotes an object existing in every world, and the constant a_0^0 will be taken to denote such an object. If none of the previous cases applies, then it may be the case that any two object domains have empty intersection, so we need a different constant to name an arbitrary object of each domain.

In Tab_G, once that an occurrence of a symbol gets a level, it never changes. In particular, new constants introduced by the δ-rule never change their level and the only symbols that may occur with different annotations in a tableau node are symbols from the base language, when we are in the case of non-rigid designation and local terms. In that case, f^n is the denotation of f in the world corresponding to the node of depth n, that can be different from the denotation of the same symbol in another world. In consideration of this fact, the notion of complementary literals does not ignore the level of symbols.

Definition 4. *Two literals P and $\neg Q$ in the language L^+ of a tableau are complementary iff P and Q are identical, including their annotations. A node is closed if it contains two complementary literals. A tableau is closed iff all its leaves are closed.*

For example, $p(c^1)$ and $\neg p(c^1)$ are complementary, but $p(c^1)$ and $\neg p(c^2)$ are not.

In the case of cumulative domains, rigid designation and local terms, the calculus is not different from the tableau system described in [9], Chapter 7 (where, however, the language does not contain any functional symbol with non null arity). In fact, every functional symbol in a tableau is annotated and $n : S$ never contains symbols of level greater than n. So every term occurring in $n : S$ is in $dom(n : S)$.

Here follow some examples. The tableau on the left below proves the validity of **CBF** in the cumulative domain logics, using the π_K-rule. Since there are no functional symbols in L_F, there is no distinction to be made about the designation of terms. The last step in the varying domain systems would not be allowed: $\forall x p(x)$ can be instantiated only with terms in depth 2. Moreover, since there are no choice points above and the last expansion can only be replaced by an application of the γ-rule on the constant a_2^2 (there are no constants in L_F), **CBF** cannot be proved valid in the varying domains systems.

$$\dfrac{\dfrac{\dfrac{\dfrac{1 : \neg(\Box\forall x p(x) \to \forall x \Box p(x))}{1 : \Box\forall x p(x),\ \neg\forall x \Box p(x)}\ (\alpha)}{1 : \Box\forall x p(x),\ \neg\Box p(c^1)}\ (\delta)}{2 : \forall x p(x),\ \neg p(c^1)}\ (\pi_K)}{2 : p(c^1),\ \forall x p(x),\ \neg p(c^1)}\ (\gamma)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{1 : \neg(\forall x \Box p(x) \to \Box\forall x p(x))}{1 : \forall x \Box p(x),\ \neg\Box\forall x p(x)}\ (\alpha)}{1 : \Box p(a^1),\ \forall x \Box p(x),\ \neg\Box\forall x p(x)}\ (\gamma)}{2 : p(a^1),\ \neg\forall x p(x)}\ (\pi_k)}{2 : p(a^1),\ \neg p(c^2)}\ (\delta)$$

A similar reasoning shows that no instance of the Barcan formula (**BF**) $\forall x \Box A \to \Box\forall x A$, that characterizes constant domains, is provable. The tableau above on the right is an attempt, using again the π_K-rule and the instance where $A = p(x)$. In this case there is no difference between the cumulative and varying domains variants of the calculus. Since with the application of the π_K-rule the universal formula is necessarily lost, the tableau cannot be closed.

Below, on the left, there is a proof of $\Box(\forall x p(x) \to p(c))$ (in any propositional basis) with rigid designation (either varying or cumulative domains: in both cases the term c^0 is in any depth) and local terms, showing that, in this variant of the systems, the rule of instantiation is necessarily valid. The same formula has a different proof (on the right, below) in the case of non rigid designation (and local terms).

$$\dfrac{\dfrac{\dfrac{1 : \neg\Box(\forall x p(x) \to p(c^0))}{2 : \neg(\forall x p(x) \to p(c^0))}\ (\pi)}{2 : \forall x p(x),\ \neg p(c^0)}\ (\alpha)}{2 : p(c^0),\ \forall x p(x),\ \neg p(c^0)}\ (\gamma)$$

$$\dfrac{\dfrac{\dfrac{1 : \neg\Box(\forall x p(x) \to p(c))}{2 : \neg(\forall x p(x) \to p(c^2))}\ (\pi)}{2 : \forall x p(x),\ \neg p(c^2)}\ (\alpha)}{2 : p(c^2),\ \forall x p(x),\ \neg p(c^2)}\ (\gamma)$$

And it cannot be proved when terms are not necessarily local. Here follows a failed attempt:

$$\dfrac{\dfrac{\dfrac{1 : \neg\Box(\forall x p(x) \to p(c))}{2 : \neg(\forall x p(x) \to p(c))}\ (\pi)}{2 : \forall x p(x),\ \neg p(c)}\ (\alpha)}{2 : p(a_2^2),\ \forall x p(x),\ \neg p(c)}\ (\gamma)$$

The next example is a (failed) attempt to show that $p(c), \forall x(p(x) \to \Box p(x)) \vdash \Box p(c)$ in the case of non-rigid designation and local terms:

$$\dfrac{\dfrac{\dfrac{1 : \forall x(p(x) \to \Box p(x)),\ p(c^1),\ \neg\Box p(c)}{1 : \forall x(p(x) \to \Box p(x)),\ p(c^1) \to \Box p(c^1),\ p(c^1),\ \neg\Box p(c)}\ (\gamma)}{1 : \forall x(p(x) \to \Box p(x)),\ \neg p(c^1),\ p(c^1),\ \neg\Box p(c) \qquad \dfrac{1 : \forall x(p(x) \to \Box p(x)),\ \Box p(c^1),\ p(c^1),\ \neg\Box p(c)}{2 : p(c^1),\ \neg p(c^2)}\ (\pi_k)}\ (\beta)$$

The leftmost leaf is closed, but the rightmost is not, because $p(c^1)$ and $\neg p(c^2)$ are not complementary. And in fact $\Box p(c)$ is not a logical consequence of $p(c)$ and $\forall x(p(x) \to \Box p(x))$ in the case of non-rigid designation and local terms. If terms are not local, even the first inference is not correct. In fact, the tableau is initialized without any annotation on symbols and the γ-rule can never be applied to a term with no levels.

3 Free Variable Tableaux

In this section we introduce Tab_{FV}, i.e. free-variable tableaux for the considered logics. The definition of quantifier rules is similar to the classical case, but for the annotation on symbols that is taken into account in the definition of modal substitutions. First of all, we define a further extension of the labelled extension of a language (Definition 1), by addition of *parameters* and *Skolem functions*:

Definition 5. *Let L be a modal language, $\mathcal{P} = \{z_1^{n_1}, z_2^{n_2}, ...\}$ an infinite denumerable set of new annotated symbols, called parameters, and $\mathcal{F}$ a denumerable set of new annotated function symbols such that, for each $i, j \in \mathbb{N}$, $\mathcal{P}$ contains infinitely many parameters of level i and $\mathcal{F}$ contains infinitely many function symbols of level i and arity j. Then the* labelled free variable extension *of L is the language L^{fv} such that $L_P^{fv} = L_P$ and*

$$L_F^{fv} = L_F^+ \cup \mathcal{P} \cup \mathcal{F}$$

The levels of parameters will be shown only when relevant. The definition of the depth of a term (Definition 3) carries along to terms in L^{fv} with no modification. The next definition introduces the notion of modal substitution, where a parameter z can be mapped to a term t in L^{fv} only if the annotation of symbols in t "agrees" with the level of z. Since the agreement depends on which depths the term is, the different DDE variants of QML correspond to different restrictions on modal substitutions.

Definition 6. *A modal substitution σ is a function from $\mathcal{P}$ to terms in L^{fv}, that is the identity almost everywhere, and such that if $\sigma(z^n) = t$ (and z has level n) then t is in depth n. Substitutions are denoted as usual by expressions of the form $\{t_1/z_1, ..., t_m/z_m\}$.*

Unifiers, i.e. solutions of unification problems, and most general unifiers (*m.g.u.*) are defined like in the classical case. The following definition is also borrowed from classical tableau calculi.

Definition 7. *Let $n_1 : S_1, ..., n_k : S_k$ be all the leaves of a free variable tableau $\mathcal{T}$, and, for each $i = 1, ..., k$, let P_i and $\neg Q_i$ be a pair of literals in $\mathcal{S}_i$. If the modal substitution σ is a solution of the unification problem $P_1 = Q_1, ..., P_k = Q_k$, then σ is an* atomic closure substitution for $\mathcal{T}$. *If it is a most general solution for such a unification problem, then it is a* most general atomic closure substitution for $\mathcal{T}$.

Besides the propositional expansion rules, that are the same as in Tab_G (but for the fact that parameters and Skolem functions can occur in tableau nodes), the expansion rules of Tab_{FV} are:

Quantifier expansion rules :

(γ)	$\dfrac{n : \forall x A, S}{n : A[z^n/x], \forall x A, S}$	(δ)	$\dfrac{n : \exists x A, S}{n : A[f^n(z_1^n, ..., z_k{}^n)/x], S}$

In the γ-rule, z is a new parameter. In the δ-rule, f^n is a new Skolem function, i.e. a symbol of level n in $\mathcal{F}$ that does not occur in $\{\exists x A\} \cup S$, and $z_1{}^n, ..., z_k{}^n$ are all the parameters with level n in $\exists x A, S$.

Most General Atomic Closure Substitution Rule :

If $\mathcal{T}$ is a tableau for a set S of sentences in L and the modal substitution σ is a most general atomic closure substitution for $\mathcal{T}$, then the tableau $\mathcal{T}\sigma$, obtained by applying σ to $\mathcal{T}$, is a tableau for S.

Note that the Skolem term $f^n(z_1^n, ..., z_k{}^n)$ introduced by an application of the δ-rule contains only parameters with level n, and the Skolem function f is required to be "locally" new (i.e. new in the node). This is enough to ensure that the rule is sound. In fact, the role of the restriction on f and the parameters in the term is to prevent a modal substitution σ to make $f^n(z_1^n, ..., z_k{}^n)$ equal to some other parameter z occurring in the premise of the δ-rule, $n : \exists x A, S$. If z has level n then $\sigma(f^n(z_1^n, ..., z_k{}^n)) = \sigma(z)$ is impossible, because z is one of $z_1^n, ..., z_k{}^n$. The level m of z cannot be greater than n, since a node never contains any symbol with level greater than its depth. Therefore, if $m \neq n$, then m is necessarily strictly less than n, and, by definition, a modal substitution cannot map a variable with level $m < n$ to a term containing a symbol of level n, that is not in depth m in any DDE variant of QML.

The definition of closed tableau (Definition 4) stays unchanged for Tab_{FV} (modulo the possible presence of parameters and Skolem functions in terms). Note that in the calculi Tab_{FV} defined here, closed tableaux are built up by first applying expansion rules, then computing an m.g.u. allowing to simultaneously close all the leaves, and applying the substitution rule to the tableau so far

constructed just once. Assuming a single, final application of the substitution rule makes the soundness proof simpler. However, like in the classical case, an alternative version of the rule can be considered, where any m.g.u. of a pair of literals in a single leaf can be applied. Since any most general atomic closure substitution can be found by repeated applications of such an alternative rule, this second version of the calculi is equivalent to the first one.

As an example, below, we give again a proof of the validity of **CBF** in the cumulative domain logics, using the π_K-rule, but this time in the calculus Tab_{FV}.

$$\dfrac{\dfrac{\dfrac{\dfrac{1 : \neg(\Box\forall x p(x) \to \forall x \Box p(x))}{1 : \Box\forall x p(x), \neg\forall x \Box p(x)}\,(\alpha)}{1 : \Box\forall x p(x), \neg\Box p(c^1)}\,(\delta)}{2 : \forall x p(x), \neg p(c^1)}\,(\pi_K)}{2 : p(z_1^2), \forall x p(x), \neg p(c^1)}\,(\gamma)$$

Here, c^1 is a (0-ary) Skolem function. Since c^1 is in depth 2, it suffices to apply the substitution $\{c^1/z_1^2\}$ to close the tableau. In the case of varying domain systems, such a substitution is illegal, because c^1 is not in depth 2.

As a further example, the tableau below is a Tab_{FV} proof of the validity of the formula $\Box\forall x \exists y p(x,y) \to \exists v \Box \exists w p(v,w)$ in the logic **K** with cumulative domains (since L_F is empty, there is no difference to be made about the designation of terms).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{1 : \neg(\Box\forall x\exists y p(x,y) \to \exists v\Box\exists w p(v,w))}{1 : \Box\forall x\exists y\ p(x,y),\ \neg\exists v\Box\exists w p(v,w)}\,(\alpha)}{1 : \Box\forall x\exists y\ p(x,y),\ \neg\Box\exists w p(z_1^1,w)}\,(\gamma)}{2 : \forall x\exists y\ p(x,y),\ \neg\exists w p(z_1^1,w)}\,(\pi_K)}{2 : \exists y\ p(z_2^2,y),\ \forall x\exists y\ p(x,y),\ \neg\exists w p(z_1^1,w)}\,(\gamma)}{2 : p(z_2^2,g^2(z_2^2)),\ \forall x\exists y\ p(x,y),\ \neg\exists w p(z_1^1,w)}\,(\delta)}{2 : p(z_2^2,g^2(z_2^2)),\ \forall x\exists y\ p(x,y),\ \neg\exists w p(z_1^1,w),\ \neg p(z_1^1,z_3^2)}\,(\gamma)$$

Since domains are cumulative, the term z_1^1 is in any depth $k \geq 1$, thus, in particular, it is in depth 2. Hence $\{z_1^1/z_2^2,\ g^2(z_1^1)/z_3^2\}$ is a modal substitution, and constitutes a most general atomic closure substitution. Its application yields a closed tableau. Note that in the varying domains option, no substitution can close the tableau, since z_1^1 is not in depth 2, and neither $\{z_1^1/z_2^2\}$ nor $\{z_2^2/z_1^1\}$ are legal substitutions. Indeed, the considered formula is not valid in this case.

4 Soundness and Completeness of Tab_G and Tab_{FV}

In this section we briefly sketch the main ideas underlying the soundness and completeness proofs for both systems Tab_G and Tab_{FV}.

In order to show that Tab_G is sound, we inductively show that every tableau for a satisfiable set of formulae S has always an open leaf, i.e. a satisfiable leaf,

where a node $n : S$ is satisfied by a world w in $\mathcal{M}$ iff $\mathcal{M}, w \models S$ (ignoring annotations) and for every ground term $t \in dom(n : S)$, $\mathcal{M}(w, t) \in \delta(w)$. The latter requirement is exploited to show that if the premise of a γ-rule is satisfiable, then so is its expansion. And it must be guaranteed to hold both in the initial tableau and when a dynamic rule is applied.

In the completeness proof for Tab_G, we first consider a calculus where the γ-rule is replaced by its "wasteful" variant γ^*:

$$(\gamma^*)\ \frac{n : \forall xA, S}{n : A[t/x], \forall xA, S} \qquad \begin{array}{l}\text{where } t \text{ is any ground term in depth} \\ n \text{ of the language of the tableau } L^+\end{array}$$

A tableau in this modification of the calculus will be called a γ^*-tableau. Completeness is proved by construction of a "syntactical model" of any set of sentences that has no closed γ^*-tableau, and the construction obviously exploits the wasteful γ^*-rule. Then we show that any γ^*-expansion in a closed γ^*-tableau can be replaced by a γ-expansion (using a suitable term from the set of symbols $\mathcal{A}$ introduced in Definition 1), thus obtaining a closed tableau in the original ground calculus Tab_G.

The syntactical model of the unrefutable set of formulae S is built in two steps. First of all an interpretation $\mathcal{M}$ of the extension L^+ of the language L of S is constructed, where however annotations are not ignored: a symbol f^i is considered different from f^j when $i \neq j$. The frame of $\mathcal{M}$ is a (possibly infinite and infinitely branching) tree, whose elements are saturated sets of formulae (the definition of saturation is similar to the standard one, obviously referring to the calculi we are considering). The initial world w_0 is a saturation of S^*, where $1 : S^*$ is the initial tableau for S. Each element w of the frame is associated with a natural number $depth(w)$, that measures its distance from the root.

The elements δ and ϕ of $\mathcal{M}$ make each possible world w a kind of Herbrand interpretation: the domain $\delta(w)$ is the set of ground terms in $dom(depth(w), w)$, where terms that differ (even only) in the annotation of their symbols are different elements of the domain. Annotated functional symbols are always given a rigid interpretation, in such a way that, if a term t contains only annotated symbols, then $\phi(w, t) = t$ for all $w \in W$. The treatment of non-annotated symbols differs in the cases of local and non-local terms. Finally, an atom P is true at w iff $P \in w$. It can be shown that, for all $w \in W$, $\mathcal{M}, w \models A$ for all $A \in w$, where, like before, symbols occurring with different levels are considered different. For this reason, $\mathcal{M}$ is not yet an interpretation of the language L, and we cannot directly conclude that $\mathcal{M} \models S$. From the interpretation of the (possibly infinite) set of symbols $f^1, f^2, f^3, ...$ where $f \in L_F$, *the* interpretation of f is extracted for every $w \in W$. Such a new interpretation satisfies all the requirements of the different DDE variants of QML we are considering, and is indeed a model of S.

The soundness and completeness of Tab_{FV} is established by showing the equivalence between the calculi Tab_{FV} and Tab_G. Therefore, since Tab_G is sound and complete, S is unsatisfiable if and only if there is a closed Tab_{FV} tableau for S. The proof is based on a syntactical transformation of Tab_G-tableau proofs into tableau proofs in Tab_{FV} and viceversa.

5 Concluding Remarks

In this paper we provide calculi treating different variants of first order modal logic (varying and cumulative domains, rigid and non rigid designation, local and non local terms), by addition of a minimal semantical annotation to tableau methods obtained by extending propositional modal systems with the classical rules for quantifiers. We give both a ground formulation of the calculi (Tab_G) and free variable tableau systems (Tab_{FV}). In both cases, the different DDE variants are obtained in a simple and modular way, just by choosing among alternative definitions of a few basic operations. In particular, in the case of Tab_{FV} calculi, a first order variant is essentially characterized by the notion of modal substitution. Differently from other approaches such as for instance [15,2], however, term unification does not depend on the propositional kernel of the logic (i.e. on the accessibility relation) and does not involve any complex or strongly semantically dependent operation: what is additionally needed, with respect to classical unification, is some numerical "greater than" test.

Although we have treated each DDE variant as a different logic, some mixed approaches are possible with respect to the denotation of symbols: for example, it can be established that constant symbols (proper names) are rigid while functional symbols are not, so that definite descriptions do not necessarily denote rigidly. Each symbol is then to be treated according to the corresponding variant in the construction of the initial tableau and in the application of dynamic rules. We believe that equality can easily be treated in the same style.

In the case of the (proof-theoretically) easiest DDE variant assuming cumulative domains, rigid designation and local terms, the calculi Tab_G are actually the same as what is obtained by a simple addition of classical quantifier rules to modal propositional ones (such as, for example, in [9], Chapter 7). However, their Tab_{FV} counterpart is not exactly the same as proposed in [10], where a tableau calculus with free variables, dynamic skolemization and unification is defined, that is correct and complete for this variant of QML. There, the new Skolem term $f(z_1, ..., z_n)$ introduced by an application of the δ-rule is such that $z_1, ..., z_n$ are all the parameters occurring in the branch. Exploiting the annotation on symbols, the δ-rule in Tab_{FV} is somewhat more liberal, requiring only parameters having the same level as the depth of the node to be the arguments of the Skolem functions. Probably, the δ-rule can be further relaxed, in the style of [14,6,4]. The details, however, still have to be worked out.

The (ground) prefixed tableaux presented in Chapter 8 of [9], in the three different variants of constant, cumulative and varying domains, associate prefixes with constants introduced by applications of the δ-rule in a way that bears some resemblance with Tab_G and its labelling mechanism. The present approach, however, is not in the style of prefixed tableaux, where prefixes are associated to each formula (not the whole set of "active" formulae), are more complex structures than integers and play a fundamental role in modal expansion rules. A stricter relation links the present work with [7,8], where the same kind of annotation is used to restrict unification in the context of modal resolution. However, being the Skolem-like transformation static in that case, it works well

with logics such as **K** and **D**, where formulae cannot move from their modal context, but requires a preliminary propositional transformation of formulae when dealing with reflexive and/or transitive logics, where the modal degree of a formula is subject to change. The dynamic introduction of parameters and Skolem terms in tableau systems makes things easier: a quantifier is treated only when it comes to the surface, i.e. when, after possible changes due to applications of the ν_T or π_4-rule, its modal context is settled.

References

1. M. Abadi and Z. Manna. Modal theorem proving. *CADE-86*, pages 172–189, 1986. Springer, LNCS 230.
2. A. Artosi, P. Benassi, G. Governatori, and A. Rotolo. Shakespearian modal logic: A labelled treatment of modal identity. *Advances in Modal Logic*, pages 1–21, 1998.
3. Y. Auffray and P. Enjalbert. Modal theorem proving: an equational viewpoint. *J. of Logic and Computation*, 2:247–297, 1992.
4. M. Baaz and C.G. Fermüller. Non-elementary speedups between different versions of tableaux. *Tableaux '95*, pages 217–230, 1995. Springer, LNCS 918.
5. D. Basin, M. Matthews, and L. Viganò. Labelled modal logics: Quantifiers. *J. of Logic, Language and Information*, 7(3):237–263, 1998.
6. B. Beckert, R. Hähnle, and P. H. Schmitt. The even more liberalized δ-rule in free variable semantic tableaux. In *Proc. of the 3rd Kurt Gödel Colloquium KGC'93*, pages 108–119, 1993. Springer, LNCS 713.
7. M. Cialdea. Resolution for some first order modal systems. *Theoretical Computer Science*, 85:213–229, 1991.
8. M. Cialdea and L. Fariñas del Cerro. A modal Herbrand's property. *Z. Math. Logik Grundlag. Math.*, 32:523–539, 1986.
9. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics.* Reidel Publ. Co., 1983.
10. M. Fitting. First-order modal tableaux. *J. of Automated Reasoning*, 4:191–213, 1988.
11. M. Fitting. On quantified modal logic. *Fundamenta Informaticae*, 39:105–121, 1999.
12. J. W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, vol. II, 249–307. Reidel Publ. Co., 1984.
13. R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, G. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of tableau methods.* Kluwer, 1999.
14. R. Hähnle and P. H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *J. of Automated Reasoning*, 13:211–222, 1994.
15. P. Jackson and H. Reichgelt. A general proof method for first-order modal logic. IJCAI '87, pages 942–944. Morgan Kaufmann, 1987.
16. K. Konolige. Resolution and quantified epistemic logics. *CADE 86*, pages 199–208, 1986. Springer, LNCS 230.
17. G. Mints. Indexed systems of sequents and cut-elimination. *J. of Philosophical Logic*, 26:671–696, 1997.
18. H. J. Ohlbach. Semantics based translation methods for modal logics. *J. of Logic and Computation*, 1(5):691–746, 1991.
19. L. A. Wallen. *Automated Deduction in Nonclassical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics.* MIT Press, 1990.

Complexity of Simple Dependent Bimodal Logics

Stéphane Demri

Laboratoire LEIBNIZ-CNRS, U.M.R. 5522
46 Avenue Felix Viallet, 38031 Grenoble, France
Email: demri@imag.fr

Abstract. We characterize the computational complexity of simple dependent bimodal logics. We define an operator $\oplus_{\subseteq}$ between logics that almost behaves as the standard joint operator $\oplus$ except that the inclusion axiom $[2]\mathtt{p} \Rightarrow [1]\mathtt{p}$ is added. Many multimodal logics from the literature are of this form or contain such fragments. For the standard modal logics K,T,B,$S4$ and $S5$ we study the complexity of the satisfiability problem of the joint in the sense of $\oplus_{\subseteq}$. We mainly establish the **PSPACE** upper bounds by designing tableaux-based algorithms in which a particular attention is given to the formalization of termination and to the design of a uniform framework. Reductions into the packed guarded fragment with only two variables introduced by M. Marx are also used. E. Spaan proved that $K \oplus_{\subseteq} S5$ is **EXPTIME**-hard. We show that for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{K, T, B\} \times \{S4, S5\}$, $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ is also **EXPTIME**-hard.

1 Introduction

Combining logics The combination of modal logics has deserved in the past years a lot of attention (see e.g. [12,15,21,16,2,24]) and this is an exciting area. Indeed, not only there are many ways to combine logics (fusion, product, ...) but also many properties of the combined logics deserve to be studied (completeness, compactness, finite model property, interpolation, decidability, complexity, ...). In this paper, we are mainly concerned with computational complexity issues and as a side-effect with the design of tableaux-based decision procedures. The simplest way to combine two logics is to take their fusion, that is to obtain a bimodal logic which has no axioms that use both of the operators. For two normal modal logics $\mathcal{L}_1$ and $\mathcal{L}_2$, we write $\mathcal{L}_1 \oplus \mathcal{L}_2$ to denote the smallest bimodal logic with two independent modal operators, say [1] and [2]. The complexity of such logics has been analyzed in [19] and from [22,18], we know that for instance the logics K $\oplus$ K, S5 $\oplus$ S5, S4 $\oplus$ S5 and K $\oplus$ S5 have **PSPACE**-complete satisfiability problems.

Other combinators for modal logics are relevant (see e.g. [15,16]). We write $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ to denote the smallest bimodal logic containing $\mathcal{L}_1 \oplus \mathcal{L}_2$ and the axiom schema $[2]\mathtt{p} \Rightarrow [1]\mathtt{p}$. It is not very difficult to design new operators since each recursive set of bimodal formulae potentially induces a way to combine two logics. The fusion operator $\oplus$ is simply associated to the empty set of formulae. In the paper, we investigate the complexity of bimodal logics obtained

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 190–204, 2000.

from monomodal logics by application of $\oplus_{\subseteq}$. To be more precise, we adopt a semantics-oriented definition since we define an operator $\oplus_{\subseteq}$ on classes of monomodal frames. The logics of the form $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ are said to be *simple dependent* bimodal logics. For instance, adding a universal modal operator to certain monomodal logics corresponds exactly to operating with $\oplus_{\subseteq}$. Unlike $\oplus$, $\oplus_{\subseteq}$ does not preserve decidability since by [30], a Horn modal logic whose satisfiability is in **NP**, is shown to be undecidable when extended with the universal modal operator. Complexity neither transfers. Indeed, $K \oplus_{\subseteq} S5$ has an **EXPTIME**-hard satisfiability problem [30] although K-satisfiability is **PSPACE**-complete and $S5$-satisfiability is **NP**-complete [22]. In this paper, we analyze the complexity of the logics $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ for $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B, S4, S5\}$. To establish the **PSPACE** upper bounds, we design Ladner-like algorithms [22,18,31] (see also [20,26,35] for proof-theoretical analyses) that are known to be close to tableau-based procedures. We invite the reader to consult [8] for understanding how the semantical analysis in [22] can be given a proof-theoretical interpretation. Furthermore, the (semantical) analysis developed in the paper can be plug into a labelled tableaux calculus for the logics. Actually, such a calculus is not difficult to define for such logics following for instance [1].

One may wonder why the operator $\oplus_{\subseteq}$ deserves some interest. After all, any bimodal formula generates an operator on logics. Actually, many logics can be explained in terms of $\oplus_{\subseteq}$. Below are few examples:

- the propositional linear temporal logic PLTL with future F and next X: PLTL-satisfiability is **PSPACE**-complete whereas the fragment with F only [resp. with X only] is in **NP** [29];
- the logic S4+5 is shown to have a satisfiability problem equal to the satisfiability problem for $S5 \oplus_{\subseteq} S4$ [33];
- many other logics have valid formulae of the form $[2]\mathtt{p} \Rightarrow [1]\mathtt{p}$, from epistemic logics to provability bimodal logics (see e.g. [36]) passing via variants of dynamic logic approximating the Kleene star operator [7];
- information logics derived from information systems (see e.g. [34]).

Our contribution. The technical contribution of the paper is to characterize the computational complexity of the satisfiability problem for the logics $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ with $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B, S4, S5\}$ (see e.g. [27] for a thourough introduction to complexity theory). The choice of the logics is a bit arbitrary since many other standard modal logics would deserve such an analysis (the standard modal logics D, K4, G, S4.3, S4.3.1 to quote a few of them). However, we felt that with the present sample, we could reasonably show the peculiarity of $\oplus_{\subseteq}$ and how the Ladner-like algorithms are precious to establish **PSPACE** upper bounds. Moreover, many proofs can be adapted to other logics not explicitly studied here. By way of example, $D \oplus_{\subseteq} K4$ can be shown to be **EXPTIME**-complete. In Table 1, we summarize the results. In the table, each problem in a given class

is complete for the class with respect to logarithmic space transformations[1]. A generalization from the bimodal case to the case with $n \geq 2$ modal connectives is sketched in Section 6 and is planned to be fully treated in a longer version.

Table 1. Worst-case complexity of simple dependent bimodal logics

$\oplus_{\subseteq}$	K	T	B	S4	S5
K	**PSPACE**	**PSPACE**	**PSPACE**	**EXPTIME**	**EXPTIME**
T	**PSPACE**	**PSPACE**	**PSPACE**	**EXPTIME**	**EXPTIME**
B	**PSPACE**	**PSPACE**	**PSPACE**	**EXPTIME**	**EXPTIME**
S4	**PSPACE**	**PSPACE**	**PSPACE**	**PSPACE**	**PSPACE**
S5	**PSPACE**	**PSPACE**	**PSPACE**	**PSPACE**	**NP**

The second contribution consists in comparing the operators $\oplus$ and $\oplus_{\subseteq}$. Although we know that $\oplus$ preserves for instance decidability (see e.g. [21]), $\oplus_{\subseteq}$ does not, even if we restrict ourselves to logics with **NP**-complete satisfiability problems (see e.g. [30]). It is known that for $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B, S4, S5\}$, $\mathcal{L}_1 \oplus \mathcal{L}_2$ has a **PSPACE**-complete satisfiability problem. By contrast, we show that for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{K, T, B\} \times \{S4, S5\}$, $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ is **EXPTIME**-complete. In a sense, the **EXPTIME**-hardness of the minimal normal modal logic K augmented with the universal modal connective, should not necessarily be explained by the presence of the universal modal connective but rather as due to the *dependent combinaison* of a logic between K and B and a logic between K4 and S5. This reinterpretation is another original aspect of our investigation.

The last but not least contribution stems in the design of parametrized Ladner-like algorithms that allows to establish **PSPACE** upper bounds. We shall come back to this point throughout the paper.

2 Bimodal Logics

For any set X, we write X^* to denote the set of finite strings built from elements of X. λ denotes the empty string. For any finite string s, we write $|s|$ [resp. $last(s)$] to denote its length [resp. the last element of s, if any]. For $s \in X^*$, for $j \in \{1, \ldots, |s|\}$, we write $s(j)$ [resp. $s[j]$] to denote the jth element of s [resp. to denote the initial substring of s of length j]. By convention $s[0] = \lambda$. For any $s \in X^*$, we write s^k to denote the string composed of k copies of s. For instance, $(1.2)^2 = 1.2.1.2$ and $|(1.2)^2| = 4$.

Given a countably infinite set $\mathtt{For}_0 = \{\mathtt{p}_0, \mathtt{p}_1, \mathtt{p}_2, \ldots\}$ of *propositional variables* the *bimodal formulae* ϕ are inductively defined as follows: $\phi ::= \mathtt{p}_k \mid$

[1] In Table 1, we consider the satisfiability problem for $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ (or equivalently the consistency problem). In order to get the table for the validity problem (or equivalently the theoremhood problem), replace **NP** by **coNP**.

$\phi_1 \wedge \phi_2 \mid \neg\phi \mid [1]\phi \mid [2]\phi$ for $\mathtt{p}_k \in \mathtt{For}_0$. The *monomodal formulae* are bimodal formulae without occurrences of [2]. We write $|\phi|$ to denote the *length* of the formula ϕ, that is the length of the string ϕ. We write $md(\phi)$ to denote the modal degree of ϕ, that is the modal depth of ϕ. md is naturally extended to finite sets of formulae, understood as conjunctions and by convention $md(\emptyset) = 0$. For $j \in \{1, 2\}$, $[j]^i\phi$ is inductively defined as follows: $[j]^0\phi = \phi$ and $[j]^{i+1}\phi = [j][j]^i\phi$ for $i \geq 0$. For $s \in \{[1], [2]\}^*$, an s-formula is defined as a formula prefixed by s.

A *monomodal* [resp. *bimodal*] *frame* $\mathcal{F}$ is a structure of the form $\langle W, R_1\rangle$ [resp. $\langle W, R_1, R_2\rangle$] such that W is a non-empty set, R_1 is a binary relation on W [resp. R_1 and R_2 are binary relations on W]. A *monomodal* [resp. *bimodal*] *model* $\mathcal{M}$ is a structure of the form $\langle W, R_1, m\rangle$ [resp. $\langle W, R_1, R_2, m\rangle$] such that $\langle W, R_1\rangle$ [resp. $\langle W, R_1, R_2\rangle$] is a monomodal [resp. bimodal] frame and m is a map $m : \mathtt{For}_0 \to \mathcal{P}(W)$. $\mathcal{M}$ is said to be *based on* the frame $\langle W, R_1\rangle$ [resp. $\langle W, R_1, R_2\rangle$]. Most of the time we omit the terms 'monomodal' and 'bimodal' when it is clear from the context what kind of objects we are dealing with.

As is usual, the formula ϕ is *satisfied by the world* $w \in W$ *in* $\mathcal{M}$ $\overset{\text{def}}{\Leftrightarrow}$ $\mathcal{M}, w \models \phi$ where the satisfaction relation $\models$ is inductively defined as follows:

- $\mathcal{M}, w \models \mathtt{p} \overset{\text{def}}{\Leftrightarrow} w \in m(\mathtt{p})$, for every propositional variable $\mathtt{p}$;
- $\mathcal{M}, w \models [1]\phi \overset{\text{def}}{\Leftrightarrow}$ for every $w' \in R_1(w)$, $\mathcal{M}, w' \models \phi$;
- $\mathcal{M}, w \models [2]\phi \overset{\text{def}}{\Leftrightarrow}$ for every $w' \in R_2(w)$, $\mathcal{M}, w' \models \phi$ when $\mathcal{M}$ is bimodal.

We omit the standard conditions for the propositional connectives. Let $\mathcal{C}$ be a class of monomodal [resp. bimodal] frames. A monomodal [resp. bimodal] formula is said to be $\mathcal{C}$-*satisfiable* $\overset{\text{def}}{\Leftrightarrow}$ there is a model $\mathcal{M}$ based a frame $\mathcal{F} \in \mathcal{C}$ such that $\mathcal{M}, w \models \phi$ for some $w \in W$. We write $SAT(\mathcal{C})$ to denote the class of $\mathcal{C}$-satisfiable formulae. We write $\mathcal{F} \models \phi$ to denote that for any model $\mathcal{M}$ based on $\mathcal{F}$ and for w in $\mathcal{M}$, $\mathcal{M}, w \models \phi$. The following standard classes of monomodal frames shall be used:

- $\mathcal{C}_K$ is the class of all the frames; $\mathcal{C}'_K$ [resp. $\mathcal{C}_T$, $\mathcal{C}_{K4}$] is the class of frames $\langle W, R_1\rangle$ such that R_1 is irreflexive [resp. reflexive, transitive];
- $\mathcal{C}_B$ [resp. $\mathcal{C}_{S4}$] is the class of frames $\langle W, R_1\rangle$ such that R_1 is reflexive and symmetric [resp. and transitive];
- $\mathcal{C}_{S5}$ [resp. $\mathcal{C}'_{S5}$] is the class of frames $\langle W, R_1\rangle$ such that R_1 is an equivalence relation [resp. $R_1 = W \times W$].

It is known that $SAT(\mathcal{C}_K) = SAT(\mathcal{C}'_K)$ and $SAT(\mathcal{C}_{S5}) = SAT(\mathcal{C}'_{S5})$. Let $\mathcal{C}_i$ for $i = 1, 2$ be classes of monomodal frames. We write $\mathcal{C}_1 \oplus \mathcal{C}_2$ [resp. $\mathcal{C}_1 \oplus_{\subseteq} \mathcal{C}_2$] to denote the class of bimodal frames $\langle W, R_1, R_2\rangle$ such that $\langle W, R_i\rangle \in \mathcal{C}_i$ for $i = 1, 2$ [resp. and $R_1 \subseteq R_2$].

A normal monomodal [resp. bimodal] logic is a set $\mathcal{L}$ of monomodal [resp. bimodal] formulae such that: $\mathcal{L}$ contains all propositional tautologies, $\mathcal{L}$ is closed under substitution, $[j](\mathtt{p} \Rightarrow \mathtt{q}) \Rightarrow ([j]\mathtt{p} \Rightarrow [j]\mathtt{q})$ for $j \in \{1\}$ [resp. for $j \in \{1, 2\}$], $\mathcal{L}$ is closed under modus ponens and $\mathcal{L}$ is closed under generalization, i.e. if $\phi \in \mathcal{L}$, then $[j]\phi$ for $j \in \{1\}$ [resp. for $j \in \{1, 2\}$]. Let K be the minimal normal monomodal logic. Below are other standard normal modal logics: $T \overset{\text{def}}{=}$

$K + [1]\mathtt{p} \Rightarrow \mathtt{p}$, $B \stackrel{\text{def}}{=} T + \mathtt{p} \Rightarrow [1]\neg[1]\neg\mathtt{p}$, $S4 \stackrel{\text{def}}{=} T + [1]\mathtt{p} \Rightarrow [1][1]\mathtt{p}$, $S5 \stackrel{\text{def}}{=} S4 + \mathtt{p} \Rightarrow [1]\neg[1]\neg\mathtt{p}$. A monomodal [resp. bimodal] logic $\mathcal{L}$ is said to be *complete* with respect to the class $\mathcal{C}$ of monomodal [resp. bimodal] frames $\stackrel{\text{def}}{\Leftrightarrow}$ for any monomodal [resp. bimodal] formula ϕ, $\phi \in SAT(\mathcal{C})$ iff $\neg\phi \notin \mathcal{L}$. For any modal logic $\mathcal{L}$, we write $\mathcal{C}_{\mathcal{L}}$ to denote the class of frames $\mathcal{F}$ such that for $\phi \in \mathcal{L}$, $\mathcal{F} \models \phi$. This notation $\mathcal{C}_{\mathcal{L}}$ is consistent with the classes of frames defined above.

Here is a first important difference between $\oplus$ and $\oplus_{\subseteq}$. Corollary 3.1.3 in [30] states that if $\mathcal{C}_1, \mathcal{C}'_1, \mathcal{C}_2, \mathcal{C}'_2$ are classes of monomodal frames closed under disjoint unions and if $SAT(\mathcal{C}_1) = SAT(\mathcal{C}'_1)$ and $SAT(\mathcal{C}_2) = SAT(\mathcal{C}'_2)$, then $SAT(\mathcal{C}_1 \oplus \mathcal{C}_2) = SAT(\mathcal{C}'_1 \oplus \mathcal{C}'_2)$. By constrast, $SAT(\mathcal{C}_T \oplus_{\subseteq} \mathcal{C}'_K) \neq SAT(\mathcal{C}_T \oplus_{\subseteq} \mathcal{C}_K)$: $\mathcal{C}_T \oplus_{\subseteq} \mathcal{C}'_K$ is empty whereas $SAT(\mathcal{C}_T \oplus_{\subseteq} \mathcal{C}_K)$ is **PSPACE**-hard.

We write $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ to mean the set $\mathcal{C}_{\mathcal{L}_1} \oplus_{\subseteq} \mathcal{C}_{\mathcal{L}_2}$ for any monomodal logic $\mathcal{L}_1, \mathcal{L}_2$. By way of example, $S5 \oplus_{\subseteq} S5$ equals $\mathcal{C}_{S5} \oplus_{\subseteq} \mathcal{C}_{S5}$ but not $\mathcal{C}'_{S5} \oplus_{\subseteq} \mathcal{C}'_{S5}$.

Lemma 1. *For* $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in (\{K, T, B, S4, S5\} \times \{K, T, B, S4\}) \cup \langle S4, S5 \rangle$*, the problem* $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ *is* **PSPACE***-hard.*

We invite the reader to consult [9] for further topics on modal logic.

3 PSPACE Ladner-Like Algorithms

In this section, we show that $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ is in **PSPACE** for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{S4, S5\} \times \{T, B, S4, S5\}$. Observe that $S4 \oplus_{\subseteq} K = S4 \oplus_{\subseteq} T$ and $S5 \oplus_{\subseteq} K = S5 \oplus_{\subseteq} T$. In the rest of this section we assume that $\mathcal{L}_1 \in \{S4, S5\}$ and $\mathcal{L}_2 \in \{T, B, S4, S5\}$. We shall define Ladner-like algorithms.

3.1 Preliminaries

We introduce a (very simple) closure operator for sets of bimodal formulae. Let X be a set of bimodal formulae. Let $sub(X)$ be the smallest set of formulae including X, closed under subformulae and such that if $[2]\phi \in sub(X)$, then $[1]\phi \in sub(X)$. A set X of formulae is said to be *sub-closed* $\stackrel{\text{def}}{\Leftrightarrow}$ $sub(X) = X$. Observe that for any finite set X of formulae, $md(sub(X)) = md(X)$ and for any formula ϕ, $card(sub(\{\phi\})) < 2 \times |\phi|$.

In order to determine the satisfiability of some formula ϕ, we need to handle sets of formulae. Actually all those sets shall be subsets of $sub(\{\phi\})$ and that is why $sub(\{\phi\})$ has been introduced. In establishing the **PSPACE** complexity upper bound, the fact that not only $sub(\{\phi\})$ is finite but also its cardinality is polynomial in the size of ϕ plays an important role. In the present case, the cardinality of $sub(\{\phi\})$ is even linear in $|\phi|$.

In order to check whether ϕ is $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-satisfiable, we build sequences of the form $X_0\ x_0\ X_1\ x_1\ X_2\ x_2 \ldots$ where $\phi \in X_0 \subseteq sub(\{\phi\})$ and for $i \in \omega$, X_i is a *consistent* subset of $sub(\{\phi\})$ and $x_i \in \{1, 2\}$. We extend a finite sequence $X_0\ x_0\ X_1 x_1 \ldots x_{i-1} X_i$ with $x_i X_{i+1}$ whenever we need a witness of $[x_i]\psi \notin X_i$ for some formula ψ (and $\psi \notin X_{i+1}$). The intention is to build paths in some

$\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model $\mathcal{M} = \langle W, R_1, R_2, m\rangle$ such that for $i \in \omega$, there is $w_i \in W$ such that $\mathcal{M}, w_i \models \psi$ iff $\psi \in X_i$ and $\langle w_i, w_{i+1}\rangle \in R_{x_i}$.

In order to establish termination which is a necessary step to obtain the **PSPACE** complexity upper bound, we shall define subsets $sub(s, \phi) \subseteq sub(\{\phi\})$ for $s \in \{1, 2\}^*$ such that for $i \in \omega$, $X_i \subseteq sub(x_0 \dots x_{i-1}, \phi)$. For $x_i \in \{1, 2\}$, $sub(x_0 \dots x_{i-1} \cdot x_i, \phi)$ contains all the formulae ψ which we could possibly be put in X_{i+1} for $\psi \in sub(x_0 \dots x_{i-1}, \phi)$.

We are on the good track to get termination if there is some computable map $f : \omega \to \omega$ such that for $|s| \geq f(|\phi|)$, $sub(s, \phi) = \emptyset$. To establish the **PSPACE** complexity upper bound, f should preferably be bounded by a polynomial. Those general principles may look quite attractive but in concrete examples of bimodal logics they are seldom sufficient to show that the satisfiability problem is in **PSPACE**. In $S4 \oplus_{\subseteq} S4$, since transitivity of R_2 is required, if $[2]\psi \in X_i$, then $\mathcal{M}, w_i \models [2]\psi$, $\mathcal{M}, w_i \models [2][2]\psi$ and therefore one can expect that $[2]\psi \in X_{i+1}$ if $x_i = 2$. So the formula $[2]\psi \in X_i$ should be propagated for any "2" transition. However, this does not guarantee termination. Actually, as already known from [22,31,8], duplicates can be identified in $X_0\ x_0\ X_1\ x_1\ X_2\ x_2 \dots$ which corresponds to a cycle detection (see also [13]). Since $card(\mathcal{P}(sub(\{\phi\})))$ is in $\mathcal{O}(2^{|\phi|})$, a finer analysis is necessary to establish the **PSPACE** complexity upper bound as done in [22] (see also [31] for the tense extension of Ladner's solution).

In order to conclude this introductory part that motivates the existence of the sets of the form $sub(s, \phi)$, let us say that once the set X_i of formulae is built and x_i is chosen, the set X_{i+1} of formulae satisfies the following conditions: X_{i+1} is a consistent subset of $sub(x_0 \dots x_i, \phi)$ and $\langle X_i, X_{i+1}\rangle$ satisfies a syntactic condition $\mathtt{C}_{x_i}$ that guarantees that $\mathcal{M}$ is an $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model and $\langle w_i, w_{i+1}\rangle \in R_{x_i}$.

Let ϕ be a bimodal formula. For $s \in \{1, 2\}^*$, let $sub(s, \phi)$ be the smallest set such that:

1. $sub(\lambda, \phi) = sub(\{\phi\})$; $sub(s, \phi)$ is sub-closed;
2. if $[i]\psi \in sub(s, \phi)$ for some $i \in \{1, 2\}$, then $\psi \in sub(s.i, \phi)$;
3. if $[1]\psi \in sub(s, \phi)$, then $[1]\psi \in sub(s.1, \phi)$;
4. if $\mathcal{L}_2 \in \{S4, S5\}$ and $[2]\psi \in sub(s, \phi)$, then $[2]\psi \in sub(s.2, \phi)$ and $[2]\psi \in sub(s.1, \phi)$.

Observe that for any initial substring s' of the string $s \in \{1, 2\}^*$, $sub(s, \phi) \subseteq sub(s', \phi)$; for $k \geq 1$, $sub(s{\cdot}1, \phi) = sub(s{\cdot}1^k, \phi)$ and if $\mathcal{L}_2 \in \{T, B\}$ and s contains more than $k \geq md(\phi) + 1$ occurrences of 2, then $sub(s, \phi) = \emptyset$.

Definition 1. *Let X, Y be subsets of $sub(\{\phi\})$. The binary relation $\mathtt{C}_1$ on the set $\mathcal{P}(sub(\{\phi\}))$ is defined as follows: $X\mathtt{C}_1Y \stackrel{def}{\Leftrightarrow}$*

1.1. $\mathcal{L}_2 \in \{T, B\}$: $X\mathtt{C}_2Y$ (see below);
1.2. $\mathcal{L}_2 \in \{S4, S5\}$: for all $[2]\psi \in X$, $[2]\psi \in Y$ and for all $[2]\psi \in Y$, $[2]\psi \in X$;
2.1. $\mathcal{L}_1 = S4$: for all $[1]\psi \in X$, $[1]\psi \in Y$;
2.2. $\mathcal{L}_1 = S5$: for all $[1]\psi \in X$, $[1]\psi \in Y$ and for all $[1]\psi \in Y$, $[1]\psi \in X$.

The binary relation $\mathtt{C}_2$ on $\mathcal{P}(sub(\{\phi\}))$ is defined as follows: $X\mathtt{C}_2Y \stackrel{def}{\Leftrightarrow}$

1. $\mathcal{L}_2 = T$: *for all* $[2]\psi \in X$, $\psi \in Y$;
2. $\mathcal{L}_2 = B$: *for all* $[2]\psi \in X$, $\psi \in Y$ *and for all* $[2]\psi \in Y$, $\psi \in X$;
3. $\mathcal{L}_2 = S4$: *for all* $[2]\psi \in X$, $[2]\psi \in Y$;
4. $\mathcal{L}_2 = S5$: *for all* $[2]\psi \in X$, $[2]\psi \in Y$ *and for all* $[2]\psi \in Y$, $[2]\psi \in X$.

Let $\texttt{clos}$ be the set of subsets Y of $sub(\{\phi\})$ such that for $i \in \{1,2\}$, $[i]\psi \in Y$ implies $\psi \in Y$. Observe that if $\mathcal{L}_i \in \{T, B, S4, S5\}$ [resp. $\mathcal{L}_i \in \{B, S5\}$, $\mathcal{L}_i \in \{S4, S5\}$], then $\texttt{C}_i$ restricted to $\texttt{clos}$ is reflexive [resp. $\texttt{C}_i$ is symmetric, $\texttt{C}_i$ is transitive]. The logic $\mathcal{L}_1$ is anyhow in $\{S4, S5\}$ throughout this section. The careful reader may be puzzled by the point 1.2. in the definition of $\texttt{C}_1$ when $\mathcal{L}_2 = S4$. Indeed, this seems to give an S5 flavour to [2]. However, observe that for any bimodal frame $\langle W, R_1, R_2\rangle$ such that R_1 is symmetric, R_2 is transitive and $R_1 \subseteq R_2$, for $\langle w, w'\rangle \in R_1$, we have $R_2(w) = R_2(w')$ which implies that w and w' satisfy the same set of [2]-formulae in any model based on $\langle W, R_1, R_2\rangle$. This gives us an additional reason to present the Ladner-like constructions for the logics studied in this section since this provides a rather uniform presentation.

Let X be a subset of $sub(s, \phi)$ for some $s \in \{1,2\}^*$ and for some formula ϕ. The set X is said to be *s-consistent* $\stackrel{\text{def}}{\Leftrightarrow}$ for $\psi \in sub(s, \phi)$:

1. if $\psi = \neg\varphi$, then $\varphi \in X$ iff not $\psi \in X$;
2. if $\psi = \varphi_1 \wedge \varphi_2$, then $\{\varphi_1, \varphi_2\} \subseteq X$ iff $\psi \in X$;
3. if $\psi = [i]\varphi$ for some $i \in \{1,2\}$ [resp. $\psi = [2]\varphi$] and $\psi \in X$, then $\varphi \in X$ [resp. $[1]\varphi \in X$].

Roughly speaking, the s-consistency entails the maximal propositional consistency with respect to $sub(s, \phi)$. The condition 3. above takes into account reflexivity and the inclusion $R_1 \subseteq R_2$.

Lemma 2. *Let* $\mathcal{M} = \langle W, R_1, R_2, m\rangle$ *be an* $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$*-model,* $w, w' \in W$, $s \in \{1,2\}^*$, $i, i' \in \{\lambda, 1, 2\}$ *and* ϕ *be a bimodal formula. Let* $X_w \stackrel{def}{=} \{\psi \in sub(s.i, \phi) : \mathcal{M}, w \models \psi\}$ *and* $X_{w'} \stackrel{def}{=} \{\psi \in sub(s.i', \phi) : \mathcal{M}, w' \models \psi\}$. *Then,* X_w *is* $s.i$*-consistent,* $X_{w'}$ *is* $s.i'$*-consistent and if* $\langle i, i'\rangle \in \{\langle\lambda,\lambda\rangle, \langle\lambda, j\rangle\}$ *and* $\langle w, w'\rangle \in R_j$ *for some* $j \in \{1,2\}$, *then* $X_w \texttt{C}_j X_{w'}$.

The proof of Lemma 2 is by an easy verification.

Lemma 3. *Let* X_i *be an* s_i*-consistent set and* $s_i \in \{1,2\}^*$, $i = 1, 2$. *Then,*

(I) $\mathcal{L}_1 = S4$: $X_1 \texttt{C}_1^* X_2$ *and* $[2]\psi \in X_1$ *implies* $\psi \in X_2$;
(II) $\mathcal{L}_1 = S5$: $X_1(\texttt{C}_1 \cup \texttt{C}_1^{-1})^* X_2$ *and* $[2]\psi \in X_1$ *implies* $\psi \in X_2$;
(III) $\mathcal{L}_2 = S4$: $X_1(\texttt{C}_1 \cup \texttt{C}_2)^* X_2$ *and* $[2]\psi \in X_1$ *implies* $\psi \in X_2$;
(IV) $\mathcal{L}_2 = S5$: $X_1(\texttt{C}_1 \cup \texttt{C}_1^{-1} \cup \texttt{C}_2 \cup \texttt{C}_2^{-1})^* X_2$ *and* $[2]\psi \in X_1$ *implies* $\psi \in X_2$.

3.2 The Algorithms

In Figure 1, the function $\texttt{WORLD}(\Sigma, s, \phi)$ returning a Boolean is defined. Σ is a finite non-empty list of subsets of $sub(\{\phi\})$ and $s \in \{1,2\}^*$. Moreover, for any $X \subseteq sub(\{\phi\})$ and for any call $\texttt{WORLD}(\Sigma, s, \phi)$ in $\texttt{WORLD}(X, \lambda, \phi)$ (at any recursion

`function` $\texttt{WORLD}(\Sigma, s, \phi)$

if $last(\Sigma)$ is not s-consistent, then return false;
for [1]$\psi \in sub(s,\phi) \setminus last(\Sigma)$ do
  if there is no $X \in \Sigma$ such that $\Sigma = \Sigma_1 X \Sigma_2$, s is of the form $s_1.s_2$ with $|s_2| = |\Sigma_2|$ and $s_2 \in \{1\}^*$, $\psi \notin X$, $last(\Sigma)\mathtt{C}_1 X$, then
  for each $X_\psi \subseteq sub(s.1,\phi) \setminus \{\psi\}$ such that $last(\Sigma)\mathtt{C}_1 X_\psi$, call $\texttt{WORLD}(\Sigma.X_\psi, s.1, \phi)$. If all these calls return false, then return false;
for [2]$\psi \in sub(s,\phi) \setminus last(\Sigma)$ do
  $\mathcal{L}_2 \in \{T, B\}$: for each $X_\psi \subseteq sub(s.2,\phi) \setminus \{\psi\}$ such that $last(\Sigma)\mathtt{C}_2 X_\psi$, call $\texttt{WORLD}(\Sigma.X_\psi, s.2, \phi)$. If all these calls return false, then return false;
  $\mathcal{L}_2 \in \{S4, S5\}$: if there is no $X \in \Sigma$ such that $\psi \notin X$ and $last(\Sigma)\mathtt{C}_2 X$, then
  for each $X_\psi \subseteq sub(s.2,\phi) \setminus \{\psi\}$ such that $last(\Sigma)\mathtt{C}_2 X_\psi$, call $\texttt{WORLD}(\Sigma.X_\psi, s.2, \phi)$. If all these calls return false, then return false;
Return true.

Fig. 1. Algorithm `WORLD`

depth), $last(\Sigma) \subseteq sub(s,\phi)$. The function `WORLD` is actually defined on the model of the function `K-WORLD` in [22] (see also [31,26,35]).

Most of the ingenuity to guarantee that the algorithms terminate are in the definition of $sub(s,\phi)$, s-consistency and the conditions $\mathtt{C}_i$. Indeed, $sub(s.i,\phi)$ contains the formulae that can be possibly propagated from $sub(s,\phi)$. In the easiest case, $sub(s.i,\phi) \subset sub(s,\phi)$ but this is not the general case here. Then $\mathtt{C}_i$ and s-consistency further restrict the formulae that can be propagated. Still, we may be in trouble to guarantee termination. That is why the detection of cycles is introduced (see e.g. [22]). It is precisely, the appropriate combination of all these ingredients that guarantees termination and in the best case the **PSPACE** upper bound. What we present is a uniform formalization of Ladner-like algorithms based on [31] and we believe it is the proper framework to allow further extensions.

We prove that for any set $X \subseteq sub(\{\phi\})$, $\texttt{WORLD}(X, \lambda, \phi)$ always terminates and requires polynomial space in $|\phi|$. We shall take advantage of the fact that if $\texttt{WORLD}(\Sigma, s, \phi)$ calls $\texttt{WORLD}(\Sigma', s', \phi)$ (at any recursion depth), then $|s'| > |s|$.

Each subset $X \subseteq sub(\{\phi\})$ can be represented as a bitstring of length $2 \times |\phi|$. By implementing Σ as a global stack, each level of the recursion uses space in $\mathcal{O}(|\phi|)$. For instance, in the parts of `WORLD` of the form "for each $X_\psi \subseteq sub(s.i,\phi) \setminus \{\psi\}$ such that $last(\Sigma)\mathtt{C}_i X_\psi$, call $\texttt{WORLD}(\Sigma.X_\psi, s.i, \phi)$. If all these calls return false, then return false" the implementation uses a bitstring of length $2 \times |\phi|$ to encode X_ψ (this value is incremented for each new X_ψ) and a Boolean indicating whether there is a call returning true.

Theorem 1. *Let $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{S4, S5\} \times \{T, B, S4, S5\}$ and n' be the number of occurrences of $S4$ in $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$. Let $X \subseteq sub(\{\phi\})$.*

(I) $\texttt{WORLD}(X, \lambda, \phi)$ terminates and requires at most space in $\mathcal{O}(|\phi|^{3+n'})$;

(II) Let $\texttt{WORLD}(\Sigma, s, \phi)$ *be a call in the computation of* $\texttt{WORLD}(X, \lambda, \phi)$. *Then,* $|\Sigma| \leq \alpha$ *and* $|s| \leq \alpha$ *with* $\alpha = (2 \times |\phi| + 1)^{2-n'} \times (4 \times |\phi|^2 + 1)^{n'}$.

The bounds in Theorem 1 are really rough since many optimizations can be designed. Because of lack of place, this is omitted here.

Theorem 1 is certainly an important step to prove that satisfiability is in **PSPACE** but this is not sufficient. Indeed, until now we have no guarantee that WORLD is actually correct. This shall be shown in the next two lemmas.

Lemma 4. *Let* ϕ *be a bimodal formula and* $Y \subseteq sub(\{\phi\})$ *such that* $\phi \in Y$. *If* $\texttt{WORLD}(Y, \lambda, \phi)$ *returns true, then* ϕ *is* $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$*-satisfiable.*

Proof. Let n' be the number of $S4$ in $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle$. Assume that $\texttt{WORLD}(Y, \lambda, \phi)$ returns true. Let us build an $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model $\mathcal{M} = \langle W, R_1, R_2, m \rangle$ for which there is $w \in W$ such that for all $\psi \in sub(\{\phi\})$, $\mathcal{M}, w \models \psi$ iff $\psi \in Y$.

Let S be the set of strings s in $\{1, 2\}^*$ such that $|s| \leq (2 \times |\phi| + 1)^{2-n'} \times (4 \times |\phi|^2 + 1)^{n'}$. We define W as the set of pairs $\langle X, s \rangle \in \texttt{clos} \times S$ for which there is a finite sequence $\langle \Sigma_1, s_1 \rangle, \ldots, \langle \Sigma_k, s_k \rangle$ $(k \geq 1)$ such that

1. $\Sigma_1 = Y$; $s_1 = \lambda$; $last(\Sigma_k) = X$; $s_k = s$;
2. for $i \in \{1, \ldots, k\}$, $\texttt{WORLD}(\Sigma_i, s_i, \phi)$ returns true;
3. for $i \in \{1, \ldots, k-1\}$, $\texttt{WORLD}(\Sigma_i, s_i, \phi)$ calls directly $\texttt{WORLD}(\Sigma_{i+1}, s_{i+1}, \phi)$.

The conditions 2. and 3. state that we only record the pairs $\langle X, s \rangle \in \texttt{clos} \times S$ that contribute to make $\texttt{WORLD}(Y, \lambda, \phi)$ true. $\langle Y, \lambda \rangle \in W$ by definition. Furthermore, for all $\langle X, s \rangle \in W$, $X \subseteq sub(s, \phi)$ and X is s-consistent.

Let us define the auxiliary binary relations R'_1 [resp. R'_2] on W as follows: $\langle X, s \rangle R'_1 \langle X', s' \rangle$ [resp. $\langle X, s \rangle R'_2 \langle X', s' \rangle$] $\stackrel{\text{def}}{\Leftrightarrow}$ there is a call $\texttt{WORLD}(\Sigma, s, \phi)$ in $\texttt{WORLD}(Y, \lambda, \phi)$ (at any depth of the recursion) such that

1. either
   a) $last(\Sigma) = X$;
   b) $\texttt{WORLD}(\Sigma, s, \phi)$ calls $\texttt{WORLD}(\Sigma', s', \phi)$ in the "1" [resp. "2"] segment of $\texttt{WORLD}(\Sigma, s, \phi)$; $last(\Sigma') = X'$;
2. or there is a finite sequence $\langle \Sigma_1, s_1 \rangle, \ldots, \langle \Sigma_k, s_k \rangle$ such that:
   a) $last(\Sigma_k) = X$; $last(\Sigma_1) = X'$; $\Sigma_k = \Sigma$; $s_k = s$; $s_1 = s'$;
   b) for $i \in \{1, \ldots, k\}$, $\langle last(\Sigma_i), s_i \rangle \in W$;
   c) for $i \in \{1, \ldots, k-1\}$, $\texttt{WORLD}(\Sigma_i, s_i, \phi)$ calls $\texttt{WORLD}(\Sigma_{i+1}, s_{i+1}, \phi)$ in the "1" [resp. in either the "1" or the "2"] segment of WORLD;
   d) the call $\texttt{WORLD}(\Sigma_k, s_k, \phi)$ enters in the "1" [resp. "2"] segment of WORLD and for some formula $[1]\psi \in sub(s, \phi) \setminus X$ [resp. $[2]\psi \in sub(s, \phi) \setminus X$], no recursive call to WORLD is necessary thanks to Σ_1, $\psi \notin X'$, $X \texttt{C}_1 X'$ [resp. $X \texttt{C}_2 X'$].

If $\mathcal{L}_2 \in \{T, B\}$, the second possibility in the definition of R'_2 above should not be taken into account. The definition of $\mathcal{M}$ can be now completed:

– $\mathcal{L}_1 = S4$: $R_1 \stackrel{\text{def}}{=} (R'_1)^*$; $\mathcal{L}_1 = S5$: $R_1 \stackrel{\text{def}}{=} (R'_1 \cup R_1'^{-1})^*$;

– $\mathcal{L}_2 = T$: $R_2 \stackrel{\text{def}}{=} R_1 \cup R_2'$; $\mathcal{L}_2 = B$: $R_2 \stackrel{\text{def}}{=} R_1 \cup R_2' \cup (R_1)^{-1} \cup (R_2')^{-1}$;
– $\mathcal{L}_2 = S4$: $R_2 \stackrel{\text{def}}{=} (R_1 \cup R_2')^*$; $\mathcal{L}_2 = S5$: $R_2 \stackrel{\text{def}}{=} (R_1 \cup R_2' \cup (R_1)^{-1} \cup (R_2')^{-1})^*$;
– for $\mathtt{p} \in \mathtt{For}_0$, $m(\mathtt{p}) \stackrel{\text{def}}{=} \{\langle X, s\rangle \in W : \mathtt{p} \in X\}$.

$\mathcal{M}$ is an $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model. One can show (i) $\langle X, s\rangle R_1' \langle X', s'\rangle$ implies $X \mathtt{C}_1 X'$ and (ii) $\langle X, s\rangle R_2' \langle X', s'\rangle$ implies $X \mathtt{C}_2 X'$. So, (iii) for $j \in \{1, 2\}$, $\langle X, s\rangle R_j \langle X', s'\rangle$ implies for all $[j]\psi \in X$, $\psi \in X'$ (by Lemma 3). By induction on the structure of ψ we show that for all $\langle X, s\rangle \in W$, for all $\psi \in sub(s, \phi)$, $\psi \in X$ iff $\mathcal{M}, \langle X, s\rangle \models \psi$. The case when ψ is a propositional variable is by definition of m.
Induction Hypothesis: for all $\psi \in sub(\phi)$ such that $|\psi| \leq n$, for all $\langle X, s\rangle \in W$, if $\psi \in sub(s, \phi)$, then $\psi \in X$ iff $\mathcal{M}, \langle X, s\rangle \models \psi$.
Let ψ be a formula in $sub(\phi)$ such that $|\psi| \leq n+1$. The cases when the outermost connective of ψ is Boolean is a consequence of the s-consistency of X and the induction hypothesis. Let us treat the other cases.
Case 1: $\psi = [1]\psi'$. Let $\langle X, s\rangle \in W$ such that $\psi \in sub(s, \phi)$. By definition of W, there is Σ such that $last(\Sigma) = X$ and $\mathtt{WORLD}(\Sigma, s, \phi)$ returns true. If $\psi \in X$, then by (iii), for all $\langle X', s'\rangle \in R_1(\langle X, s\rangle)$, $\psi' \in X'$. One can show that $\psi' \in sub(s', \phi)$. By the induction hypothesis, $\mathcal{M}, \langle X', s'\rangle \models \psi'$ and therefore $\mathcal{M}, \langle X, s\rangle \models \psi$. Now, if $\psi \notin X$, two cases are distinguished.
Case 1.1: there is X' in Σ such that $X \mathtt{C}_1 X'$, $\psi' \notin X'$ and $\Sigma = \Sigma' X' \Sigma_2$, s is of the form $s'.s_2$ with $|\Sigma_2| = |s_2|$ and $s_2 \in \{1\}^*$. By definition of W, $\mathtt{WORLD}(\Sigma'.X', s', \phi)$ returns true (see the conditions 2. and 3. defining W). Hence, $\langle X, s\rangle R_1' \langle X', s'\rangle$ by definition and therefore $\langle X, s\rangle R_1 \langle X', s'\rangle$. One can show that $\psi' \in sub(s', \phi)$ since s is of the form $s'.1^k$ for some $k \geq 0$. By induction hypothesis, $\mathcal{M}, \langle X', s'\rangle \not\models \psi'$ and therefore $\mathcal{M}, \langle X, s\rangle \not\models \psi$.
Case 1.2: $\mathtt{WORLD}(\Sigma, s, \phi)$ calls successfully $\mathtt{WORLD}(\Sigma', s', \phi)$ in the "1" segment of $\mathtt{WORLD}$, $last(\Sigma') = X'$ and $\psi' \notin last(\Sigma')$, $X \mathtt{C}_1 X'$, and $X' \subseteq sub(s', \phi)$. Moreover, we have $s' = s.1$. This is so since $\mathtt{WORLD}(\Sigma, s, \phi)$ returns true. By definition of R_1', $\langle X, s\rangle R_1' \langle X', s'\rangle$. Furthermore, one can easily show that $\psi' \in sub(s', \phi)$. By the induction hypothesis, $\mathcal{M}, \langle X', s'\rangle \not\models \psi'$ and therefore $\mathcal{M}, \langle X, s\rangle \not\models \psi$.
Case 2: $\psi = [2]\psi'$. This is analogous to the Case 1.

As a conclusion, since $\phi \in Y$ and $\mathtt{WORLD}(Y, \lambda, \phi)$ returns true, $\mathcal{M}, \langle Y, \lambda\rangle \models \phi$ and therefore ϕ is $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-satisfiable.

The proof of Lemma 4 can be viewed as a way to transform a successful call $\mathtt{WORLD}(Y, \lambda, \phi)$ into a *quasi* $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model by analyzing the computation tree of $\mathtt{WORLD}(Y, \lambda, \phi)$. Then, this *quasi* $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model is appropriately completed in order to get an $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-model. The idea to construct a (standard) model from different *coherent* pieces is very common to establish decidability and complexity results for modal logics (see e.g. [22,28,5,23]). Mosaics technique uses such an approach (see e.g. [23]).

Lemma 5. *Let ϕ be a bimodal formula. If ϕ is $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$-satisfiable, then there is $Y \subseteq sub(\{\phi\})$ such that $\phi \in Y$ and $\mathtt{WORLD}(Y, \lambda, \phi)$ returns true.*

Since $\mathtt{WORLD}$ is correct, the proof of Lemma 4 provides the finite model property for $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ and an exponential bound for the size of the models exists.

Theorem 2. *For* $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{S4, S5\} \times \{T, B, S4, S5\}$, $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ *is in* **PSPACE**.

For $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{S4, S5\} \times \{T, B, S4\} \cup \langle S4, S5 \rangle$, $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ is **PSPACE**-hard. So, in particular $SAT(S5 \oplus_{\subseteq} S4)$ is **PSPACE**-complete and the logic S4+5 introduced in [33] has consequently a **PSPACE**-complete satisfiability problem. Until now, we have not yet established that $SAT(S5 \oplus_{\subseteq} S5)$ is **PSPACE**-hard as $SAT(S5 \oplus S5)$ [18]. It is unlikely since by [11, Proposition 4.8] $SAT(S5 \oplus_{\subseteq} S5)$ is **NP**-complete.

4 Reduction into a PSPACE Guarded Fragment

By FO2 we mean the fragment of first-order logic without equality or function symbols using only two variables. In this section we show that for $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B\}$, $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ can be linearly translated into a known **PSPACE** fragment of FO2, say WLGF2 standing for weak loosely guarded fragment with two variables. The vocabulary of WLGF2 consists of: the symbols $\neg$, $\wedge$, $\Rightarrow$, $\forall$ for propositional connectives and universal quantification; a countable set $\{\mathtt{P}_i : i \in \omega\}$ of unary predicate symbols; a set $\{\mathtt{R}_1, \mathtt{R}_2\}$ of binary predicate symbols and a set $\{\mathtt{x}_0, \mathtt{x}_1\}$ of individual variables. The set of WLGF2-formulae is the smallest set containing the set of atomic formulae built over this vocabulary, closed under the standard rules for Boolean connectives and under the rule below: if $\phi(\mathtt{x}_i)$ and $\psi(\mathtt{x}_i, \mathtt{x}_{1-i})$ are WLGF2-formulae for some $i \in \{0, 1\}$ such that,

- the only variable free in $\phi(\mathtt{x}_i)$ is $\mathtt{x}_i$;
- $\psi(\mathtt{x}_i, \mathtt{x}_{1-i})$ is a conjunction of atomic formulae of the form $\mathtt{R}(\mathtt{x}, \mathtt{y})$ such that for at least one conjunct $\{\mathtt{x}, \mathtt{y}\} = \{\mathtt{x}_0, \mathtt{x}_1\}$;

then $\forall \mathtt{x}_i \ (\psi(\mathtt{x}_i, \mathtt{x}_{1-i}) \Rightarrow \phi(\mathtt{x}_i))$ is a WLGF2-formula. WLGF2 is a fragment of the loosely guarded fragment LGF (see e.g. [4]). Actually WLGF2 is even a fragment of $\mathcal{PGF}_2$ defined in [23] and shown to be in **PSPACE** [23]. None of the obvious FO2-formulae capturing reflexivity, symmetry and inclusion are WLGF2-formulae. Instead of using such axioms, we introduce $\mathcal{PGF}_2$-modalities in the sense of [23, Section 4.1.1].

For $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B\}$, we define a map $T_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}$ such that $T_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\phi)$ is of the form $init_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2} \wedge ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\phi, \mathtt{x}_0)$ where $init_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}$ is a fixed WLGF2-formula. Analogously to the standard translation ST [3], $ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}$ encodes the quantification in the interpretation of $[i]$ into the language of WLGF2. We allow ourselves only a restricted form of universal quantification that encodes appropriately the properties of the bimodal frames. The main idea of $ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}$ is to visit only the successor worlds that satisfy the *local constraints* on the relations of the frames. Indeed, reflexivity, symmetry and inclusion can be checked *locally*. $ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}$ is defined inductively as follows ($i \in \{0, 1\}$);

- $ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\mathtt{p}_j, \mathtt{x}_i) \stackrel{\text{def}}{=} \mathtt{P}_j(\mathtt{x}_i)$;
- $ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\phi_1 \wedge \phi_2, \mathtt{x}_i) \stackrel{\text{def}}{=} ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\phi_1, \mathtt{x}_i) \wedge ST_{\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2}(\phi_2, \mathtt{x}_i)$;

– $ST_{\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2}([j]\phi, \mathtt{x}_i) \stackrel{\text{def}}{=} \forall\, \mathtt{x}_{1-i}((\bigwedge_{k\in\{j,2\}} \mathtt{R}_k(\mathtt{x}_i, \mathtt{x}_{1-i}) \wedge \varphi^k_{\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2}(\mathtt{x}_i, \mathtt{x}_{1-i})) \Rightarrow ST_{\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2}(\phi, \mathtt{x}_{1-i}))$ for $j \in \{1,2\}$ where the $\varphi^k_{\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2}(\mathtt{x}_i, \mathtt{x}_{1-i})$s are defined in the table below.

$\mathcal{L}_1$	$\mathcal{L}_2$	$\varphi^1_{\mathcal{L}_1 \oplus_\subset \mathcal{L}_2}(\mathtt{x}_i, \mathtt{x}_{1-i})$	$\varphi^2_{\mathcal{L}_1 \oplus_\subset \mathcal{L}_2}(\mathtt{x}_i, \mathtt{x}_{1-i})$
K	K	$\top$	$\top$
K	T	$\top$	$\mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
K	B	$\top$	$\mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_i) \wedge \mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
T	K	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
T	T	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
T	B	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_i)$
B	K	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_i) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
B	T	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_i) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$
B	B	$\mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_i) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i})$	$\mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_1(\mathtt{x}_{1-i}, \mathtt{x}_{1-i}) \wedge \mathtt{R}_2(\mathtt{x}_{1-i}, \mathtt{x}_i)$

Let us define the initial formulae: $init_{K\oplus_\subseteq K} \stackrel{\text{def}}{=} \top$; $init_{T\oplus_\subseteq K} \stackrel{\text{def}}{=} init_{B\oplus_\subseteq K} \stackrel{\text{def}}{=} \mathtt{R}_1(\mathtt{x}_0, \mathtt{x}_0)$; $init_{K\oplus_\subseteq T} \stackrel{\text{def}}{=} init_{K\oplus_\subseteq B} \stackrel{\text{def}}{=} \mathtt{R}_2(\mathtt{x}_0, \mathtt{x}_0)$; for $\mathcal{L}_1, \mathcal{L}_2 \in \{T, B\}$, $init_{\mathcal{L}_1\oplus_\subseteq \mathcal{L}_2} \stackrel{\text{def}}{=} \mathtt{R}_1(\mathtt{x}_0, \mathtt{x}_0) \wedge \mathtt{R}_2(\mathtt{x}_0, \mathtt{x}_0)$.

Lemma 6. *For $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B\}$, for any formula ϕ, $\phi \in SAT(\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2)$ iff $init_{\mathcal{L}_1\oplus_\subseteq \mathcal{L}_2} \wedge ST_{\mathcal{L}_1\oplus_\subseteq \mathcal{L}_2}(\phi, \mathtt{x}_0)$ is WLGF2-satisfiable.*

Since $T_{\mathcal{L}_1\oplus_\subseteq \mathcal{L}_2}$ is in linear-time,

Theorem 3. *For $\mathcal{L}_1, \mathcal{L}_2$ in $\{K, T, B\}$, $SAT(\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2)$ is in* **PSPACE**.

5 EXPTIME-Complete Bimodal Logics

It remains to characterize the complexity of $SAT(\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2)$ for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{K, T, B\} \times \{S4, S5\}$. By using logarithmic space transformations into converse-PDL (that is known to be in **EXPTIME**), for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{K, T, B\} \times \{S4, S5\}$, $SAT(\mathcal{L}_1 \oplus_\subseteq \mathcal{L}_2)$ can be shown to be in **EXPTIME**. Let $\mathcal{C}$ be a class of monomodal frames. We write $GSAT(\mathcal{C})$ to denote the set of monomodal formulae ϕ such that there is a $\mathcal{C}$-model $\mathcal{M} = \langle W, R_1, m \rangle$ satisfying for all $w \in W$, $\mathcal{M}, w \models \phi$.

Lemma 7. *Let $\mathcal{C}$, $\mathcal{C}'$ be classes of monomodal frames such that $\mathcal{C}_{S5} \subseteq \mathcal{C} \subseteq \mathcal{C}_{K4}$ and $\mathcal{C}'$ is closed under generated subframes, disjoint unions and isomorphic copies. Then, for any monomodal formula ϕ, $\phi \in GSAT(\mathcal{C}')$ iff $[2]\phi \wedge \phi \in SAT(\mathcal{C}' \oplus_\subseteq \mathcal{C})$.*

Proof. The idea of the proof has its origin in the proof of [32, Proposition 7] where it is shown that the respective *global* satisfiability problems for S4 and S5 are identical, that is $GSAT(S4) = GSAT(S5)$. One can show that since $\mathcal{C}'$ is closed under generated subframes, disjoint union and isomorphic copies, $SAT(\mathcal{C}' \oplus_\subseteq \mathcal{C}_{S5}) = SAT(\mathcal{C}' \oplus_\subseteq \mathcal{C}'_{S5})$. So in $\mathcal{C}' \oplus_\subseteq \mathcal{C}_{S5}$, the modal connective [2] behaves as a universal modal connective.

Let ϕ be a monomodal formula. Assume that $\phi \in GSAT(\mathcal{C}')$. So, there is an $\mathcal{C}' \oplus_\subseteq \mathcal{C}_{S5}$-model $\mathcal{M} = \langle W, R_1, R_2, m \rangle$ and $w \in W$ such that $\mathcal{M}, w \models [2]\phi$.

Since R_2 is reflexive, $\mathcal{M}, w \models [2]\phi \wedge \phi$. By hypothesis, $\mathcal{C}' \oplus_{\subseteq} \mathcal{C}_{S5} \subseteq \mathcal{C}' \oplus_{\subseteq} \mathcal{C}$, so $[2]\phi \wedge \phi \in SAT(\mathcal{C}' \oplus_{\subseteq} \mathcal{C})$.

Now assume that $[2]\phi \wedge \phi \in SAT(\mathcal{C}' \oplus_{\subseteq} \mathcal{C})$. So, there is a $\mathcal{C}' \oplus_{\subseteq} \mathcal{C}$-model $\mathcal{M} = \langle W, R_1, R_2, m\rangle$ such that for some $w \in W$, $\mathcal{M}, w \models [2]\phi \wedge \phi$. Since R_2 is transitive, for $w' \in R_2^*(w)$, $\mathcal{M}, w' \models \phi$. In particular, for $w' \in R_1^*(w)$, $\mathcal{M}, w' \models \phi$. Let $\mathcal{M} = \langle W', R_1', R_2', m'\rangle$ be the $\mathcal{C}' \oplus_{\subseteq} \mathcal{C}_{S5}$-model such that $W' = R_1^*(w)$ and, R_1' and m' are the respective restrictions of R_1 and m to W' and $R_2' = W' \times W'$. Since $\mathcal{C}'$ is closed under generated subframes, $\langle W', R_1'\rangle \in \mathcal{C}'$. So, $\phi \in GSAT(\mathcal{C}')$.

Many examples of classes of frames between $\mathcal{C}_{K4}$ and $\mathcal{C}_{S5}$ can be found in [17, Figure 4].

Theorem 4. *Let $\mathcal{L}_1, \mathcal{L}_2$ be monomodal logics such that $K \subseteq \mathcal{L}_1 \subseteq B$, $K4 \subseteq \mathcal{L}_2 \subseteq S5$ and for $i \in \{1, 2\}$, $\mathcal{L}_i$ is complete with respect to $\mathcal{C}_{\mathcal{L}_i}$. Then, $SAT(\mathcal{C}_{\mathcal{L}_1} \oplus_{\subseteq} \mathcal{C}_{\mathcal{L}_2})$ is* **EXPTIME***-hard.*

Proof. By [10, Theorem 1] (see also [30]), $GSAT(\mathcal{C}_{\mathcal{L}_1})$ is **EXPTIME**-hard. By Lemma 7, $SAT(\mathcal{C}_{\mathcal{L}_1} \oplus_{\subseteq} \mathcal{C}_{\mathcal{L}_2})$ is **EXPTIME**-hard ($\mathcal{C}_{\mathcal{L}_1}$ is closed under generated subframes, disjoint unions and isomorphic copies).

Hence, for $\langle \mathcal{L}_1, \mathcal{L}_2\rangle \in \{K, T, B\} \times \{S4, S5\}$, $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ is **EXPTIME**-hard. Since $K \oplus_{\subseteq} S4$ is a fragment of the logic $\mathcal{A}$ introduced in [7], $\mathcal{A}$-satisfiability is **EXPTIME**-hard. $\mathcal{A}$-satisfiability can be also translated in logarithmic space into PDL (see also [6]).

6 Concluding Remarks

We have characterized the computational complexity of simple dependent bimodal logics. Table 1 summarizes the main results. As a side-effect, we have established that S4+5 [33] is **PSPACE**-complete whereas the logic $\mathcal{A}$ in [7] is **EXPTIME**-complete. Unlike the fusion operator $\oplus$, the situation with $\oplus_{\subseteq}$ is not uniform since **NP**-complete, **PSPACE**-complete and **EXPTIME**-complete dependent bimodal logics have been found. The only case of **NP**-complete logic is $S5 \oplus_{\subseteq} S5$ and we conjecture that this can be generalized to extensions of S4.3.

The most interesting proofs are related to **PSPACE** upper bounds. We used two proof techniques. The first one consists in translation $SAT(\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2)$ for $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B\}$ into satisfiability for a fragment of M. Marx's packed guarded fragment with only two individual variables $\mathcal{PGF}_2$. This approach has obvious limitations as soon as transitive relations are involved.

The second technique consists in defining Ladner-like decision procedures for $\langle \mathcal{L}_1, \mathcal{L}_2\rangle \in \{S4, S5\} \times \{T, B, S4, S5\}$ extending Ladner technique following [31] and we have presented a uniform framework that can be easily reused to study other polymodal logics. Indeed, it is the appropriate definitions of the sets $sub(s, \phi)$, the notion of s-consistency, the conditions $\mathtt{C}_i$ and possibly the mechanism of cycle detection that allows to obtain the **PSPACE** upper bounds. This technique can be also used for $\mathcal{L}_1, \mathcal{L}_2 \in \{K, T, B\}$. We took the decision to

use $\mathcal{PGF}_2$ instead since it is an interesting fragment to equip with an analytic tableau-style calculus (see in [25] a labelled tableaux calculus for the modal logic MLR_2 of binary relations that corresponds roughly to WLGF2 augmented with other binary predicate symbols). Last but not least, the decision procedures we have defined could be straightforwardly (and more efficiently) reused in a tableaux calculus for $\langle \mathcal{L}_1, \mathcal{L}_2 \rangle \in \{S4, S5\} \times \{T, B, S4, S5\}$. For instance, we can show that in a prefixed calculus one not need to consider prefixes of length greater than $(4 \times |\phi|^2 + 1)^2$. In a non-prefixed version, one does not need to apply the "π-rule" more than $(4 \times |\phi|^2 + 1)^2$ times on a branch to show that ϕ is valid. An analysis similar to the one in [8] about results in [22,14] would be the right way to formally establish such results.

Besides, it is natural to extend the operator $\oplus_{\subseteq}$ to an n-ary operator $n \geq 2$. Let $(\mathcal{C}_i)_{i \in \{1,\ldots,n\}}$ be $n \geq 2$ classes of monomodal frames. The class $\mathcal{C}_1 \oplus_{\subseteq} \ldots \oplus_{\subseteq} \mathcal{C}_n$ of n-modal frames is defined as the class of frames $\langle W, R_1, \ldots, R_n \rangle$ such that for $i \in \{1, \ldots, n\}$, $\langle W, R_i \rangle \in \mathcal{C}_i$ and $R_1 \subseteq \ldots \subseteq R_n$. All the other notions can be naturally defined. One can show the following generalization:

Theorem 5. *Let $\mathcal{L}_1, \ldots, \mathcal{L}_n$ be in $\{K, T, B, S4, S5\}$, $n \geq 2$. If there exist $i < j \in \{1, \ldots, n\}$ such that $SAT(\mathcal{L}_i \oplus_{\subseteq} \mathcal{L}_j)$ is* **EXPTIME***-hard then $SAT(\mathcal{L}_1 \oplus_{\subseteq} \ldots \oplus_{\subseteq} \mathcal{L}_n)$ is* **EXPTIME***-complete. Otherwise, if for $i \in \{1, \ldots, n\}$, $\mathcal{L}_i = S5$, then $SAT(\mathcal{L}_1 \oplus_{\subseteq} \ldots \oplus_{\subseteq} \mathcal{L}_n)$ is* **NP***-complete otherwise $SAT(\mathcal{L}_1 \oplus_{\subseteq} \ldots \oplus_{\subseteq} \mathcal{L}_n)$ is* **PSPACE***-complete.*

Acknowledgments. The author thanks the anonymous referees for useful remarks and suggestions. Special thanks are due to one of the referees for finding a mistake in the submitted version.

References

1. M. Baldoni. *Normal Multimodal Logics: Automated Deduction and Logic Programming.* PhD thesis, Università degli Studi di Torino, 1998.
2. B. Beckert and D. Gabbay. Fibring semantic tableaux. In H. de Swart, editor, *TABLEAUX-8*, pages 77–92. LNAI, Springer-Verlag, 1998.
3. J. van Benthem. *Modal logic and classical logic.* Bibliopolis, 1985.
4. J. van Benthem. Modal logic in two Gestalts. Technical report, ILLC, 1998.
5. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic.* 2000. to appear.
6. M. Castilho, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic I: the frame problem. *J. of Logic and Computation*, 1998. to appear.
7. M. Castilho and A. Herzig. An alternative to the iteration operator of propositional dynamic logic. Technical Report 96-05-R, IRIT, 1996.
8. S. Cerrito and M. Cialdea Mayer. A polynomial translation of S4 into T and contraction-free tableaux for S4. *Logic Journal of the IGPL*, 5(2):287–300, 1997.
9. A. Chagrov and M. Zakharyaschev. *Modal Logic.* Clarendon Press, Oxford, 1997.
10. C. Chen and I. Lin. The complexity of propositional modal theories and the complexity of consistency of propositional modal theories. In A. Nerode and Yu. V. Matiyasevich, editors, *LFCS-3, St. Petersburg*, pages 69–80. Springer-Verlag, LNCS 813, 1994.
11. S. Demri. A class of decidable information logics. *TCS*, 195(1):33–60, 1998.

12. K. Fine and G. Schurz. Transfer theorems for multimodal logics. In *Logic and Reality. Essays in Pure and Applied Logic. In Memory of Arthur Prior*. OUP, 1992. To appear.
13. M. Fitting. *Proof methods for modal and intuitionistic logics*. D. Reidel Publishing Co., 1983.
14. M. Fitting. First-order modal tableaux. *JAR*, 4:191–213, 1988.
15. D. Gabbay. Fibred semantics and the weaving of logics Part 1: modal and intuitionistic logics. *The Journal of Symbolic Logic*, 61(4):1057–1120, 1996.
16. D. Gabbay and V. Shehtman. Products of modal logics, Part I. *Logic Journal of the IGPL*, 6(1):73–146, 1998.
17. R Goré. Tableaux methods for modal and temporal logics. In M. d'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableaux Methods*, pages 297–396. Kluwer, 1999.
18. J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
19. E. Hemaspaandra. Complexity transfer for modal logic (extended abstract). In *LICS-9*, pages 164–173, 1994.
20. A. Heuerding. *Sequent Calculi for Proof Search in Some Modal Logics*. PhD thesis, University of Bern, 1998.
21. M. Kracht and F. Wolter. Simulation and transfer results in modal logic - A survey. *Studia Logica*, 59:149–1997, 1997.
22. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.
23. M. Marx. Complexity of modal logics of relations. Technical report, ILLC, 1997. To appear in Annals of Pure and Applied Logic.
24. M. Marx. Complexity of products of modal logics. *Journal of Logic and Computation*, 9(2):197–214, 1999.
25. M. Marx, S. Mikulás, and S. Schlobach. Tableau calculus for local cubic modal logic and its implementation. *Logic Journal of the IGPL*, 7(6):755–778, 1999.
26. F. Massacci. Single steps tableaux for modal logics. *JAR*, 1999. to appear.
27. Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.
28. V. Pratt. Models of program logics. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 115–122, 1979.
29. A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *JACM*, 32(3):733–749, 1985.
30. E. Spaan. *Complexity of Modal Logics*. PhD thesis, ILLC, 1993.
31. E. Spaan. The complexity of propositional tense logics. In M. de Rijke, editor, *Diamonds and Defaults*, pages 287–309. Kluwer, 1993.
32. M. Tiomkin and M. Kaminsky. Nonmonotonic default modal logics. *JACM*, 38:963–984, 1991.
33. D. Vakarelov. Modal characterization of the classes of finite and infinite quasi-ordered sets. In P. Petkov, editor, *Mathematical Logic*, pages 373–387. Plenum Press, New-York, 1990.
34. D. Vakarelov. Modal logics for knowledge representation systems. *TCS*, 90:433–456, 1991.
35. L. Viganò. *Labelled Non-Classical Logics*. Kluwer, 2000.
36. A. Visser. A course on bimodal provability logic. *Annals of Pure and Applied Logic*, 73:109–142, 1995.

Properties of Embeddings from Int to S4

Uwe Egly

Institut für Informationssysteme 184/3, TU Wien
Favoritenstr. 9–11, A–1040 Wien, Austria
email: uwe@kr.tuwien.ac.at

Abstract. In this paper, we compare the effect of different embeddings of the intuitionistic logic Int in the modal logic S4. We select two different embeddings from the literature and show that translated (propositional) intuitionistic formulae have sometimes exponentially shorter minimal proofs in a cut-free Gentzen system for S4 than the original formula in a cut-free standard Gentzen system for Int. A similar relation on minimal proof length is shown for two variants of multi-succedent cut-free Gentzen calculi for Int.

1 Introduction

The translation of formulae from one logical representation into another is a common practice in many fields of logic and automated deduction. To mention just one example, consider the translation of (the negation of) a propositional formula into a satisfiability-equivalent 3-CNF (conjunctive normal form where each clause has at most three literals). Besides this example, where the latter representation is a restricted form of the former one, more general translations are possible. In this paper, we consider translations (embeddings) of formulae from intuitionistic logic Int into modal logic S4 and investigate the effect of different translations on the length of the resulting (minimal) proofs.

The first embedding of this kind was introduced by Gödel [8] in 1933;[1] many others were introduced in the forties and fifties [10,11,13]. Historically, one goal was to characterize intuitionistic truth in terms of classical provability. More recent considerations focus, for instance, on the generation of calculi for Int which are well-suited for automated deduction (see, e.g., [16] where the connection calculus for Int is developed with the corresponding calculus for S4 in mind).

The paper is organized as follows. In the next section, we introduce the cut-free sequent calculi for Int and S4 together with some notation. In Section 3, we consider two embeddings taken from the literature. The first one, $(\cdot)^\circ$, is an adaption of Girard's embedding of Int into classical linear logic; the second one, $(\cdot)^\Box$, is a variant of an embedding of Rasiowa and Sikorski, which in turn is a first-order generalization of the embedding T of McKinsey and Tarski. Both embeddings are *faithful*, i.e., the original formula is provable in Int iff the translated

[1] Artemov mentioned in [1] that there was a similar embedding of Orlov [12] into a weaker system than S4.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 205–219, 2000.

$$F, \Gamma \Longrightarrow \Delta, F \ \text{Ax} \qquad \bot, \Gamma \Longrightarrow \Delta \ \bot l$$

$$\frac{A, B, \Gamma \Longrightarrow \Delta}{A \wedge B, \Gamma \Longrightarrow \Delta} \wedge l \qquad \frac{\Gamma \Longrightarrow \Delta, A \qquad \Gamma \Longrightarrow \Delta, B}{\Gamma \Longrightarrow \Delta, A \wedge B} \wedge r$$

$$\frac{A, \Gamma \Longrightarrow \Delta \qquad B, \Gamma \Longrightarrow \Delta}{A \vee B, \Gamma \Longrightarrow \Delta} \vee l \qquad \frac{\Gamma \Longrightarrow \Delta, A, B}{\Gamma \Longrightarrow \Delta, A \vee B} \vee r$$

$$\frac{\Gamma \Longrightarrow \Delta, A \qquad B, \Gamma \Longrightarrow \Delta}{A \to B, \Gamma \Longrightarrow \Delta} \to l \qquad \frac{A, \Gamma \Longrightarrow \Delta, B}{\Gamma \Longrightarrow \Delta, A \to B} \to r$$

$$\frac{\forall x\, A, A^x_t, \Gamma \Longrightarrow \Delta}{\forall x\, A, \Gamma \Longrightarrow \Delta} \forall l \qquad \frac{\Gamma \Longrightarrow \Delta, A^x_y}{\Gamma \Longrightarrow \Delta, \forall x\, A} \forall r$$

$$\frac{A^x_y, \Gamma \Longrightarrow \Delta}{\exists x\, A, \Gamma \Longrightarrow \Delta} \exists l \qquad \frac{\Gamma \Longrightarrow \Delta, A^x_t, \exists x\, A}{\Gamma \Longrightarrow \Delta, \exists x\, A} \exists r$$

Fig. 1. Axioms and inference rules of G3c.

formula is provable in S4. In Section 4, we provide hard formulae for cut-free standard Gentzen systems for the propositional fragment of Int. The effect of the two embeddings on the length of minimal proofs is investigated. It turns out that embeddings are sometimes beneficial yielding an exponential decrease of minimal proof length. Additionally, a comparison is performed between different multi-succedent variants of cut-free Gentzen systems for the propositional fragment of Int.

2 Preliminaries

Throughout this paper we use a first-order language consisting of *variables, function symbols, predicate symbols, logical connectives, falsum* ($\bot$)*, quantifiers, modalities* $\Box$ *and* $\Diamond$*, and punctuation symbols. Terms* and *formulae* are defined according to the usual formation rules. Negation is a defined concept, i.e., $\neg F$ is defined to be $F \to \bot$. We will identify 0-ary predicate symbols with *propositional atoms*, and 0-ary function symbols with (*object*) *constants*.

We use Gentzen systems (sequent systems) in which weakening and contraction are absorbed into axioms and rules (such systems are often called G3-systems; see, e.g., [9] or [15]). These systems are closely related to tableau systems. Although our results require the propositional fragment only, we introduce the calculi and embeddings for first-order logic.

The formal objects of sequent systems are *sequents* $S := \Gamma \Longrightarrow \Sigma$, where Γ, Σ are finite sets of first-order formulae. (Γ is the *antecedent* of S, and Σ is the *succedent* of S.) Recall that the informal meaning of a sequent

$$A_1, \ldots, A_n \Longrightarrow B_1, \ldots, B_m$$

is the same as the informal meaning of the formula $(\bigwedge_{i=1}^{n} A_i) \to (\bigvee_{i=1}^{m} B_i)$. A sequent system for classical first-order logic, called G3c, is depicted in Fig. 1. A sequent S is called *intuitionistic* if the succedent of S consists of at most one formula. A sequent calculus for intuitionistic first-order logic, called G3i, is depicted in Fig. 2. G3m is the corresponding sequent calculus for minimal logic (i.e., intuitinistic logic *without* $\bot l$). In G3i and G3m, the formal objects are intuitionistic sequents, i.e., in Fig. 2, Δ consists of at most one formula.

In all the mentioned calculi, the *eigenvariable condition* applies, i.e., in $\forall r$ and $\exists l$, the *eigenvariable* y does not occur (free) in $\forall x A$, $\exists x A$, Γ, and Δ, respectively. A multi-succedent intuitionistic calculus G3i' is obtained from G3c by replacing $\to r$ and $\forall r$ by the following *critical* rules.

$$\frac{A,\, \Gamma \implies B}{\Gamma \implies \Delta,\, A \to B}\ {\to}r \qquad \frac{\Gamma \implies A^x_y}{\Gamma \implies \Delta,\, \forall x\, A}\ \forall r$$

Additionally, the $\to l$-inference of G3c in Fig. 1 is replaced by the inference

$$\frac{A \to B, \Gamma \implies \Delta, A \quad B, \Gamma \implies \Delta}{A \to B, \Gamma \implies \Delta}\ {\to}l$$

Obviously, arbitrary sequents are (temporarily) allowed in proofs in G3i'; an application of a critical rule forces an intuitionistic sequent in its premise.

Other variants of multi-succedent intuitionistic calculi are known (e.g., Beth tableaux; see [7] for the details and references). A variant which will be considered later is Dragalin's calculus GHPC [2] which uses

$$\frac{A \to B,\, \Gamma \implies A \qquad B,\, \Gamma \implies \Delta}{A \to B,\, \Gamma \implies \Delta}\ {\to}l$$

instead of the $\to l$-inference of G3i' given above.

Finally, a sequent calculus G3s for modal logic S4 is obtained by *extending* G3c by the following rules where $\Box\Gamma$ ($\Diamond\Gamma$) means that each formula in Γ starts with $\Box$ ($\Diamond$).

$$\frac{\Box A,\, A,\, \Gamma \implies \Delta}{\Box A,\, \Gamma \implies \Delta}\ \Box l \qquad \frac{\Box\Gamma \implies \Diamond\Delta,\, A}{\Gamma',\, \Box\Gamma \implies \Diamond\Delta,\, \Delta',\, \Box A}\ \Box r$$

$$\frac{A,\, \Box\Gamma \implies \Diamond\Delta}{\Diamond A,\, \Gamma',\, \Box\Gamma \implies \Diamond\Delta,\, \Delta'}\ \Diamond l \qquad \frac{\Gamma \implies \Delta,\, A,\, \Diamond A}{\Gamma \implies \Delta,\, \Diamond A}\ \Diamond r$$

We use the terms *principal formula* and *proof* in a sequent system in the usual way. The principal formula of an inference q is sometimes denoted by F_q. By $\vdash_{\mathsf{G3s}} S$, $\vdash_{\mathsf{G3c}} S$, $\vdash_{\mathsf{G3i}} S$, $\vdash_{\mathsf{G3m}} S$, we mean provability of S in G3s, G3c, G3i, and G3m, respectively. According to the uniform tableau notation, we call unary inferences (without quantifier inferences) *α-rules*, and binary inferences *β-rules*. Principal formulae of a φ-rule ($\varphi \in \{\alpha, \beta\}$) are called *$\varphi$-formulae*.

The *length*, $|\alpha|$, of a proof α (in any of the systems used in this paper) is defined as the number of axioms occurring in it.

$$F, \Gamma \Longrightarrow F \ \text{Ax} \qquad \bot, \Gamma \Longrightarrow \Delta \ \bot l$$

$$\frac{A, B, \Gamma \Longrightarrow \Delta}{A \wedge B, \Gamma \Longrightarrow \Delta} \wedge l \qquad \frac{\Gamma \Longrightarrow A \quad \Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \wedge B} \wedge r$$

$$\frac{A, \Gamma \Longrightarrow \Delta \quad B, \Gamma \Longrightarrow \Delta}{A \vee B, \Gamma \Longrightarrow \Delta} \vee l \qquad \frac{\Gamma \Longrightarrow A_i}{\Gamma \Longrightarrow A_0 \vee A_1} \vee r \quad (i = 0, 1)$$

$$\frac{A \to B, \Gamma \Longrightarrow A \quad B, \Gamma \Longrightarrow \Delta}{A \to B, \Gamma \Longrightarrow \Delta} \to l \qquad \frac{A, \Gamma \Longrightarrow B}{\Gamma \Longrightarrow A \to B} \to r$$

$$\frac{\forall x\, A, A_t^x, \Gamma \Longrightarrow \Delta}{\forall x\, A, \Gamma \Longrightarrow \Delta} \forall l \qquad \frac{\Gamma \Longrightarrow A_y^x}{\Gamma \Longrightarrow \forall x\, A} \forall r$$

$$\frac{A_y^x, \Gamma \Longrightarrow \Delta}{\exists x\, A, \Gamma \Longrightarrow \Delta} \exists l \qquad \frac{\Gamma \Longrightarrow A_t^x}{\Gamma \Longrightarrow \exists x\, A} \exists r$$

Fig. 2. Axioms and inference rules of G3i.

The following definition of a polynomial simulation is adapted from [3]. A calculus P_1 can *polynomially simulate* (p-simulate) a calculus P_2 if there is a polynomial p such that the following holds. For every proof of a formula (or sequent) F in P_2 of length n, there is a proof of (the translation of) F in P_1, whose length is not greater than $p(n)$.

We use the terms *propositional formula*, *propositional sequent* etc. for a classical propositional formula, sequent etc. without $\Box, \Diamond$. If modalities are allowed, the term *modal* is added.

Definition 1. *The* classical kernel, $\mathsf{ck}(\cdot)$*, of a formula F and a sequent S is defined as follows. If F is atomic, then $\mathsf{ck}(F) = F$. If F is of the form $Qx\,A$ ($Q \in \{\forall, \exists\}$) then $\mathsf{ck}(F) = Qx\,\mathsf{ck}(A)$. If F is of the form $A \circ B$ ($\circ \in \{\vee, \wedge, \to\}$) then $\mathsf{ck}(F) = \mathsf{ck}(A) \circ \mathsf{ck}(B)$. If F is of the form $m\,A$ ($m \in \{\Box, \Diamond\}$) then $\mathsf{ck}(F) = \mathsf{ck}(A)$. If S is a sequent of the form*

$$A_1, \ldots, A_n \Longrightarrow B_1, \ldots, B_m.$$

then $\mathsf{ck}(S)$ is of the form

$$\mathsf{ck}(A_1), \ldots, \mathsf{ck}(A_n) \Longrightarrow \mathsf{ck}(B_1), \ldots, \mathsf{ck}(B_m).$$

Obviously, the classical kernel of a (modal) formula or a sequent is constructed by deleting all occurrences of $\Box$, $\Diamond$.

3 Different Embeddings of Int to S4

There is a number of slightly differing embeddings of Int to S4 (see, e.g., [8], [10], [11], [13]); we consider two of them (see [15] p. 230). The first mapping, $(\cdot)^\circ$, is

$$
\begin{array}{rlrl}
P^\circ &= P & P^\Box &= \Box P \\
\bot^\circ &= \bot & \bot^\Box &= \bot \\
(A \wedge B)^\circ &= A^\circ \wedge B^\circ & (A \wedge B)^\Box &= A^\Box \wedge B^\Box \\
(A \vee B)^\circ &= \Box A^\circ \vee \Box B^\circ & (A \vee B)^\Box &= A^\Box \vee B^\Box \\
(A \to B)^\circ &= (\Box A^\circ) \to B^\circ & (A \to B)^\Box &= \Box(A^\Box \to B^\Box) \\
(\forall x A)^\circ &= \forall x A^\circ & (\forall x A)^\Box &= \Box \forall x A^\Box \\
(\exists x A)^\circ &= \exists x \Box A^\circ & (\exists x A)^\Box &= \exists x A^\Box
\end{array}
$$

Fig. 3. The two embeddings used in this paper.

derived from Girard's embedding of intuitionistic logic into classical linear logic, whereas the second mapping, $(\cdot)^\Box$, is a slight variant of the embedding in [13] which in turn is the first-order generalization of the embedding T in [11]. The two embeddings are shown in Fig. 3, where P denotes an atomic formula.

Roughly speaking, $(\cdot)^\circ$ encodes intuitionistic restrictions of the rules $\vee r$, $\to l$, and $\exists r$, whereas $(\cdot)^\Box$ encodes restrictions of the critical rules $\to r$ and $\forall r$. However, according to $\Box r$, non-boxed antecedent formulae are deleted with the consequence that both embeddings are more restrictive than necessary.

Obviously, both translations are structure-preserving in the sense that the classical kernel of a translated formula is the source formula.

Let Γ be of the form $\{ B_1, \ldots, B_n \}$. For $\star \in \{ \circ, \Box \}$, we use $\Gamma^\star$ in order to denote $\{ (B_1)^\star, \ldots, (B_n)^\star \}$.

An important property of these embeddings is *faithfulness*:

$$\vdash_{\mathsf{G3s}} \Box\Gamma^\circ \Longrightarrow A^\circ \quad \text{iff} \quad \vdash_{\mathsf{G3s}} \Gamma^\Box \Longrightarrow A^\Box \quad \text{iff} \quad \vdash_{\mathsf{G3i}} \Longrightarrow (\bigwedge \Gamma) \to A$$

where $\Box\Gamma^\circ$ means $\{ \Box(B_1)^\circ, \ldots, \Box(B_n)^\circ \}$ if $\Gamma = \{ B_1, \ldots, B_n \}$.

We extend the calculus G3s by the two translations. A formula F is provable in $\star$G3s ($\star \in \{ \circ, \Box \}$) if there is a G3s–proof of the sequent $\Longrightarrow F^\star$.

4 Comparisons of Embeddings and Calculi

4.1 Hard Formulae for G3i and G3m

We present a class of propositional formulae for which any G3i-proof is exponential. For $n > 0$, we define $F_n \equiv L_n \to C$ where

$$
\begin{array}{l}
L_n \equiv A_n \wedge (O_0 \wedge (O_1 \wedge (\cdots (O_{n-1} \wedge O_n) \cdots)), \\
O_i \equiv \begin{cases} (A_0 \to C) & \text{if } i = 0, \\ (A_i \to ((A_{i-1} \vee C) \vee A_{i-1})) & \text{if } 1 \leq i \leq n, \end{cases}
\end{array}
$$

and $A_0, \ldots, A_n, C$ are atomic formulae. In contrast to the sequence of (more complicated) formulae given in [5], even different multi-succedent calculi for Int can be separated (with respect to p-simulation) with the sequence $(F_i)_{i>0}$.

Definition 2. *Let $S \equiv \Gamma \vdash \Delta$ be a propositional sequent and let $\mathcal{A}_S$ be the set of atoms occurring in S. We define an* evaluation function $\mathrm{val}_I(\cdot)$ *for propositional formulae and sequents which is based on a classical interpretation $I\colon \mathcal{A}_S \mapsto \{\mathbf{t}, \mathbf{f}\}$:*
$\mathrm{val}_I(\bot) = \mathbf{f}$,
$\mathrm{val}_I(A) = I(A)$, *for all* $A \in \mathcal{A}_S$,
$\mathrm{val}_I(F \wedge G) = \mathbf{f}$ *iff* $\mathrm{val}_I(F) = \mathbf{f}$ *or* $\mathrm{val}_I(G) = \mathbf{f}$,
$\mathrm{val}_I(F \vee G) = \mathbf{f}$ *iff* $\mathrm{val}_I(F) = \mathbf{f}$ *and* $\mathrm{val}_I(G) = \mathbf{f}$,
$\mathrm{val}_I(F \to G) = \mathbf{f}$ *iff* $\mathrm{val}_I(F) = \mathbf{t}$ *and* $\mathrm{val}_I(G) = \mathbf{f}$,
$\mathrm{val}_I(\Gamma \Longrightarrow \Delta) = \mathbf{f}$ *iff for every* $F \in \Gamma$, $G \in \Delta$, $\mathrm{val}_I(F) = \mathbf{t}$ *and* $\mathrm{val}_I(G) = \mathbf{f}$.
A sequent S is false *iff there exists an interpretation I such that* $\mathrm{val}_I(S) = \mathbf{f}$.

False (classical) sequents cannot be proven in the classical sequent calculus (because of its correctness). Therefore, these sequents are neither provable in G3i nor in G3s. We first show that any of the A_n, O_i $(0 \leq i \leq n)$, C are required for a classical (and hence, intuitionistic) proof of F_n.

Lemma 1. *Let F_n be as defined above. Let S_n be of the form*

$$S_n \;\equiv\; A_n, O_0, \ldots, O_n \;\Longrightarrow\; C$$

Removing one of the indicated A_n, O_i, or C from S_n yields a false *sequent S'_n.*

Proof. We show for any of the above deletions that the resulting sequent S'_n is *false.*
Case 1. Removing C from S_n results in a new sequent S'_n

$$A_n, O_0, \ldots, O_n \;\Longrightarrow .$$

Define $I(A_j) = \mathbf{t}$ $(0 \leq j \leq n)$, and $I(C) = \mathbf{t}$. Then $\mathrm{val}_I(S'_n) = \mathbf{f}$.
Case 2. Removing the indicated occurrence of A_n from S_n results in a new sequent S'_n

$$O_0, \ldots, O_n \;\Longrightarrow C.$$

Define $I(A_i) = \mathbf{f}$ $(0 \leq i \leq n)$ and $I(C) = \mathbf{f}$. Then $\mathrm{val}_I(S'_n) = \mathbf{f}$.
Case 3. Removing O_j $(0 \leq j \leq n)$ from S_n results in a new sequent S'_n

$$A_n, O_0, \ldots, O_{j-1}, O_{j+1}, \ldots, O_n \;\Longrightarrow C.$$

Define $I(A_i) = \mathbf{f}$ $(0 \leq i \leq j-1)$, $I(A_i) = \mathbf{t}$ $(j \leq i \leq n)$, and $I(C) = \mathbf{f}$. Then $\mathrm{val}_I(S'_n) = \mathbf{f}$. □

We introduce the notion of a reduction ordering.

Definition 3. *Let S be a sequent and $\mathcal{F}_S$ be the set of all (sub)formulae occurring in S (different occurrences of identical subformulae are distinguished by indexing). We define:*

1. *Let $\ll \subseteq \mathcal{F}_S \times \mathcal{F}_S$ be the subformula ordering of S, i.e., $(F, G) \in \ll$ iff G is an immediate subformula of F.*

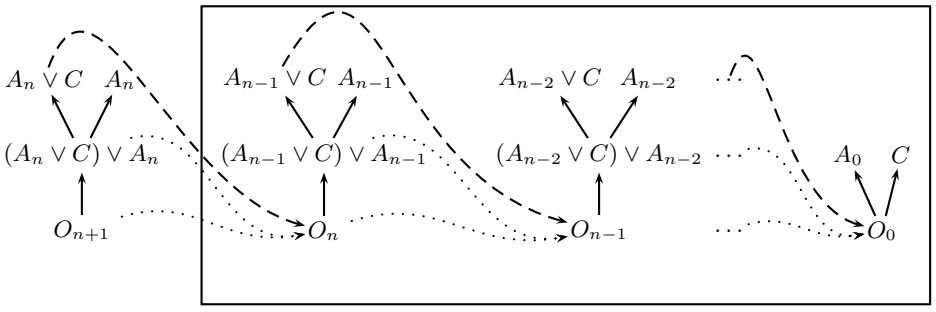


Fig. 4. Partial formula trees with some elements of $\sqsubset$ and $\lhd$.

2. *Let* $\sqsubset \subseteq \mathcal{F}_S \times \mathcal{F}_S$ *be a binary relation, different from* $\ll$.
3. *Let* $\lhd = (\ll \cup \sqsubset)^+$, *where* '$^+$' *denotes the transitive closure. The relation* $\lhd$ *is a* reduction ordering *for* S *iff* $\lhd$ *is acyclic.*

Let α *be a proof of* S *and* $\lhd$ *be a reduction ordering for* S. *A branch* b *of* S *in* α *is said to* satisfy $\lhd$ *iff for every two inferences* p, q *from* b *it holds: If* p *occurs somewhere above* q *on* b, *then* $(F_p, F_q) \notin \lhd$, *i.e., either* $F_q \lhd F_p$ *or the formulae* F_p, F_q *are not ordered. Otherwise, the two inferences* p, q *are said to* violate $\lhd$. α *is said to* satisfy $\lhd$ *iff every branch in* α *satisfies* $\lhd$.

Definition 4. *Let* $S_n \equiv A_n, O_0, \ldots, O_n \implies C$ $(n > 0)$ *and let* $\ll$ *be the subformula ordering of* S_n. *We define a relation* $\lhd = (\ll \cup \sqsubset)^+$, *where* $\sqsubset$ *is defined as*

$$\sqsubset \;=\; \bigcup_{0 \le j < n} \{\, (A_j \vee C,\ O_j) \,\}.$$

In the following, we depict (partial) formula trees together with some elements of the relations $\ll$, $\sqsubset$, and $\lhd$ (an example is given in Fig. 4). We use $F \longrightarrow G$ to denote $(F, G) \in \ll$ (i.e., G is an immediate subformula of F), $F \dashrightarrow G$ to denote $(F, G) \in \sqsubset$, and $F \cdots\!\!\rightarrow G$ to denote $(F, G) \in \lhd$.

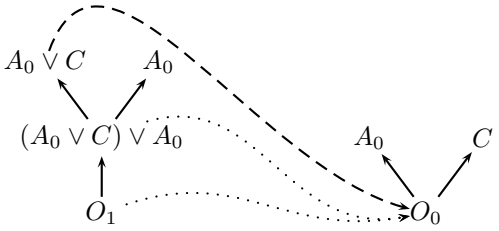
Lemma 2. *There is exactly one sequence* M_n

$$O_n,\ (A_{n-1} \vee C) \vee A_{n-1},\ A_{n-1} \vee C,\ \ldots,\ O_1,\ (A_0 \vee C) \vee A_0,\ A_0 \vee C,\ O_0$$

of non-atomic formulae from S_n *such that*

$$(*)\ \ F \text{ occurs before } G \text{ in } M_n \text{ iff } (F, G) \in \lhd.$$

Proof. The proof is by induction on n.
Base: $n = 1$. Consider $S_1 \equiv A_1, O_0, O_1 \implies C$. The only possibility for M_1 is $O_1,\ (A_0 \vee C) \vee A_0,\ A_0 \vee C,\ O_0$.



IH: Assume that, for all $m \leq n$, there exists exactly one sequence M_n

$$O_n,\ (A_{n-1} \vee C) \vee A_{n-1},\ A_{n-1} \vee C,\ \ldots,\ O_1,\ (A_0 \vee C) \vee A_0,\ A_0 \vee C,\ O_0$$

satisfying $(*)$.

Step. Consider the partial formula trees in Fig. 4. IH provides a unique sequence M_n for the partial formula trees in the box. The following three items follow from the mentioned pairs by $(F, G) \in \circ$ $(\circ \in \{\ll, \sqsubset\})$ implies $(F, G) \in \lhd$ and transitivity:

1. $(O_{n+1},\ A_n \vee C) \in \lhd$, because $(O_{n+1},\ (A_n \vee C) \vee A_n) \in \ll$ and $((A_n \vee C) \vee A_n,\ A_n \vee C) \in \ll$.
2. $((A_n \vee C) \vee A_n,\ O_n) \in \lhd$, because $((A_n \vee C) \vee A_n,\ A_n \vee C) \in \ll$ and $(A_n \vee C,\ O_n) \in \sqsubset$.
3. $(O_{n+1},\ O_n) \in \lhd$, because $(O_{n+1},\ A_n \vee C) \in \lhd$ and $(A_n \vee C,\ O_n) \in \sqsubset$.

Hence, $O_{n+1},\ (A_n \vee C) \vee A_n,\ A_n \vee C,\ M_n$ is the only sequence satisfying $(*)$. □

Corollary 1. *$\lhd$ is acyclic, i.e., it is a reduction ordering for S_n.*

In the following lemma, we show that every proof of S_n in G3i has to obey the reduction ordering $\lhd$.

Lemma 3. *Let $S_n \equiv A_n, O_0, \ldots, O_n \implies C$ $(n > 0)$ and let $\lhd$ be the reduction ordering from Definition 4. Then, $\lhd$ forms a reduction ordering for S_n such that* every G3i*-proof α of S_n has to satisfy $\lhd$.*

Proof. First, recall that $\lhd$ is acyclic, i.e., $\lhd$ is a reduction ordering (Corollary 1). The lemma is proven by contradiction. Assume that α_n is a proof of S_n in G3i, but $\lhd$ is not satisfied. We show that, under this circumstance, at least one sequent in α_n is classically false with the consequence that α_n is *not* a proof of S_n in G3i.

We consider the following elements of $\lhd$: for every j with $0 \leq j < n$ and m with $0 \leq m \leq j$,

$$(A_j \vee C,\ O_m) \in \lhd.$$

This follows immediately from Lemma 2.

We first show that the restriction to the above pairs does not violate generality. More precisely, we prove that, whenever there is a violation of $\lhd$, then there is a violation involving two inferences $\vee l$ and $\rightarrow l$ with principal formulae $A_j \vee C$ and O_m, respectively. Two important observations follow.

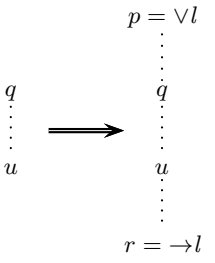


Fig. 5.

1. Any *positive* subformula A in S_n (i.e., C or any negative subformula of $O_0, \ldots, O_n$) is atomic. Hence, all axioms of α_n must be of the form $\Gamma, A \Longrightarrow A$. A reduction of non-atomic formulae in the antecedent is therefore required.
2. Since all formulae in S_n are needed for a proof (Lemma 1), especially all O_i must be reduced. Since all non-atomic subformulae of O_i (including O_i itself) are β-formulae, each non-atomic subformula of O_i has to be reduced. Otherwise, O_i could be removed from S_n without harm.

Consider the left part in Fig. 5. Let q and u denote two inferences from $\{\vee l,\ \rightarrow l\}$, let F_q and F_u be the corresponding principal formulae, let q occur somewhere above u, and let $(F_q, F_u) \in \lhd$. Hence, q and u violate $\lhd$. Moreover, let R_{k_l} be a (non-atomic) subformula of O_{k_l} in the following cases.
Case 1: $F_q = O_{k_q}$ and $F_u = O_{k_u}$.
Because of observation 2 and $(O_{k_q}, A_{k_q-1} \vee C) \in \ll$, there is an inference $p = \vee l$ (with $F_p = A_{k_q-1} \vee C$) above q which violates $(A_{k_q-1} \vee C, F_u) \in \lhd$.
Case 2: $F_q = (A_{k_q-1} \vee C) \vee A_{k_q-1}$ and $F_u = O_{k_u}$.
Similar to case 1 but with $(A_{k_q-1} \vee C) \vee A_{k_q-1}$ instead of O_{k_q}.
Case 3: $F_q = O_{k_q}$ and $F_u = R_{k_u}$.
Because of $(O_{k_u}, R_{k_u}) \in \ll$, there is an inference $r = \rightarrow l$ below u such that $F_r = O_{k_u}$. Similar arguments as in case 1 apply but with inference r instead of inference u.
Case 4: $F_q = (A_{k_q-1} \vee C) \vee A_{k_q-1}$ and $F_u = R_{k_u}$.
Choose the inference r as in case 3 and the inference p as in case 1. Then r and p violate $\lhd$ because $F_r = O_{k_u}$, $F_p = A_{k_q-1} \vee C$, and r occurs below p.

Let $\circ$ denote the last inference in α. Consider a branch b of S_n in α for which $\lhd$ is not satisfied. Starting from $\circ$, we select a pair of inferences (p, r) from b such that

(i) p occurs somewhere above r on b with $F_p = A_j \vee C$;
(ii) p and r violate $\lhd$ since $F_p \lhd F_r$.

In the following, Γ consists of exactly one of $A_j, C, A_j \vee C, (A_j \vee C) \vee A_j, O_{j+1}$ (for every j, $0 \leq j \leq n-1$) except for explicitly indicated formulae in the sequent which do not occur in Γ. Additionally, O_0 might occur in Γ. The exact value of Γ depends on the considered subbranch of b.

$(A_j \vee C,\ O_m) \in \lhd$ is violated. Consider a sequent with an antecedent of the form $A_n, \Gamma, F[A_j \vee C], O_m$, where $F[A_j \vee C]$ denotes a superformula of $A_j \vee C$. Moreover, O_m is reduced first (by the inference $r = \to l$) and $A_j \vee C$ is reduced later (by the inference $p = \vee l$). We obtain

$$S \ \equiv\ A_n,\ \Gamma,\ F[A_j \vee C],\ O_m \implies A_m$$

as the left premise of r. Let $\mathcal{B} = \{\, A_q \mid A_q \in \Gamma \,\} = \{\, A_{k_1}, \ldots, A_{k_l} \,\}$. There are the following subcases:
(a) If $\mathcal{B} = \emptyset$ then define $I(C) = \mathbf{t}$, $I(A_n) = \mathbf{t}$ and $I(A_m) = \mathbf{f}$. Computing $\mathrm{val}_I(S)$ yields $\mathbf{f}$.
(b) If $\mathcal{B} \neq \emptyset$ then select the element A_{k_l} with the greatest index k_l from $\mathcal{B}$. Then there exists an inference $q = \to l$ with principal formula O_{k_l+1} ($k_l + 1 < n$) below r on b, because this reduction is the only possibility to get a sequent with (a superformula of) A_{k_l} in the antecedent. Let

$$T \ \equiv\ A_n,\ \Gamma',\ O_{k_l+1} \implies A_{k_l+1}$$

be the left premise of q. Observe that all A's which occur in Γ' must also occur in Γ. Since $A_{k_l+1} \notin \Gamma$ (because A_{k_l} is the one with the greatest index in Γ), $A_{k_l+1} \notin \Gamma'$. Therefore, T is not an axiom. Define $I(A_{k_l+1}) = \mathbf{f}$, $I(C) = \mathbf{t}$, $I(A_l) = \mathbf{t}$ for all other A_l's. Computing $\mathrm{val}_I(T)$ yields $\mathbf{f}$.

In both cases, we get a false sequent which is not provable. Hence, α cannot be an G3i-proof which contradicts our assumption. □

Lemma 4. *Let n be a fixed, but arbitrary constant. Let axioms of the form $A_0, \Gamma \implies A_0$ be called* A-axioms. *Then 2^n A-axioms are required for any* G3i*-proof of the sequent $S_n \equiv A_n, O_0, \ldots, O_n \implies C$.*

Proof. Let N_k denote $A_{k-1} \vee C$ and let M_k denote $N_k \vee A_{k-1}$. Moreover, O^k denotes $\{\, O_0, \ldots, O_k \,\}$. By Lemma 3, any G3i-proof α_n of S_n has to obey $\lhd$. With Lemma 2, it follows that there is exactly one reduction sequence in any branch of α_n.

The proof is by induction on n.
Base. For $n = 1$, the only possibility for α_1 is as follows ($\Gamma_1 = \{\, A_1, O_0 \,\}$).

$$\dfrac{\Gamma_1, O_1 \implies A_1 \qquad \dfrac{\dfrac{\dfrac{\Gamma_1, A_0 \implies A_0 \quad A_1, C, A_0 \implies C}{\Gamma_1, A_0 \implies C}\to l \qquad \Gamma_1, C \implies C}{\Gamma_1, N_1 \implies C}\vee l \qquad \alpha_1'}{\Gamma_1, M_1 \implies C}\vee l}{A_1, O_0, O_1 \implies C}\to l$$

Let α_1' denote the leftmost subproof of α_1 (marked gray) with the end sequent $A_1,\ O_0,\ A_0 \implies C$, containing exactly one A-axiom. A copy of α_1' occurs above the right premise of the low-most $\vee l$-inference. Hence, there are two A-axioms in α_1.
IH: Assume that, for all $m \leq n$, any G3i-proof of S_m has 2^m A-axioms, each in a separate occurrence of α_m' (with the end sequent $A_m,\ O^{m-1},\ A_{m-1} \implies C$).

Step. Consider S_{n+1}. By IH, we have an G3i-proof α_n of S_n with 2^n A-axioms in 2^n separate occurrences of α'_n. Take α_n and add A_{n+1} to the antecedent of each sequent resulting in α'_{n+1} with end sequent $A_{n+1}, O^n, A_n \implies C$. Since A_{n+1} occurs only in antecedents of α'_{n+1}, no part of the proof becomes redundant due to new axioms. Take two copies of α'_{n+1} and derive:

$$\dfrac{A_{n+1},\, O^{n+1} \implies A_{n+1} \qquad \dfrac{\dfrac{\alpha'_{n+1} \quad A_{n+1},\, O^n,\, C \implies C}{A_{n+1},\, O^n,\, N_{n+1} \implies C}\lor l \qquad \dfrac{\alpha'_{n+1}}{A_{n+1},\, O^n,\, M_{n+1} \implies C}}{A_{n+1},\, O_0, \ldots, O_{n+1} \implies C}\to l$$

According to Lemma 1, O_{n+1} is relevant in any G3i-proof α_{n+1} of S_{n+1}. There is only one possibility to get the desired end sequent because of Lemma 2 and Lemma 3. □

Theorem 1. *Any proof of F_n in* G3i *or* G3m *has length $\geq 2^n$.*

Proof. The sequent $\implies F_n$ is deterministically reduced to S_n without introducing axioms. By Lemma 4, any G3i-proof α_n of S_n requires 2^n A-axioms. Inspecting α_n reveals that it is indeed a G3m-proof of S_n because no axiom of the form $\bot, \Gamma \implies G$ occurs in α_n. Since G3m is a "subcalculus" of G3i, it is impossible to get a shorter proof in G3m as in G3i. □

4.2 Hard Formulae Made Easy by Translations

We consider $(F_n)^\circ \equiv (\Box(L_n)^\circ) \to C$ where:

$$(L_n)^\circ \equiv A_n \land ((O_0)^\circ \land ((O_1)^\circ \land (\cdots((O_{n-1})^\circ \land (O_n)^\circ))\cdots),$$
$$(O_i)^\circ \equiv \begin{cases} (\Box A_0) \to C & \text{if } i = 0, \\ (\Box A_i) \to (\Box(\Box A_{i-1} \lor \Box C) \lor \Box A_{i-1}) & \text{if } 1 \leq i \leq n. \end{cases}$$

Lemma 5. *Let n be an arbitrary but fixed constant and let S^n_k be the sequent*

$$\Box(L_n)^\circ,\, A_n,\, (O_{m+k})^\circ, \ldots, (O_{m+1})^\circ \implies C,\, \Box A_0, \ldots, \Box A_{n-k}$$

where $m \geq 0$ and $n = m + k$. Then there exists a proof α^n_k of S^n_k in G3s *with length $3k + 1$.*

Proof. By induction on k. We abbreviate $\{\, \Box A_0, \ldots, \Box A_l \,\}$ by $\Box\Delta_l$ $(0 \leq l \leq n)$ and $\Gamma_n = \{\, \Box(L_n)^\circ,\, A_n \,\}$.
Base: $k = 1$. Then α^n_1 has 4 axioms and is

$$\dfrac{\dfrac{\dfrac{\dfrac{A_n, \bigwedge_{i=0}^n (O_i)^\circ \implies A_{m+1}}{A_n \land \bigwedge_{i=0}^n (O_i)^\circ \implies A_{m+1}}\land l}{\Box(L_n)^\circ \implies A_{m+1}}\Box l}{\Gamma_n \implies C, \Box\Delta_{n-k}, \Box A_{m+1}}\Box r \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Ax}(\Box A_m) \quad \dfrac{\mathrm{Ax}(C)}{\Gamma_n, \Box C \implies C, \Box\Delta_{n-k}}\Box l}{\Gamma_n, \Box A_m \lor \Box C \implies C, \Box\Delta_{n-k}}\lor l}{\Gamma_n, \Box(\Box A_m \lor \Box C) \implies C, \Box\Delta_{n-k}}\Box l \quad \mathrm{Ax}(\Box A_m)}{\Gamma_n, \Box(\Box A_m \lor \Box C) \lor \Box A_m \implies C, \Box\Delta_{n-k}}\lor l}{\Gamma_n, (O_{m+1})^\circ \implies C, \Box\Delta_{n-k}}\to l$$

where $\mathrm{Ax}(F)$ denotes an axiom of the form $F, \Gamma \Longrightarrow \Delta, F$. Observe that the leftmost top sequent is an axiom because $n = m + k$.
IH: Assume that $k > 1$ and, for all $d < k$, S^n_d has a proof α^n_d in G3s of length $3d + 1$.
Step. Consider S^n_k and α^n_k. We abbreviate $\{ (O_{m+k})^\circ, \ldots (O_{m+1})^\circ \}$ by $(\Gamma^n_k)^\circ$.

$$\dfrac{\dfrac{\alpha^n_{k-1}}{\Gamma_n, (\Gamma^n_{k-1})^\circ \Longrightarrow C, \Box\Delta_{n-k+1}} \quad \dfrac{\text{as in the base case with 3 axioms}}{\Gamma_n, (\Gamma^n_{k-1})^\circ, \Box(\Box A_m \vee \Box C) \vee \Box A_m \Longrightarrow C, \Box\Delta_{n-k}}}{\Gamma_n, (\Gamma^n_k)^\circ \Longrightarrow C, \Box\Delta_{n-k}} \to l$$

Apparently, the number of axioms of α^n_k is $3k + 1$ as desired. □

Lemma 6. *There exists a proof α^n of $(F_n)^\circ$ in* G3s *with length $3n + 2$.*

Proof. By Lemma 5, S^n_n has a proof α^n_n in G3s of length $3n + 1$. An additional $\to l$ with a further axiom $\Box(L_n)^\circ, A_n, (O_n)^\circ, \ldots, (O_1)^\circ, C \Longrightarrow C$ yields

$$\Box(L_n)^\circ, A_n, (O_n)^\circ, \ldots, (O_0)^\circ \Longrightarrow C.$$

Additional applications of unary inferences $\wedge l$, $\Box l$, and $\to r$ complete the proof of $(F_n)^\circ$ in G3s without adding more axioms. Hence, the length of α^n is $3n + 2$. □

We consider $(F_n)^\Box \equiv \Box((L_n)^\Box \to \Box C)$ where:

$$(L_n)^\Box \equiv (\Box A_n \wedge ((O_0)^\Box \wedge ((O_1)^\Box \wedge (\cdots((O_{n-1})^\Box \wedge (O_n)^\Box)) \cdots),$$
$$(O_i)^\Box \equiv \begin{cases} \Box((\Box A_0) \to \Box C) & \text{if } i = 0, \\ \Box((\Box A_i) \to ((\Box A_{i-1} \vee \Box C) \vee \Box A_{i-1})) & \text{if } 1 \leq i \leq n. \end{cases}$$

The proof of the next lemma is analogous to the proof of Lemma 6.

Lemma 7. *There exists a proof α^n of $(F_n)^\Box$ in* G3s *with length $3n + 2$.*

Theorem 2. G3i *cannot p-simulate* ∘G3s *and* □G3s.

Proof. By Theorem 1, Lemma 6 and Lemma 7.

Surprisingly, both translations enable short proofs of the embedded formula F_n in S4. Although $(\cdot)^\circ$ roughly coincides with $\vee r$ in G3i, the temporary use of more than one formula in the succedent enables a short proof in ∘G3s. A similar reason applies to $(\cdot)^\Box$ which roughly coincides with $\to r$ in G3i'.
In Section 2, we mentioned different multi-succedent intuitionistic calculi, namely G3i' and Dragalin's calculus GHPC. The sequence $F_1, F_2, \ldots$ of propositional formulae can be used to separate the two calculi with respect to polynomial simulation.

Corollary 2. *Dragalin's calculus GHPC cannot p-simulate* G3i'.

Proof. Exponential length of any proof of F_n in Dragalin's calculus GHPC follows from Theorem 1 and the following structural properties of F_n:

1. the top level symbol of F_n (viewed as a formula tree) is an implication;
2. the righthandside of this implication is a propositional variable;
3. all other implications occur negatively in F_n (hence, they are eventually reduced by $\to l$);
4. the lefthandside of all other implications are propositional variables.

Therefore, any proof of F_n in Dragalin's calculus GHPC is essentially a G3i–proof of the same formula. Short G3i'–proofs of F_n are easily established from the short proofs of F_n in $\Box$G3s. $\Box$

4.3 Comparing Different Translations

For $n > 0$, we define $G_n \equiv \overline{A}_1 \to (\overline{A}_2 \to \ldots (\overline{A}_n \to B_n) \cdots)$ where, for $1 \leq i \leq n$,

$$\overline{A}_i \equiv A_{i,1} \vee A_{i,2}$$
$$B_n \equiv \bigwedge_{i=1}^{n} \overline{A}_i.$$

Then we have:

1. $(G_n)^\circ \equiv ((\Box(\overline{A}_1)^\circ) \to ((\Box(\overline{A}_2)^\circ) \to \ldots\ ((\Box(\overline{A}_n)^\circ) \to (B_n)^\circ) \cdots))$ where, for $1 \leq i \leq n$,
$$(\overline{A}_i)^\circ \equiv \Box A_{i,1} \vee \Box A_{i,2}$$
$$(B_n)^\circ \equiv \bigwedge_{i=1}^{n} (\overline{A}_i)^\circ.$$
2. $(G_n)^\Box \equiv \Box((\overline{A}_1)^\Box \to \Box((\overline{A}_2)^\Box \to \ldots\ \Box((\overline{A}_n)^\Box \to (B_n)^\Box) \cdots))$ where, for $1 \leq i \leq n$,
$$(\overline{A}_i)^\Box \equiv \Box A_{i,1} \vee \Box A_{i,2}$$
$$(B_n)^\Box \equiv \bigwedge_{i=1}^{n} (\overline{A}_i)^\Box.$$

Lemma 8. *There exists a proof α^n of $(G_n)^\circ$ in* G3s *with length n.*

Proof. Apply n-times $\to r$ and get $\Box(\overline{A}_1)^\circ, \ldots, \Box(\overline{A}_n)^\circ \Longrightarrow (B_n)^\circ$. Then apply n-times $\Box l$ and n-times $\wedge r$ resulting in the desired proof α^n with n axioms. $\Box$

Lemma 9. *Let n be an arbitrary but fixed constant. For $1 \leq k < n$ and $n > 1$, any proof of the sequent S^n_{n-k} of the form*

$$\Box A_{1,i_1}, \ldots, \Box A_{k,i_k} \Longrightarrow \Box((\overline{A}_{k+1})^\Box \to \ldots\ \Box((\overline{A}_n)^\Box \to (B_n)^\Box) \cdots)$$

in G3s *has length $\geq 2^{n-k}$.*

Proof. First observe that an application of $\Box l$ below any $\Box r$ dominating an implication is redundant.

The proof is by induction on $d := n - k$.
Base: $d = 1$. Then we have to consider the sequent S_1^n of the form

$$\Box A_{1,i_1}, \ldots, \Box A_{n-1,i_{n-1}} \implies \Box((\overline{A}_n)^{\Box} \to (B_n)^{\Box}).$$

Then any proof α_1^n of S_1^n in G3s has length $\geq n$ ($\geq 2^d$).
IH: Assume that $d > 1$ and, for all $e < d$, S_e^n has a proof α_e^n in G3s of length $\geq 2^e$.
Step. Consider the sequent S_d^n of the form

$$\Box A_{1,i_1}, \ldots, \Box A_{k,i_k} \implies \Box((\overline{A}_{k+1})^{\Box} \to \ldots \Box((\overline{A}_n)^{\Box} \to (B_n)^{\Box}) \cdots).$$

We have to apply $\Box r$ and $\to r$ in order to avoid redundant sequents. Moreover, we have to apply $\vee l$ in order to avoid unprovable sequents. Now, we have two sequents (instead of one) of the form

$$\Box A_{1,i_1}, \ldots, \Box A_{k,i_k}, \Box A_{k+1,i_{k+1}} \implies \Box((\overline{A}_{k+2})^{\Box} \to \ldots \Box((\overline{A}_n)^{\Box} \to (B_n)^{\Box}) \cdots)$$

where i_{k+1} is either 1 or 2. By the induction hypothesis, each of the new two sequents is of the form S_{d-1}^n and has only proofs of length $\geq 2^{d-1}$. Hence, the new proof has length $\geq 2^d$. □

The following lemma is an immediate consequence of Lemma 9.

Lemma 10. *Any proof of* $(G_n)^{\Box}$ *in* G3s *has length* $\geq 2^n$.

Although the translation $(\cdot)^{\Box}$ seems to closely correspond to the multi-succedent version of G3i, the behavior with respect to proof complexity is rather different. The reason for this surprising behavior is the deletion of antecedent formulae by $\Box r$ and the "protection" of atomic formulae by a $\Box$–prefix. Consequently, left rules (especially $\vee l$) have to be applied before $\Box r$ resulting in the generation of many (unnecessary) branches.

Lemma 11. *There exists a proof* α^n *of* G_n *in* G3i *and* G3i' *with length* n.

Proof. Apply n-times $\to r$ and get $\overline{A}_1, \ldots, \overline{A}_n \implies B_n$. Then apply n-times $\wedge r$ resulting in the desired proof α^n with n axioms. □

Theorem 3. $\Box$G3s *cannot p-simulate* $\circ$G3s, G3i, *and* G3i'.

5 Conclusion

In this paper, we showed that some classes of formulae, which have only rather long proofs in standard Gentzen systems for Int (like, e.g., G3i) have rather short proofs in G3i' or $\circ$G3s and $\Box$G3s. The important property which enables the short proofs is the temporary use of more than one formula in the succedent.

Applications of $\Box r$ (or critical rules in case of multi-succedent calculi for Int) force an intuitionistic sequent in the rule's premise. Moreover, we compared two variants of multi-succedent calculi for Int and observed that, for a certain class of formulae, all GHPC-proofs are exponentially longer than minimal G3i'-proofs. The reason for the exponential increase of proof length is the weaker $\rightarrow l$-rule in GHPC which forces an intuitionistic sequent in its left premise.

It would be interesting to investigate the opposite translation from propositional S4 to propositional Int. Such a translation must exist because the "provability problem" is PSPACE-complete for both propositional logics. The question is whether there exists translations which are conceptually as simple as the embedding of Int to S4.

References

1. S. N. Artemov. Understanding Constructive Semantics. FoLLI Spinoza Lecture, 1999. `http:www.math.cornell.edu/~artemov/publ.html`.
2. A. G. Dragalin. *Mathematical Intuitionism. Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, 1988. Russian original 1979.
3. E. Eder. *Relative Complexities of First Order Calculi*. Vieweg, Braunschweig, 1992.
4. U. Egly and S. Schmitt. Intuitionistic Proof Transformations and their Application to Constructive Program Synthesis. In J. Calmet and J. Plaza, editors, *Proc. AISC*, volume 1476 of *LNAI*, pages 132–144. Springer Verlag, 1998.
5. U. Egly and S. Schmitt. On Intuitionistic Proof Transformations, Their Complexity and Application to Constructive Program Synthesis. *Fundamenta Informaticae*, 39:59–83, 1999.
6. S. Feferman, J. H. Dawson, Jr., S. C. Kleene, G. H. Moore, R. M. Solovay and J. van Heijenoort, editors. *Kurt Gödel Collected Works*, volume 1, New York, 1986, Oxford University Press.
7. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel, Dordrecht, 1983.
8. K. Gödel. Eine Interpretation des intuitionistischen Aussagenkalküls. In *Ergebnisse eines Mathematischen Kolloquiums*, volume 4, pages 39–40. 1933. English translation in [6], pp 300–303.
9. S. C. Kleene. *Introduction to Metamathematics*. North-Holland Publishing Company, Amsterdam, 1952.
10. S. Maehara. Eine Darstellung der intuitionistischen Logik in der klassischen. *Nagoya Mathematical Journal*, 7:45–64, 1954.
11. J. C. McKinsey and A. Tarski. Some Theorems About the Sentential Calculus of Lewis and Heyting. *Journal of Symbolic Logic*, 13(1):1–15, 1948.
12. I. E. Orlov. The Calculus of Compatibility of Propositions. *Mathematics of the USSR, Sbornik*, 35:263–286, 1928. In Russian.
13. H. Rasiowa and R. Sikorski. Algebraic Treatment of the Notion of Satisfiability. *Fundamenta Mathematicae*, 40:62–95, 1953.
14. A. S. Troelstra. Introductory Note to 1933f. In [6], pp. 296–299.
15. A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1996.
16. L. A. Wallen. *Automated Deduction in Nonclassical Logics*. MIT Press, 1990.

Term-Modal Logics

Melvin Fitting[1], Lars Thalmann[2], and Andrei Voronkov[3]

[1] City University of New York
fitting@alpha.lehman.cuny.edu
[2] Uppsala University
talman@csd.uu.se
[3] University of Manchester
voronkov@cs.man.ac.uk

Abstract. Many powerful logics exist today for reasoning about multi-agent systems, but in most of these it is hard to reason about an infinite or indeterminate number of agents. Also the naming schemes used in the logics often lack expressiveness to name agents in an intuitive way.
To obtain a more expressive language for multi-agent reasoning and a better naming scheme for agents, we introduce a family of logics called *term-modal logics*. A main feature of our logics is the use of modal operators indexed by the terms of the logics. Thus, one can *quantify over variables occurring in modal operators*. In term-modal logics agents can be represented by terms, and knowledge of agents is expressed with formulas within the scope of modal operators.
This gives us a flexible and uniform language for reasoning about the agents themselves and their knowledge. This paper gives examples of the expressiveness of the languages and provides sequent-style and tableau-based proof systems for the logics. Furthermore we give proofs of soundness and completeness with respect to the possible world semantics.

1 Introduction

1.1 Term-Modal Logics

In this paper, we describe a new family of modal logics, namely the term-modal first-order logics, where we by *term-modal* mean that any term can be used as a modality. The specific logics we discuss are the term-modal versions of the modal logics $\mathsf{K}, \mathsf{K4}, \mathsf{D}, \mathsf{D4}, \mathsf{T}$ and $\mathsf{S4}$. Sequent-style and tableau-style proof systems for the logics are given, and their soundness and completeness are shown.

1.2 Motivations

Many researchers have been interested in the use of multi-modal logics for knowledge representation, although most of them have been investigating the use of a finite set of modalities, indexed by the first n natural numbers, usually denoted either $[1], [2], \ldots, [n]$ or $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n$. When instead using an infinite set of

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 220–236, 2000.

modalities we can reason about a dynamic society of agents, where some agents might vanish and new agents may join the group.

In the family of multi-modal logics presented in this paper, any term can denote an agent. This makes naming of agents easy and the logics expressive. The use of complex names, possibly involving variables, for agents makes it easy to model a society of agents, and give names to new agents by their relationship to already existing agents. For example, to express that the agent $mother(x)$ thinks (or knows, or believes) that the agent x is good, we can write $[mother(x)]good(x)$.

The standard multi-modal logics allow us to reason about beliefs of particular agents, but provide very limited facilities to reason about believes of groups of agents or agents themselves. In our language, we can distinguish a group of agents by specifying their properties. For example, to express that every Christian believes in the existence of God, we can write $\forall x(christian(x) \supset [x]\exists y God(y))$.

An example of a society of agents is the collection of computer processes running on some system. Here the logic with its infinite complex naming mechanism can be used to specify requirements of the system as a whole and the proof system can be used to check that these requirements are satisfied.

When the computer processes spawn other processes, the society of agents (i.e. the number of processes) grows, and the naming mechanism can be used to refer to the newly created processes. As the number of processes spawned by the program may not be known beforehand, it is convenient to have an unlimited set of names for these new agents.

Many researchers have investigated multi-modal logic with a finite set of modalities, and many have discussed naming. We here combine these two aspects into one general family of logics, the family of term-modal logics.

1.3 Background

In Fitting [3], proofs of soundness and completeness of (single-)modal logics are given. In this paper we introduce some new definitions, and extend the proofs for the term-modal logics.

Fagin et al. [2] use modal logics to describe multi-agent systems. Their approach is based on a finite set of agents, and they also discuss the use of common and distributed knowledge. By using the logic presented in this paper, their work might be extended to handle dynamic agent societies with an easy naming mechanism, where quantification over agents is possible.

1.4 Overview

The rest of this paper is structured as follows. Section 2 defines the syntax of term-modal logics, and Section 3 their semantics. In Section 4 we introduce sequent calculi and in Section 5 tableau calculi for these logics. Soundness and completess of the sequent calculi are proved in Fitting, Thalmann, and Voronkov [5]. As a step toward automated reasoning in term-modal logic in Section 6 we introduce free-variable versions of tableau calculi for term-modal logics. In

Section 7 we give an example refutation in such a calculus, in order to illustrate some distinctive features of free-variable calculi for term-modal logics.

Proofs not included in this paper can be found in Fitting, Thalmann, and Voronkov [5].

2 Syntax

The term-modal logics are obtained from the standard predicate modal logics by adding modal operators indexed by terms. In this section we give a formal definition of the syntax of term-modal logics.

We assume a *signature* Σ consisting of three disjoint sets of constants, function symbols and relation symbols. Usually, the signature is assumed to be fixed, but in some situation we will vary it for technical convenience. In addition to the symbols of Σ, we will use various infinite *sets P of parameters* disjoint from the symbols in Σ. In some situations parameters will behave as new constants, in others as elements of a domain on which formulas are evaluated. The signature is not necessarily finite or countable. For every signature Σ, we denote by Σ^- the signature obtained from Σ by omitting all constants and function symbols.

Definition 1 (term) Suppose P is a set of parameters and V a set of *variables* disjoint from the set of parameters. The set of *terms of the signature Σ with parameters in P* and variables in V, denoted $\mathcal{T}(\Sigma \cup P, V)$ is defined inductively as follows.

1. Each constant in Σ, variable in V, and symbol in P is a term.
2. If $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a term.

In this paper, we can restrict ourselves to a fixed set of variables, however the set of parameters (and sometimes the signature) will vary. So we will use a simpler notation $\mathcal{T}(\Sigma \cup P)$. A term is called *ground* if it has no occurrences of variables.

Definition 2 (formula) Let P be a set of parameters. The set of *formulas of the signature Σ with parameters in P*, denoted $\mathcal{F}(\Sigma \cup P)$, is defined inductively as follows.

1. If R is a relation symbol of arity n and $t_1, \ldots, t_n$ are terms in $\mathcal{T}(\Sigma \cup P)$, then $R(t_1, \ldots, t_n)$ is an *atomic formula*.
2. If A and B are formulas, then so are $(A \wedge B)$, $(A \vee B)$, $(A \supset B)$ and $\neg A$.
3. If A is a formula and t is a term in $\mathcal{T}(\Sigma \cup P)$, then $[t]A$ and $\langle t \rangle A$ are formulas.
4. If A is a formula and x is a variable, then $\forall x A$ and $\exists x A$ are formulas.

The notions of *free and bound occurrences of variables* are defined as usual, with the exception of the following item:

- The free occurrences of variables in $[t]A$ and $\langle t \rangle A$ are all occurrences of variables in t plus all free occurrences of variables in A.

A formula is called *closed*, or a *sentence* if it has no free occurrences of variables (but note that it can contain parameters). A $\exists$-*formula* is any formula $\exists x A$. A *literal* is either an atomic formula A or its negation $\neg A$. Literals A and $\neg A$ are called *complementary* to each other.

Intuitively, when interpreting the formulas in a multi-agent context, the meaning of the formula $[t]A$ is that the agent denoted by the term t knows (believes etc.) the information represented by the formula A. The formula $\langle t \rangle A$ intuitively means that the agent denoted by t considers it possible that A holds, i.e., it is not the case that the agent knows the contrary (which can also be expressed by $\neg[t]\neg A$).

In the proof systems introduced later we make no assumptions about the kind of knowledge expressed. The knowledge could in fact be just beliefs, i.e., an agent might believe something which is false.

It is easy to add axioms of knowledge, if one is interested in describing a specific kind of knowledge. An example of this is the knowledge axiom, $[x]A \supset A$, which intuitively means that if an agent x knows something, then it is true. More about different epistemic interpretations of modal logic can be found in e.g. Hintikka [11].

The distinctive feature of our logics is the possibility to express knowledge of agents and properties of agents themselves in one language. For example, we can write $\forall x(\mathit{human}(x) \supset [x]\mathit{good}(x))$ to express that everyone knows that he/she is good. Note the use of quantification over agents.

For the rest of this paper, we will denote

- variables by x, y, z, u, v;
- terms by s, t;
- domain elements by d (i.e. elements in the set $\mathbf{D}$, which is defined later);
- formulas by A, B, C;
- sets of parameters of the language by P;
- parameters by p;
- literals by L;
- logics $\mathsf{K}, \mathsf{K4}, \mathsf{D}, \mathsf{D4}, \mathsf{T}, \mathsf{S4}$ by the generic symbol $\mathcal{L}$.

We write $A(x)$ to denote a formula A with (zero or more) free occurrences of the variable x and write $A(t)$ to denote the replacement of all free occurrences of x by a term t. Before the replacement, we rename in $A(x)$ all bound occurrences of variables that have free occurrences in t.

2.1 Related Work

The current trend in modal and description logics is to define expressive but still decidable logics. Our logics are undecidable since they contain first-order classical logic. Moreover, the expressiveness of our logics is, in a way, higher than that of the standard first-order modal logics, since first-order modal logics can be interpreted in our logics by using a single constant in modal operators (at least for cumulative domains, but our results can be extended to the constant domain versions of the logics as well). Logics with modalities indexed by terms were studied by Grove and Halpern [10,9]. These logics are more expressive in some aspects and less expressive in other aspects than ours. Namely, these

logics can handle equality and agents with special properties. However, there are restrictions on how formulas can be built in these logics, so some well-formed formulas of our logics cannot be used as formulas in Grove and Halpern [10,9].

3 Semantics

3.1 Frames

In this section, we describe a possible world semantics for the logics. The semantics is defined through the notions of *frames* and *structures*. It differs from the standard semantics of first-order modal logics by the treatment of the reachability relation on worlds: the reachability relation is indexed by elements of the domain. We assume all definitions be given w.r.t. a nonempty set $\mathbf{D}$, called the *domain*.

Definition 3 (frame) A *frame* over $\mathbf{D}$ is a triple $\langle \mathcal{W}, \mathcal{D}, \longrightarrow \rangle$, where

1. $\mathcal{W}$ is a non-empty set, called the *set of possible worlds*.
2. $\mathcal{D}$ is a mapping from $\mathcal{W}$ to the set of subsets of $\mathbf{D}$. The set $\mathcal{D}(w)$ is denoted by $\mathcal{D}_w$ and called the *domain of* w.
3. $\longrightarrow$ is a relation on $\mathcal{W} \times \mathbf{D} \times \mathcal{W}$, called the *accessibility relation*. If $\longrightarrow (w_1, d, w_2)$, then we say that w_2 is *d-reachable* from w_1 and write $w_1 \stackrel{d}{\longrightarrow} w_2$.

We require the following *monotonicity condition* to be satisfied in all frames:[1] if $w_1 \stackrel{d}{\longrightarrow} w_2$, then $\mathcal{D}_{w_1} \subseteq \mathcal{D}_{w_2}$. The monotonicity condition corresponds to *cumulative domains*, see e.g. Wallen [15] (also known as *nested domains*, see Garson [6]).

Definition 4 ($\mathcal{L}$-frame) We specialize the concept of frames to six different classes:

K. All frames are K-*frames*.

T. If $w \stackrel{d}{\longrightarrow} w$ holds for all w and d (i.e. the accessibility relation is reflexive in its 1st and 3rd arguments), then the frame is a T-*frame*.

D. If for all d and w there exists w' such that $w \stackrel{d}{\longrightarrow} w'$ (i.e. the accessibility relation satisfies seriality in its 1st and 3rd arguments) then the frame is a D-*frame*.

K4. If from $w \stackrel{d}{\longrightarrow} w'$ and $w' \stackrel{d}{\longrightarrow} w''$ it follows that $w \stackrel{d}{\longrightarrow} w''$ for all d and $w, w', w'' \in \mathcal{W}$, (i.e. the accessibility relation is transitive in its 1st and 3rd arguments) then it is a K4-*frame*.

S4. If the accessibility relation is both reflexive and transitive in its 1st and 3rd arguments, then the frame is an S4-*frame*.

D4. If the accessibility relation is both reflexive and satisfies seriality in its 1st and 3rd arguments, then the frame is a D4-*frame*.

[1] The monotonicity condition can be replaced by a weaker condition: if $w_1 \stackrel{d}{\longrightarrow} w_2$ and $d \in w_1$ then $\mathcal{D}_{w_1} \subseteq \mathcal{D}_{w_2}$. The reason is that the first-order language cannot express properties of worlds d-reachable from w, when $d \notin \mathcal{D}_w$.

3.2 First-Order Modal Structures

The first-order modal (Kripke) structures are introduced in the standard way, except for the case of modal operators.

DEFINITION 5 (structure for $\mathcal{L}$) Let $\mathcal{L}$ be one of K, T, D, $\mathsf{K4}$, $\mathsf{S4}$, $\mathsf{D4}$. A *first-order modal structure* for $\mathcal{L}$, or simply *$\mathcal{L}$-structure* over a domain $\mathbf{D}$ is a tuple $\mathfrak{S} = \langle \mathcal{W}, \mathcal{D}, \longrightarrow, I, \Vdash \rangle$, where

1. $\langle \mathcal{W}, \mathcal{D}, \longrightarrow \rangle$ is an $\mathcal{L}$-frame over $\mathbf{D}$.
2. $\Vdash$ is a binary relation between worlds and atomic sentences in $\mathcal{F}(\Sigma^- \cup \mathbf{D})$. (Note that elements of $\mathbf{D}$ are treated as parameters in $\mathcal{F}(\Sigma^- \cup \mathbf{D})$).
3. I, called the *interpretation function*, is a mapping that maps every constant c of Σ to an element of $\mathbf{D}$ and every function symbol f of Σ of arity n to an n-place function on $\mathbf{D}$. The corresponding element of $\mathbf{D}$ and function on $\mathbf{D}$ are called the *interpretations* of c and f respectively. We require the interpretation of any constant and function symbol to be totally defined in every world: this means that $I(c)$ belongs to $\mathcal{D}_w$ for every world $w \in \mathcal{W}$ and for every $d_1, \ldots, d_n \in \mathcal{D}_w$ we have $I(f)(d_1, \ldots, d_n) \in \mathcal{D}_w$.

Note that $\Vdash$ is only defined on formulas without function symbols or constants, but with parameters in $\mathbf{D}$.

We call a *valuation V in a structure* $\mathfrak{S}$ any mapping $V : P \to \mathbf{D}$ from a set of parameters to the domain $\mathbf{D}$ of $\mathfrak{S}$. Any valuation V can be extended to the set of all ground terms by defining

$$V(c) = I(c);$$
$$V(f(t_1, \ldots, t_n)) = I(f)(V(t_1), \ldots, V(t_n)).$$

Now we can give the central notion of satisfiability of formulas in structures. Given a first-order modal structure $\langle \mathcal{W}, \mathcal{D}, \longrightarrow, I, \Vdash \rangle$, we change the relation $\Vdash$ into a ternary relation between worlds in $\mathcal{W}$, valuations, and sentences in $\mathcal{F}(\Sigma \cup \mathbf{D})$ as given below. We write $\mathfrak{S}, w, V \Vdash A$ when this relation holds on $\mathfrak{S}, w, V, A$ and denote by $\nVdash$ the complement of $\Vdash$. When we use this notation, we can omit one or both of $\mathfrak{S}, V$, when they are clear from the context.

DEFINITION 6 (relation $\Vdash$) Given $\mathfrak{S}$ and V, we define the relation $\Vdash$ as follows.

1. $w, V \Vdash R(t_1, \ldots, t_n)$ if $w \Vdash R(V(t_1), \ldots, V(t_n))$.
2. $w, V \Vdash A \wedge B$ if $w, V \Vdash A$ and $w, V \Vdash B$.
3. $w, V \Vdash A \vee B$ if $w, V \Vdash A$ or $w, V \Vdash B$.
4. $w, V \Vdash A \supset B$ if $w, V \nVdash A$ or $w, V \Vdash B$.
5. $w, V \Vdash \neg A$ if $w, V \nVdash A$.
6. $w, V \Vdash [t]A$ if for all w' such that $w \xrightarrow{V(t)} w'$ we have $w', V \Vdash A$.
7. $w, V \Vdash \langle t \rangle A$ if there exists w' such that $w \xrightarrow{V(t)} w'$ and $w', V \Vdash A$.
8. $w, V \Vdash \forall x A(x)$ if $w, V \Vdash A(d)$, for all $d \in \mathcal{D}_w$.
9. $w, V \Vdash \exists x A(x)$ if $w, V \Vdash A(d)$, for some $d \in \mathcal{D}_w$.

Definition 7 (truth, satisfiability)
Let $\mathfrak{S} = \langle \mathcal{W}, \mathcal{D}, \longrightarrow, I, \Vdash \rangle$ be a structure. We say a sentence A is *true*, or *holds*, or *is locally satisfied* in $\mathfrak{S}$ at a world $w \in \mathcal{W}$ *under a valuation* V if $\mathfrak{S}, w, V \Vdash A$. A formula A is *globally satisfied* in a structure $\mathfrak{S}$ under a valuation V if it is satisfied at every world of $\mathfrak{S}$ under V. A formula A is called *locally (respectively, globally) satisfiable in* $\mathfrak{S}$ if it is locally (respectively, globally) satisfied in $\mathfrak{S}$ under some valuation. If A is locally satisfiable in $\mathfrak{S}$ we also say that $\mathfrak{S}$ is a *model* of A.

Note that the truth of a formula A under a valuation V only depends on the value of V on the parameters occurring in A. Thus, if A is a sentence of Σ, its truth does not depend on the valuation at all.

Definition 8 ($\mathcal{L}$-model, validity) Let $\mathcal{L}$ be one of K, T, D, $\mathsf{K4}$, $\mathsf{S4}$, $\mathsf{D4}$. We call a model $\langle \mathcal{W}, \mathcal{D}, \longrightarrow, I, \Vdash \rangle$ of a formula A an *$\mathcal{L}$-model* if its frame $\langle \mathcal{W}, \mathcal{D}, \longrightarrow \rangle$ is an $\mathcal{L}$-frame. A formula A is called *$\mathcal{L}$-satisfiable* if it has an $\mathcal{L}$-model. A formula A is called *$\mathcal{L}$-valid* if it is true in every world of every $\mathcal{L}$-structure under every valuation.

It is not hard to argue that satisfiability and validity are dual notions in the following sense: a formula A is unsatisfiable if and only if $\neg A$ is valid. In view of this duality we will formulate our results in terms of (un)satisfiability only.

When we speak of a *logic* $\mathcal{L}$ in this paper, we understand the set of $\mathcal{L}$-valid formulas. So, we will speak of *logics* K, T, D, $\mathsf{K4}$, $\mathsf{S4}$, $\mathsf{D4}$. Another standard way of introducing a logic is to define a suitable *calculus* deriving valid formulas in this logic. In the next section we introduce such calculi for all these logics.

Formulas A and B are called *$\mathcal{L}$-equivalent* if the formulas $A \supset B$ and $B \supset A$ are valid. It is evident that in any context when we speak about worlds, structures, valuations, and satisfiability, we can replace formulas by equivalent ones.

We will now introduce the negation normal form of formulas, which will simplify our proofs considerably.

Definition 9 (negation normal form) A formula A is said to be in *negation normal form* if it is constructed from literals using $\wedge$, $\vee$, $\forall$, $\exists$, $[t]$ and $\langle t \rangle$. A formula B is called a *negation normal form of a formula* A, if B is in negation normal form and B is equivalent to A.

It is not hard to argue that every formula has a negation normal form, so we will only consider formulas in negation normal form.

4 Sequent Calculi

In this section we define sequent calculi for the family of term-modal logics. There are several essentially equivalent notions of sequent giving rise to different calculi. The original definition of Gentzen [7] defines sequents as expressions $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$, where $A_1, \ldots, A_n, B_1, \ldots, B_m$ are formulas.

Smullyan [13] represents such a sequent as a collection $A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m$ of formulas or a collection $T\,A_1, \ldots, T\,A_n, F\,B_1, \ldots, F\,B_m$ of *signed formulas* with the intended meaning that all formulas A_i are true and all formulas B_j are false and introduces a *uniform notation* to group together inference rules with similar behavior. We will use the approach due to Schütte [12]. Instead of using arbitrary formulas we will use only formulas in *negation normal form*. Every rule in the uniform notation corresponds to an inference rule introducing a particular connective on formulas in negation normal form. Then we do not need the unifying notation anymore, since we can label inference rules by the corresponding connectives. We will use parameters instead of free variables, and therefore only deal with sentences.

Definition 10 (sequent) A *sequent* is a set of sentences. Let S be a sequent and $\mathfrak{S}$ be a structure. We say that a sequent S is *is locally satisfied* in $\mathfrak{S}$ at a world $w \in \mathcal{W}$ *under a valuation* V if $\mathfrak{S}, w, V \Vdash A$ for all $A \in S$. A sequent S is *globally satisfied* in a structure $\mathfrak{S}$ under a valuation V if S is satisfied at every world of $\mathfrak{S}$ under V. A sequent S is called *locally (respectively, globally) satisfiable in* $\mathfrak{S}$ if it is locally (respectively, globally) satisfied in $\mathfrak{S}$ under some valuation. If S is locally satisfiable in $\mathfrak{S}$ we also say that $\mathfrak{S}$ is a *model* of S.

Thus, a sequent is understood as a (possibly infinite) conjunction of its members.

For a formula A and a set of formulas S we use A, S or S, A to denote the set $S \cup \{A\}$. Likewise, we write S_1, S_2 to denote the union of two sequents $S_1 \cup S_2$.

Sequent calculi for logics K, K4, D, D4, T, S4 are shown in Figure 1.

Definition 11 (inference, derivation, refutation) *The inference rules* of the sequent calculi are shown in Figure 1. We call an *inference* any particular instance of an inference rule. The *premises* of any inference or inference rule are the sequents above the bar, its *conclusion* is the sequent below the bar. An *axiom* is any conclusion of (ax). A *derivation* of a sequent S is a tree made of inferences and having S as the root. A derivation is called a *refutation* if all leaves in it are axioms.

We use the term *refutation* instead of a proof because the sequent calculi used in this paper establish unsatisfiability rather than validity.

Example 12 Suppose that we wish to establish K-validity of the formula

$$\forall z([z]\forall x A(x) \supset \forall y[z]A(y)).$$

We turn this formula into its negation $\neg\forall z([z]\forall x A(x) \supset \forall y[z]A(y))$ and establish the unsatisfiability of the latter. To this end, we transform this formula into its negation normal form

$$\exists z([z]\forall x A(x) \wedge \exists y\langle z\rangle \neg A(y))$$

and try to find a refutation in the sequent calculus for K. An example refutation is as follows.

For all logics $\mathsf{K}, \mathsf{K4}, \mathsf{D}, \mathsf{D4}, \mathsf{T}, \mathsf{S4}$:

$$\frac{}{S, A, \neg A}\ (\text{ax}) \qquad \frac{S, A \quad S, B}{S, A \vee B}\ (\vee) \qquad \frac{S, A, B}{S, A \wedge B}\ (\wedge)$$

$$\frac{S, A(p)}{S, \exists x A(x)}\ (\exists) \qquad \frac{S, \forall x A(x), A(t)}{S, \forall x A(x)}\ (\forall) \qquad \frac{S^{[t]}, A}{S, \langle t \rangle A}\ (\langle t \rangle)$$

For logics with seriality (D, $\mathsf{D4}$):

$$\frac{S^{[t]}}{S}\ ([t])$$

For reflexive logics (T, $\mathsf{S4}$):

$$\frac{S, A}{S, [t]A}\ ([t])$$

Logic $\mathcal{L}$	**Definition of $S^{[t]}$**
K, T, D	$S^{[t]} = \{A \mid [t]A \in S\}$
S4	$S^{[t]} = \{[t]A \mid [t]A \in S\}$
K4, D4	$S^{[t]} = \{A \mid [t]A \in S\} \cup \{[t]A \mid [t]A \in S\}$

The rule ($\exists$) satisfies the *parameter condition*: p is a parameter having no occurrences in the conclusion of the rule.

Fig. 1. Sequent calculi

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x A(x), A(q), \neg A(q)}\ (\text{ax})}{\forall x A(x), \neg A(q)}\ (\forall)}{[p]\forall x A(x), \langle p \rangle \neg A(q)}\ (\langle p \rangle)}{[p]\forall x A(x), \exists y \langle p \rangle \neg A(y)}\ (\exists)}{[p]\forall x A(x) \wedge \exists y \langle p \rangle \neg A(y)}\ (\wedge)}{\exists z([z]\forall x A(x) \wedge \exists y \langle z \rangle \neg A(y))}\ (\exists)$$

This refutation is also a valid refutation in T and D.

To obtain a refutation in $\mathsf{K4}$ and $\mathsf{D4}$, we have to modify the top part of this refutation because of the difference in the definition of $S^{[t]}$:

$$\dfrac{\dfrac{\dfrac{}{[p]\forall x A(x), \forall x A(x), A(q), \neg A(q)}\ (\text{ax})}{[p]\forall x A(x), \forall x A(x), \neg A(q)}\ (\forall)}{[p]\forall x A(x), \langle p \rangle \neg A(q)}\ (\langle p \rangle)$$

A refutation in $\mathsf{S4}$ follows a different strategy because of the difference in the ($[t]$) rule:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x A(x), A(q), \neg A(q)}\,(\text{ax})}{\forall x A(x), \neg A(q)}\,(\forall)}{[p]\forall x A(x), \neg A(q)}\,([p])}{[p]\forall x A(x), \langle p\rangle \neg A(q)}\,(\langle p\rangle)$$

Since every formula has a negation normal form, we can restrict ourselves to negation normal forms.

We will augment the logics defined above with so-called *global assumptions*. Let Ψ be a set of sentences and $\mathcal{L}$ be one of the logics defined so far. We call a *sequent calculus for $\mathcal{L}$ with global assumptions Ψ* the calculus obtained from $\mathcal{L}$ by adding the rule $\dfrac{S, A}{S}\ (\Psi)$, where $A \in \Psi$.

Soundness and completeness of sequent calculi are proven in Fitting, Thalmann, and Voronkov [5]. The proofs are quite lengthy and not included. They are based on the method of Fitting [3]. Completeness is formulated as follows:

THEOREM 13 (Completeness of sequent calculi) *Let S be a set of sentences. If S has no refutation in the sequent calculus for $\mathcal{L}$ with the global assumptions Ψ, then there exists an $\mathcal{L}$-structure $\mathfrak{S}$ and a valuation V under which S is locally satisfied and all formulas in Ψ are globally satisfied.*

5 Tableau Systems

Tableau systems formalize proof-search in sequent calculi. Tableaux are often introduced as trees of formulas, with inference rules on tableaux formulated in terms of *branches*. To simplify the presentation, we introduce tableaux as multisets of branches.

DEFINITION 14 (tableau, branch) A *tableau* is a finite multiset $S_1, S_2, \ldots$ of sequents, denoted $S_1 \mid S_2 \mid \cdots$. The *empty tableau* is denoted by $\#$. Every sequent S_i is called a *branch* of this tableau.

The tableau calculus for each logic studied in this paper can be obtained by a simple transformation of the corresponding sequent calculus. For every inference rule $\dfrac{S_1 \quad \ldots \quad S_n}{S}$ of the sequent calculus, the corresponding tableau rule has the form $\dfrac{S \mid \mathcal{T}}{S_1 \mid \ldots \mid S_n \mid \mathcal{T}}$ where $\mathcal{T}$ is any tableau. Note the reverse order of the sequents. The tableau calculus rules have the following intuitive meaning: suppose that we search for a refutation of S and all sequents in $\mathcal{T}$. Then, since there is a sequent calculus rule reducing S to the sequents $S_1, \ldots, S_n$, it is enough to find a refutation of $S_1, \ldots, S_n$ and all sequents in $\mathcal{T}$. To find a refutation for a formula A, we begin with a tableau consisting of one branch A and try to apply the tableau rules until no (unrefuted) branches remain.

Formally, the *tableau calculi* for $\mathcal{L}$ are shown in Figure 2.

For all logics $\mathsf{K}, \mathsf{K4}, \mathsf{D}, \mathsf{D4}, \mathsf{T}, \mathsf{S4}$:

$$\frac{S, A, \neg A \mid \mathcal{T}}{\mathcal{T}}\ |\text{ax}| \qquad \frac{S, A \vee B \mid \mathcal{T}}{S, A \mid S, B \mid \mathcal{T}}\ |\vee| \qquad \frac{S, A \wedge B \mid \mathcal{T}}{S, A, B \mid \mathcal{T}}\ |\wedge|$$

$$\frac{S, \exists x A(x) \mid \mathcal{T}}{S, A(p) \mid \mathcal{T}}\ |\exists| \qquad \frac{S, \forall x A(x) \mid \mathcal{T}}{S, \forall x A(x), A(t) \mid \mathcal{T}}\ |\forall| \qquad \frac{S, \langle t \rangle A \mid \mathcal{T}}{S^{[t]}, A \mid \mathcal{T}}\ |\langle t \rangle|$$

For logics with seriality (D, $\mathsf{D4}$): For reflexive logics (T, $\mathsf{S4}$):

$$\frac{S \mid \mathcal{T}}{S^{[t]} \mid \mathcal{T}}\ |[t]| \qquad \frac{S, [t]A \mid \mathcal{T}}{S, A \mid \mathcal{T}}\ |[t]|$$

For logics with global assumptions Ψ:

$$\frac{S \mid \mathcal{T}}{S, A \mid \mathcal{T}}\ |\Psi|$$

Here $S^{[t]}$ is defined in the same way as for the sequent calculi. *Parameter condition*: in the rule $|\exists|$ p is a parameter having no occurrences in the premise of the rule. In the rule $|\Psi|$, $A \in \Psi$.

Fig. 2. Tableau calculi

Theorem 15 (Equivalence of tableau calculi and sequent calculi) *A sequent S has a refutation in the sequent calculus for $\mathcal{L}$ (with global assumptions Ψ) if and only if there exists a derivation of # from S in the tableau calculus for $\mathcal{L}$ (with the global assumptions Ψ).* □

6 Free-Variable Tableaux

In this section we change the tableau systems introduced in Section 5 into free-variable tableau systems. We will use the definitions introduced so far, except that we now allow free variables to occur in sequents and tableaux.

To avoid problems with the parameter condition in ($\forall$) rules, we introduce the *occurrence constraints* similar to those used in Voronkov [14], but we could use the "dynamic skolemization" technique introduced in Fitting [4] as well.

In this section we assume knowledge of the standard notions of substitutions and (idempotent, most general) unifiers see, e.g., Eder [1]. The application of a substitution σ to a term or formula E is denoted $E\sigma$. As usual, we may need to rename bound variables in a formula before we apply a substitution to it. Any idempotent most general unifier of n expressions $E_1, \ldots, E_n$ is denoted by $mgu(E_1, \ldots, E_n)$. The set of free variables of any expression E (e.g. formula or set of formulas) is denoted by $vars(E)$.

Definition 16 (occurrence constraint) We call a *simple occurrence constraint* either $\bot$ or an expression $p \notin X$, where p is a parameter and X is a finite set of variables. An *occurrence constraint* is a conjunction of zero or more simple occurrence constraints. A conjunction of zero simple occurrence constraints is denoted by $\top$.

For any substitution σ and simple occurrence constraint $\mathcal{C} = (p \notin X)$, we denote by $\mathcal{C}\sigma$ the following simple occurrence constraint:

$$\mathcal{C}\sigma = \begin{cases} \bot, & \text{if } p \text{ occurs in } X\sigma; \\ p \notin \mathit{vars}(X\sigma), & \text{otherwise.} \end{cases}$$

When $\mathcal{C}$ is a conjunction $\mathcal{C}_1 \wedge \ldots \wedge \mathcal{C}_n$ of simple occurrence constraints, we denote by $\mathcal{C}\sigma$ the following occurrence constraint:

$$\mathcal{C}\sigma = \begin{cases} \bot, & \text{if } \mathcal{C}_i = \bot \text{ for some } i; \\ \mathcal{C}_1\sigma \wedge \ldots \wedge \mathcal{C}_n\sigma, & \text{otherwise.} \end{cases}$$

An occurrence constraint $\mathcal{C}$ is called *satisfiable* if $\mathcal{C}$ is not $\bot$. A *solution* to an occurrence constraint $\mathcal{C}$ is any substitution σ such that $\mathcal{C}\sigma \neq \bot$ and $x\sigma$ is ground for every variable occurring in $\mathcal{C}$. Evidently, an occurrence constraint $\mathcal{C}$ is satisfiable if and only if it has a solution: indeed, one can take as a solution any substitution mapping all variables of $\mathcal{C}$ into any ground term not containing parameters in $\mathcal{C}$.

We call a *constrained tableau* any pair consisting of a tableau $\mathcal{T}$ and constraint $\mathcal{C}$, denoted $\mathcal{T} \cdot \mathcal{C}$. Let $\mathcal{L}$ be one of the logics $\mathsf{K}, \mathsf{K4}, \mathsf{D}, \mathsf{D4}, \mathsf{T}$ and $\mathsf{S4}$. The *free-variable tableau calculi* for $\mathcal{L}$ are shown in Figure 3.

Theorem 17 (Equivalence of free-tableau calculi and sequent calculi)
Let S be a set of sentences of the signature Σ. Then S has a refutation in the sequent calculus for $\mathcal{L}$ (with global assumptions Ψ) if and only if there exists a derivation of $\# \cdot \mathcal{C}$ from $S \cdot \top$ in the tableau calculus for $\mathcal{L}$ (with the global assumptions Ψ) such that $\mathcal{C}$ is satisfiable.

In order to prove this theorem, we will prove two results showing bisimulation between tableau derivations and free-variable tableau derivations.

Let $\mathcal{T}\cdot\mathcal{C}$ be a constrained tableau and σ be a substitution. We call the tableau $\mathcal{T}\sigma$ the *σ-instance* of $\mathcal{T} \cdot \mathcal{C}$ if $\mathcal{C}\sigma$ is satisfiable. A tableau $\mathcal{T}'$ is called an *instance* of $\mathcal{T} \cdot \mathcal{C}$ if it is a σ-instance of $\mathcal{T} \cdot \mathcal{C}$ for some σ.

Lemma 18 *Suppose there exists a derivation of $\mathcal{T}_2 \cdot \mathcal{C}$ from $\mathcal{T}_1 \cdot \top$ in the free-variable tableau calculus for $\mathcal{L}$ with global assumptions Ψ. Then any instance of $\mathcal{T}_2 \cdot \mathcal{C}$ has a derivation from $\mathcal{T}_1$ in the tableau calculus for $\mathcal{L}$ with the global assumptions Ψ.*

Proof. The proof is by induction on the length of derivations in the free-variable tableau calculus. When the derivation is of length 0, the claim is obvious, since $\mathcal{T}_1$ is the only instance of $\mathcal{T}_1 \cdot \top$, when $\mathcal{T}_1$ has no free variables. For derivations

$$\frac{S, A(\bar{s}), \neg A(\bar{t}) \mid \mathcal{T} \cdot \mathcal{C}}{\mathcal{T} mgu(\bar{s}, \bar{t}) \cdot \mathcal{C} mgu(\bar{s}, \bar{t})} \ |\text{ax}| \qquad \frac{S, A \vee B \mid \mathcal{T} \cdot \mathcal{C}}{S, A \mid S, B \mid \mathcal{T} \cdot \mathcal{C}} \ |\vee| \qquad \frac{S, A \wedge B \mid \mathcal{T} \cdot \mathcal{C}}{S, A, B \mid \mathcal{T} \cdot \mathcal{C}} \ |\wedge|$$

$$\frac{S, \exists x A(x) \mid \mathcal{T} \cdot \mathcal{C}}{S, A(p) \mid \mathcal{T} \cdot \mathcal{C} \wedge p \notin vars(S, \exists x A(x))} \ |\exists| \qquad \frac{S, \forall x A(x) \mid \mathcal{T} \cdot \mathcal{C}}{S, \forall x A(x), A(y) \mid \mathcal{T} \cdot \mathcal{C}} \ |\forall|$$

$$\frac{S, \langle t \rangle A \mid \mathcal{T} \cdot \mathcal{C}}{S^{[t_1]}, \ldots, S^{[t_n]}, A \mid \mathcal{T}\sigma \cdot \mathcal{C}\sigma} \ |\langle t, t_1, \ldots, t_n \rangle|$$

For logics with seriality (D, D4): $\dfrac{S \mid \mathcal{T} \cdot \mathcal{C}}{S^{[t_1]}, \ldots, S^{[t_n]}, \mid \mathcal{T}\sigma \cdot \mathcal{C}\sigma} \ |[t_1, \ldots, t_n]|$

For reflexive logics (T, S4): $\dfrac{S, [t]A \mid \mathcal{T} \cdot \mathcal{C}}{S, A \mid \mathcal{T} \cdot \mathcal{C}} \ |[t]|$

For logics with global assumptions Ψ:

$$\frac{S \mid \mathcal{T} \cdot \mathcal{C}}{S, A \mid \mathcal{T} \cdot \mathcal{C}} \ |\Psi|$$

In the rule $|\langle t, t_1, \ldots, t_n \rangle|$, $\sigma = mgu(t, t_1, \ldots, t_n)$. In the rule $|[t_1, \ldots, t_n]|$, $\sigma = mgu(t_1, \ldots, t_n)$. In the rule $|\exists|$, p is new parameter, not occurring in the premise. In the rule $|\forall|$, y is a new variable, not occurring in the premise. In the rule $|\Psi|$, $A \in \Psi$.

Fig. 3. Free-variable tableau with constraints calculi

with at least one inference, consider the last inference of the derivation. We will consider only two cases, other cases are similar.

Case: *the last inference is* $|\text{ax}|$. Then it has the form $\dfrac{S, A(\bar{s}), \neg A(\bar{t}) \mid \mathcal{T} \cdot \mathcal{C}}{\mathcal{T}\sigma \cdot \mathcal{C}\sigma} \ |\text{ax}|$, where $\sigma = mgu(\bar{s}, \bar{t})$.
Take any instance of $\mathcal{T}\sigma \cdot \mathcal{C}\sigma$, then this instance has the form $\mathcal{T}\sigma\tau$ for some substitution τ such that $\mathcal{C}\sigma\tau$ is satisfiable. We have to prove that $\mathcal{T}\sigma\tau$ is derivable from $\mathcal{T}_1$.
We claim that the following is a valid inference in the tableau calculus:

$$\frac{(S, A(\bar{s}), \neg A(\bar{t}) \mid \mathcal{T})\sigma\tau}{\mathcal{T}\sigma\tau} \ |\text{ax}|. \tag{1}$$

Indeed, since σ is a unifier of $\bar{s}$ and $\bar{t}$, then $A(\bar{s})\sigma = A(\bar{t})\sigma$, hence $A(\bar{s})\sigma\tau = A(\bar{t})\sigma\tau$.
Since $\mathcal{C}\sigma\tau$ is satisfiable, $(S, A(\bar{s}), \neg A(\bar{t}) \mid \mathcal{T})\sigma\tau$ is a $\sigma\tau$-instance of $S, A(\bar{s}), \neg A(\bar{t}) \mid \mathcal{T} \cdot \mathcal{C}$. By the induction hypothesis, this instance has a derivation from $\mathcal{T}_1$. Add to this derivation the inference (1), then we obtain a required derivation of $\mathcal{T}\sigma\tau$.
Case: *the last inference is* $|\exists|$.

$$\frac{S, \exists x A(x) \mid \mathcal{T} \cdot \mathcal{C}}{S, A(p) \mid \mathcal{T} \cdot \mathcal{C} \wedge p \notin vars(S, \exists x A(x))} \; |\exists|.$$

Take any instance of $S, A(p) \mid \mathcal{T} \cdot \mathcal{C} \wedge p \notin vars(S, \exists x A(x))$, then there is a substitution τ such that this instance has the form $S\tau, A(p)\tau \mid \mathcal{T}\tau$ and $(\mathcal{C} \wedge p \notin vars(S, \exists x A(x)))\tau$ is satisfiable. We have to prove that $S\tau, A(p)\tau \mid \mathcal{T}\tau$ is derivable from $\mathcal{T}_1$.

By the definition of constraint satisfiability, since $(\mathcal{C} \wedge p \notin vars(S, \exists x A(x)))\tau$ is satisfiable, then $\mathcal{C}\tau$ is satisfiable and p does not occur in $S\tau, \exists x A(x)\tau$. Since p does not occur in $S\tau, \exists x A(x)\tau$, the following is a valid inference in the tableau calculus:

$$\frac{S\tau, \exists x A(x)\tau \mid \mathcal{T}\tau}{S\tau, A(p)\tau \mid \mathcal{T}\tau} \; |\exists|. \tag{2}$$

By the induction hypothesis, every instance of $S, \exists x A(x) \mid \mathcal{T} \cdot \mathcal{C}$ has a derivation from $\mathcal{T}_1$. Since $\mathcal{C}\tau$ is satisfiable, we can take its τ-instance $S\tau, \exists x A(x)\tau \mid \mathcal{T}\tau$, this instance has a derivation from $\mathcal{T}_1$. Add to this derivation inference (2) and we obtain a required derivation of $S\tau, A(p)\tau \mid \mathcal{T}\tau$ from $\mathcal{T}_1$.

Now we want to prove a simulation result in the inverse direction. If we defined a sequent as a multiset of formulas, we could use an argument similar to the previous lemma. The use of sets instead of multisets causes some technical problems because the notion of instance does not work properly any more. To avoid these technical problems we give a definition of generalization that is nearly inverse to the notion of instance but takes into account some specific problems in the inverse simulation proof.

Let $\mathcal{T} = S_1 \mid \ldots \mid S_n$ and $\mathcal{T}' = S'_1 \mid \ldots \mid S'_n$ be two tableaux. We write $\mathcal{T} \sqsubseteq \mathcal{T}'$ if (i) for every $i = 1 \ldots n$ we have $S_i \subseteq S'_i$ and (ii) each parameter occurring in some $\mathcal{T}'$ also occurs in $\mathcal{T}$. Let $\mathcal{T} \cdot \mathcal{C}$ be a constrained tableau and $\mathcal{T}'$ a tableau. We call $\mathcal{T} \cdot \mathcal{C}$ a *σ-generalization* of $\mathcal{T}'$ if $\mathcal{T}' \sqsubseteq \mathcal{T}\sigma$ and $\mathcal{C}\sigma$ is satisfiable. We call $\mathcal{T} \cdot \mathcal{C}$ a *generalization* of $\mathcal{T}'$ if $\mathcal{T} \cdot \mathcal{C}$ is a σ-generalization of $\mathcal{T}'$ for some σ.

LEMMA 19 *Suppose there exists a derivation of $\mathcal{T}_2$ from $\mathcal{T}_1$ in the tableau calculus for $\mathcal{L}$ with global assumptions Ψ. Then some generalization of $\mathcal{T}_2$ has a derivation from $\mathcal{T}_1 \cdot \top$ in the free-variable tableau calculus for $\mathcal{L}$ with the global assumptions Ψ.*

PROOF. The proof is by induction on the length of derivations in the tableau calculus. When the derivation is of length 0, the claim is obvious, since $\mathcal{T}_1 \cdot \top$ is a generalization of $\mathcal{T}_1$. For derivations with at least one inference, consider the last inference of the derivation. We will consider only two cases, other cases are similar.

Case: *the last inference is* $|\mathrm{ax}|$. Then it has the form $\dfrac{S, A, \neg A \mid \mathcal{T}}{\mathcal{T}}\ |\mathrm{ax}|$. By the induction hypothesis, some σ-generalization of $S, A, \neg A \mid \mathcal{T}$ is derivable from $\mathcal{T}_1 \cdot \top$. Then this generalization has a form $S', A', \neg B' \mid \mathcal{T}' \cdot \mathcal{C}$ such that $A'\sigma = A$, $B'\sigma = A$, $\mathcal{T} \sqsubseteq \mathcal{T}'\sigma$ and $\mathcal{C}\sigma$ is satisfiable. Then σ is a unifier of A' and B', therefore, there exists a most general unifier τ of A' and B' and a substitution δ such that $\tau\delta = \sigma$. Consider the following inference in the free-variable tableau calculus.

$$\frac{S', A', \neg B' \mid \mathcal{T}' \cdot \mathcal{C}}{\mathcal{T}'\tau \cdot \mathcal{C}\tau}\ |\mathrm{ax}|.$$

We claim that the conclusion of this inference is a generalization of $\mathcal{T}$, this will complete the proof of this case. To prove the claim, we have to find a substitution δ' such that (i) $\mathcal{T} \sqsubseteq \mathcal{T}'\tau\delta'$ and (ii) $\mathcal{C}\tau\delta'$ is satisfiable. Well, take δ' to be δ, then both (i) and (ii) follow from $\tau\delta' = \sigma$.

Case: *the last inference is* $|\exists|$.

$$\frac{S, \exists x A(x) \mid \mathcal{T}}{S, A(p) \mid \mathcal{T}}\ |\exists| \tag{3}$$

By the induction hypothesis, some σ-generalization of $S, \exists x A(x) \mid \mathcal{T}$ is derivable from $\mathcal{T}_1 \cdot \top$. Then this generalization has a form $S', \exists x A'(x) \mid \mathcal{T}' \cdot \mathcal{C}$ such that (i) $S \subseteq S'\sigma$, (ii) every parameter occurring in $(S', \exists x A'(x))\sigma$ also occurs in $S, \exists x A(x) \mid \mathcal{T}$, (iii) $\exists x A'(x)\sigma = \exists x A(x)$, (iv) $\mathcal{T} \sqsubseteq \mathcal{T}'\sigma$, and (v) $\mathcal{C}\sigma$ is satisfiable. Consider the following inference in the free-variable tableau calculus.

$$\frac{S', \exists x A'(x) \mid \mathcal{T}' \cdot \mathcal{C}}{S', A'(p) \mid \mathcal{T}' \cdot \mathcal{C} \wedge p \notin \mathit{vars}(S', \exists x A'(x))}\ |\exists|.$$

Let us check that the parameter condition is satisfied. Suppose, by contradiction, that p occurs in $S', \exists x A'(x) \mid \mathcal{T}'$, then it also occurs in $(S', \exists x A'(x) \mid \mathcal{T}')\sigma$, hence also in $S, \exists x A(x) \mid \mathcal{T}$. This violates the parameter condition of (3).

We claim that the conclusion of this inference is a generalization of $S, A(p) \mid \mathcal{T}$, this will complete the proof of this case. We actually claim that the conclusion is the σ-generalization of $S, A(p) \mid \mathcal{T}$. All conditions on σ-generalization except for constraint satisfaction immediately follow from (i)–(iv) above. It remains to verify that $(\mathcal{C} \wedge p \notin \mathit{vars}(S', \exists x A'(x)))\sigma$ is satisfiable. $\mathcal{C}\sigma$ is satisfiable by (v) above, so it remains to check that p does not occur in $(S', \exists x A'(x))\sigma$. If p occurred in $(S', \exists x A'(x))\sigma$, then by (ii) above p would also occur in $S, \exists x A(x) \mid \mathcal{T}$, but this is impossible because of the parameter condition in (3).

Now we can prove soundness and completeness of the free-variable calculi.

1. Suppose S has a refutation in the sequent calculus for $\mathcal{L}$ with global assumptions Ψ. Then by Theorem 15 there exists a derivation of # from S in the tableau calculus for $\mathcal{L}$ with the global assumptions Ψ. Hence, by Lemma 19 there exists a derivation of some generalization of # from $S \cdot \top$ in the free-variable tableau calculus for $\mathcal{L}$. But any generalization of # has the form $\# \cdot \mathcal{C}$ for a satisfiable $\mathcal{C}$.

2. Suppose there exists a derivation of $\# \cdot \mathcal{C}$ from $S \cdot \top$ in the tableau calculus for $\mathcal{L}$ with the global assumptions Ψ such that $\mathcal{C}$ is satisfiable. By Lemma 18 any instance of $\# \cdot \mathcal{C}$ has a derivation from S in the tableau calculus for $\mathcal{L}$ with the global assumptions Ψ. Obviously, $\#$ is such an instance, so it is derivable from S as well. By Theorem 15 S has a refutation in the sequent calculus for $\mathcal{L}$ with the global assumptions Ψ.

7 Example Refutation

Consider the following formula valid in term-modal K:

$$\forall x \exists y([y]P(y,y) \wedge [f(y)](P(f(y),f(y)) \supset P(y,f(y))) \supset [f(x)]P(x,f(x))).$$

For better readability, we will omit parenthesis in terms like $f(x)$ and write fx instead. We will establish the validity of this formula, i.e. unsatisfiability of its negation using the free-variable tableau calculus for K. First, we negate the formula and transform it into negation normal form:

$$\exists x \forall y([y]P(y,y) \wedge [fy](\neg P(fy,fy) \vee P(y,fy)) \wedge \langle fx \rangle \neg P(x,fx)),$$

and then show its refutation. The refutation is given in Figure 4. In the refutation we do not show the constraint, since it always has the form $p \notin \emptyset$ and is satisfiable. For better readability, we denote the inference steps by $\rightarrow$ followed by the name of the inference rule. We also group several similar inferences into one. For example, by $|\forall|^*$ we denoted a sequence of $|\forall|$ inferences, and by $|\wedge|^*$ a sequence of $|\wedge|$ inferences.

References

1. E. Eder. Properties of substitutions and unifications. *Journal of Symbolic Computations*, 1(1):31–48, 1985.
2. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, 1995.
3. M. Fitting. *Proof methods for modal and intuitionistic logics*, volume 169 of *Synthese Library*. Reidel Publ. Comp., 1983.
4. M. Fitting. First-order modal tableaux. *Journal of Automated Reasoning*, 4:191–213, 1988.
5. M. Fitting, L. Thalmann, and A. Voronkov. Term-modal logics. Technical Report UMCS-2000-6-4, Department of Computer Science, University of Manchester, January 2000.
6. J.W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenther, editors, *Handbook in Philosophical Logic*, volume II, chapter II.5, pages 249–307. D. Reidel Publishing Company, 1984.
7. G. Gentzen. Untersuchungen über das logische Schließen. *Mathematical Zeitschrift*, 39:176–210, 405–431, 1934. Translated as [8].
8. G. Gentzen. Investigations into logical deduction. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North Holland, Amsterdam, 1969. Originally appeared as [7].

$$\exists x \forall y([y]P(y,y) \wedge [fy](\neg P(fy,fy) \vee P(y,fy)) \wedge \langle fx\rangle \neg P(x,fx)) \rightarrow \quad |\exists|$$

$$\forall y([y]P(y,y) \wedge [fy](\neg P(fy,fy) \vee P(y,fy)) \wedge \langle fp\rangle \neg P(p,fp)) \rightarrow \quad |\forall|^*$$

$$\begin{array}{l}\forall y([y]P(y,y) \wedge [fy](\neg P(fy,fy) \vee P(y,fy)) \wedge \langle fp\rangle \neg P(p,fp)),\\ [z]P(z,z) \wedge [fz](\neg P(fz,fz) \vee P(z,fz)) \wedge \langle fp\rangle \neg P(p,fp),\\ [u]P(u,u) \wedge [fu](\neg P(fu,fu) \vee P(u,fu)) \wedge \langle fp\rangle \neg P(p,fp) \rightarrow \quad |\wedge|^*\end{array}$$

$$\begin{array}{l}\forall y([y]P(y,y) \wedge [fy](\neg P(fy,fy) \vee P(y,fy)) \wedge \langle fp\rangle \neg P(p,fp)),\\ [z]P(z,z), \quad [fz](\neg P(fz,fz) \vee P(z,fz)), \quad \langle fp\rangle \neg P(p,fp),\\ [u]P(u,u), \quad [fu](\neg P(fu,fu) \vee P(u,fu)) \rightarrow \quad |\langle fp,z,fu\rangle|\end{array}$$

$$P(fp,fp), \quad \neg P(p,fp), \quad \neg P(fp,fp) \vee P(p,fp) \rightarrow \quad |\vee|$$

$$\begin{array}{l}P(fp,fp), \neg P(p,fp), \neg P(fp,fp) \ \ |\\ P(fp,fp), \neg P(p,fp), P(p,fp) \rightarrow \quad |\mathrm{ax}|^*\end{array}$$

$$\#$$

Fig. 4. Example refutation in the free-variable calculus

9. A.J. Grove. Naming and identity in epistemic logics part II: A first-order logic for naming. *Artificial Intelligence*, 74:311–350, 1995.
10. A.J. Grove and J.Y. Halpern. Naming and identity in a multi-agent epistemic logic. In J. Allen, R. Fikes, and E. Sandewall, editors, *KR'91. Proc. of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 301–312, Cambridge, Massachusets, April 1991. Morgan Kaufmann.
11. J. Hintikka. *Knowledge and Belief.* Cornell University Press, Ithaca, New York, 1962.
12. K. Schütte. *Beweistheorie (in German).* Springer Verlag, 1960.
13. R.M. Smullyan. A unifying principle in quantification theory. In *Proc. Nat. Acad. Sci. U.S.A.*, volume 49, pages 828–832, 1963.
14. A. Voronkov. Proof search in intuitionistic logic based on constraint satisfaction. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods. 5th International Workshop, TABLEAUX '96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 312–329, Terrasini, Palermo Italy, May 1996.
15. L.A. Wallen. *Automated Deduction in Nonclassical Logics.* The MIT Press, 1990.

A Subset-Matching Size-Bounded Cache for Satisfiability in Modal Logics

Enrico Giunchiglia and Armando Tacchella

DIST, Università di Genova, Viale Causa, 13 – 16145 Genova, Italy
{enrico,tac}@dist.unige.it

Abstract. We present a data type —that we call "bit matrix"— for caching the (in)consistency of sets of formulas. Bit matrices have three distinguishing features: (*i*) they can be queried for subsets and supersets; (*ii*) they can be bounded in size; and (*iii*) if bounded, the latest obtained (in)consistency results can be kept. We have implemented a caching mechanism based on bit matrices in *SAT. Experimenting with the TANCS 2000 benchmarks for modal logic K, we show that bit matrices (*i*) allow for considerable speedups, and (*ii*) lead to better performances both in space and time than the natural alternative, i.e., hash tables.

1 Introduction

The implementation of efficient decision procedures for modal logics is a major research problem in automated deduction. Several systems have been developed, like KRIS [1,2], KSAT [3,4], LWB [5], FACT [6], and more recently *SAT [7,8], KK [9], DLP [10], HAM-ALC [11], KTSEQC [12], and MSPASS [13]. A competition is run in conjunction with the Tableaux conferences, aiming at stimulating the development of effective systems. At the last competition, the last four of the above cited systems participated, and —as the organizer Fabio Massacci of the competition writes [14]

> The "winner" is undoubtedly the DLP prover by Peter Patel-Schneider.

As reported in [15], one of the reasons for DLP effectiveness is its ability to cache —for later reuse— the result of intermediate consistency checks of sets of formulas. The DLP caching mechanism is based on hash tables.

In this paper, we present a data type —that we call "bit matrix"— for caching intermediate consistency checks of sets of formulas. A caching mechanism based on bit matrices has been implemented in *SAT. Bit matrices have three distinguishing features:

1. they can be queried for subsets and supersets;
2. they can be bounded in size; and
3. if bounded, then the latest obtained (in)consistency results can be kept.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 237–251, 2000.

The first feature is important because it allows, e.g., to determine the consistency of any subset of a set of formulas whose consistency has been cached. The second feature allows for limiting (either before or during the execution of the algorithm) the dimension of the data structure. This is important because in many modal logics (e.g. K) the number of consistency checks (and of results to store) can be exponential in the dimension of the input formula. The third feature is important because —given that *SAT search algorithm is based on chronological backtracking— we expect the oldest results to the least useful in the future.

Experimenting with the TANCS 2000 benchmarks for modal logic K, we show that

- caching yields considerable speedups, and
- bit matrices allow for less consistency checks and better timings than hash tables.

The paper is structured as follows. In Section 2, we review some notions that are necessary to comprehend the rest of the paper. In particular —besides some basic logic notions— we review the structure of the decision procedure for K implemented in *SAT. In Section 3, we discuss some design issues behind caching for modal logics, and for K in particular. We also show how *SAT decision procedure for K has been modified in order to implement caching. In Section 4, we present bit matrices and how they meet the issues raised in the preceding Section. Section 5 describes how caching mechanisms based on bit matrices and hash tables have been implemented in *SAT. Then, the results of the experimental analysis are reported. We end the paper in Section 6 with some conclusions.

2 Basics

2.1 Definitions and Notation

The set of formulas is constructed starting from a given set of propositional letters and applying the 0-ary operators $\top$ and $\bot$ (representing truth and falsity respectively); the unary operators $\neg$ and $\Box$; and the binary operators $\wedge$, $\vee$, $\supset$ and $\equiv$.

The *modal logic* K is the smallest set of formulas (called *theorems*) closed under tautological consequence and the rule

$$\frac{(\varphi_1 \wedge \ldots \wedge \varphi_n) \supset \psi}{(\Box\varphi_1 \wedge \ldots \wedge \Box\varphi_n) \supset \Box\psi}\, RK$$

with $n \geq 0$ [16].

We say that a conjunction μ of propositional literals and formulas of the form $\Box\varphi$ or $\neg\Box\varphi$ is an *assignment* if, for any pair ψ, ψ' of conjuncts in μ, it is not the case that $\psi = \neg\psi'$. An assignment μ *satisfies* a formula φ if μ entails φ by propositional reasoning. A formula φ is *consistent* or *satisfiable* if $\neg\varphi$ is not a theorem, i.e., if $\neg\varphi \notin$ K.

```
function KSAT(φ)
  return KSAT_DPL(cnf(φ), ⊤).

function KSAT_DPL(φ, μ)
  if φ = {} then return KCONSIST(μ);                    /* base */
  elsif {} ∈ φ then return False;                       /* backtrack */
  elsif { a unit clause {l} belongs to φ } then         /* unit propagation */
    return KSAT_DPL(assign(l, φ),μ ∧ l);
  elsif not KCONSIST(μ) then return False;              /* early pruning */
  else
    l := choose-literal(φ, μ);                          /* split */
    return KSAT_DPL(assign(l, φ),μ ∧ l) or
           KSAT_DPL(assign(l̄, φ),μ ∧ l̄).
```

Fig. 1. KSAT and KSAT_{DPL}

As notational conventions, we use variously decorated lowercase and uppercase greek letters (e.g., $\varphi, \alpha, \alpha_1, \Delta, \Gamma, \Gamma'$) to denote formulas and sets of formulas respectively. Given a formula φ,

- ψ is a *modal subformula* of φ if $\Box\psi$ occurs in φ,
- $d(\varphi)$ is the *modal depth* (or simply *depth*) of φ, i.e., the maximum number of nested $\Box$ operators occurring in φ, and
- the *size* of φ is the number of distinct subformulas.

2.2 DPL-Based Algorithm for Modal Satisfiability

Consider a formula φ.

Following the SAT-based approach (see, e.g., [3,8]) the problem of determining whether φ is consistent is decomposed in two steps:

- *generate* a set of assignments, each satisfying φ, and
- *test* whether at least one of the generated assignment is consistent.

Testing the consistency of an assignment μ amounts to determining the consistency of other formulas whose depth is strictly smaller than $d(\mu)$. This implies that we can check the consistency of these other formulas by recursively applying the above methodology, at the same time ensuring the termination of the overall process. From an implementation point of view, this amounts to two mutually recursive procedures:

- KSAT(φ) for the generation of assignments satisfying φ, and
- KCONSIST(μ) for testing the consistency of each generated assignment μ.

Although the generation of assignments satisfying φ can be based on any procedure for SAT (see [8]), the KSAT procedure implemented in *SAT (represented in Figure 1) is based on the Davis-Putnam-Longemann-Loveland (DPL) procedure [17]. In Figure 1:

function KCONSIST(μ)
  $\Delta := \{\alpha \mid \Box\alpha$ is a conjunct of $\mu\}$;
  $\Gamma := \{\beta \mid \neg\Box\beta$ is a conjunct of $\mu\}$;
  foreach $\beta \in \Gamma$ **do**
    if not KSAT($\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta$) **then return** *False*
  return *True*.

Fig. 2. KCONSIST

- *cnf*(φ) is the set of clauses obtained from φ by applying Plaisted and Greenbaum's conversion [18].
- *choose-literal*(φ, μ) returns a literal occurring in φ and chosen according to some heuristic criterion.
- if l is a literal, $\overline{l}$ stands for A if $l = \neg A$, and for $\neg A$ if $l = A$;
- for any literal l and formula φ, *assign*(l, φ) is obtained from φ by
  - deleting the clauses in which l occurs as a disjunct, and
  - eliminating $\overline{l}$ from the others.

As can be observed, the procedure KSAT_{DPL} in Figure 1 is the DPL-procedure modulo

- the call to KCONSIST(μ) when it finds an assignment μ satisfying the input formula ($\varphi = \{\}$), and
- the *early pruning* step, i.e., a call to KCONSIST(μ) that forces backtracking after each unit propagation when incomplete assignments are not consistent.

The procedure KSAT is correct and complete, i.e., it returns *True* if the input formula is consistent, and *False* otherwise. See [8].

3 Caching in Modal Logics

Consider the procedure KCONSIST in Figure 2. Given two assignments μ and μ', it may be the case that KCONSIST(μ) and KCONSIST(μ') perform some equal subtests, i.e., recursive calls to KSAT. This is the case, e.g., when μ and μ' differ only for the propositional conjuncts and there is at least one conjunct of the form $\neg\Box\beta$. To prevent recomputation, the standard solution is to store both the formula whose consistency is being checked and the result of the check. Then, the cache is consulted before performing each subtest, to determine whether its result can be assessed on the basis of the formulas already checked and stored.

In the following, we assume to have two different caching mechanisms, each using a separate caching structure:

- S-cache to store and query about consistent formulas, and
- U-cache to store and query about inconsistent formulas.

In this way storing a subtest amounts to storing the formula in the appropriate caching structure. Of course, the issue is how to implement effective caching mechanisms enabling to reduce the number of subtests as much as possible. To this extent, the following considerations are in order:

1. if a formula $\bigwedge_{\alpha\in\Delta'} \alpha\wedge\neg\beta$ has already been determined to be consistent then, if $\Delta \subseteq \Delta'$, we can conclude that also $\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta$ is consistent, and
2. if a formula $\bigwedge_{\alpha\in\Delta'} \alpha \wedge \neg\beta$ has already been determined to be inconsistent then, if $\Delta \supseteq \Delta'$, we can conclude that also $\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta$ is inconsistent.

The above observations suggest caching mechanisms which can store sets of formulas and can be efficiently queried about subsets or supersets. In other words, given a subtest

$$\textsc{Ksat}(\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta), \tag{1}$$

we want to be able to query our S-cache about the presence of a formula

$$\bigwedge_{\alpha\in\Delta'} \alpha \wedge \neg\beta \tag{2}$$

with $\Delta \subseteq \Delta'$ (query for subsets or *subset-matching*). Analogously, given the subtest (1), we want to be able to query our U-cache about the presence of a formula (2) with $\Delta \supseteq \Delta'$ (query for supersets or *superset-matching*). In this way, caching a subtest avoids the recomputation of the very same subtest, *and* of the possibly many "subsumed" subtests.

Observations 1 and 2 are independent of the particular modal logic being considered. They are to be taken into account when designing caching structures for satisfiability in any modal logic. Of course, depending on the particular modal logic considered, some other considerations might be in order. For example, in K, we observe that in KCONSIST there is a natural unbalance between satisfiable subtests and unsatisfiable ones. In fact, with reference to Figure 2, when testing an assignment μ

3. many subtests can be determined to be satisfiable, all sharing the same set Δ, and
4. at most one subtest may turn out to be unsatisfiable.

Observation 3 suggests that S-cache should be able to store satisfiable subtests sharing a common set Δ in a compact way. Therefore, S-cache associates the set Δ to the set $\Gamma' \subseteq \Gamma$, representing the "computed" satisfiable subtests $\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta$ for each $\beta \in \Gamma'$. Observation 4 suggests that U-cache should not care about subtests sharing a common Δ. Therefore, U-cache associates Δ to the single β for which the subtest $\bigwedge_{\alpha\in\Delta} \alpha \wedge \neg\beta$ failed.

Given the design issues outlined above, we modified KCONSIST to yield the procedure $\textsc{Kconsist}_{\textsc{C}}$ shown in Figure 3. In the Figure:

- U-cache_get(Δ, Γ) returns *True* if U-cache contains a set Δ' such that $\Delta \supseteq \Delta'$, Δ' is associated with β and $\beta \in \Gamma$;

```
function KCONSIST_C(μ)
  Δ := {α | □α is a conjunct of μ};
  Γ := {β | ¬□β is a conjunct of μ};
  if U-cache_get(Δ, Γ) return False;
  Γ_r := S-cache_get(Δ, Γ);
  Γ_s := ∅;
  foreach β ∈ Γ_r do
    if not KSAT(⋀_{α∈Δ} α ∧ ¬β) then
      if Γ_s ≠ ∅ then S-cache_store(Δ, Γ_s);
      U-cache_store(Δ, β);
      return False
    else Γ_s := Γ_s ∪ {β};
  if Γ_s ≠ ∅ then S-cache_store(Δ, Γ_s);
  return True.
```

Fig. 3. KCONSIST$_C$: consistency checking for K with caching

- S-cache_get(Δ, Γ) returns the set $\Gamma \setminus \Gamma'$ where Γ' is the union over all the sets Γ'' such that for some set $\Delta' \supseteq \Delta$, Γ'' is associated to Δ' in S-cache.
- U-cache_store(Δ, β) stores in U-cache the set Δ and associates β to it;
- S-cache_store(Δ, Γ) stores in S-cache the set Δ and associates to it the set Γ.

The new issue is now to implement effective data structures for S-cache and U-cache supporting the above functions. Clearly, we expect that the computational costs associated to the above functions will be superior to the computational costs associated to other caching structures designed for "equality-matching", i.e., effectively supporting the functions obtained from the above by substituting "$\supseteq$" with "=". There is indeed a trade-off between "smart but expensive" and "simple but efficient" data-structures for caching. Of course, depending on

- the particular logic being considered, and
- the characteristics of the particular formula being tested,

we expect that one caching mechanism will lead to a faster decision process than the others.

Independently from the data-structure being used, the following (last) observation needs to be taken into account when dealing with modal logics whose decision problem is not in NP (e.g. K, S4):

5. testing the consistency of a formula may require an exponential number of subtests.

This is the case for the Halpern and Moses formulas presented in [19] for various modal logics. Observation 5 suggests that it may be necessary to bound the size of the cache, and introduce mechanisms for deciding which formulas to discard when the bound is reached.

		1	$\hat{2}$	3	$\cdots$	32
	ψ_1	1	0	0	$\cdots$	0
	ψ_2	0	0	0	$\cdots$	0
	ψ_3	1	0	0	$\cdots$	0
(i)	ψ_4	0	0	0	$\cdots$	0
	ψ_5	0	0	0	$\cdots$	0
	ψ_6	1	0	0	$\cdots$	0
	ψ_7	0	0	0	$\cdots$	0
	ψ_8	1	0	0	$\cdots$	0

		1	2	$\hat{3}$	$\cdots$	32
	ψ_1	1	1	0	$\cdots$	0
	ψ_2	0	1	0	$\cdots$	0
	ψ_3	1	0	0	$\cdots$	0
(ii)	ψ_4	0	0	0	$\cdots$	0
	ψ_5	0	0	0	$\cdots$	0
	ψ_6	1	1	0	$\cdots$	0
	ψ_7	0	0	0	$\cdots$	0
	ψ_8	1	0	0	$\cdots$	0

Fig. 4. Bit matrix B with $N = 8$, (i) before and (ii) after storing the set $\psi_1 \wedge \psi_2 \wedge \psi_6$ ("$\hat{}$" denotes the current column).

4 Bit Matrices for Modal Satisfiability

4.1 Bit Matrices

A *vector* is a sequence of elements of the same kind. An *N-vector* is a vector with N elements. In the following, latin letters A,B,C ... are used to denote vectors and, given an N-vector A and a number $i \leq N$, we write A[i] to denote its i-th element.

A $N \times W$ *bit matrix* is an N-vector B, where each B[i] is a bit vector of length W (i.e., is a vector of W bits). Given a $N \times W$ bit matrix A, we write A[i][k] (with $i \leq N$ and $k \leq W$) to denote the k-th bit (k-th column) of the i-th vector (i-th row). Intuitively, N is the cardinality of the set from where the objects are extracted, and W (called the *window size* of B) is the number of subsets that can be stored. W can be either fixed or dynamically increased, allowing different types of storage policies.

To understand storage and retrieval algorithms for bit matrices, consider a finite set Σ of formulas. Let $\psi_1, \psi_2, \ldots \psi_N$ be the elements of Σ, listed according to a fixed enumeration (thus, the cardinality of Σ is $|\Sigma| = N$). Then, we have an $N \times W$ bit matrix B. We say that a bit B[i][k] with $i \leq N$ and $k \leq W$ is *set* when B[i][k]$= 1$ and that it is *clear* otherwise. The *current column* $\hat{k}$ tells us where the incoming set is going to be stored. Initially, $\hat{k} = 1$ and each B[i][k] is clear.

Storing a set Δ: Given that a $N \times W$ bit matrix can contain up to W subsets, there are two cases depending on whether the matrix is full and new subtests need to be stored, or not. In order to distinguish these two cases, we use a boolean variable *full*. If *full* is false, then we set each bit B[i][$\hat{k}$] if and only if $\psi_i \in \Delta$. Then, $\hat{k}$ is incremented and, if $\hat{k} = W + 1$, *full* is set to true. If *full* is true then, we can

- either dynamically allocate more space, and allow more sets to be cached without overwriting previous data. In this case, we set *full* to false, $\hat{k}$ to $W + 1$ and W to the new value (determined by the dynamically allocated space). Then we apply the procedure above;

– or overwrite previously cached data with some policies, generally depending on the contents of B. In other words, we simply fix $\hat{k}$ to be $f(B)$, where f is a given function that decides which sets are to be overwritten. Then, we set each bit B$[i][\hat{k}]$ such that $\psi_i \in \Delta$, and clear the others.

For example, assume that Σ consists of 8 formulas $\psi_1, \psi_2, \ldots \psi_8$ listed according to a fixed enumeration. Intuitively, we consider an 8×32 bit matrix B, in which each $B[i]$ corresponds to ψ_i. Figure 4 - (i), (in which each row index i has been replaced by the corresponding formula ψ_i) shows the case in which the set $\{\psi_1, \psi_3, \psi_6, \psi_8\}$ is already stored and $\hat{k} = 2$. Assume that we are to store $\{\psi_1, \psi_2, \psi_6\}$ in B. As depicted in Figure 4 - (ii) we consider the current column ($\hat{k} = 2$) and we set the bits B[1][2], B[2][2] and B[6][2]. The current column becomes $\hat{k} + 1$.

Subset matching for a set Δ: For the retrieval, we first define the intersection between two bit vectors A and B as the bit vector C = (A ∩ B) such that C$[i]$ is set if and only if both A$[i]$ and B$[i]$ are set. To see if there exists a set Δ' such that $\Delta \subseteq \Delta'$ is stored in B, it is sufficient to consider the conjunction C of all the bit vectors B$[i]$ such that $\psi_i \in \Delta$, i.e., $C = \bigcap_{\psi_i \in \Delta}$ B$[i]$. C is a bit vector and, if at least one of its elements is set, then $\Delta \subseteq \Delta'$ for some set Δ' previously stored. We call this a *hit*. Otherwise, if every element of C is clear we have a *miss*. Notice how bit matrices naturally implement subset-matching. This capability comes at no additional cost since it is an intrinsic property of the retrieval algorithm. As an example of hit, considering B in Figure 4 - (ii) and $\Delta = \{\psi_1, \psi_6\}$, we have that C = (B[1] ∩ B[6]). Since C[1] and C[2] are set, we conclude that B contains two supersets of Δ and this is clearly the case in Figure 4 - (ii).

Superset matching for a set Δ: Superset-matching can be reduced to subset-matching for Δ by exploiting the following set-theoretic property: given a finite set Σ and two sets $\Delta, \Delta' \subseteq \Sigma$ we have that $\Delta \supseteq \Delta'$ if and only if $(\Sigma \setminus \Delta) \subseteq (\Sigma \setminus \Delta')$.

In the following, we assume that the routine KCONSIST$_C$ is used in KSAT to establish consistency of assignments and we describe how to use bit matrices for S-cache and U-cache respectively.

4.2 Bit Matrices for S-Cache

Consider a formula φ. Let $\psi_1, \ldots, \psi_N$ be a listing of the modal subformulas of φ. Let μ be an assignment satisfying φ. (We assume that each conjunct in μ is a subformula of φ.) Δ, Γ and Γ_r have the same meaning as in Figure 3. We write α_i (resp. β_j) for ψ_i (resp. ψ_j).

In order to take into account observation 3, S-cache can be implemented as a bit matrix B with $2N$ rows. Every column of B encodes the satisfiable subtests resulting from a single call to KCONSIST$_C$: given a column k and $i \leq N$, B$[i][k]$ is set if and only if $\alpha_i \in \Delta$, while B$[N + j][k]$ is set if and only if $\beta_j \in \Gamma_r$ and

the subtest $\bigwedge_{\alpha_i \in \Delta} \alpha_i \wedge \neg\beta_j$ is known to be satisfiable. Rows in the range $[1, N]$ encode members of Δ, while rows in the range $[N+1, 2N]$ encode members of Γ_r.

The bit matrix based algorithm for S-cache_get works as follows:

1. the conjunction C of all the bit vectors B[i] such that $\alpha_i \in \Delta$ is computed, i.e. C= $\bigcap_{\alpha_i \in \Delta}$ B[i];
2. initialize $\Gamma_r := \Gamma$;
3. for each $\beta_j \in \Gamma$ and for each k such that C[k] is set, test if B[$N+j$][k] is set and, if so, $\Gamma_r := \Gamma_r \setminus \{\beta_j\}$;
4. return Γ_r.

For S-cache_store we have:

1. take the current column $\hat{k}$ and clear each bit B[i][$\hat{k}$] for $i \leq 2N$;
2. set each bit B[i][$\hat{k}$] such that $\alpha_i \in \Delta$:
3. while Δ does not change, i.e., while in the same KCONSIST$_C$ call, set B[$N+j$][$\hat{k}$] whenever $\bigwedge_{\alpha_i \in \Delta} \wedge \neg\beta_j$ is satisfiable.

The time required to cache *all* the satisfiable subtests of a given assignment μ is proportional to the number of modal subformulas in μ.

4.3 Bit Matrices for U-Cache

Consider the hypotheses written in the first paragraph of the preceding subsection.

In order to take into account observation 4, U-cache can be implemented using a bit matrix B of N rows, and and a vector D of W elements. Every column of B encodes part of an unsatisfiable subtests resulting from a call to KCONSIST$_C$: given a column k and $i \leq N$, B[i][k] is set if and only if $\alpha_i \in \Delta$. The index j such that $\bigwedge_{\alpha_i \in \Delta} \supset \beta_j$, is stored in D[$k$]. As before, rows in the range $[1, N]$ encode members of Δ, but we need a single number to remember β_j.

The bit matrix based algorithm for U-cache_get works as follows:

1. the conjunction C of all the bit vectors B[i] such that $\alpha_i \notin \Delta$ is computed, C= $\bigcap_{\alpha_i \notin \Delta}$ B[i];
2. for each $\beta_j \in \Gamma$ and for each k such that C[k] is set, test if D[k]= j and, if so, return *True*;
3. return *False*;

For U-cache_store we have:

1. take the current column $\hat{k}$ ($k < W$) and for $i \leq N$ set each bit B[i][$\hat{k}$];
2. clear each bitB[i][$\hat{k}$] such that $\alpha_i \in \Delta$;
3. D[k] := j since $\bigwedge_{\alpha_i \in \Delta} \supset \beta_j$.

The time required to cache an unsatisfiable subtest from a given assignment μ is proportional to the number of modal subformulas *not appearing* in μ.

5 Implementation and Experimental Analysis

5.1 Caching in *SAT

We implemented in our system *SAT the data structures and algorithms presented in sections 3 and 4. *SAT is a platform for building and experimenting SAT-based decision procedures for modal logics (see [7] and [8]). The core of *SAT is a C implementation of the procedures KSAT and KCONSIST in Tables 1 and 2.

The current version of *SAT [1] features a caching otpimization based on the algorithm KCONSIST_C described in section 3. Bit matrices, as well as hash tables, are available to support such optimization. In the following, φ is the formula in input, and N is the number of subformulas in φ.

Caching with bit matrices in *SAT is implemented according to the data structures and the algorithms presented in section 4, with the following assumptions:

1. only a small number of subtest results is useful, and
2. latest obtained results are more useful.

Assumption 1 suggests a size-bounded cache, i.e., W is fixed to a value defined by the user. A fixed window size implies that, whenever the number of subtests exceeds W, we must resolve which of the previously stored results need to be overwritten. Assumption 2 suggests that we can tackle this situation by accessing the bit matrix with a first in - first out policy. Results are stored in chronological order and, when the window size is exceeded, the latest obtained result owerwrites the first, then the second and so on. With bit matrices, this access policy is implemented simply by incrementing $\hat{k}$ modulo W, i.e., whenever $\hat{k} > W$ the next current column is $\hat{k} := 1$. One last point to consider, is the (real) memory allocated by bit matrices in *SAT. We say that a formula ψ *occurs at level* l *in* φ if ψ is under the scope of exactly l nested boxes in φ ($l \geq 0$). Now we can state the following assumption:

3. it is unlikely that a subtest involving formulas of a given level is recomputed with the same formulas occurring at a different level.

The data structure for S-cache and U-cache in *SAT is a vector of $d(\varphi)$ bit matrices. Each one holds the results of subtests involving formulas occurring at a given level in φ.

Caching with hash tables in *SAT is implemented by replacing subset and superset-matching with equality matching in the algorithms presented in section 3. The data structure is a standard hash table: a vector H of lenght N whose elements are lists called *buckets*. A suitable function $1 \leq h(\Delta) \leq N$ is used to calculate the hash code associated to each set Δ. Two sets $\Delta \neq \Delta'$ may share

[1] You can find *SAT latest version and available documentation at:
`http://www.mrg.dist.unige.it/~tac/StarSAT.html`

the same hash code, i.e., $h(\Delta) = h(\Delta')$ and they are both stored in the bucket $H[h(\Delta)]$. In S-cache (resp. U-cache) a set Γ is associated to each Δ such that for each $\beta \in \Gamma$ the subtest $\bigwedge_{\alpha \in \Delta} \wedge \neg\beta$ is consistent (resp. inconsistent). Notice that, according to assumption 3 above, *SAT features a vector of length d of hash tables.

The selection of caching methods in *SAT includes the following options that we used in our experimental analisys:

kSAT denotes *SAT with all caching optimization turned off;
kSAT+USB denotes *SAT plus caching with bounded bit matrices: the window size is $W = 512$ unless otherwise specified, and both S-cache and U-cache are active;
kSAT+USH denotes *SAT plus caching with unbounded hash tables: as in the case of bit matrices, both S-cache and U-cache are active.

5.2 Experimental Analysis

To test the effectiveness of the caching mechanisms that we have described and implemented in *SAT, we consider the TANCS 2000 benchmarks problems, category "MODAL PSPACE Division", sub-category "Problems for Modal QBF", class "Medium/Hard".[2] These formulas are generated starting from Quantified Boolean Formulas (QBFs). QBFs are generated according to the following parameters: the number of clauses C, the alternation depth D, and the maximum number of variables allowed for each alternation V. By default, each clause consists of four literals, and care is taken in order to avoid the generation of trivial tests. Then, each QBF is translated into a modal formula using a polynomial encoding based on Ladner original translation ([20]), enhanced with some smart tricks so that no extra variable is introduced. Each formula, can be also "modalized": each propositional variable is replaced with a special modal formula with only one propositional variable. See [21] for more details. We will refer to the formulas resulting after Ladner encoding as "unbounded QBFs", and to their modalized version as "unbounded modalized QBFs".

Table 1 shows the results for kSAT, kSAT+USB, and kSAT+USH on a collection of "unbounded QBFs" (top table) and "unbounded modalized QBFs" (bottom table) benchmarks. For each selected value of C,V, and D the four samples found in the TANCS2000 repository were run and we report the geometric mean of the size (column "Sz"), the depth (column "d"), and the number of modal subformulas ("Ms") of the samples. For practical reasons, a time limit of 1200 seconds of CPU time is enforced for each run. For the systems, we report the number of failures ("F"), i.e. the number of samples for which either the time limit was exceeded or the physical memory was exhausted by *SAT.[3] The average

[2] They can be retrieved at `ftp://ftp.dis.uniroma1.it/pub/tancs/problems/qbf-cnf-tancs.tar.gz`.

[3] All the tests have been run on PCs `PII350MHz` with `256 MbRAM`, running `Linux SUSE 6.2`.

Table 1. kSAT, kSAT+USB, and kSAT+USH performances

						kSAT			kSAT+USB			kSAT+USH		
C	V	D	Sz	d	Ms	F	Cpu	Cons	F	Cpu	Cons	F	Cpu	Cons
10	4	4	7202	20	1013	4	–	–	0	102.05	26663	0	133.82	41898
20	4	4	8471	20	1259	4	–	–	0	321.15	75967	0	493.66	131996
30	4	4	9289	20	1418	1	87.50	53472	0	18.08	3953	0	29.36	8253
10	8	4	31281	40	2728	4	–	–	4	–	–	4	–	–
20	8	4	247845	80	11275	4	–	–	4	–	–	4	–	–
30	8	4	308422	80	14553	4	–	–	4	–	–	4	–	–
10	4	6	15698	28	1717	4	–	–	3	981.05	154892	4	–	–
20	4	6	19395	28	2284	4	–	–	4	–	–	4	–	–
10	4	4	20452	40	973	4	–	–	0	30.20	20398	0	101.46	98330
20	4	4	27137	41	1311	4	–	–	0	28.99	12796	0	176.98	134319
30	4	4	30140	41	1479	4	–	–	0	48.49	28928	1	338.78	197789
10	8	4	459144	159	7056	4	–	–	1	704.37	128334	4	–	–
20	8	4	95742	79	2761	4	–	–	4	–	–	4	–	–
30	8	4	140703	81	4138	4	–	–	3	727.93	96175	4	–	–
10	4	6	46054	57	1773	4	–	–	1	64.98	17100	0	155.37	140415
20	4	6	59526	57	2312	4	–	–	3	295.58	213999	3	875.62	664688

CPU time ("Cpu") in seconds, and the average number of consistency checks ("Cons"), are calculated on the samples that did not fail. When the system exceeds the resources on all the samples, the entry "–" replaces the average CPU time and number of consistency checks.

As it can be seen, caching produces dramatic speedups. Many tests that are not solved within the time limit by kSAT, are solved when using a caching mechanism. For the unbounded QBFs, we have a one order of magnitude speedup. For the modalized version, we have a speedup of two orders of magnitude. Interestingly, we see that kSAT+USB performs better than kSAT+USH. The difference in performances is not dramatic. However, we recall that kSAT+USB uses a cache bounded in size, while kSAT+USH does not: kSAT+USH failed on some tests because it exhausted the physical memory of the computer. Also notice that kSAT performs always many more subtests than kSAT+USH: this means that some equal subtests are indeed repeated by kSAT. Moreover, kSAT+USH performs more subtest than kSAT+USB, even if kSAT+USB uses a size bounded cache. This means that a subset-matching and superset-matching cache is indeed a winning bet at least in these tests which involve pretty big formulas, both in terms of depth and size.

To evaluate whether it is the case that only a small number of results is useful, and the latest obtained results are the most important, we run kSAT+USB on previous benchmarks, but with different window sizes. Table 2 shows the results. As it can be observed, on the first test, having $W = 32$ leads to the best computation times. Indeed, by incrementing the window size, the number of performed subtests decreases. However, the time saved does not compensate

Table 2. kSAT+USB performances when varying the window size

			W=32			W=256			W=512			W=1024		
C	V	D	F	Cpu	Cons	F	Cpu	Cons	F	Cpu	Cons	F	Cpu	Cons
10	4	4	0	89.85	27860	0	93.96	26663	0	102.05	26663	0	120.88	26663
20	4	4	0	352.54	98572	0	306.45	78459	0	321.15	75967	0	372.04	75081
30	4	4	0	18.25	4554	0	17.26	3970	0	18.08	3953	0	19.94	3931
10	8	4	4	–	–	4	–	–	4	–	–	4	–	–
20	8	4	4	–	–	4	–	–	4	–	–	4	–	–
30	8	4	4	–	–	4	–	–	4	–	–	4	–	–
10	4	6	3	882.31	159339	3	924.92	155948	3	981.05	154892	3	1128.69	154892
20	4	6	4	–	–	4	–	–	4	–	–	4	–	–
10	4	4	2	98.06	95035	0	32.63	24461	0	30.20	20398	0	36.00	18982
20	4	4	3	237.01	128555	0	28.76	13836	0	28.99	12796	0	33.28	12321
30	4	4	2	389.78	304689	0	57.03	40008	0	48.49	28928	0	55.70	25995
10	8	4	4	–	–	2	804.62	145312	1	704.37	128334	1	620.89	89416
20	8	4	4	–	–	4	–	–	4	–	–	4	–	–
30	8	4	4	–	–	4	–	–	3	727.93	96175	3	730.64	92482
10	4	6	4	–	–	1	60.16	17475	1	64.98	17100	1	73.64	16616
20	4	6	4	–	–	3	308.19	237896	3	295.58	213999	1	648.46	287018

the additional costs for querying a bigger cache. However, the most interesting result is that (by observing these few data), it seems that beyond a certain value for W, the number of performed subtests decreases very slowly with W. This seems to be the case for all the tests which do not exceed the available resources. Of course, this points out that

- beyond a certain value for W, furtherly increasing W will not pay off, and
- our strategy for forgetting results is good, in the sense that we are not throwing away useful data.

6 Conclusions

We have presented bit matrices. They are a simple caching structure that features subset and superset-matching. They can be bounded in size, and different policies for forgetting results can be easily implemented. Bit matrices have been implemented in *SAT and are available for experimentation. Following what has been done in DLP, we have also implemented a caching mechanism based on hash tables. Hash tables are not bounded in size. Experimenting with the TANCS 2000 benchmarks for modal K, we have showed that

- caching allows for considerable speedups, and
- bit matrices allow for less consistency checks and better timings than hash tables.

There is a huge literature focusing on data-structures for efficiently manipulating sets. Most of the these data-structures are variations of the familiar trees from standard textbooks, and it is not easy to use them for subset and superset-matching. Recently, [22] presents a data structure, called UBTree, that allows for subset and superset-matching, and study its application in IPP, a state-of-the-art planning system. Besides the different application domain, the main difference between UBTree and bit matrices is that for the first it is not clear how to implement effective strategies for storing new elements when the cache is bounded. Bit matrices can be seen as a tree in which each row corresponds to a path. The particular implementation of bit matrices makes queries particular effective.

For the lack of space we reported only a few test results: a much wider set of benchmarks was run and the results on kSAT+USB are reported in [23]. In this work, the performance is evaluated on the full set of TANCS2000 benchmarks, thus including formulas of the "Medium" and "Easy/Medium" categories (as opposed to the "Medium/Hard" category that we analyzed here). The results of this evaluation plus some unpublished preliminary results that we have on kSAT+USH essentially confirm the results that we presented. We also remark that only some of the potentials of bit matrices have been explored. For example, having a different window size at each depth might lead to better performances. Using a different strategy when forgetting subtests (e.g., before overwriting, we might look whether some cached subtest is subsumed by some other cached subtest) might improve performances. Of course, there is always a trade-off between "smart but expensive" and "simple but efficient" strategies.

References

1. B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption Algorithms for Concept Description Languages. In *Proc. 8th European Conference on Artificial Intelligence*, pages 348–353, 1990.
2. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.
3. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal K. In *Proc. CADE-96*, Lecture Notes in Artificial Intelligence, New Brunswick, NJ, USA, August 1996. Springer Verlag.
4. E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 1998.
5. A. Heuerding, G. Jager, S. Schwendimann, and M. Seyfried. The Logics Workbench LWB: A Snapshot. *Euromath Bulletin*, 2(1):177–186, 1996.
6. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

7. Armando Tacchella. *SAT system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Collected Papers from the International Description Logics Workshop (DL'99)*. CEUR, July 1999.
8. E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT-Based Decision Procedures for Classical Modal Logics. *Journal of Automated Reasoning*, 2000. To appear.
9. Andrei Voronkov. KK: a theorem prover for K. In Harald Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, volume 1632 of *LNAI*, pages 383–387, Berlin, July 7–10, 1999. Springer-Verlag.
10. P. F. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*, pages 87–89. CEUR, May 1998.
11. V. Haarslev and R. Moeller. HAM-ALC. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, C. A. Welty, and F. Sebastiani, editors, *Collected Papers from the International Description Logics Workshop (DL'98)*. CEUR, May 1998.
12. Vijay Boyapati and Rajeev Goré. KtSeqC: System description. In Neil V. Murray, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-99)*, volume 1617 of *LNAI*, pages 29–31, Berlin, June 07–11 1999. Springer.
13. U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption testing with SPASS. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Collected Papers from the International Description Logics Workshop (DL'99)*, pages 136–137. CEUR, July 1999.
14. Fabio Massacci. Results of the TANCS comparison. `http://www.dis.uniroma1.it/~tancs/results.shtml`, 1999. Accessed March 9, 2000.
15. I. Horrocks and P. F. Patel-Schneider. Advances in propositional modal satisfiability, 1999. Manuscript.
16. B. F. Chellas. *Modal Logic – an Introduction*. Cambridge University Press, 1980.
17. M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Journal of the ACM*, 5(7), 1962.
18. D.A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, 2:293–304, 1986.
19. Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
20. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comp.*, 6(3):467–480, 1977.
21. Fabio Massacci. Design and results of the Tableaux-99 Non-classical (Modal) Systems comparison. In Neil V. Murray, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-99)*, volume 1617 of *LNAI*, pages 14–18, Berlin, June 07–11 1999. Springer.
22. Jörg Hoffman and Jana Koehler. A new method to index and query sets. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 462–467, 1999.
23. Armando Tacchella. Evaluating *SAT on TANCS2000 benchmarks. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-2000)*, *LNAI*, Berlin, July 2000. Springer.

Dual Intuitionistic Logic Revisited

Rajeev Goré*

Automated Reasoning Group
Computer Sciences Laboratory
Res. Sch. of Inf. Sci. and Eng.
Institute of Advanced Studies

Formal Methods Group
Dept. of Computer Science
Fac. of Eng. and Inf. Tech.
The Faculties

Australian National University

rpg@arp.anu.edu.au http://arp.anu.edu.au/~rpg

Abstract. We unify the algebraic, relational and sequent methods used by various authors to investigate "dual intuitionistic logic". We show that restricting sequents to "singletons on the left/right" cannot capture "intuitionistic logic with dual operators", the natural hybrid logic that arises from intuitionistic and dual-intuitionistic logic. We show that a previously reported generalised display framework does deliver the required cut-free display calculus. We also pinpoint precisely the structural rule necessary to turn this display calculus into one for classical logic.

1 Introduction and Motivation

"Dual intuitionistic logic" has been investigated to varying degrees of success using algebraic, relational, axiomatic and sequent perspectives. The ones I am aware of are listed below, there may well be others. McKinsey and Tarski [14] investigate algebraic properties of "closure" or Brouwerian algebras, the algebraic duals to Heyting algebras. Curry [3] presents what he called "absolute implicational lattices" and "absolute subtractive lattices". Curry calls them both "Skolem lattices" on the grounds that the two dual operations of these lattices were introduced side-by-side by Skolem. Rauszer [15,16] uses algebraic, Hilbert-style and relational methods to study "intuitionistic logic with dual operators" where the connective of "pseudo-difference" is the dual to intuitionistic implication. Rauszer does not consider Gentzen calculi. Czermak [4] investigates "dual intuitionistic logic" by restricting Gentzen's **LK** to "singletons on the left" which is the natural dual notion to Gentzen's "singletons on the right" restriction for **LJ**. Goodman [8] uses Brouwerian algebras to investigate the "logic of contradictions". Goodman mentions that Kripke semantics also exist in which "any formula, once false, remains false", but he annoyingly fails to give the crucial clause for satisfiability for his "pseudo-difference" connective. He also gives a "sequentcalculus" for his logic but does not investigate cut-elimination at all. Urbas [17] highlights several deficiencies of Goodman's analysis and defines several Gentzen calculi with the "singletons on the left" restriction, but adds rules

* Supported by a Queen Elizabeth II Fellowship from the Australian Research Council.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 252–267, 2000.

for incorporating both implication and its dual connective. Urbas also proves cut-elimination for his systems and points out a beautiful duality.

None of these authors gives a full picture, so we first present a unified proof-theoretic picture of what is known about "dual intuitionistic logic". We view a logic as a syntax and semantics, and begin with the Kripke "dual" and algebraic "dual" approach of Rauszer for "intuitionistic logic with dual operators", dubbing this logic **BiInt** (short for Bi-Intuitionistic logic). We then consider the Gentzen-dual approach for obtaining "dual intuitionistic logic" **DualInt** by restricting **LK** to "singletons on the left". We explain the differences between the calculi of Czermak, Goodman and Urbas, and show that Urbas' $\mathbf{LDJ}^{-\prec}_{\not\supset}$ is sound and complete for **DualInt**. We argue that a good calculus should allow the formation of natural hybrid logics like **BiInt**, and show that the "singletons on the left/right" restriction *cannot* provide the hybrid calculus we seek.

We then give a cut-free display calculus δ**BiInt** for **BiInt**, obtained from the general display framework of [12], and show that δ**BiInt** does indeed capture Rauszer's "intuitionistic logic with dual operators" by proving that δ**BiInt** is sound with respect to Rauszer's relational semantics and complete with respect to Rauszer's algebraic semantics. The display calculus δ**BiInt** also captures both **Int** and **DualInt** separately.

A first attempt to understand "dual intuitionistic logic" using display calculi can be found in [9]. But the framework proposed there was not general enough to cater for substructural logics. Such a framework can be found in [12], and we dub this framework **SLD**. The current paper arose out of attempts to understand "dual intuitionistic logic" as a case-study of the generalised **SLD** framework, and to answer the question of a semantics for the display calculus of [9]. As far as I am aware, [9] and δ**BiInt** are the only cut-free sequent formulations of the logic **BiInt**, but see also Section 6.

Acknowledgements: I am grateful to Hiroakira Ono, Jean Goubault-Larrecq, and an anonymous referee for informing me of Rauszer's work, Crolard's work, and useful comments respectively.

2 Bi-Intuitionistic Logic

We use the term **Bi-Intuitionistic** logic (**BiInt** for short) to refer to a logic obtained from intuitionistic logic **Int** by the addition of a connective ($\prec$) which is "dual" to intuitionistic implication ($\rightarrow$). Most authors use ($\div$) for this connective and call it "pseudo-difference", but the left-right symmetry of the symbol ($\div$) hides the "direction" of the pseudo-difference operation. We prefer ($\prec$) which should be viewed as an arrow without a head, but with a tail. The argument at the left end is in a "positive" position, and the argument at the right end is in a "negative" position. Rauszer called this logic Heyting-Brouwer logic and gave algebraic, relational and topological semantics, as well as two different, yet dual, Hilbert-style calculi [16,15]. She did not investigate Gentzen methods. In the following subsections we first set the scene by stating Rauszer's algebraic and relational semantics for **BiInt**.

$$
\begin{array}{lll}
w \models \top & & \text{for every } w \in W \\
w \models A \wedge B & \text{if} & w \models A \ \& \ w \models B \\
w \models A \to B & \text{if} & (\forall v \geq w)(v \models A \Rightarrow v \models B) \\
w \models A \prec B & \text{if} & (\exists v \leq w)(v \models A \ \& \ v \not\models B)
\end{array}
\qquad
\begin{array}{lll}
w \models \bot & & \text{for no } w \in W \\
w \models A \vee B & \text{if} & w \models A \text{ or } w \models B \\
w \models \neg A & \text{if} & (\forall v \geq w)(v \not\models A) \\
w \models \sim A & \text{if} & (\exists v \leq w)(v \not\models A)
\end{array}
$$

Fig. 1. Rauszer's Semantics for **BiInt**

Every primitive proposition from $\mathtt{PRP} = \{p_0, p_1, p_3, \cdots\}$ is a **formula** of **BiInt**, the verum and falsum constants $\top, \bot$ are formulae of **BiInt**, and if A and B are formulae of **BiInt** then so are each of: $(A \to B)$, $(A \wedge B)$, $(A \vee B)$, and $(A \prec B)$. We can define two negations $\neg A := (A \to \bot)$ and $\sim A := (\top \prec A)$. Thus we use A, B, C to stand for formulae. An **Int**-formula is any formula built from PRP and $\top$ and $\bot$ using only $\wedge$, $\vee$, and $\to$. A **DualInt**-formula is any formula built from PRP and $\top$ and $\bot$ using only $\wedge$, $\vee$ and $\prec$.

2.1 Rauszer's Relational Semantics for BiInt

Rauszer's [16] semantics for **BiInt** extend the traditional Kripke semantics for **Int**, as Kripke semantics for tense logics extend those for modal logics. Rauszer actually considers the case of predicate-intuitionistic logic. We give a precise definition of the propositional parts to make this paper self-contained.

We use the classical first-order meta-level connectives &, "or", "not", $\Rightarrow$, $\forall$ and $\exists$ to state Rauszer's semantics. We often use the abbreviations:

$$
\begin{array}{lcl}
(\forall v \geq w)(\cdots) & := & (\forall v \in W, (w \leq v \Rightarrow (\cdots))) \\
(\forall v \leq w)(\cdots) & := & (\forall v \in W, (v \leq w \Rightarrow (\cdots))) \\
(\exists v \geq w)(\cdots) & := & (\exists v \in W, (w \leq v \ \& \ (\cdots))) \\
(\exists v \leq w)(\cdots) & := & (\exists v \in W, (v \leq w \ \& \ (\cdots)))
\end{array}
$$

A Kripke **frame** is pair $\langle W, \leq \rangle$ where W is a non-empty set (of possible worlds) and $\leq \subseteq (W \times W)$ is a reflexive and transitive binary relation over W. A Kripke **model** is a triple $\langle W, \leq, V \rangle$ where $\langle W, \leq \rangle$ is a Kripke frame and V is a mapping from $\mathtt{PRP} \cup \{\top, \bot\}$ to 2^W such that $V(\top) = W$, $V(\bot) = \emptyset$ and V obeys **persistence:** $(\forall v \geq w)[w \in V(p) \Rightarrow v \in V(p)]$.

Given a model $\langle W, \leq, V \rangle$, we say that $w \in W$ **satisfies** p if $w \in V(p)$, and write this as $w \models p$. Note that $w \models p$ if $(\forall v \geq w)(v \in V(p))$. We write $w \not\models p$ to mean $(not)(w \models p)$; that is, $(\exists v \geq w)(v \notin V(p))$. The satisfaction relation is then extended to the verum and falsum constants and compound formulae as given in Figure 1 [16].

As usual, a **BiInt**-formula [**Int**-formula, **DualInt**-formula] A is **BiInt-valid** [**Int**-valid, **DualInt**-valid] if it is satisfied by every world in every Kripke model.

Proposition 1. *For all $w \in W$, and all formulae A we can prove by induction upon the structure of A that it is impossible to have $(w \models A) \& (w \not\models A)$, that we must have $(w \models A)$ or $(w \not\models A)$, that $(not)(w \models A)$ is exactly $(w \not\models A)$, and that $(not)(w \not\models A)$ is exactly $(w \models A)$. That is, the meta-logic used to write the*

semantic clauses for satisfiability is just classical first-order logic. The crucial point is that $(w \models \neg A)$ *is* different *from* $(not)(w \models A)$.

2.2 Rauszer's Algebraic Semantics for BiInt

A **quasi-ordered set** is a pair $\langle U, \leq \rangle$ where U is a non-empty set (of points) and $\leq \subseteq (U \times U)$ is a reflexive and transitive binary relation over U.

Following Curry [3], a **lattice** $\langle U, \leq, \wedge, \vee \rangle$ is a quasi-ordered set with two (as it turns out associative, commutative and isotonic) binary operations meet $(\wedge)$ and join $(\vee)$ where, for all $a, b, c \in U$:

(1a) $a \wedge b \leq a$ (1b) $a \leq a \vee b$
(2a) $a \wedge b \leq b$ (2b) $b \leq a \vee b$
(3a) $c \leq a \;\&\; c \leq b \Rightarrow c \leq a \wedge b$ (3b) $a \leq c \;\&\; b \leq c \Rightarrow a \vee b \leq c$

Following Curry [3] but using different symbols, a **Heyting algebra** $\langle U, \leq, \wedge, \vee, \rightarrow, \mathbf{0} \rangle$ and a **Brouwerian algebra** $\langle U, \leq, \wedge, \vee, \prec, \mathbf{1} \rangle$ are lattices with extra binary operations $(\rightarrow)$ and $(\prec)$ respectively where, for all $a, b, c \in U$:

Heyting Algebra	Brouwerian Algebra
(4a) $a \wedge (a \rightarrow b) \leq b$	(4b) $a \leq b \vee (a \prec b)$
(5a) $a \wedge b \leq c \Rightarrow b \leq a \rightarrow c$	(5b) $a \leq b \vee c \Rightarrow a \prec c \leq b$
(6a) $\mathbf{0} \leq a$	(6b) $a \leq \mathbf{1}$

Following Rauszer [15,16], an abstract algebra $\langle U, \leq, \wedge, \vee, \rightarrow, \prec, \mathbf{0}, \mathbf{1} \rangle$ is an **HB-algebra** (for Heyting and Brouwer) if $\langle U, \leq, \wedge, \vee, \rightarrow, \mathbf{0} \rangle$ is a Heyting algebra and $\langle U, \leq, \wedge, \vee, \prec, \mathbf{1} \rangle$ is a Brouwerian algebra. Rauszer actually uses the name "semi-Boolean" algebras, defines $\mathbf{0} := (a \prec a)$, $\mathbf{1} := (a \rightarrow a)$, and for every $a \in U$ defines two negations $\neg a := (a \rightarrow \mathbf{0})$ and $\sim a := (\mathbf{1} \prec a)$. Rauszer also uses a symbol different from our $(\sim)$ for the latter.

Following Rauszer [16, page 9], let $\mathcal{G} = \langle G, \leq \rangle$ be a quasi-ordered set. A subset $F \subset G$ is **open** if $(\forall x, y \in G)((x \in F \;\&\; x \leq y) \Rightarrow y \in F)$. Let $\mathcal{O}(G)$ be the collection of all open subsets of $\mathcal{G}$, let $\subseteq$ be set inclusion, $\cap$ be set intersection, $\cup$ be set union, and $\emptyset$ be the empty set. Define two operations on the members of $\mathcal{O}(G)$ as below, for all $F, H \in \mathcal{O}(G)$:

(7a) $F \Rightarrow H = \{x \in G : (\forall y \geq x)(y \in F \Rightarrow y \in H)\}$
(7b) $F \nRightarrow H = \{x \in G : (\exists y \leq x)(y \in F \;\&\; y \notin H)\}$

Theorem 1 ((Rauszer)). *The algebra* $\mathfrak{O}(\mathcal{G}) = \langle \mathcal{O}(G), \subseteq, \cap, \cup, \Rightarrow, \nRightarrow, \emptyset, G \rangle$ *is a complete* **HB***-algebra [16, page 9].*

Rauszer calls a structure of the form $\mathfrak{O}(\mathcal{G})$ an **order-topology** and proves the following representation theorem for **HB**-algebras.

Theorem 2 ((Rauszer)). *For every* **HB***-algebra* $\mathfrak{U}$ *there exists an order-topology* $\mathfrak{O}(\mathcal{G})$ *and a monomorphism* h *from* $\mathfrak{U}$ *to* $\mathfrak{O}(\mathcal{G})$ *[16, page 9].*

Theorem 1 implies that every Kripke model gives rise to an **HB**-algebra, while Theorem 2 implies that every **HB**-algebra gives rise to a Kripke model. Rauszer makes the connection explicit in [16, Section 10].

Urbas' $\mathbf{LJ}^{-\subset}$ is Gentzen's **LJ** plus

$$(-\subset\vdash_{\mathbf{LJ}})\ \frac{A, \Gamma \vdash \Delta}{A \mathrel{-\!\subset} B, \Gamma \vdash \Delta} \qquad \frac{\Gamma \vdash B}{A \mathrel{-\!\subset} B, \Gamma \vdash} \qquad (\vdash_{\mathbf{LJ}} -\subset)\ \frac{\Gamma \vdash A \quad B, \Delta \vdash}{\Gamma, \Delta \vdash A \mathrel{-\!\subset} B}$$

Urbas' $\mathbf{LDJ}^{-\prec}$ is **LDJ** plus

$$(\prec \vdash)\ \frac{A \vdash \Delta, B}{A \prec B \vdash \Delta} \qquad (\vdash \prec)\ \frac{\Gamma \vdash \Theta, A \quad B \vdash \Delta}{\Gamma \vdash \Theta, \Delta, A \prec B}$$

Urbas' **LDJ**

$$\text{(id)}\ \ A \vdash A \qquad (\vdash \text{Exch})\ \frac{\Gamma \vdash \Delta, A, B, \Theta}{\Gamma \vdash \Delta, B, A, \Theta}$$

$$\text{(cut)}\ \frac{\Gamma \vdash \Delta, A \quad A \vdash \Theta}{\Gamma \vdash \Delta, \Theta} \qquad (\vdash \text{Ctr})\ \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}$$

$$(\text{Wk}\vdash)\ \frac{\vdash \Delta}{A \vdash \Delta} \qquad (\vdash \text{Wk})\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}$$

$$(\wedge\vdash)\ \frac{A \vdash \Delta}{A \wedge B \vdash \Delta} \quad \frac{B \vdash \Delta}{A \wedge B \vdash \Delta} \qquad (\vdash \wedge)\ \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}$$

$$(\vee\vdash)\ \frac{A \vdash \Delta \quad B \vdash \Delta}{A \vee B \vdash \Delta} \qquad (\vdash \vee)\ \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B} \quad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \vee B}$$

$$(\sim\vdash)\ \frac{\vdash \Delta, A}{\sim A \vdash \Delta} \qquad (\vdash\sim)\ \frac{A \vdash \Delta}{\vdash \Delta, \sim A}$$

$$(\supset\vdash)\ \frac{\vdash \Delta, A \quad B \vdash \Theta}{A \supset B \vdash \Delta, \Theta} \qquad (\vdash \supset)\ \frac{A \vdash \Delta}{\vdash \Delta, A \supset B} \quad \frac{\Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \supset B}$$

Fig. 2. Urbas' Systems for Dual Intuitionistic Logic

3 Dual Gentzen Systems

We now turn to the Gentzen-dual view of "dual intuitionistic logic" investigated by Czermak [4], Goodman [7,8] and Urbas [17] where Gentzen's **LK** is restricted to "singletons on the left", in a manner dual to the way Gentzen's **LJ** arises from **LK** using the "singletons on the right" restriction. We assume that the **LK** connectives are $(\wedge)$, $(\vee)$, $(\rightarrow)$, $(\neg)$, and that the sequent arrow is written as $(\vdash)$. In what follows, it may seem as if quite a few different languages of symbols are used interchangeably. But the notational differences have been deliberately chosen and are explained along the way.

Czermak adds the "singletons on the left" restriction to Gentzen's **LK** to obtain a "dual-intuitionistic calculus **DJ**". Czermak states that cut-elimination holds but does not give details, and makes no attempts to give rules for $(\prec)$. An important property of **Int** known as Glivenko's Theorem is that all classically

valid formulae of the form $\neg A$ are also **Int**-valid. Czermak mentions that an embedding into modal logic **S4** gives the restricted dual-Glivenko result below:

Restricted-Dual-Glivenko: each classically valid formula without implication is derivable in **DJ**.

Referring to Figure 2, let $\mathbf{LDJ}_{\not\emptyset}$ [$\mathbf{LDJ}^{\prec}_{\not\emptyset}$] be **LDJ** [$\mathbf{LDJ}^{\prec}$] minus the rules for $(\supset)$. Then Czermak's **DJ** is essentially Urbas' $\mathbf{LDJ}^{\prec}_{\not\emptyset}$, where Czermak's $(\neg) := (\sim)$. Czermak also adds rules for a universal quantifier but we ignore these aspects. Czermak gives a syntactic proof of the Restricted-Dual-Glivenko property; "Restricted" in that it does not extend to formulae containing implication since Czermak does not give "singletons on the left" rules for implication.

Goodman, apparently independently [7], also attempts a "sequentcalculus" which we dub **GDJ**, whose core rules are the same as Czermak's **DJ** and Urbas' $\mathbf{LDJ}_{\not\emptyset}$, where this time, Goodman's $(\neg) := (\sim)$. Goodman's **GDJ** also contains the following rules for $(\prec)$, which he writes as the $(\dot{-})$ symbol:

$$(\prec \vdash) \quad \frac{A \vdash \Delta, B}{A \prec B \vdash \Delta} \qquad\qquad (\prec \vdash \mathbf{Inv}) \quad \frac{A \prec B \vdash \Delta}{A \vdash \Delta, B}$$

Goodman states that **GDJ** is sound and complete w.r.t. Brouwerian algebras.

Goodman's rule $(\prec \vdash \mathbf{Inv})$ actually *eliminates* $A \prec B$ instead of introducing it. Moreover, there is no explicit rule for introducing $(\prec)$ into the succedent. Hence a formula of the form $A \prec B$ can enter the succedent of the endsequent of a derivation only via some initial sequent of the form $A \prec B \vdash A \prec B$. Urbas refers to Goodman's calculus but does not discuss this aspect. Goodman allows further initial sequents $\top \vdash C$ where $C \in \Sigma$ for some theory (set of **DualInt**-formulae) and states a general dual-Glivenko result for **GDJ** using a translation gd(.) that replaces every occurrence of $(\sim)$ with $(\neg)$ and replaces every occurrence of $(B \prec C)$ with $(B \wedge \neg C)$:

Goodman-Dual-Glivenko: if the classical logic formula $\mathrm{gd}(A)$ obtained from some **DualInt**-formula A is a classical logical consequence of a set of classical formulae $\mathrm{gd}(\Sigma)$, obtained from a set of **DualInt**-formulae Σ, then $\top \vdash A$ is derivable in **GDJ** using initial sequents $\{\ \top \vdash C \mid C \in \Sigma\}$.

Notice that the classical consequence part uses formulae translated by gd(.) whereas the **GDJ** derivation uses the original **DualInt** formulae.

It is well-known that Gentzen calculi do not obey cut-elimination when theories are involved. In any case, Goodman does not investigate cut-elimination, which is probably the reason why he can retain $(\prec \vdash \mathbf{Inv})$: this rule breaks the subformula property and will hinder cut-elimination. Goodman shows that $(A \supset B) := ((\sim A) \vee B)$ is one way of obtaining a sort of implication but points out that this implication fails the deduction theorem.

Urbas picks up from the work of Czermak and Goodman and points out that Goodman uses a verum constant $\top$, which Gentzen did not, and also points out that Czermak does not give "singletons in the antecedent" rules for implication. Urbas then rectifies these deficiencies by giving the calculus shown in Figure 2.

Urbas uses the symbol $(\neg)$ whereas we have used $(\sim)$ in defining **LDJ**. Since $\sim\sim A \vdash A$ is derivable in **LDJ**, the $(\sim)$ symbol is definitely not the negation $(\neg)$ from **Int**. In fact, it is the negation from **DualInt**. Also, as Urbas points out, the rules for $(\supset \vdash)$ and $(\vdash \supset)$ are derivable in **LDJ** by putting $(A \supset B) := ((\sim A) \vee B)$. Thus the $(\supset)$ connective of **LDJ** is definitely *not* the implication $(\rightarrow)$ from **Int**. Rauszer's semantics for this connective would be:

$$w \models A \supset B \text{ if } (\exists v \leq w)[v \not\models A] \text{ or } w \models B$$

We therefore concentrate upon $\mathbf{LDJ}_{\not\supset}^{\prec}$.

Lemma 1. *(1)* $(\top \rightarrow A)$ *is* **BiInt***-valid iff (2)* A *is* **BiInt***-valid iff (3)* $(\top \prec A)$ *is* **BiInt***-unsatisfiable.*

Proof. That (1) iff (2) is trivial. We show that (3) implies (2).

(3)	$(\forall w)[$	$w \not\models (\top \prec A)$			$]$
(4)	$(\forall w)(not)(\exists v \leq w))[$	$v \models \top$	&	$v \not\models A$	$]$
(5)	$(\forall w)(not)(\exists v \leq w))[$			$v \not\models A$	$]$
(6)	$(\forall w)(\forall v \leq w)[$			$v \models A$	$]$
(2)	$(\forall w)[$			$w \models A$	$]$

That (2) implies (6), and hence (3) is also obvious. □

Theorem 3 ((Soundness)). $\vdash A$ *derivable in* $\mathbf{LDJ}_{\not\supset}^{\prec}$ *implies* A **DualInt***-valid.*

Proof. We define a translation $\mathtt{t}(.)$ from $\mathbf{LDJ}_{\not\supset}^{\prec}$ to **DualInt**-formulae such that $\mathtt{t}(\vdash A) = (\top \prec A)$. Induction on the length of the given derivation of $\vdash A$ gives that $\top \prec A$ is **DualInt**-unsatisfiable. Lemma 1 then implies that A is **BiInt**-valid. Since $\mathbf{LDJ}_{\not\supset}^{\prec}$ can introduce only **DualInt**-formulae, A is **DualInt**-valid.

Let Γ be the list of formulae $A_1, A_2, \cdots, A_n$ with $n \geq 0$. Define $\hat{\Gamma} := (A_1 \wedge A_2 \wedge \cdots \wedge A_n)$ and $\check{\Gamma} := (A_1 \vee A_2 \vee \cdots \vee A_n)$, with $\hat{\emptyset} := (\top)$ and $\check{\emptyset} := (\bot)$ and $\mathtt{t}(\Gamma \vdash \Delta) := \hat{\Gamma} \prec \check{\Delta}$. For every rule (ρ) of $\mathbf{LDJ}_{\not\supset}^{\prec}$ we must now show that: if the $\mathtt{t}(.)$-translation of the premisses of (ρ) are **DualInt**-unsatisfiable, then so is the $\mathtt{t}(.)$-translation of the conclusion of (ρ).

We consider the case of the $(\prec \vdash)$ rule from $\mathbf{LDJ}_{\not\supset}^{\prec}$ only. The hypothesis is that $\mathtt{t}(A \vdash \Delta, B) = A \prec (\check{\Delta} \vee B)$ is **DualInt**-unsatisfiable. We have to show that $\mathtt{t}(A \prec B \vdash \Delta) = ((A \prec B) \prec \check{\Delta})$ is also **DualInt**-unsatisfiable.

Semantically, the hypothesis means that in any Kripke model we must have $(\forall w)[w \not\models A \prec (\check{\Delta} \vee B)]$. After expanding this out further we obtain $(\forall w)(\forall v \leq w)[v \not\models A$ or $v \models B$ or $v \models \check{\Delta}]$. Putting $v = w$ and rewriting gives

(a) $(\forall w)[(w \models A \ \&\ w \not\models B) \Rightarrow w \models \check{\Delta}]$

The semantic expansion of the desired conclusion is $(\forall w)[w \not\models (A \prec B) \prec \check{\Delta}]$, which expands to $(\forall w)(\forall v \leq w)[(\exists u \leq v)(u \models A \ \&\ u \not\models B) \Rightarrow v \models \check{\Delta}]$.

Choose some arbitrary Kripke model and choose arbitrary worlds $v_0 \leq w_0$. Suppose the premiss of the desired conclusion is true: that is, suppose $[(\exists u_0 \leq v_0)(u_0 \models A \ \&\ u_0 \not\models B)]$. The world u_0 must satisfy (a) so $[u_0 \models \check{\Delta}]$. Since $(u_0 \leq$

v_0), we must have $[v_0 \models \check{\Delta}]$ by persistence. Since v_0 and w_0 were arbitrary points in an arbitrary Kripke model we can generalise this to $(\forall w)(\forall v \leq w)[(\exists u \leq v)(u \models A \;\&\; u \not\models B) \Rightarrow v \models \check{\Delta}]$. □

Theorem 4 ((Completeness)). *A* **DualInt***-valid* $\Rightarrow \vdash A$ $\mathbf{LDJ}^{-\prec}_{\not\supset}$*-derivable.*

Proof. The Lindenbaum algebra formed from the equiprovable formulae of $\mathbf{LDJ}^{-\prec}_{\not\supset}$ form a Brouwerian algebra. See [11] for a similar proof. □

Corollary 1. *A formula A is* **DualInt***-valid iff* $\vdash A$ *is derivable in* $\mathbf{LDJ}^{-\prec}_{\not\supset}$.

Corollary 2. *A sequent* $\vdash A$ *without* $(\prec)$ *is derivable in* **LDJ** *iff* $A[(B \supset C) := (\sim B \vee C)]$ *is* **DualInt***-valid.*

A formula A is a **counter-theorem** of a sequent calculus [display calculus] if the sequent $A \vdash$ $[A \vdash \mathrm{I}]$ is derivable in that calculus.

Theorem 5 ((Dual-Glivenko)). *A sequent* $\vdash A$ *is derivable in* **LDJ** *iff* $\vdash A[(\sim) := (\neg)]$ *is derivable in* **LK** *[4,8,17].*

Theorem 6 ((Goodman, Urbas)). *There is no connective* $\sharp$ *definable in* **LDJ** [$\mathbf{LDJ}^{-\prec}$] *such that* $A \vdash B$ *is derivable iff* $\vdash A \sharp B$ *is derivable [8,17].*

Urbas uses $(\prec)$ where we have used $(-\subset)$ in Figure 2. Our reason is that $(-\subset)$ is really not the "pseudo-difference" operation from **DualInt**. To prove a dual version of Theorem 6 for **LJ** and $\mathbf{LJ}^{-\subset}$, we make use of the fact that $\mathbf{LJ}^{-\subset}$ is really just a definitional extension of **LJ**, and hence **LK**, where $(A - \subset B) := (A \wedge \neg B)$. We also use the following version of Glivenko's Theorem.

Theorem 7 ((Glivenko)). *The sequent* $\vdash \neg A$ *is derivable in* **LJ** [$\mathbf{LJ}^{-\subset}$] *iff it is derivable in* **LK** [*when* $(B - \subset C) := (B \wedge \neg C)$].

Theorem 8. *There is no connective* $\sharp$ *definable in* **LJ** [$\mathbf{LJ}^{-\subset}$] *such that* $A \vdash B$ *is derivable iff* $A \sharp B \vdash$ *is derivable.*

Proof. This is a slight variation of Urbas' [17] proof of Theorem 6, which itself is a variation of Goodman's [8] algebraic argument.

Suppose there were such a connective. If $A \vdash B$ is derivable in **LJ** or $\mathbf{LJ}^{-\subset}$ then it is derivable in **LK**. In the case of $\mathbf{LJ}^{-\subset}$, this is because $(A - \subset B) \dashv\vdash (A \wedge \neg B)$ in $\mathbf{LJ}^{-\subset}$. By supposition, $A \sharp B \vdash$ would then be derivable in **LJ** and $\mathbf{LJ}^{-\subset}$, and hence in **LK**. So $A \sharp B$ would be definable in **LK**. Since $A \vdash A$ is derivable in **LJ** and $\mathbf{LJ}^{-\subset}$, we would then have $A \sharp A \vdash$ derivable in **LJ** and $\mathbf{LJ}^{-\subset}$, and hence also in **LK**. But replacing any occurrence of A by $\neg\neg A$ in any counter-theorem of **LK** gives another counter-theorem of **LK**, which means that $(\neg\neg A) \sharp A \vdash$ would be derivable in **LK**. By Theorem 7, **LK**, **LJ** and $\mathbf{LJ}^{-\subset}$ share counter-theorems, so $(\neg\neg A) \sharp A \vdash$ would also be derivable in **LJ** and $\mathbf{LJ}^{-\subset}$. By supposition, this would mean that $(\neg\neg A) \vdash A$ would also be derivable in **LJ** and $\mathbf{LJ}^{-\subset}$; which we know to be false. □

In summary we have the following situation. The "singletons on the left" restriction in $\mathbf{LDJ}^{-\prec}_{\not\supset}$ does capture **DualInt** faithfully, but the connective ($\supset$) in $\mathbf{LDJ}^{-\prec}$ is just a definition $(A \supset B) := ((\sim A) \vee B)$. Indeed, by Theorem 6, it is impossible to extend $\mathbf{LDJ}^{-\prec}_{\not\supset}$ with any "implication" connective that obeys the deduction theorem. Dually, the "singletons on the right" restriction in **LJ** does capture **Int** faithfully, but the connective $(-\!\subset)$ in $\mathbf{LJ}^{-\subset}$ is just a definition $(A -\!\subset B) := (A \wedge \neg B)$. Indeed, by Theorem 8, it is impossible to extend $\mathbf{LJ}^{-\subset}$ with any "pseudo-difference" connective that obeys the dual-deduction theorem. So how can we hope to obtain a Gentzen calculus for the hybrid logic **BiInt** ?

We cannot merge $\mathbf{LJ}^{-\subset}$ and $\mathbf{LDJ}^{-\prec}_{\not\supset}$ since the two restrictions cannot be both applied simultaneously. For example, the **LJ** restriction blocks all the (Ctr) and (Wk) rules of $\mathbf{LDJ}^{-\prec}_{\not\supset}$, and vice-versa. We clearly need some other way to keep track of the notions that "only one formula is in the negative [positive] context".

Gentzen calculi for **Int** that permit multiples in the succedent do exist [5,6]. A Gentzen calculus for **DualInt** with multiples in the antecedent has been explored by Tristan Crolard [2] who gives a calculus $\mathbf{SLK^1}$ for **BiInt**, which he calls "subtractive logic". The calculus $\mathbf{SLK^1}$ puts a "singleton on the right" ["singleton on the left"] restriction on its rule for introducing implication [subtraction] into the right [left] hand side. Crolard [2, Section 4.8] states that "although we conjecture cut-elimination in **SLK**, we did not consider this issue in this paper".

Another possible way is to use two turnstiles $\vdash_{LDJ}$ and $\vdash_{LJ}$ and to nest them in some appropriate way; see [12, Section 13.3]. In the next section we present a cut-free display calculus for **BiInt** that solves this problem.

4 A Display Calculus for Bi-Intuitionistic Logic

For brevity we assume the reader is familiar with display calculi; see [18]. The display calculus δ**BiInt** we are about to define arises naturally from the generalised display calculus reported in [12]. There, the connectives $\otimes$ and $\oplus$ are used for intensional conjunction and disjunction, while $\wedge$ and $\vee$ are used for extensional conjunction and disjunction. As explained in [12], the addition of structural rules causes a gradual collapse of the intensional conjunction [disjunction] and extensional conjunction [disjunction] connectives. In the case of δ**BiInt**, $\otimes$ is indistinguishable from $\wedge$, and $\oplus$ is indistinguishable from $\vee$. So we retain the rules for $\otimes$ and $\oplus$ to help the reader switch to and from [12].

Display calculi extend the language of Gentzen's sequents by using multiple complex structural connectives rather than just Gentzen's single comma. Every formula of **BiInt** is a **structure** of δ**BiInt**, I is a (nullary) structure of δ**BiInt**, and if X and Y are structures of δ**BiInt** then so are $(X\ ;\ Y)$ and $(X < Y)$ and $(X > Y)$. Thus we use X, Y, Z, W to stand for structures.

The **sequents** of δ**BiInt** take the form $X \vdash Y$ where X and Y are structures of δ**BiInt**, with X the **antecedent** and Y the **succedent**. From δ**BiInt** sequents we build a rule of δ**BiInt** in the usual Gentzen way with its **premisses** above the line and its **conclusion** below the line; see Figure 3. The double lines in some rules indicate that the rules can also be applied in a *downward* manner to obtain

Display Postulates

$$\text{(dp)}\ \dfrac{\dfrac{\dfrac{X \vdash Z < Y}{\Longrightarrow}}{X\ ;\ Y \vdash Z}}{\dfrac{\Longrightarrow}{Y \vdash X > Z}} \qquad \text{(dp)}\ \dfrac{\dfrac{\dfrac{Z < Y \vdash X}{\Longrightarrow}}{Z \vdash X\ ;\ Y}}{\dfrac{\Longrightarrow}{X > Z \vdash Y}}$$

Cut Rule

$$\text{(cut)}\ \dfrac{X \vdash A \quad A \vdash Y}{X \vdash Y}$$

where A is a formula

Basic Structural Rules

$$\text{(Ver-I)}\ \dfrac{\mathrm{I} \vdash X}{Y \vdash X} \qquad \text{(Efq-I)}\ \dfrac{X \vdash \mathrm{I}}{X \vdash Y}$$

$$\text{(I}^{+}_{-}\vdash\text{)}\ \dfrac{\dfrac{\dfrac{X\ ;\ \mathrm{I} \vdash Y}{\Longrightarrow}}{X \vdash Y}}{\dfrac{\Longrightarrow}{\mathrm{I}\ ;\ X \vdash Y}} \qquad \text{(}\vdash \mathrm{I}^{+}_{-}\text{)}\ \dfrac{\dfrac{\dfrac{X \vdash \mathrm{I}\ ;\ Y}{\Longrightarrow}}{X \vdash Y}}{\dfrac{\Longrightarrow}{X \vdash Y\ ;\ \mathrm{I}}}$$

Further Structural Rules

$$\text{(Wk}\vdash\text{)}\ \dfrac{X \vdash Z}{X\ ;\ Y \vdash Z} \quad \dfrac{Y \vdash Z}{X\ ;\ Y \vdash Z} \qquad \text{(}\vdash \text{Wk)}\ \dfrac{Z \vdash X}{Z \vdash X\ ;\ Y} \quad \dfrac{Z \vdash Y}{Z \vdash X\ ;\ Y}$$

$$\text{(Ctr}\vdash\text{)}\ \dfrac{X\ ;\ X \vdash Z}{X \vdash Z} \qquad \text{(}\vdash \text{Ctr)}\ \dfrac{Z \vdash X\ ;\ X}{Z \vdash X}$$

$$\text{(Ass}\vdash\text{)}\dfrac{X\ ;\ (Y\ ;\ Z) \vdash W}{\Longrightarrow} \qquad \text{(}\vdash \text{Ass)}\dfrac{W \vdash (X\ ;\ Y)\ ;\ Z}{\Longrightarrow}$$

$$\text{(Com}\vdash\text{)}\ \dfrac{Y\ ;\ X \vdash Z}{X\ ;\ Y \vdash Z} \qquad \text{(}\vdash \text{Com)}\ \dfrac{Z \vdash Y\ ;\ X}{Z \vdash X\ ;\ Y}$$

Logical Introduction Rules

$$\text{(}\bot\vdash\text{)}\ \bot \vdash \mathrm{I} \qquad \text{(id)}\ p \vdash p \qquad \text{(}\vdash \bot\text{)}\ \dfrac{Z \vdash \mathrm{I}}{Z \vdash \bot}$$

$$\text{(}\top\vdash\text{)}\ \dfrac{\mathrm{I} \vdash Z}{\top \vdash Z} \qquad \text{(}\vdash \top\text{)}\ \mathrm{I} \vdash \top$$

$$\text{(}\otimes\vdash\text{)}\ \dfrac{A\ ;\ B \vdash Z}{A \otimes B \vdash Z} \qquad \text{(}\vdash \otimes\text{)}\ \dfrac{X \vdash A \quad Y \vdash B}{X\ ;\ Y \vdash A \otimes B}$$

$$\text{(}\oplus\vdash\text{)}\ \dfrac{A \vdash X \quad B \vdash Y}{A \oplus B \vdash X\ ;\ Y} \qquad \text{(}\vdash \oplus\text{)}\ \dfrac{Z \vdash A\ ;\ B}{Z \vdash A \oplus B}$$

$$\text{(}\prec\vdash\text{)}\ \dfrac{A < B \vdash Z}{A \prec B \vdash Z} \qquad \text{(}\vdash \prec\text{)}\ \dfrac{X \vdash A \quad B \vdash Y}{X < Y \vdash A \prec B}$$

$$\text{(}\rightarrow\vdash\text{)}\ \dfrac{X \vdash A \quad B \vdash Y}{A \rightarrow B \vdash X > Y} \qquad \text{(}\vdash\rightarrow\text{)}\ \dfrac{Z \vdash A > B}{Z \vdash A \rightarrow B}$$

Fig. 3. Rules of the display calculus δ**BiInt**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{I}\vdash\top}\,(\vdash\top)\quad\dfrac{}{A\vdash A}\,\textit{Lemma 2}}{\dfrac{\mathrm{I}<A\vdash\top\prec A}{\mathrm{I}\vdash A\;;(\top\prec A)}\,(dp)}\,(\vdash\prec)}{(A\to\bot)\;;\mathrm{I}\vdash A\;;(\top\prec A)}\,(dp)+(\vdash\mathrm{Com})}{((A\to\bot)\;;\mathrm{I})<(\top\prec A)\vdash A}\,(\mathrm{Wk}\vdash)\quad\dfrac{}{\bot\vdash\mathrm{I}}\,(\bot\vdash)}{A\to\bot\vdash[((A\to\bot)\;;\mathrm{I})<(\top\prec A)]>\mathrm{I}}\,(dp)\quad(\to\vdash)}{(A\to\bot)\;;\mathrm{I}\vdash[((A\to\bot)\;;\mathrm{I})<(\top\prec A)]>\mathrm{I}}\,(\mathrm{I}^+_-\vdash)}{(A\to\bot)\;;\mathrm{I}\vdash([((A\to\bot)\;;\mathrm{I})<(\top\prec A)]>\mathrm{I})\;;(\top\prec A)}\,(\vdash\mathrm{Wk})}{((A\to\bot)\;;\mathrm{I})<(\top\prec A)\vdash[((A\to\bot)\;;\mathrm{I})<(\top\prec A)]>\mathrm{I}}\,(dp)}{[((A\to\bot)\;;\mathrm{I})<(\top\prec A)]\;;(((A\to\bot)\;;\mathrm{I})<(\top\prec A))\vdash\mathrm{I}}\,(dp)}{((A\to\bot)\;;\mathrm{I})<(\top\prec A)\vdash\mathrm{I}}\,(\mathrm{Ctr}\vdash)}{(A\to\bot)\;;\mathrm{I}\vdash\mathrm{I}\;;(\top\prec A)}\,(dp)}{A\to\bot\vdash\top\prec A}\,(\vdash\mathrm{I}^+_-)\textit{and}(\mathrm{I}^+_-\vdash)}{\neg A\vdash\sim A}\,(\mathrm{defn})$$

Fig. 4. Derivation of $\neg A \vdash\sim A$ in δ**BiInt**

the sequent *above* the lines *from* the sequent *below* the lines. A **derivation** in δ**BiInt** of a sequent $X \vdash Y$ is defined in the usual Gentzen sense.

The structure A [B] occurs **negatively** [**positively**] in each of the structures $(A > B)$ and $(B < A)$, while both A and B occur positively in the structure $(A\ ;\ B)$. If X occurs negatively [positively] in Y then this occurrence of X occurs positively [negatively] in $(Y > Z)$ and $(Z < Y)$, but occurs negatively [positively] in $(Y\ ;\ Z)$ and $(Z\ ;\ Y)$, for any Z. This natural notion of polarity is extended to occurrences of structures in sequents as follows.

In a sequent $X \vdash Y$, an occurrence of Z is an **antecedent part** [**succedent part**] iff it occurs positively in X [Y] or it occurs negatively in Y [X] [1]. Two sequents $X_1 \vdash Y_1$ and $X_2 \vdash Y_2$ are **structurally equivalent** iff there is a derivation of the first sequent from the second (and vice-versa) using only display postulates from Figure 3.

Due to the commutativity of the semicolon (;) we really only require one of (<) and (>). That is, the two rules shown below are easily derivable in δ**BiInt** using (Com $\vdash$) and ($\vdash$ Com) respectively:

$$(\ <\mathrm{dp}>\ \vdash)\dfrac{Y < X \vdash Z}{X > Y \vdash Z} \qquad\qquad (\vdash\ <\mathrm{dp}>\ )\dfrac{Z \vdash Y < X}{Z \vdash X > Y}$$

Lemma 2. *Induction on the form of A gives $A \vdash A$ is derivable in* δ**BiInt**.

The sequent $\neg A \vdash\sim A$ is (cut-free) derivable in δ**BiInt**; see Figure 4. The following two theorems are important characteristics of display calculi.

Theorem 9 ((Belnap)). *For every antecedent* [*succedent*] *part Z of every sequent $X \vdash Y$, there is a structurally equivalent sequent $Z \vdash Y'$ $[X' \vdash Z]$ that has Z (alone) as its antecedent* [*succedent*]. *Z is* **displayed** *in $Z \vdash Y'$ $[X' \vdash Z]$.*

Theorem 10 ((Belnap)). *Since the rules of δ***BiInt** *satisfy Belnap's conditions C1-C8 [1], the calculus δ***BiInt** *enjoys cut-elimination: if there is a derivation of $X \vdash Y$ in δ***BiInt**, *there is a cut-free derivation of $X \vdash Y$ in δ***BiInt**.

Theorem 11 ((Soundness)). *If $\mathrm{I} \vdash A$ is δ***BiInt***-derivable then A is* **BiInt***-valid.*

Proof. We define a translation τ from δ**BiInt**-structures to **BiInt** formulae such that $\tau(\mathrm{I} \vdash A) = (\top \to A)$, and show by induction on the length of the given derivation of $\mathrm{I} \vdash A$ that $\top \to A$, and hence A itself, is **BiInt**-valid. The translation τ uses two subparts a, for antecedent part, and s, for succedent part:

$$\tau(X \vdash Y) \;:=\; a(X) \to s(Y)$$

$$\begin{array}{llllll} a(\mathrm{I}) & := & \top & s(\mathrm{I}) & := & \bot \\ a(X\ ;\ Y) & := & a(X) \otimes s(Y) & s(X\ ;\ Y) & := & s(X) \oplus s(Y) \\ a(X > Y) & := & a(Y) \prec s(X) & s(X > Y) & := & a(X) \to s(Y) \\ a(X < Y) & := & a(X) \prec s(Y) & s(X < Y) & := & a(Y) \to s(X) \end{array}$$

The translations $a(.)$ and $s(.)$ also show the overloading of the structural connectives (<) and (>) when found in antecedent and succedent positions, just as Gentzen's comma is overloaded to mean "conjunction" ["disjunction"] in the antecedent [succedent] of Gentzen sequents. Note that $a(X < Y) = a(Y > X)$ and $s(X < Y) = s(Y > X)$ in keeping with the fact that the rules (< dp > $\vdash$) and ($\vdash$ < dp >) are derivable in δ**BiInt**. Thus, both structures $(A < B)$ and $(B > A)$ in an antecedent position "mean" the formula $A \prec B$, while both structures $(A < B)$ and $(B > A)$ in a succedent position "mean" the formula $B \to A$.

For every rule (ρ) of δ**BiInt** we show that: if the τ-translation of the premisses of (ρ) are **BiInt**-valid, then so is the τ-translation of the conclusion of (ρ).

The only interesting cases are the display postulates involving $(X\ ;\ Y)$ on the right hand side of ($\vdash$) so we deal with these cases only.

Display Postulate	Components	τ-translation
(a)	$Z < Y \vdash X$	$(a(Z) \prec s(Y)) \to s(X)$
(b)	$Z \vdash X\ ;\ Y$	$a(Z) \to (s(X) \oplus s(Y))$
(c)	$X > Z \vdash Y$	$(a(Z) \prec s(X)) \to s(Y)$

Pick a Kripke model, any model. With respect to this model, the statements that each one of **(a)**, **(b)**, **(c)** is **BiInt**-valid can be stated as shown below:

(a) $(\forall w)(\forall v \geq w)[\ (\exists u \leq v)(u \models a(Z)\ \ \&\ \ u \not\models s(Y))\ \Rightarrow\ v \models s(X)]$
(b) $(\forall w)(\forall v \geq w)[\quad v \not\models a(Z) \quad \text{or}\ \ v \models s(X)\ \ \text{or}\ \ v \models s(Y)]$
(c) $(\forall w)(\forall v \geq w)[\ (\exists u \leq v)(u \models a(Z)\ \ \&\ \ u \not\models s(X))\ \Rightarrow\ v \models s(Y)]$

We show the arguments from **(a)** to **(b)** to **(c)** in Figure 5. □

Corollary 3. *If $\top \vdash A$ is derivable in δ***BiInt** *for an* **Int***-formula* [**DualInt***-formula*] *A, then A is* **Int***-valid* [**DualInt***-valid*].

(a) $(\forall w)(\forall v \geq w)[\ (\exists u \leq v)(u \models a(Z)\ \ \&\ \ u \not\models s(Y))\ \Rightarrow\ v \models s(X)]$
Putting $u = v$ in **(a)** gives
(1) $(\forall w)(\forall v \geq w)[\quad (v \models a(Z)\quad \&\ \ v \not\models s(Y))\ \Rightarrow\ v \models s(X)]$
Putting (1) into clausal-normal-form (CNF) gives
(b) $(\forall w)(\forall v \geq w)[\quad v \not\models a(Z)\quad$ or $\quad v \models s(Y)\quad$ or $\quad v \models s(X)]$
Un-CNF of **(b)** gives
(2) $(\forall w)(\forall v \geq w)[\quad (v \models a(Z)\quad \&\ \ v \not\models s(X))\ \Rightarrow\ v \models s(Y)]$
(8) Assume $w_0 \leq v_0$ and $(\exists u_0 \leq v_0)[u_0 \models a(Z)\ \&\ u_0 \not\models s(X)]$
(9) Putting $w = v = u_0$ into (2) gives $u_0 \models s(Y)$
(10) Since $u_0 \leq v_0$, persistence gives $v_0 \models s(Y)$.
(11) Since w_0 and v_0 were arbitrary, we obtain **(c)** below
(c) $(\forall w)(\forall v \geq w)[\ (\exists u \leq v)(u \models a(Z)\ \ \&\ \ u \not\models s(X))\ \Rightarrow\ v \models s(Y)]$

Fig. 5. Soundness of Display Postulates

Corollary 3 may seem odd for **DualInt** since the proof of Theorem 11 shows that $\top \to A$ is **BiInt**-valid, and $\top \to A$ is not a formula of **DualInt**. But for any **DualInt**-formulae A, the formula $\top \to A$ is **BiInt**-valid iff A is **DualInt**-valid.

Theorem 12 ((Completeness)). *A **BiInt**-valid $\Rightarrow \mathrm{I} \vdash A$ is δ**BiInt**-derivable.*

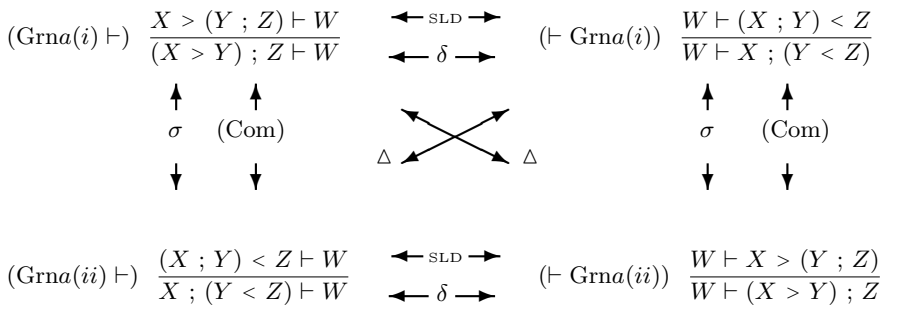
Proof. The Lindenbaum algebra formed from the equiprovable formulae of δ**BiInt** is an **HB**-algebra. □

5 Regaining Classical Logic

Figure 6 shows Grishin's rules from the generalised display calculus of [12], which we refer to as **SLD**. The symbols σ, δ and $\triangle$ are the names of three transforms from **SLD** which map formulae to formulae, structures to structures, sequents to sequents, and proofs to proofs. See [12] for details.

Figure 6 shows four horizontal pairs of **SLD** structural rules: (Grn$a(i)$), (Grn$a(ii)$), (Grn$b(i)$) and (Grn$b(ii)$). The rules in any horizontal pair are inter-derivable in the basic **SLD**-framework without further structural rules, as shown by the horizontal arrows with "SLD" attached to them in Figure 6. For example, from ($\vdash$ Grn$a(i)$) we can derive (Grn$a(i) \vdash$), and vice-versa in **SLD**.

Adding (Com) for commutativity of semicolon on both sides of turnstile allows us to inter-derive the rules connected by vertical arrows with (Com) attached to them. That is, the rules which are σ-transforms of each other. As explained in [12], the composition of the σ-transform with the δ-transform results in the $\triangle$-transform, hence the addition of (Com) also allows us to inter-derive the rules which are $\triangle$-transforms of each other. That is, (Com) causes all four of the (Grn$a$) rules to become inter-derivable, and all four of the (Grn$b$) rules to become inter-derivable. Let (Grn$a(ii) \vdash$) be a representative for the (Grn$a$) rules, and let (Grn$b(ii) \vdash$) be a representative for the (Grn$b$) rules.



Arrows with attached $\alpha \in \{\sigma, \delta, \triangle\}$ indicate rules are α-transforms from **SLD** [12].
Arrows with SLD attached indicate rules inter-derivable in **SLD**.
Arrows with (Com) attached indicate rules inter-derivable in **SLD** with (Com).

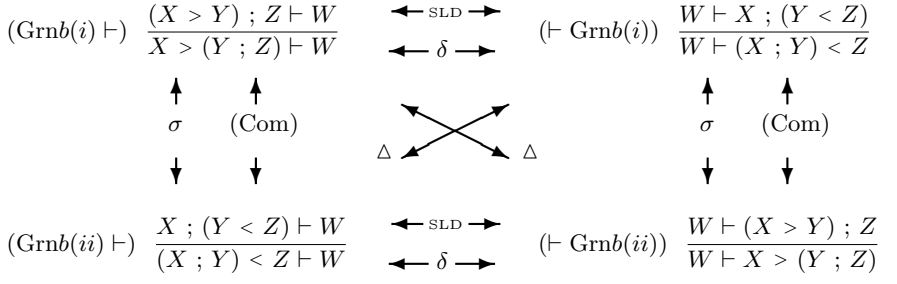


Fig. 6. Dualities, Symmetries and Inter-derivability Between Grishin's Rules

Adding all the Grishin's rules from Figure 6 converts any intuitionistic substructural logic displayable in the **SLD**-framework into its classical counterpart [12]. But adding (Ctr), and (Wk) has some subtle effects, as outlined next.

The left hand derivation from Figure 7 shows that (Grn$b(ii)\vdash$) is derivable in **SLD** when (Ctr $\vdash$) and (Wk) are present. Adding (Com) then gives us all the (Grn$b$) rules as outlined above. Thus, all (Grn$b$) rules are derivable in δ**BiInt**.

Theorem 13. *The rule* (Grn$a(ii)\vdash$) *is not derivable in* δ**BiInt**.

Proof. Suppose (Grn$a(ii)\vdash$) is derivable in δ**BiInt**. Then $\sim A \vdash \neg A$ is derivable in δ**BiInt** as shown in the bottom right of Figure 7. Since Figure 4 shows that $\neg A \vdash \sim A$ is already derivable in δ**BiInt**, we then have $\sim\ A\ \dashv\vdash\ \neg\ A$ in δ**BiInt**. But as shown in the top right of Figure 7, the sequent $\sim\sim A \vdash A$ is also derivable in δ**BiInt**, so putting $\neg\neg A$ for $\sim\sim A$ gives a derivation of $\neg\neg A \vdash A$ in δ**BiInt**. The soundness of δ**BiInt** with respect to Rauszer's semantics then implies that the formula $\neg\neg A \to A$ is **BiInt**-valid. A particular instance of this formula is $\neg\neg p \to p$, which is then **Int**-valid; contrary to what we know about **Int**. Hence (Grn$a(ii)\vdash$) cannot be derivable in δ**BiInt**.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X\ ;\ (Y\ <\ Z)\vdash W}{X\vdash W\ <\ (Y\ <\ Z)}(\mathrm{dp})}{X\ ;\ Y\vdash (W\ <\ (Y\ <\ Z))\ ;\ Z}(\mathrm{Wk}\vdash)+(\vdash \mathrm{Wk})}{(X\ ;\ Y)\ <\ Z\vdash W\ <\ (Y\ <\ Z)}(\mathrm{dp})}{[(X\ ;\ Y)\ <\ Z]\ ;\ (Y\ <\ Z)\vdash W}(\mathrm{dp})}{Y\ <\ Z\vdash [(X\ ;\ Y)\ <\ Z]\ >\ W}(\mathrm{dp})}{Y\vdash ([(X\ ;\ Y)\ <\ Z]\ >\ W)\ ;\ Z}(\mathrm{dp})}{X\ ;\ Y\vdash ([(X\ ;\ Y)\ <\ Z]\ >\ W)\ ;\ Z}(\mathrm{Wk}\vdash)}{(X\ ;\ Y)\ <\ Z\vdash [(X\ ;\ Y)\ <\ Z]\ >\ W}(\mathrm{dp})}{[(X\ ;\ Y)\ <\ Z]\ ;\ ((X\ ;\ Y)\ <\ Z)\vdash W}(\mathrm{dp})}{(X\ ;\ Y)\ <\ Z\vdash W}(\mathrm{Ctr}\vdash)$$

Derivation of $(\mathrm{Grn}b(ii)\vdash)$ in δ**BiInt**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{I}\vdash\top}(\vdash\top)}{\top\vdash\top}(\top\vdash)\quad\dfrac{}{A\vdash A}\ \text{Lemma 2}}{\top\ <\ A\vdash\top\prec A}(\vdash\prec)}{\top\vdash(\top\prec A)\ ;\ A}(\mathrm{dp})}{\top\vdash A\ ;\ (\top\prec A)}(\vdash\mathrm{Com})}{\top\ <\ (\top\prec A)\vdash A}(\mathrm{dp})}{\top\prec(\top\prec A)\vdash A}(\prec\vdash)}{\sim\sim A\vdash A}(\mathrm{defn})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A\vdash A}\ \text{Lemma 2}}{A\vdash A}(\mathrm{Wk}\vdash)+(\vdash\mathrm{Wk})}{A\ ;\ \top\vdash\bot\ ;\ A}(\mathrm{dp})}{(A\ ;\ \top)\ <\ A\vdash\bot}(\mathrm{Grn}a(ii)\vdash)}{A\ ;\ (\top\ <\ A)\vdash\bot}(\mathrm{dp})}{\top\ <\ A\vdash A\ >\ \bot}(\prec\vdash)}{\top\prec A\vdash A\ >\ \bot}(\vdash\!\rightarrow)}{\sim A\vdash\neg A}(\mathrm{defn})$$

Fig. 7. More Derivations in δ**BiInt**

Theorem 14. *To obtain a display calculus for classical logic from* δ**BiInt**, *it suffices to add any one of the* (Grn$a$) *rules, e.g.* (Grn$a(ii)\vdash$).

Corollary 4. *Adding any semi-equation from Table1 to an* **HB**-*algebra gives a Boolean algebra.*

Corollary 5. *A (reflexive and transitive) Kripke frame validates the formulae from Table1 iff it is symmetric.*

6 Concluding Remarks

My goal is to extend δ**BiInt** to bi-intuitionistic modal and tense logics. Wolter [19] studies intuitionistic modal logics obtained by extending **BiInt** with tense-logical modalities, and also extends the Gödel-McKinsey-Tarski embedding of **Int** into **S4** to give an embedding of **BiInt** into $\mathbf{K_tS4}$; see also [13], which contains a typographical error in the translation. Consequently, we can use the method of [10] to give a cut-free display calculus for **BiInt** using the classical modal display logic framework of Wansing [18].

Table 1. Semi-equations and formulae for classicality

$a \wedge (b \prec c)$	$\leq$	$(a \wedge b) \prec c$	$A \otimes (B \prec C)$	$\rightarrow$	$(A \otimes B) \prec C$
$(b \prec a) \wedge c$	$\leq$	$(b \wedge c) \prec a$	$(B \prec A) \otimes C$	$\rightarrow$	$(B \otimes C) \prec A$
$c \rightarrow (a \vee b)$	$\leq$	$(c \rightarrow a) \vee b$	$C \rightarrow (A \oplus B)$	$\rightarrow$	$(C \rightarrow A) \oplus B$
$a \rightarrow (b \vee c)$	$\leq$	$(a \rightarrow b) \vee c$	$A \rightarrow (B \oplus C)$	$\rightarrow$	$(A \rightarrow B) \oplus C$

Although Crolard does not prove cut-elimination for his **SLK**, it seems highly likely that **SLK** does enjoy cut-elimination, so $\mathbf{SLK^1}$ is a good candidate as an alternative cut-free Gentzen system for **BiInt**.

References

1. N D Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.
2. T Crolard. Subtractive logic. *Theor. Comp. Sci.* , to appear.
3. H B Curry. *Foundations of Mathematical Logic*. Dover, 1976.
4. J Czermak. A remark on Gentzen's calculus of sequents. *Notre Dame Journal of Formal Logic*, 18(3):471–474, 1977.
5. A G Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. AMS, USA, 1988.
6. R Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *JSL*, 57(3), 1992.
7. N D Goodman. The logic of contradiction (abstract). *Notices of the American Mathematical Society*, 25, A-24, 1978. Abstract number 752-02-2.
8. N D Goodman. The logic of contradiction. *ZML*, 27:119–126, 1981.
9. R Goré. A uniform display system for intuitionistic and dual intuitionistic logic. TR-ARP-6-95, Automated Reasoning Project, Australian Nat. Uni.
10. R Goré. Intuitionistic logic redisplayed. TR-ARP-1-95, Automated Reasoning Project, Australian National University, 1995.
11. R Goré. Cut-free display calculi for relation algebras. In CSL96, LNCS 1258:198–210. Springer, 1997.
12. R Goré. Substructural logics on display. *Log. J. IGPL*, 6(3):451–504, 1998.
13. P Łukowski. Modal Interpretation of Heyting-Brouwer Logic. *Bulletin of the Section of Logic*, 25:80–83, 1996.
14. J C C McKinsey and A Tarski. On closed elements in closure algebras. *Annals of Mathematics*, 47:122–162, 1946.
15. C Rauszer. Semi-boolean algebras and their applications to intuitionistic logic with dual operators. *Fundamenta Mathematicae*, 1974.
16. C Rauszer. *An algebraic and Kripke-style approach to a certain extension of intuitionistic logic*, volume CLXVII of *Dissertationes Mathematicae*. Institute of Mathematics, Polish Academy of Sciences, 1980.
17. I Urbas. Dual intuitionistic logic. *NDJFL*, 37(3):440–451, 1996.
18. H Wansing. *Displaying Modal Logic*, Kluwer, Trends in Logic, 1998.
19. F Wolter. On logics with coimplication. *J. Phil. Logic*, 27:353–387, 1998.

Model Sets in a Nonconstructive Logic of Partial Terms with Definite Descriptions*

Raymond D. Gumb

Computer Science Department, University of Massachusetts, Lowell, MA 01854.
Email: gumb@cs.uml.edu.

Abstract. The logic of partial terms (**LPT**) is a variety of negative free logic. In **LPT**, functions, as well as predicates, are strict, and free variables are given the generality interpretation. Both nonconstructive (classical) and intuitionist brands of negative free logic have served in foundational investigations, and Hilbert-style axiomatizations, natural deduction systems, and Gentzen-style sequents have been developed for them. This paper focuses on nonconstructive **LPT** with definite descriptions, called **LPD**, lays the foundation for tableaux systems by defining the concept of an **LPD** model system and establishing Hintikka's Lemma, and summarizes the corresponding tableaux proof rules.

Philosophical Roots of Negative Free Logics

... not even with these (contraries 'Socrates is well' and 'Socrates is sick') is it necessary always for one to be true and the other false. For if Socrates exists one will be true and the other false, but if he does not both will be false.... (Aristotle, **Categories**, x, 13b12)

A robust sense of reality is necessary in framing a correct analysis of propositions about ... round squares and other such pseudo-objects....we shall insist that in the analysis of propositions, nothing "unreal" is to be admitted. (Bertrand Russell, **Introduction to Mathematical Philosophy**)

1 Negative Free Logic and the Logic of Partial Terms

This paper describes a nonconstructive logic of partial terms with definite descriptions (**LPD**). In the philosophical literature, **LPD** and similar logics are called "negative free logics" as explained below. Philosophers have long been concerned with singular terms in natural language that refer to no existent, for example, definite descriptions such as "the present King of France" or "the greatest natural number" and grammatically proper names such as "Pegasus".

* Karel Lambert helped me understand better the nature and scope of free logics. All of my previous work had been with positive free logics, and he introduced me to negative free logics. I am grateful to him for a number of useful suggestions.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 268–278, 2000.

Beginning in the 1950's, free logics were developed to handle anomalies arising from singular terms that might refer to no existent in an otherwise classical predicate logic (perhaps with identity), as chronicled in the work of some of the field's pioneers [18,20]. The relevance of free logic to some branches of mathematics and related areas of mathematical computer science was explicitly argued beginning in the 1960's [29,30,24,23]. As noted elsewhere [15], free logic has been implicit from the beginning in a central area of computer science, program specification and verification [17]. The term "free logic" has not been current in the mathematics and computer science literature, however, because some authors object to the term "free logic" (free logics have nothing to do with free algebras), and others were unaware of the philosophical literature at the time that they rediscovered free logic.

Since **LPD** and similar logics draw from diverse philosophical, mathematical, and computer science traditions, the terminology can be confusing. By a (nonconstructive or classical) *free logic*, I mean an extension of standard first-order logic. That is, when attention is restricted to closed formulas containing only variables as singular terms, the theorems and derivations of a free logic coincide with those of standard logic. Free logics are unlike standard logic in that not all singular terms (other than bound variables) need have existential import. Similarly, free brands of intuitionist logic are extensions of the usual intuitionist logic.

Among species of free logics, the most common are positive and negative free logics. *Positive free logics* allow some atomic statements containing singular terms that do not refer to existents to be true. On the other hand, *negative free logics* count as false any atomic statement containing a singular term that does not refer to an existent.[1] The concept of "strictness" in computer science is closely related to the concerns of negative free logic. An n-ary function f is said to be *strict* in its i-th argument provided $f^n(t_1, \ldots, t_n)$ is undefined if t_i is undefined, and n-ary predicate P is *strict* in its i-th argument provided $P^n(t_1, \ldots, t_n)$ is false if t_i is undefined. If a function (predicate) is strict in all of its arguments then it is *strict*, otherwise it is *nonstrict*. A logic containing only strict predicates is, then, a negative free logic. The *logic of partial terms* (**LPT**)[2] is called a *strict* negative free logic because it is a negative free logic that countenances only functions that are strict in every argument. **LPT** typically imposes the generality interpretation on free variables. In **LPT**, there is no "outer domain of nonexistents" as there are in the semantics of many positive free logics, and constants and variables always denote.

By using **LPT**, as is the case with other free logics, the axioms of a mathematical theory can be stated without cluttering the axiomatization with error conditions, as would be required using restricted quantification in standard logic.

[1] Rolf Schock developed the foundations of negative free logic in the 1960's and pioneered work in definition theory and definite descriptions, [26,27,28]. A number of other logicians have built on his work since [25,4,31].

[2] There are some variations in what we call **LPT** [22,1,33,32,5,6,21,10], and another common name for **LPT** is "$\mathbf{E}^+$-logic". I follow Beeson's terminology [1] in this paper.

For example, Peano's axiom:

$$\forall X \forall Y (\mathrm{succ}(X) = \mathrm{succ}(Y) \rightarrow X = Y)$$

can be stated in the usual way, whereas with restricted quantification (understanding that error objects are not natural numbers), the axiom takes on the more awkward form:

$$\forall X : \mathrm{Natno}\, \forall Y : \mathrm{Natno}\, (\mathrm{succ}(X) = \mathrm{succ}(Y) \rightarrow X = Y)$$

or, translated into standard first-order logic, the even more verbose form:

$$\forall X \forall Y (\mathrm{Natno}(X) \rightarrow (\mathrm{Natno}(Y) \rightarrow (\mathrm{succ}(X) = \mathrm{succ}(Y) \rightarrow X = Y)))$$

Moreover, **LPT** is concise, because in **LPT** there is no need for strictness axioms, which are verbose. Other negative free logics also dispense with strictness axioms for predicates. In contrast, most nonstrict positive free logics [30,7,14,15,16] require strictness axioms. However, note that although **LPT** is adequate for handling partial functions in most if not all traditional mathematical applications, it is not in general adequate in computer science, because every practical programming language contains nonstrict constructs that must be formalized for program specification and verification. A nonstrict positive free logic is required [14,15,16].

The logic **LPD**, the subject of the present paper, is **LPT** outfitted with 0-ary function symbols and definite descriptions. Having definite descriptions makes a logic easier to use. Definite descriptions are analogous to "anonymous functions" in programming languages, whereas definitions are analogous to "named functions". It is convenient to have both anonymous and named functions in a programming language, even though in principal either can play the role of the other. Similar remarks apply to definite descriptions and definitions in an applied mathematical logic, especially one underlying an applied logic of programming. **LPD** is similar to systems found in the literature, and my sketch of its syntax and semantics will be brief. I concentrate on defining **LPD** model systems (for sentences) and Herbrand models and establishing Hintikka's Lemma, laying the foundation for **LPD** tableaux systems

2 Syntax

The logical symbols of **LPD** include the propositional connectives for negation ($\neg$) and material implication ($\rightarrow$), a countably infinite set of individual variables ($x, y, z, \ldots$), identity ($=$), the universal quantifier ($\forall$), and the definite description symbol (ι). Free and bound occurrences of variables are to be understood as usual. The set of nonlogical symbols consists of:

- A countable (finite or countably infinite) set **R** of predicate symbols $P, Q, R, \ldots$. Associated with each predicate symbol P in **R** is a natural number $n_P \geq 1$ giving its arity.

- A countable set $\mathbf{F}$ of function symbols $f, g, h, \ldots$, with each function symbol f assigned an arity $n_f \geq 0$.
- A countable set $\mathbf{C}$ of constant symbols $a, b, c, \ldots$.

In **LPT** (but not **LPD**), 0-ary function symbols are excluded, because they cannot be identified with with either free variables or constant symbols, since a 0-ary function symbol need not denote. Variables and constants also have different semantics, as will be described in Section 3.

Understand $\mathbf{L}(\mathbf{R}, \mathbf{F}, \mathbf{C})$ to be the **LPD** *language determined by* $\mathbf{R}$, $\mathbf{F}$, and $\mathbf{C}$ [8]. The *syntax* of terms $t, \ldots$ and formulas $A, B, \ldots$ of $\mathbf{L}(\mathbf{R}, \mathbf{F}, \mathbf{C})$ is defined by mutual recursion:

1. A variable x is a term.
2. A constant $c \in \mathbf{C}$ is a term.
3. $f(t_1, \ldots, t_{n_f})$ is a term if $f \in \mathbf{F}$ and $t_1, \ldots, t_{n_f}$ are terms.
4. $t_1 = t_2$ is a(n atomic) formula if t_1 and t_2 are terms.
5. $P(t_1, \ldots, t_{n_P})$ is a(n atomic) formula if $P \in \mathbf{R}$ and $t_1, \ldots, t_{n_P}$ are terms.
6. $\neg A$ is a formula if A is.
7. $(A \to B)$ is a formula if A and B are.
8. $\forall x A$ is a formula if x is a variable, A is a formula, and x does not occur bound in A.
9. $\iota x A$ is a term if x is a variable, A is a formula, and x does not occur bound in A.

A *literal* is understood to be an atomic formula or the negation of one. A *sentence* is a formula with no free variables. Regarding substitution, write $\equiv$ for *syntactic identity*, $A(x)$ for any formula having only x as a free variable, and $A(a)$ for the sentence $A(x)\,[a/x]$ resulting from substituting a for every occurrence of x in $A(x)$. A *theory* $\mathcal{T}$ of **LPD** is a pair $< L, T >$, where L is an **LPD** language and T is a set of sentences of L.

A number of abbreviations are common in the literature. The other propositional connectives $(\wedge, \vee, \leftrightarrow)$ and the existential quantifier $(\exists)$ are understood as usual. The following abbreviations are current in the **LPT** and free logic literature:

Definedness: $t{\downarrow} \equiv t = t$
Undefinedness: $t{\uparrow} \equiv \neg t{\downarrow}$
Quasi-equality: $t_1 \doteq t_2 \equiv t_1{\downarrow} \vee t_2{\downarrow} \to t_1 = t_2$
Undefined: $\bot \equiv \iota x \neg(x = x)$

3 Semantics

A *model* $\mathbf{M}$ for an **LPD** language $\mathbf{L(R,F,C)}$ is a pair $\mathbf{M} =< D, I >$, where the domain D is a nonempty set and I is an interpretation function satisfying the following conditions:

1. $c^I \in D$ for each constant $c \in \mathbf{C}$.

2. f^I is a partial function from D^{n_f} to D for each $f \in \mathbf{F}$. (Either $f^I \in D$ or f^I is *undefined* when $n_f = 0$).
3. $[\![=]\!]^I = \{<d, d>: d \in D\}$.
4. $P^I \subseteq D^{n_P}$ for $P \in \mathbf{R}$.

Turning to the semantics of individual variables, a *state* σ in a model $\mathbf{M}$ is a total function mapping the individual variables into the domain D. Understand x^σ to be the value that σ assigns the variable x. Take a state σ' to be an *x-variant* of a state σ provided $\sigma'(y) = \sigma(y)$ for every individual variable $y \not\equiv x$, and write $\sigma\{d/x\}$ for the x-variant of σ such that $x^{\sigma\{d/x\}} = d \in D$. The *value* of a term (a formula) of $\mathbf{L}(\mathbf{R}, \mathbf{F}, \mathbf{C})$ in model $\mathbf{M}$ in the state σ is defined as follows:

1. $x^{I,\sigma} = x^\sigma$.
2. $c^{I,\sigma} = c^I$.
3. $[\![f(t_1, \ldots, t_{n_f})]\!]^{I,\sigma} = f^I([\![t_1]\!]^{I,\sigma}, \ldots, [\![t_{n_f}]\!]^{I,\sigma})$ if
   a) $[\![t_i]\!]^{I,\sigma} \in D$ for each i from 1 to n_f, and
   b) $f^{I,\sigma}$ is defined at $<[\![t_1]\!]^{I,\sigma}, \ldots, [\![t_{n_f}]\!]^{I,\sigma}>$
   undefined otherwise
4. $[\![\iota x A]\!]^{I,\sigma} = d$ if there is a unique d such that $A^{I,\sigma\{d/x\}} = \mathbf{true}$
   undefined otherwise
5. $[\![t_1 = t_2]\!]^{I,\sigma} = \mathbf{true}$ if $t_1^{I,\sigma} = t_2^{I,\sigma}$
   $= \mathbf{false}$ otherwise
6. $[\![P(t_1, \ldots, t_{n_P}]\!]^{I,\sigma} = \mathbf{true}$ if
   a) $[\![t_i]\!]^{I,\sigma} \in D$ for each i from 1 to n_P, and
   b) $<t_1^{I,\sigma}, \ldots, t_{n_P}^{I,\sigma}> \in P^I$
   $= \mathbf{false}$ otherwise
7. $[\![\neg A]\!]^{I,\sigma} = \neg [\![A]\!]^{I,\sigma}$
8. $[\![A \to B]\!]^{I,\sigma} = [\![A]\!]^{I,\sigma} \to [\![B]\!]^{I,\sigma}$
9. $[\![\forall x A]\!]^{I,\sigma} = \mathbf{true}$ if $A^{I,\sigma'} = \mathbf{true}$ for every x-variant σ' of σ
   $= \mathbf{false}$ otherwise

Note that by clause 5, an undefined term cannot be "equal" to any term, including itself.

We come to the definitions of the key semantic concepts. A formula A is *true* in model $\mathbf{M}$, written $\mathbf{M} \models A$, if $A^{I,\sigma} = \mathbf{true}$ for all states σ. A model $\mathbf{M}$ *satisfies* a set of formulas S if, for every formula $A \in S$, $\mathbf{M} \models A$, and S is *satisfiable* if there is a model that satisfies S. A sentence A is a *logical consequence* of a set of sentences S, written $S \models A$, if $S \cup \{\neg A\}$ is unsatisfiable. Finally, a sentence A is *valid*, written $\models A$, if $\emptyset \models A$.[3]

[3] To accommodate the generality interpretation of free variables, the following definition replaces the one in the text, above: A formula A is *valid* if, for every model $\mathbf{M}$, $\mathbf{M} \models A$, and it follows that a sentence A is valid in this sense if and only if $\emptyset \models A$. Hintikka sets are defined in terms of sentences, and consequently I have little to say about open formulas.

4 Model Sets

Let t and t' be any terms, A any sentence, and $\prec$ be a well-ordering of the terms of $\mathbf{L} = \mathbf{L}(\mathbf{R}, \mathbf{F}, \mathbf{C})$ that sets $\bot$ as the first element and that otherwise preserves the natural complexity ordering. Understand an **LPD** *model set* (also known as a *Hintikka set*) to be a set of sentences S[4] (in the language $\mathbf{L}$) satisfying the following conditions:

(I.$\overline{\emptyset}$) $\exists x(x\downarrow) \in S$
(C.$\neg$) If $A \in S$, then $\neg A \notin S$.
(C.$\bot\uparrow$) $\bot\downarrow \notin S$
(D.c) If $c \in \mathbf{C}$, then $c\downarrow \in S$
(D.f_s) If $f \in \mathbf{F}$ and $f(t_1, \ldots, t_n)\downarrow \in S$, then $t_1\downarrow, \ldots, t_n\downarrow \in S$.
(D.$=_s$) If $t_1 = t_2 \in S$, then $t_1\downarrow, t_2\downarrow \in S$.
(D.P_s) If $P \in \mathbf{R}$ and $P(t_1, \ldots, t_n) \in S$, then $t_1\downarrow, \ldots, t_n\downarrow \in S$.
(D.=) If $A(t)$ is a literal, $A(t) \in S$, $t' \prec t$ and either $t = t' \in S$ or $t' = t \in S$, then $A(t') \in S$
(D.$\neg\neg$) If $\neg\neg A \in S$, then $A \in S$
(D.$\neg\rightarrow$) If $\neg(A \rightarrow B) \in S$, then $A, \neg B \in S$
(D.$\rightarrow$) If $A \rightarrow B \in S$, then either $\neg A \in S$ or $B \in S$
(D.$\neg\forall$) If $\neg\forall x A(x) \in S$, then $\neg A(t), t\downarrow \in S$ for some term t.
(D.$\forall$) If $\forall x A(x) \in S$, then $A(t) \in S$ for each term t such that $t\downarrow \in S$.
(D.ι) If $B(\iota x A(x)) \in S$ is a literal, then $\exists y(\forall x(A(x) \leftrightarrow x = y) \wedge B(y)) \in S$.

The term $\bot$ must be the first element in the well-ordering $\prec$ so that there is no $t \prec \bot$, blocking $\{t\downarrow, t = \bot\}$ from being included in a model set. As it is, we have $\{t\downarrow, t = \bot, \bot\downarrow\}$ by condition (**D**.=), which condition (**C**.$\bot$) blocks. The motivation of the other rules is much as usual. Note that condition (**D**.ι) is a negative free logic rendering of Lambert's Law [19,2].

Let S be a(n **LPD**) model set in the language $\mathbf{L}$. The binary relation $\cong_S$, written simply as $\cong$ when no ambiguity can arise, is defined on the terms of $\mathbf{L}$ as follows. For each pair of terms t and t' of $\mathbf{L}$, take $t \cong t'$ if

1. $t \equiv t'$, or
2. $t = t' \in S$, or
3. $t' = t \in S$, or
4. $t\downarrow \notin S$ and $t'\downarrow \notin S$, or
5. there is a term t'' of $\mathbf{L}$ such that $t \cong t''$, $t'' \cong t'$, $t'' \prec t$, and $t'' \prec t'$.

Proof of the following is similar to proofs found in the literature [11]:

Proposition 1. $\cong$ *is an equivalence relation on the terms of* $\mathbf{L}$.

[4] Permitting open formulas in model sets is not very useful, because, in full first-order logic (without resorting to Skolem functions or the like), key semantic concepts such as *logical consequence* cannot be defined in a natural manner. In Fitting's treatment of free variable tableaux, open formulas are permitted in tableaux so that unification can be used for more efficient proofs, but the formulas to be proved must be sentences [8]. In his completeness proof, the model sets consist solely of sentences.

By the definitions of $\cong$ and$\downarrow$ and conditions $(\mathbf{D}.=_s)$ and $(\mathbf{D}.=)$, we have

Proposition 2. *If $t\downarrow \in S$ and $t \cong t'$, then there is some term t'' of $\mathbf{L}$ such that $t'' \preceq t, t'' \preceq t'$ and either $t'' = t, t'' = t' \in S$, $t'' = t, t' = t'' \in S$, $t = t'', t'' = t' \in S$, or $t = t'', t' = t'' \in S$.*

The *length* $\ell(A)$ of a formula A is defined inductively as follows:

1. $\ell(A) = 1$ if A is atomic,
2. $\ell(\neg A) = \ell(A) + 1$,
3. $\ell(A \to B) = \ell(A) + \ell(B) + 1$,
4. $\ell(\forall x A) = \ell(A) + 1$,

For purposes of the next definition, each sentence A of $\mathbf{L}$ is written in the form $B(t_1, \ldots, t_n)$, where the t_i's are all of the distinct closed terms occurring in A. Understand $B(t'_1, \ldots, t'_n)$ to be an $\cong$-*variant* of A if $t_1 \cong t'_1, \ldots, t_n \cong t'_n$, and the $\cong$-*variation count* $v(A, A')$ to be the number of i $(1 \leq i \leq n)$ such that $t_i \ncong t'_i$.

Lemma 1. *Let S be a model set. If $A \in S$ and A' is a $\cong$-variant of A, then $\neg A' \notin S$.*

Proof: The proof runs by mathematical induction on $\ell(A)$.

Basis: $\ell(A) = 1$. $A \in S$ is atomic, and $A' \equiv P(t'_1, \ldots, t'_n)$ $(P \in \mathbf{R})$ is an $\cong$-variant of $A \equiv P(t_1, \ldots, t_n)$. The proof continues by a subsidiary induction on $v(A, A')$. If $v(A, A') = 0$, then we have $A \equiv A'$, and $\neg A' \notin S$ by $(\mathbf{C}.\neg)$. So, proceeding inductively, suppose $v(A, A') \geq 1$ and that j is the smallest i such that $t_i \ncong t'_i$. Since $A \equiv P(t_1, \ldots, t_n)$ is atomic, $t_j\downarrow \in S$ by either $(\mathbf{D}.=_s)$ or $(\mathbf{D}.P_s)$. Let t''_j be as in Proposition 2. Since $A, t_j\downarrow \in S$ and $t_j \cong t'_j$, we have $A[t''_j/t_j] \in S$ by $(\mathbf{D}.=)$. If $\neg A' \in S$, then by a similar argument $\neg A'[t''_j/t'_j] \in S$, contradicting the inner induction hypothesis since $v(A[t''_j/t], A'[t''_j/t'_j]) = v(A, A') - 1$. Hence, $\neg A' \notin S$.

Induction Step: $\ell(A(t)) \geq 2$. The proof is as in the literature [11].

5 Herbrand Models

Let S be a model set in the language $\mathbf{L} = \mathbf{L}(\mathbf{R}, \mathbf{F}, \mathbf{C})$. A *Herbrand model* $\mathbf{M} = < D, I >$ for S is defined as follows. Understand the *equivalence class for a defined term* t of $\mathbf{L}$ to be $\ll t \gg = \{t' : t = t' \in S \vee t' = t \in S\}$ if $t\downarrow \in S$ and $\ll t \gg$ to be *undefined* otherwise. Set the domain of the Herbrand model $\mathbf{M}$ to be $D = \{\ll t \gg : t\downarrow \in S\}$.

The specification of the interpretation I is as follows:

1. $c^I = \ll c \gg$ for each $c \in \mathbf{C}$,
2. $f^I = \{< \ll t_1 \gg, \ldots, \ll t_{n_f} \gg, \ll f(t_1, \ldots, t_{n_f}) \gg >: f(t_1, \ldots, t_{n_f})\downarrow \in S\}$,
3. $[\,=\,]^I = \{< d, d >: d \in D\}$, and
4. $P^I = \{< \ll t_1 \gg, \ldots, \ll t_{n_P} \gg >: P(t_1, \ldots, t_{n_P}) \in S\}$,

Proposition 3. *1. If* $\ll t_1 \gg = \ll t'_1 \gg, \ldots, \ll t_{n_f} \gg = \ll t'_{n_f} \gg$ *and* $\ll f(t_1, \ldots, t_{n_f}) \gg \in D$, *then* $\ll f(t'_1, \ldots, t'_{n_f}) \gg \in D$ *and* $\ll f(t_1, \ldots, t_{n_f}) \gg = \ll f(t'_1, \ldots, t'_{n_f}) \gg$.

2. If $\ll t_1 \gg = \ll t'_1 \gg, \ldots, \ll t_{n_P} \gg = \ll t'_{n_P} \gg$ *and* $<\ll t_1 \gg, \ldots, \ll t_{n_P} \gg> \in P^I$, *then* $<\ll t'_1 \gg, \ldots, \ll t'_{n_P} \gg> \in P^I$.

3. If there is some i *such that* $1 \leq i \leq n_f$ *and* $t_i^{I,\sigma} \notin D$, *then* $[\![f(t_1, \ldots, t_{n_f})]\!]^{I,\sigma} \notin D$.

4. If there is some i *such that* $1 \leq i \leq n_P$ *and* $t_i^{I,\sigma} \notin D$, *then* $[\![P(t_1, \ldots, t_{n_P})]\!]^{I,\sigma} =$ **false**.

6 Hintikka's Lemma

The following result is the key ingredient in proof of the completeness of a tableaux system (tree method) for **LPD**:

Theorem 1. Hintikka's Lemma*: An* **LPD** *model set is satisfiable.*

Proof: Let S be an **LPD** model set in the language **L**. Understand $d(A)$ to be the number of distinct descriptions occurring in sentence A of **L**, the *d-complexity* of A to be the ordered pair $<d(A), \ell(A)>$, and $\prec$ to be the ordering on the sentences of **L** determined by d-complexity. The proof proceeds by using lexicographic induction on the d-complexity $<d(A), \ell(A)>$ of $A \in S$ to demonstrate that S is satisfied on its Herbrand model.

Basis: $<d(A), \ell(A)> \preceq <0, 2>$.

Case 1: $<d(A), \ell(A)> = <0, 1>$.

Suppose $A \equiv P(t_1, \ldots, t_{n_P})$, where $P \in \mathbf{R}$ and $P(t_1, \ldots, t_{n_P})$ is atomic, contains no descriptions. We have (for each σ) $[\![P(t_1, \ldots, t_{n_P})]\!]^{I,\sigma} =$ **true** because $\ll\ll t_1 \gg, \ldots, \ll t_{n_P} \gg\gg \in P^I$ by $(\mathbf{D}.P_s)$ and the definitions of the Herbrand domain D and P^I. The case when $A \equiv t_1 = t_2$ is similar.

Case 2: $<d(A), \ell(A)> = <0, 2>$.

$A \equiv \neg P(t_1, \ldots, t_{n_P})$ ($P \in \mathbf{R}$ or $P \equiv =$) is the negation of an atomic sentence that contains no descriptions. By Lemma 1, $P(t_1, \ldots, t_{n_P})$ is not an $\cong$-variant of some sentence $B' \in S$. So, by the definition of the Herbrand interpretation I, $[\![P(t_1, \ldots, t_{n_P})]\!]^{I,\sigma} =$ **false**, and $[\![\neg P(t_1, \ldots, t_{n_P})]\!]^{I,\sigma} =$ **true**.

Induction Step: $<d(A), \ell(A)> \succ <0, 2>$.

Case 1: $<d(A), \ell(A)> = <m+1, 1>$ or $<d(A), \ell(A)> = <m+1, 2>$.

We have $A \equiv C(\iota x B(x))$, where $C(x)$ is a literal. By condition $(\mathbf{D}.\iota)$ on **LPD** model sets, $\exists y(\forall x(B(x) \leftrightarrow x = y) \wedge C(y)) \in S$. Since $d(\exists y(\forall x(B(x) \leftrightarrow x = y) \wedge C(y))) = d(A) - 1 = m$, we have $[\![\exists y(\forall x(B(x) \leftrightarrow x = y) \wedge C(y))]\!]^{I,\sigma} =$ **true** by the induction hypothesis, and $[\![\iota x B(x)]\!]^{I,\sigma} \in D$ Hence, $[\![C(\iota x B(x))]\!]^{I,\sigma} =$ **true**.

Case 2: $<d(A), \ell(A)> \succeq <m, n+3>$.

A is one of the following forms: $\neg\neg B$, $\neg(B \rightarrow C)$, $(B \rightarrow C)$, $\forall x B$, or $\neg\forall x B$. The proof is by cases, with the ℓ but not the d values decreasing in

appeals to the induction hypothesis. I illustrate the case when $A \equiv \neg\neg B$, as details for the other cases are similar to proofs that can be retrieved from the literature.[5] We have $B \in S$ by condition (**D.** $\neg\neg$), and $\ell(B) = \ell(\neg\neg B) - 2 = \ell(A) - 2 = n + 1$. Hence, $A^{I,\sigma} = [\neg\neg B]^{I,\sigma} = B^{I,\sigma} = \mathbf{true}$ by the induction hypothesis since $<d(B), \ell(B)> \prec <d(A), \ell(A)>$.

7 Tableaux

Rules of the tableaux system (tree method) are read-off from the clauses defining a model set as usual [11,13,16]. Corresponding adjustments in the rules of the tableaux system ([12], pp. 322-323) are needed to suit the new and modified rules. The adjustment to rule **(D.=)** is required to handle terms containing occurrences of function symbols. Roughly, we substitute one term for another term that is equal to it just when the first term is "simpler" ($\prec$) than the second. The well-ordering requirement limits the proliferation of substitutions. For example, if $P(t), Q(t'), t = t'$ occur on an open branch, and $t \prec t'$, we extend the branch by adding $Q(t)$ (if there is no redundancy), not both it and $P(t')$. The complexity preserving feature of the well-ordering blocks certain infinite loops in the tableaux system when we have, for example, $a = f(a)$ and $P(a)$ on an open branch, because $a \prec f(a)$, so these sentences would not (by themselves) require that the branch be extended. However, if it were possible for $f(a) \prec a$, we would need to extend the branch first with $P(f(a))$, then with $P(f(f(a)))$, and so on.

Condition **(I.$\overline{\emptyset}$)** for model sets translates into an initialization rule:

(I.$\overline{\emptyset}$) Add the sentence $\exists x(x\downarrow)$ as the first sentence in the enumeration of the set of sentences to be tested for inconsistency.

A slight efficiency can be had by adding instead $c = c$ for the first constant c (in the well-ordering $\prec$) that is foreign to A.

The *tableaux derivation rules*, **D**-*rules* for short, are as follows:

(D.c) If constant c occurs in a sentence on a branch, then extend that branch with the sentence $c\downarrow$.

(D.f_s) If term $f(t_1, \ldots, t_n)$ occurs on a branch, then extend that branch with the sentences $t_1\downarrow$, ..., and $t_n\downarrow$.

(D.$=_s$) If $t_1 = t_2$ occurs on a branch, then extend that branch with the sentences $t_1\downarrow$ and $t_2\downarrow$.

(D.P_s) If atomic sentence $P(t_1, \ldots, t_n)$ occurs on a branch, then extend that branch with the sentences $t_1\downarrow$, ..., and $t_n\downarrow$.

(D.=) If both $A(t)$ and either $t = t'$ or $t' = t$ occur on a branch, $A(t)$ is a literal, and $t' \prec t$, then extend that branch with the sentence $A(t')$.

(D.$\neg\neg$) If $\neg\neg A$ occurs on a branch, then extend that branch with the sentence A.

[5] The proof in these cases is adapted from the proof in [11], pp. 38-43, with modifications to suit the model-theoretic semantics as in Boolos and Jeffrey's simple completeness proof ([3], pp. 138-143).

(D.¬ →) If $\neg(A \to B)$ occurs on a branch, then extend that branch with the sentences A and $\neg B$.
(D.→) If $A \to B$ occurs on a branch, then add $\neg A$ as a left child of that branch and B as a right child of that branch.
(D.¬∀) If $\neg\forall x A(x)$ occurs on a branch, then extend that branch with the sentences $\neg A(c)$ and $c\!\downarrow$ for the first constant c foreign to that branch.
(D.∀) If $\forall x A(x)$ and $t\!\downarrow$ occur on a branch, then extend that branch with the sentence $A(t)$.
(D.ι) If literal $B(\iota x A(x))$ occurs on a branch, then extend that branch with the sentence $\exists y(\forall x(A(x) \leftrightarrow x = y) \wedge B(y))$.

In **D**-rule **(D.ι)**, a slight efficiency can be gained by, instead of extending the branch with the sentence $\exists y(\forall x(A(x) \leftrightarrow x = y) \wedge B(y))$, extending it with the sentences $\forall x(A(x) \leftrightarrow x = c)$ and $B(c))$ for the first constant c foreign to the branch.

A branch is *closed* if either of the following *branch closing rules*, **C**-*rules* for short, are violated:

(C.¬) For some sentence A, both A and $\neg A$ occur on that branch.
(C.⊥↑) $\bot\!\downarrow$ occurs on that branch.

References

1. Michael Beeson. *Foundations of Constructive Mathematics*. Springer, Berlin, 1985.
2. Ermanno Bencievenga, Karel Lambert, and Bas van Fraassen. *Logic, Bivalence, and Denotation*. Ridgeview, Atascadero, California, second edition, 1991.
3. George S. Boolos and Richard C. Jeffrey. *Computability and Logic*. Cambridge University Press, Cambridge, 1974.
4. Tyler Burge. Truth and singular terms. In [18], 1991.
5. William M. Farmer. Reasoning about partial functions with the aid of a computer. *Erkenntnis*, 43:279–294, 1995.
6. Solomon Feferman. Definedness. *Erkenntnis*, 43:295–320, 1995.
7. L. M. G. Feijs and H. B. M. Jonkers. *Formal Specification and Design*. Cambridge University Press, Cambridge, 1992.
8. Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, second edition, 1996.
9. M. P. Fourman, C. J. Mulvey, and D. S. Scott, editors. *Applications of sheaves : Proceedings of the Research Symposium on Applications of Sheaf Theory to Logic, Algebra, and Analysis, Durham*, volume 753 of *Lecture Notes in Mathematics*. Springer, Berlin, 1979.
10. ESPRIT CoFI Working Group. Cofi: The Common Framework Initiative for algebraic specification and development (various documents). Available at http://www.briks.dk/Projects/CoFI.
11. Raymond D. Gumb. *Evolving Theories*. Haven, New York, 1979.
12. Raymond D. Gumb. An extended joint consistency theorem for free logic with equality. *Notre Dame Journal of Formal Logic*, 20:321–335, 1979. Abstract in *Journal of Symbolic Logic*, 42:146, 1977.

13. Raymond D. Gumb. An extended joint consistency theorem for a family of free modal logics with equality. *Journal of Symbolic Logic*, 49:174–183, 1984. Abstract in *Journal of Symbolic Logic*, 46:435-436, 1981.
14. Raymond D. Gumb. *Programming Logics: An Introduction to Verification and Semantics*. Wiley, New York, 1989.
15. Raymond D. Gumb. Free logic in program specification and verification. In E. Morsher, editor, *New Directions in Free Logic*. Akademia Verlag, Bonn, 1996.
16. Raymond D. Gumb and Karel Lambert. Definitions in nonstrict positive free logic. *Modern Logic*, 7:25–55, 1997. Corrected version available at http://www.cs.uml.edu/∼gumb.
17. C. A. R. Hoare. An axiomatic basis of computer programming. *Communications of the ACM*, 12:576–583, 1969.
18. K. Lambert, editor. *Philosophical Applications of Free Logic*. Oxford University Press, Oxford, 1991.
19. Karel Lambert. On the philosophical foundations of free description theory. *History and Philosophy of Logic*, 8:57–66, 1987.
20. Hugues Leblanc. *Truth-Value Semantics*. North Holland, Amsterdam, 1976.
21. David L. Parnas. A logic for describing, not verifying, software. *Erkenntnis*, 43:321–338, 1995.
22. R. A. Pljuškevičus. A sequential variant of constructive logic calculi for normal formulas not containing structural rules. In *The Calculi of Symbolic Logic, I, Proceedings of the Steklov Institute for Mathematics 98*, pages 175–229. AMS Translations, Dordrecht, 1971.
23. Gordon D. Plotkin. Denotational semantics with partial functions. Lecture Notes, CSLI Summer School, Stanford University, July 1985.
24. Gordon D. Plotkin. Types and partial functions. Lecture Notes, Computer Science Department, University of Edinburgh, 1985.
25. Ronald Scales. *Attribution and Existence*. PhD thesis, University of California, Irvine, 1969.
26. Rolf Schock. Contributions to syntax, semantics, and the philosophy of science. *Notre Dame Journal of Formal Logic*, 5:241–289, 1964.
27. Rolf Schock. On definitions. *Archiv für Mathematische Logik und Grundlagenforschung*, 8:28–44, 1965.
28. Rolf Schock. *Logics without Existence Assumptions*. Almqvist and Wiksell, Stockholm, 1968.
29. Dana S. Scott. Existence and description in formal logic. In R. Schoenmann, editor, *Bertrand Russell, Philosopher of the Century*, pages 181–200. Allen and Unwin, London, 1967. Reprinted in [18].
30. Dana S. Scott. Identity and existence in intuitionistic logic. In [9], pages 660-696, 1979.
31. Neil Tennant. *Natural Logic*. Edinburgh University Press, Edinburgh, corrected edition, 1990.
32. A. S. Troelstra and D. van Dalen. *Constructivity in Mathematics*, volume I and II. North-Holland, Amsterdam, 1988.
33. Michael Unterhalt. *Kripke-Semantik für Logik mit partieller Existenz*. PhD thesis, Westfälische Wilhelms-Universität Münster, 1986.

Search Space Compression in Connection Tableau Calculi Using Disjunctive Constraints

Ortrun Ibens

Institut für Informatik, Technische Universität München, Germany
ibens@informatik.tu-muenchen.de, Phone: ++49 89 28927923

Abstract. Automated theorem proving with connection tableau calculi imposes search problems in tremendous search spaces. We present a new approach to search space reduction in connection tableau calculi. In our approach structurally similar connection tableaux are compressed with the help of symbolic disjunctive constraints. We describe the necessary changes of the calculus, and we develop elaborate techniques for an efficient constraint processing. Moreover, we extend the most important search pruning techniques for connection tableau calculi to our approach.

1 Introduction

Automated theorem proving (ATP) is an important research area in artificial intelligence. The objective of an ATP system is to find out whether or not a formally specified *query* (or *goal*) is a logical consequence of a set of formally specified *axioms*. For this purpose, system-specific *inference rules* are applied systematically. A sequence of inference rule applications (*inferences*) which shows that a query is a logical consequence of a set of axioms is a *proof*. The main strength of ATP systems is their ability to handle *declarative* descriptions of knowledge. However, this ability introduces the aspect of *search* into the deduction process. The set of all objects which can be derived from an input problem by the inference rules forms the *search space*. Usually, a tremendous search space has to be explored during the search for a proof. A large amount of the research performed in the field of ATP, therefore, focuses on search space reduction.

In this paper, we deal with search space reduction in connection tableau calculi [7] which are successfully used in first-order ATP. They can be viewed as refinements of the tableau method, namely *free-variable* tableau systems [3] with *connection condition*, and are distinguishable according to their search control (ie., *subgoal selection*), refinements (eg., *regularity*), and extensions (eg., *fold-up*) [7]. We reduce the connection tableau search space as follows: We compress certain structurally similar parts of it with the help of symbolic disjunctive constraints over first-order equations. Thus, we obtain a combined exploration of these parts. As a consequence, however, a constraint satisfiability test is necessary. We show that the effort of our constraint satisfiability testing does in general not outweigh the benefits gained from the search space compression.

In Section 2, we summarize the basics of connection tableau calculi. In Section 3, we present our new approach which extends connection tableau calculi

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 279–293, 2000.

to so-called *constrained-connection-tableau calculi*. We discuss the compression of input clauses, the necessary modifications of the calculus, and the handling of constraints. In Section 4, we deal with search space reduction in constrained-connection-tableau calculi. After that we evaluate our approach experimentally (see Section 5) and compare it with a related approach (see Section 6).

2 Connection Tableau Calculi

The proof objects generated by connection tableau calculi are specialized *literal trees*, called *connection tableaux*. Literal trees are trees where all nodes except for the root node are labelled with literals. A literal tree is a *connection tableau* for an input set S of clauses, if for each non-leaf node the OR-connected literal labels of its immediate successors form an instance of a clause in S, and if each non-root-non-leaf node has an immediate successor with a complementary literal label (*connection condition*). Figure 1 shows connection tableaux for a given set of clauses. Given a node N of a connection tableau T, the disjunction of the literals labels of the immediate successors of N in T is called the *tableau clause below N in T*. A branch of a tableau is *closed* if it contains two nodes with complementary literal labels; otherwise, it is *open*. A tableau is *closed* if all branches are closed; otherwise, it is *open*. A set S of clauses is unsatisfiable if and only if there is a closed tableau for S.

A connection tableau for a set S of clauses can be generated by first applying the *start rule* and then repeatedly applying either the *extension rule* or the *reduction rule*. Employment of the start rule means selecting a clause of S and attaching its literals as immediate successors to an unlabelled root node. Applications of the extension rule or the reduction rule are controlled by a *subgoal selection function* which assigns to each open connection tableau T a *subgoal* of T. (A *subgoal* is the literal at the end of an open branch.) In the extension rule a clause of S is selected which contains a literal whose complement is unifiable with the selected subgoal. The new tableau is obtained by applying the respective most general unifier to the clause and to the current tableau and attaching the instantiated literals of the clause as immediate successors to the instantiated selected subgoal. The reduction rule closes a branch by unifying the selected subgoal with the complement of a literal on the same branch (the respective most general unifier is applied to the whole tableau).

The basic search strategy in connection tableau calculi is *depth-first search with backtracking*. That is, the inference rules are applied repeatedly until either a closed tableau has been generated or no rule application is possible at the current tableau. In the latter case, backtracking is performed. A refinement of depth-first search which is more suitable for infinite search spaces is *iterative deepening search* which explores iteratively larger finite initial segments of the search space by depth-first search with backtracking.

Important *structural refinements* of clausal tableaux are *regularity* and *tautology-freeness* [7]. A tableau is *regular* if for each branch b and for each two different nodes N, N' of b the literal labels of N and N' are different. A tableau

is *tautology-free* if none of its tableau clauses contains two complementary literals. Regularity and tautology-freeness can be implemented by using inequality constraints [6]. These techniques avoid redundancy within a single tableau. In contrast to this, *local failure caching* [7,8] avoids the generation of certain tableaux which are redundant in the presence of other tableaux. It is applicable in combination with depth-first subgoal selection and can be described as follows: Assume a depth-first subgoal selection function which selects the subgoal L of a given connection tableau T, and let N be the node of T which is labelled with L. Let the connection tableau T' be obtained by applying inferences to T. Let L' be the literal label of N in T'. Then the most general substitution σ with $\sigma(L) = L'$ is called the *instantiation of* L *via* T'. If the sub-tableau below N is closed in T', we say L is *solved in* T'. If L is solved in T' and the remaining subgoals of T are still (possibly instantiated) subgoals of T', we say L is *solved via* T' and the instantiation of L via T' is called the *solution of* L *via* T'. Assume that σ is the solution of L via T'. After each attempt to close T' has failed, σ becomes an *anti-lemma at* N and is stored in the so-called *failure cache at* N. Until N is removed by backtracking, the anti-lemmata at N are used as follows: if in another solution attempt of L a tableau T'' is constructed where L is *not yet solved* or where L is *solved via* T'' such that there is an anti-lemma at N which is more general than the instantiation of L via T'', then backtracking is invoked. Local failure caching is compatible with regularity and tautology-freeness, provided these techniques are confined to the *open part* of the tableau (see [7]).

Local failure caching provides a restricted form of *tableau subsumption* [7] as follows: Let T, T', L, σ be as above. Then, local failure caching avoids that certain connection tableaux are derived from T whose *subgoal trees* are *subsumed* by the subgoal tree of T'. (The *subgoal tree* of an open connection tableau T is the literal tree which is obtained from T if all nodes which do not belong to open branches are removed. A literal tree T *subsumes* a literal tree T', if there is an *instance* T'' of T such that T' can be obtained by attaching a finite number of literal trees to non-leaf nodes of T''. The *instance of a literal tree* T *under a substitution* σ, denoted by $\sigma(T)$, is obtained by applying σ to each literal of T.)

3 Connection Tableau Calculi with Constraints

3.1 Outlines of the Approach

We will now describe how a certain kind of *redundancy of the search* arises from structurally similar input clauses, which cannot be tackled with the above structural refinements or subsumption techniques but can be with our approach.

In general, structurally similar input clauses lead to structurally similar tableaux. For example, in the input problem S (see Figure 1), the unit clauses $p(a_1, f(b_1)), \ldots, p(a_n, f(b_n))$ only differ from each other in certain sub-terms. When given the initial connection tableau T (see Figure 1), n inferences are possible alternatively at the literal $\neg p(X, Y)$. These possible inferences involve the input clauses $p(a_1, f(b_1)), \ldots, p(a_n, f(b_n))$ and result in the tableaux $T_1, \ldots, T_n$

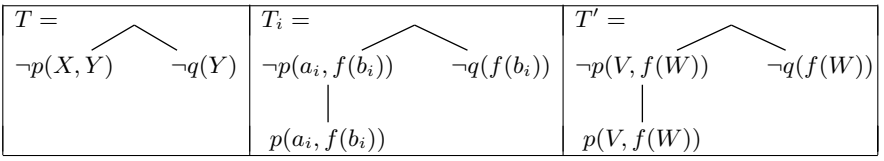


Fig. 1. T and $T_1, \ldots, T_n$, $n \geq 1$, are connection tableaux for the set $S = \{\neg p(X,Y) \vee \neg q(Y), p(a_1, f(b_1)), \ldots, p(a_n, f(b_n)), q(g(d))\}$. The connection tableau T' and the condition $\langle V, W\rangle \in \{\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\}$ represent $T_1, \ldots, T_n$.

(see Figure 1). Since $T_1, \ldots, T_n$ only differ from each other in certain sub-terms of the literals, it seems redundant to construct each of them individually.

A simultaneous processing of $T_1, \ldots, T_n$ is achieved by a compression of the structurally similar clauses $p(a_1, f(b_1)), \ldots, p(a_n, f(n_n))$ into the new clause $p(V, f(W))$ with the additional condition $\langle V, W\rangle \in \{\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\}$. The new clause expresses the common term structure of the compressed clauses, and the condition encodes their differences. When using the new clause instead of the compressed clauses, only one inference is possible at the literal $\neg p(X, Y)$ of T. The result is the connection tableau T' (see Figure 1) with the condition $\langle V, W\rangle \in \{\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\}$. Since neither at $T_1, \ldots, T_n$ nor at T' further inferences are possible, a reduction of the search space has been achieved. In general, a satisfiability test of the conditions associated with tableaux is necessary periodically to find out whether or not the current tableau and its condition still represent (encode) at least one ordinary tableau. In order not to simply move the search space from the tableau level into the satisfiability test, an intelligent representation and handling of the conditions is necessary.

In our approach, the conditions are expressed by AND-OR-connected equations over first-order terms, called *constraints*. If a clause with a constraint is attached to a tableau with a constraint, both constraints are instantiated and AND-connected, and the result is simplified. Since a constraint normal-form is not required, we use cheap simplification techniques which, on the one hand, can identify a lot of redundant or unsatisfiable sub-conditions but which, on the other hand, need not identify all redundant or unsatisfiable sub-conditions. The satisfiability test need not be performed after each inference. Since the satisfiability of a constraint follows if it has a solution (ie., an allowed instantiation of its variables), only one solution is computed in the satisfiability test. This implies that *only one* of the simultaneously processed tableaux is computed explicitly. The advantage of our approach over conventional connection tableau calculi can be seen when regarding the above example. If a satisfiability test is applied to the condition $\langle V, W\rangle \in \{\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\}$ associated with T', then one solution $\{V \leftarrow a_k, W \leftarrow b_k\}$, $1 \leq k \leq n$, (representing one connection tableau T_k) can be found at low costs. After the inference at the literal $\neg q(f(W))$ has failed, all represented tableaux $T_1, \ldots, T_n$ are backtracked simultaneously.

3.2 Compression of Clauses

In our approach, we use constraints with the following syntax (henceforth, we always refer to this definition when using the denotation *constraint*):

Definition 1 (Constraint).

1. *If t and t' are terms, then $t = t'$ is a* constraint. *We call $t = t'$ an* elementary constraint.
2. *If $c_1, \ldots, c_k$, $k \geq 0$, are constraints, then $c_1 \wedge \cdots \wedge c_k$ and $c_1 \vee \cdots \vee c_k$ are* constraints. *For $k \geq 2$, we call $c_1 \wedge \cdots \wedge c_k$ a* conjunctive constraint *and $c_1 \vee \cdots \vee c_k$ a* disjunctive constraint. *For $k = 0$, we call $c_1 \wedge \cdots \wedge c_k$ the* empty conjunction *and $c_1 \vee \cdots \vee c_k$ the* empty disjunction.

We abbreviate the empty conjunction by ϵ. If c is a conjunctive constraint $c_1 \wedge \cdots \wedge c_k$ or a disjunctive constraint $c_1 \vee \cdots \vee c_k$, then $c_1, \ldots, c_k$ are *immediate sub-constraints of c*. The notion of a *sub-constraint of a constraint* is inductively defined as follows: c itself is a *sub-constraint of c*. If $\hat{c}$ is an immediate sub-constraint of a constraint c, then $\hat{c}$ is a *sub-constraint of c*. If $\hat{c}$ is a sub-constraint of c, then each immediate sub-constraint of $\hat{c}$ is a *sub-constraint of c*.

Our constraints have the following semantics: An elementary constraint $t = t'$ is *valid* if and only if t equals t'. A constraint $c_1 \wedge \cdots \wedge c_k$ is *valid* if and only if each c_i $(1 \leq i \leq k)$ is valid. Therefore, the empty conjunction is valid. A constraint $c_1 \vee \cdots \vee c_k$ is *valid* if and only if there is a valid c_i $(1 \leq i \leq k)$. We denote the *instance* of a constraint c under a substitution σ by $\sigma(c)$. A substitution σ is called *solution* of a constraint c if $\sigma(c)$ is valid. A constraint is *satisfiable* if and only if it has a solution. Constraints c_1 and c_2 are *equivalent* if and only if each solution of c_1 is a solution of c_2 and each solution of c_2 is a solution of c_1.

We now describe the compression of structurally similar clauses. We regard clauses as structurally similar if and only if they are *generalizable*:

Definition 2 (Generalization, Generalizer).

1. *A clause C is a* generalization *of a set $\{C_1, \ldots, C_n\}$, $n \geq 1$, of clauses if and only if each clause C_i $(1 \leq i \leq n)$ is an instance of C. A set of clauses is called* generalizable *if and only if there is a generalization of it.*
2. *Let C be a generalization of a set S of generalizable clauses. C is a* most specific generalization of S *if and only if, for each generalization C' of S, there is a substitution σ such that $\sigma(C') = C$.*
3. *Let C be a generalization of a set $\{C_1, \ldots, C_n\}$ of generalizable clauses. For each i with $1 \leq i \leq n$, let $\sigma_i = \{X_{i,1} \leftarrow t_{i,1}, \ldots, X_{i,n_i} \leftarrow t_{i,n_i}\}$, $n_i \geq 0$, be a substitution with $C_i = \sigma_i(C)$. Then, we call the constraint*

$$\bigvee_{i=1}^{n} (X_{i,1} = t_{i,1} \wedge \cdots \wedge X_{i,n_i} = t_{i,n_i})$$

a generalizer for $\{C_1, \ldots, C_n\}$ with respect to C.

When compressing a set of generalizable clauses, a most specific generalization of the set and a generalizer for the set with respect to the generalization are obtained. For example, $q(g(Y, Z))$ is a most specific generalization of the set $\{q(g(a, a)), q(g(h(X), b_1)), \ldots, q(g(h(X), b_m))\}$, $m \geq 1$, and

$$(Y = a \wedge Z = a) \vee \bigvee_{i=1}^{m} (Y = h(X) \wedge Z = b_i)$$

is a respective generalizer. The result of the compression is a so-called *constrained clause* $\langle C, c\rangle$ where the generalization forms the *clause part* C and the generalizer the *constraint part* c. Constrained clauses have the following semantics: A set S of constrained clauses is *satisfiable* if and only if, for each $\langle C, c\rangle \in S$, the set $E_{\langle C,c\rangle} = \{\sigma(C) | \sigma \text{ is a solution of } c\}$ is non-empty, and the set $\bigcup_{\langle C,c\rangle \in S} E_{\langle C,c\rangle}$ is satisfiable.

In general the input set to a connection tableau calculus is not generalizable. Therefore, the input is divided into maximal sub-sets of generalizable clauses (literals may be permuted to obtain larger sub-sets). If a sub-set contains one clause C only, then the resulting constrained clause is $\langle C, \epsilon\rangle$. Let S be a set of ordinary clauses, and let S' be obtained from S by compression of sub-sets of generalizable clauses. Then, obviously S is satisfiable if and only if S' is satisfiable. In the following text, we will only deal with sets of constrained clauses which are obtained as described above, i.e., by generalizing ordinary clauses. Specifically, this implies that each constrained clause in a considered set of constrained clauses has a satisfiable constraint part.

The generalizer for a set of generalizable clauses is generally in *disjunctive normal-form*, ie., it is a disjunctive constraint where each sub-constraint is a conjunction of elementary constraints. It can be transformed into an equivalent constraint with a minimal number of elementary sub-constraints by the distributive law. The number of elementary sub-constraints has an influence on the effort of the constraint simplification and the satisfiability test (see Section 3.4). The above generalizer, for example, can be transformed into the equivalent constraint

$$(Y = a \wedge Z = a) \vee (Y = h(X) \wedge \bigvee_{i=1}^{m} Z = b_i).$$

3.3 Constrained-Connection-Tableau Calculi

We call connection tableau calculi which work on constrained clauses *constrained-connection-tableau calculi*. The input to a constrained-connection-tableau calculus is a set of constrained clauses, and the proof objects are connection tableaux with constraints, so-called *constrained connection tableaux*. A pair $\langle T, c\rangle$ is a constrained connection tableau for a set S of constrained clauses if T is a connection tableau for the set $\{C | \langle C, \hat{c}\rangle \in S\}$ and if, for each tableau clause C' of T, there is a constrained clause $\langle C, \hat{c}\rangle \in S$ such that $C' = \sigma(C)$ and c is a constraint $c_1 \wedge \sigma(\hat{c}) \wedge c_2$ (for some σ, c_1, c_2). Restrictions on c_1 and c_2 follow implicitly from the fact that the second condition has to be true for *each* tableau clause C' of T. Examples for constrained connection tableaux are given in Figure 2. We call a constrained connection tableau *closed* if its tableau part is closed and its constraint part is satisfiable.

The constrained-connection-tableau start rule generates an initial constrained connection tableau $\langle T_0, c_0\rangle$ from a constrained clause $\langle C, c\rangle$ as follows: T_0 is constructed from C according to the ordinary connection tableau start rule, and c yields c_0. The constrained-connection-tableau extension rule generates a

constrained connection tableau $\langle T_2, c_2 \rangle$ from a constrained connection tableau $\langle T_1, c_1 \rangle$ and a constrained clause $\langle C, c \rangle$ as follows: T_2 is constructed from T_1 and C according to the ordinary connection tableau extension rule, and c_2 equals the instance of $c_1 \wedge c$ under the same substitution which is applied to the tableau part. The constrained-connection-tableau reduction rule generates a constrained connection tableau $\langle T_2, c_2 \rangle$ from a constrained connection tableau $\langle T_1, c_1 \rangle$ as follows: T_2 is constructed from T_1 according to the ordinary connection tableau reduction rule, and c_2 equals the instance of c_1 under the same substitution which is applied to the tableau part.

We proceed with the soundness and completeness of constrained-connection-tableau calculi. The main part of the completeness proof is the following lemma:

Lemma 1. *Let S be a set of constrained clauses, let ξ be a subgoal selection function, and let S' be the set $\{\hat{\sigma}(C) | \langle C, \hat{c} \rangle \in S$ and $\hat{\sigma}$ is a solution of $\hat{c}\}$. Then there is a subgoal selection function ξ' such that the following holds: For each ordinary connection tableau T' for S' which can be generated by connection tableau inferences (using ξ'), a constrained connection tableau $\langle T, c \rangle$ for S can be generated by constrained-connection-tableau inferences (using ξ) such that, firstly, there is a solution σ of c with $\sigma(T) = T'$ and, secondly, for each closed branch $\sigma(b)$ of $\sigma(T)$ the respective branch b of T is closed.*

Lemma 1 can be proven by induction via the number of performed ordinary connection tableau inferences (see [4]).

Theorem 1. *A closed constrained connection tableau for a set S of constrained clauses can be generated by constrained-connection-tableau inferences (using any subgoal selection function) if and only if S is unsatisfiable.*

Proof: We only give the completeness proof. See also [4]. Let an unsatisfiable set S of constrained clauses and a subgoal selection function ξ be given. Let S' be the set $\{\hat{\sigma}(C) | \langle C, \hat{c} \rangle \in S$ and $\hat{\sigma}$ is a solution of $\hat{c}\}$, and let ξ' be a subgoal selection function according to Lemma 1. Since S is unsatisfiable, S' is also unsatisfiable. Because connection tableau calculi are complete, a closed ordinary connection tableau T' for S' can be generated by connection tableau inferences (using any subgoal selection function, in particular using ξ'). According to Lemma 1, a constrained connection tableau $\langle T, c \rangle$ for S can be generated by constrained-connection-tableau inferences (using ξ) such that, firstly, there is a solution σ of c with $\sigma(T) = T'$ and, secondly, for each closed branch $\sigma(b)$ of $\sigma(T)$ the corresponding branch b of T is closed. Since T' is closed, T is also closed. Since σ is a solution of c, c is satisfiable. Therefore, the constrained connection tableau $\langle T, c \rangle$ is closed. q.e.d.

Depth-first search with backtracking or iterative deepening search are suitable to search for a closed constrained connection tableau. Backtracking may not only be invoked if at the current constrained connection tableau no inference is possible but also if the constraint part of the current constrained connection tableau is unsatisfiable (because an unsatisfiable constraint cannot become satisfiable by applying substitutions or by conjunctively adding further constraints).

3.4 Constraint Processing

Satisfiability Test. Recall that the satisfiability of a constraint follows if it has a solution. Therefore, *one* solution of the given constraint is searched for in the satisfiability test. The computation of a solution is based on the enumeration of solutions of elementary sub-constraints by means of backtracking — until their composition is a solution of the whole constraint. The computation of a solution of an elementary constraint $s = t$ is simply a unification problem since $s = t$ has a solution if and only if s and t are unifiable (any unifier of s and t is a solution of $s = t$). Obviously, it is sufficient to enumerate one most general unifier of s and t only. [4] gives a detailed discussion of different algorithms for the satisfiability test.

Backtracking of inferences may immediately be invoked if the constraint part of the current constrained connection tableau is unsatisfiable. Due to the imposed effort, however, the frequency of the satisfiability testing should be subject to accurate considerations. If after each inference a satisfiability test is applied to the constraint part of the current constrained connection tableau, then, in case it is unsatisfiable, this is immediately recognized. However, when performing the test after each inference, the accumulated effort of the satisfiability testing may be too high (in particular if most of the tested constraints are satisfiable). Another strategy may be to test the satisfiability of the constraint part only if the tableau part is closed. Following this strategy, however, a high number of useless inferences may be applied to constrained connection tableaux with unsatisfiable constraint parts. In our approach, a satisfiability test is applied after extension steps involving a constrained clause with a unit clause part and after reduction steps. These are situations where a certain sub-tableau of the tableau part of the current constrained connection tableau has become closed. After extension steps involving a constrained clause with a non-unit clause part and a constraint part unequal to the empty conjunction only a *weak satisfiability test* is applied (see below). After the remaining inferences no satisfiability test is applied.

Weak Satisfiability Test. According to Section 3.3, the constraint part of a constrained connection tableau is in general a conjunctive constraint. A weak satisfiability test of a conjunctive constraint can be performed by testing one immediate sub-constraint only. If the tested sub-constraint is unsatisfiable, then the whole constraint is unsatisfiable. Otherwise, neither its satisfiability nor its unsatisfiability can be derived. In our approach, we test in the weak satisfiability test the immediate sub-constraint which has been added in the recent inference. For a discussion of other weak satisfiability tests refer to [4].

Constraint Simplification. According to the definition of being valid, valid sub-constraints of a given constraint can be identified by an analysis of elementary sub-constraints and a propagation of the results. An analogous propagation technique allows the identification of certain unsatisfiable sub-constraints: We call a constraint *trivially unsatisfiable* if it is either an elementary constraint

$s = t$ where s and t are not unifiable, or a conjunctive constraint with a trivially unsatisfiable immediate sub-constraint, or a disjunctive constraint where all immediate sub-constraints are trivially unsatisfiable. Obviously, a trivially unsatisfiable constraint is unsatisfiable.

In the following cases valid or unsatisfiable sub-constraints may be removed. Let c be a constraint, let $\bar{c}$ be a sub-constraint of c, and let $\hat{c}$ be an immediate sub-constraint of $\bar{c}$. If $\hat{c}$ is valid and $\bar{c}$ is not valid, we can obtain a constraint c' which is equivalent to c by the deletion of $\hat{c}$ from c. Analogously, if $\hat{c}$ is trivially unsatisfiable and $\bar{c}$ is not trivially unsatisfiable, we can obtain a constraint c' which is equivalent to c by the deletion of $\hat{c}$ from c. Since the computation of a solution is based on the enumeration of solutions of elementary sub-constraints, the deletion of valid or trivially unsatisfiable sub-constraints can significantly reduce the effort of the satisfiability testing. We therefore remove valid or trivially unsatisfiable sub-constraints from the constraint part of the current constrained connection tableau after each inference.

4 Search Space Reduction Techniques

4.1 Structural Refinements

We define a constrained connection tableau $\langle T, c\rangle$ as *regular* if there is a solution σ of c such that the connection tableau $\sigma(T)$ is regular. Constrained-connection-tableau calculi with regularity are sound and complete. The completeness proof is similar to the completeness proof of pure constrained-connection-tableau calculi (see Section 3.3). The only difference is that we infer with the help of Lemma 1 the derivability of a *regular* closed constrained connection tableau from the derivability of a *regular* closed ordinary connection tableau. In the search process the regularity of a constrained connection tableau $\langle T, c\rangle$ is tested as follows: Firstly, if T is not regular, then, for all solutions σ of c, $\sigma(T)$ is not regular. Secondly, if T is regular, the regularity test of $\langle T, c\rangle$ is combined with the satisfiability test of c. That is, in the satisfiability test of c only such a solution σ of c is admissible where $\sigma(T)$ is regular. The definitions, the completeness proof, and the testing of *tautology-freeness* are analogous. Analogous to connection tableau calculi, the structural refinements regularity and tautology-freeness of constrained-connection-tableau calculi can be implemented by means of inequality constraints.

4.2 Local Failure Caching

The following example shows that the local failure caching for ordinary connection tableau calculi has to be modified when extended to constrained-connection-tableau calculi. The constrained connection tableaux $\langle T, c\rangle$ and $\langle T', c'\rangle$ for the set S of constrained clauses (see Figure 2) can be generated by constrained-connection-tableau inferences (performing depth-first subgoal selection). $\sigma = \{X \leftarrow f(Y)\}$ is the solution of the subgoal $\neg p(X)$ of T via T'. The only possible inference at $\langle T', c'\rangle$ is an extension step involving the constrained clause $\langle q(f(d)), \epsilon\rangle$. The constraint part $d = a \vee d = b$ of the resulting constrained connection tableau

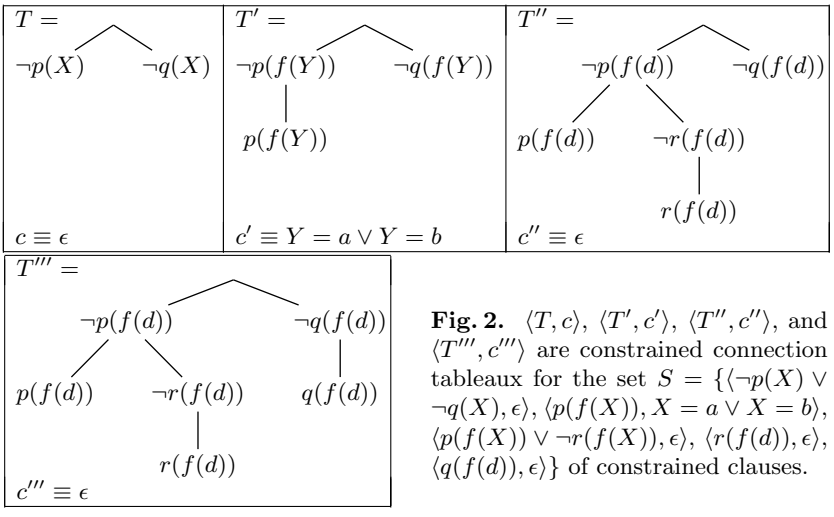


Fig. 2. $\langle T, c\rangle$, $\langle T', c'\rangle$, $\langle T'', c''\rangle$, and $\langle T''', c'''\rangle$ are constrained connection tableaux for the set $S = \{\langle \neg p(X) \vee \neg q(X), \epsilon\rangle, \langle p(f(X)), X = a \vee X = b\rangle, \langle p(f(X)) \vee \neg r(f(X)), \epsilon\rangle, \langle r(f(d)), \epsilon\rangle, \langle q(f(d)), \epsilon\rangle\}$ of constrained clauses.

is unsatisfiable. Therefore, $\langle T', c'\rangle$ cannot become closed. When applying local failure caching for ordinary connection tableau calculi, σ becomes an anti-lemma at the node N of T which is labelled with the subgoal $\neg p(X)$. In the following solution attempts of $\neg p(X)$, σ forbids an extension step at the subgoal $\neg p(X)$ of T with the constrained clause $\langle p(f(X)) \vee \neg r(f(X)), \epsilon\rangle$. Therefore, the closed constrained connection tableau $\langle T''', c'''\rangle$ (see Figure 2) cannot be generated, and the search process becomes incomplete.

The incompleteness derives from the fact that we have ignored the constraint c' which restricts the instantiation of Y to a or b. Consequently, not $\sigma = \{X \leftarrow f(Y)\}$ should become an anti-lemma at N, but both $\{X \leftarrow f(a)\}$ and $\{X \leftarrow f(b)\}$. We derive the following algorithm for anti-lemma generation from this recognition. Let constrained connection tableaux $\langle T_1, c_1\rangle$ and $\langle T_2, c_2\rangle$ for a set S_1 of constrained clauses be generated by constrained-connection-tableau inferences (using a depth-first subgoal selection function) such that $\langle T_2, c_2\rangle$ cannot become closed and there is a subgoal L of T_1 which is solved via T_2. Let σ_L be the solution of L via T_2, and let N_L be the node of T_1 which is labelled with L. Then, for each solution ρ of c_2, store $\rho \circ \sigma_L$ in the failure cache at N_L. ($\rho \circ \sigma_L$ denotes the *composition of* ρ *and* σ_L. For any term t, $(\rho \circ \sigma_L)(t)$ equals $\rho(\sigma_L(t))$.) The anti-lemmata at N_L are used as follows: Assume that in another solution attempt of L a connection tableau $\langle T_3, c_3\rangle$ has been constructed where L is not yet solved or where L is solved via T_3. Let σ'_L be the instantiation of L via T_3. Firstly, like in ordinary connection tableau calculi, if there is an anti-lemma at N_L which is more general than σ'_L, then backtracking is invoked. Secondly, if the first case does not occur, in the satisfiability test of c_3 only such a solution ρ of c_3 is admissible where, for each anti-lemmata α at N_L, α is not more general than $\rho \circ \sigma'_L$. The described generation and use of anti-lemmata preserve the completeness of the search process (see also [4]).

The above anti-lemma generation leads to storing an unbound number of anti-lemmata at N_L, because, for any substitution θ and for each solution ρ of

c_2, $\theta \circ \rho$ is also a solution of c_2. Obviously, the anti-lemma $\theta \circ \rho \circ \sigma_L$ is redundant since there is also the anti-lemma $\rho \circ \sigma_L$ in the same failure cache. We see that it is sufficient to generate anti-lemmata from the solutions of c_2 which are contained in a so-called *generating set of solutions of* c_2.

Definition 3 (Generating Set of Solutions). *A set Σ of solutions of a constraint c is a* generating set of solutions of *c, if and only if,*

1. *for each solution ρ of c, there are a solution $\sigma \in \Sigma$ and a substitution θ such that $\rho = \theta \circ \sigma$, and*
2. *for each solution $\sigma \in \Sigma$ and for each non-empty substitution θ, $(\theta \circ \sigma) \notin \Sigma$.*

A generating set of solutions of a constraint is finite and can be computed [4]. These considerations allow the following modification of the anti-lemma generation: Let Σ be a generating set of solutions of c_2. Then, for each $\rho \in \Sigma$, store $\rho \circ \sigma_L$ in the failure cache at N_L. Using this modification, only a finite number of anti-lemmata is stored in the failure cache at N_L.

In the above example, the constrained connection tableau $\langle T', c' \rangle$ cannot become closed because the only possible inference at $\langle T', c' \rangle$ instantiates c' to an unsatisfiable constraint. In the following example the fact that the regarded constrained connection tableau cannot become closed is independent from its constraint part. This situation allows an improved anti-lemma generation.

Let S' be obtained from S by removing $\langle q(f(d)), \epsilon \rangle$ and adding $\langle q(d), \epsilon \rangle$. $\langle T, c \rangle$ and $\langle T', c' \rangle$ (see Figure 2) are constrained connection tableaux for S' which can be generated by constrained-connection-tableau inferences (using a depth-first subgoal selection function). Since no inference is possible at $\langle T', c' \rangle$, it cannot become closed. Therefore, anti-lemmata may be stored at the node N of T which is labelled with the subgoal $\neg p(X)$. Since the fact that $\langle T', c' \rangle$ cannot become closed does not depend on its constraint part c' (but only on the fact that no inference is possible), the solutions of c' need not be considered in the anti-lemma generation. We may therefore directly store the solution $\sigma = \{X \leftarrow f(Y)\}$ of $\neg p(X)$ via T' in the failure cache at N. Like in the above example, the anti-lemma σ forbids the extension step at the subgoal $\neg p(X)$ of T with the constrained clause $\langle p(f(X)) \vee \neg r(f(X)), \epsilon \rangle$. In the current example, however, this does not lead to the incompleteness of the search process, since the anti-lemma σ avoids that the constrained connection tableau $\langle T'', c'' \rangle$ (see Figure 2) is generated which cannot become closed using constrained clauses from S'.

From this observation, we derive the following improved algorithm for anti-lemma generation: Let S_1, $\langle T_1, c_1 \rangle$, $\langle T_2, c_2 \rangle$, L, σ_L, and N_L be as before. If the fact that $\langle T_2, c_2 \rangle$ cannot become closed does not depend on c_2, then store σ_L in the failure cache at N_L. Otherwise, let Σ be a generating set of solutions of c_2 and, for each $\rho \in \Sigma$, store $\rho \circ \sigma_L$ in the failure cache at N_L.

Due to this improvement the pruning potential of local failure caching for constrained-connection-tableau calculi is higher than the pruning potential of local failure caching for ordinary connection tableau calculi. In order to recognize this, let again S_1, $\langle T_1, c_1 \rangle$, and $\langle T_2, c_2 \rangle$ be as before. Assume that the set S_1 of constrained clauses has been generated from a set S_0 of ordinary clauses by

clause compression as described in Section 3.2. Let Σ be a generating set of solutions of c_2. Then, for each $\rho \in \Sigma$, $\rho(T_2)$ is an ordinary connection tableau for S_0 and may be generated by connection tableau inferences using a depth-first subgoal selection function. Since $\langle T_2, c_2 \rangle$ cannot become closed using clauses from S_1, for each $\rho \in \Sigma$, $\rho(T_2)$ cannot become closed using clauses from S_0, and local failure caching for ordinary connection tableau calculi avoids that in the following search process tableaux are generated whose subgoal trees are subsumed by the subgoal tree of $\rho(T_2)$. Analogously, for each $\rho \in \Sigma$ and for each constrained connection tableau $\langle T_3, c_3 \rangle$ generated in the following constrained-connection-tableau search process, local failure caching for constrained-connection-tableau calculi eliminates from the set $\{\sigma(T_3) | \sigma \text{ is a solution of } c_3\}$ of ordinary connection tableaux which are encoded by $\langle T_3, c_3 \rangle$ all tableaux whose subgoal trees are subsumed by the subgoal tree of $\rho(T_2)$. If the fact fact that $\langle T_2, c_2 \rangle$ cannot become closed is independent from c_2, the pruning potential of local failure caching for constrained-connection-tableau calculi is even higher: Then from the set $\{\sigma(T_3) | \sigma \text{ is a solution of } c_3\}$ all tableaux are eliminated whose subgoal trees are subsumed by the subgoal tree of the *more general* tableau T_2.

Local failure caching for constrained-connection-tableau calculi is compatible with regularity and tautology-freeness, provided these are confined to the open part of the tableau part of the respective constrained connection tableau.

4.3 Structural Information from Deterministic Sub-Constraints

The following technique reduces the constrained-connection-tableau search space by moving certain non-disjunctive parts of the constraints into the tableaux. Let c be a constraint with an elementary sub-constraint $s = t$. We call $s = t$ *deterministic in* c if c is a constraint $\hat{c}_1 \wedge (s = t) \wedge \hat{c}_2$. If $s = t$ is deterministic in c and ρ is a most general unifier of s and t, then, for each solution σ of c, there is a solution θ of $\rho(\hat{c}_1 \wedge \hat{c}_2)$ such that $\theta \circ \rho$ equals σ. Deterministic elementary sub-constraints arise from the instantiation and simplification of constraints during the search process. We can make use of a constraint $s = t$ being deterministic in the constraint part c of a constrained connection tableau $\langle T, c \rangle$ if we apply a most general unifier ρ of s and t to $\langle T, c \rangle$. Since the sets $\{\sigma(T) | \sigma \text{ is a solution of } c\}$ and $\{(\theta \circ \rho)(T) | \theta \text{ is a solution of } \rho(c)\}$ of ordinary connection tableaux which are encoded by $\langle T, c \rangle$ and $\langle \rho(T), \rho(c) \rangle$, respectively, are equal, we may proceed the search process with inferences at $\langle \rho(T), \rho(c) \rangle$ instead of $\langle T, c \rangle$.

This usually achieves a search space reduction: Firstly, since the literals of $\rho(T)$ are more instantiated than the literals of T, certain inferences which are possible at T may be impossible at $\rho(T)$. When applied to T, these inferences lead to unsatisfiable constraint parts of the resulting constrained connection tableaux. Since it takes some effort to recognize the unsatisfiability of the constraint parts, it is better to avoid these inferences in advance. Secondly, if $\langle T, c \rangle$ violates regularity or tautology-freeness, $\langle \rho(T), \rho(c) \rangle$ violates them too, and the structural information provided by ρ may allow to find this out without an investigation of the constraint part. Thirdly, due to the structural information provided by

ρ, anti-lemmata may prune solution attempts of subgoals directly, ie., without forming compositions with solutions of the constraint part.

When in the following search process backtracking is invoked due to the structural information provided by ρ, we have to treat this during the generation of anti-lemmata as if the backtracking depends on the constraint part of the respective constrained connection tableau. The reason is that ρ represents the sub-constraint $s = t$ of the constraint c.

4.4 Failure Caching for Constraint Solution Attempts

The following technique reduces the search performed during the constraint satisfiability testing by passing information from one test to certain following tests. Assume that $\langle T_2, c_2 \rangle$ has been constructed by applying inferences to a constrained connection tableau $\langle T_1, c_1 \rangle$. Since c_2 is equivalent to $\sigma(c_1) \wedge c$ (for a substitution σ and a constraint c), each solution of c_2 is a solution of c_1, and consequently each substitution which is not more general than any solution of c_1 cannot be more general than any solution of c_2. This observation permits us to pass information about backtracked solution attempts from the satisfiability test of c_1 to the satisfiability test of c_2. That is, when computing a solution of c_1, we may store each computed composition of solutions of elementary sub-constraints which cannot be extended to a solution of the whole constraint as a *failure substitution*. In the satisfiability test of c_2, backtracking may be invoked immediately whenever a composition θ of solutions of elementary sub-constraints of c_2 is computed such that a stored failure substitution is more general than θ. The failure substitutions which have been generated during the satisfiability test of c_1 have to be deleted as soon as $\langle T_1, c_1 \rangle$ is removed by backtracking of inferences. This technique implements a *failure caching for constraint solution attempts*. Its compatibility with regularity and tautology-freeness is obvious. To ensure the compatibility with local failure caching, "certain" failure substitutions have to be removed in "certain" situations. Due to the limited space, we cannot go deeper into this here.

5 Evaluation

We integrated our approach into the connection tableau prover SETHEO [5,8]. Thus we obtained the system C-SETHEO. In the following experiments, the performance of SETHEO and C-SETHEO (each applying the respective search space reduction techniques described in this paper and a depth-first subgoal selection function) is compared when run on different problems from version 2.1.0 of the benchmark library TPTP [10] using a SUN Ultra 1 workstation (143 MHZ). The problems of the TPTP are formulated in first-order clause logic. When given an input problem, C-SETHEO first replaces each maximal set of structurally similar input clauses with an equivalent constrained clause. Then, it processes the result of this transformation. SETHEO tries to solve the original input problem directly. We investigate problems from the field, planning and geometry domains. In each of these domains, not only unit clauses but also non-unit clauses can be compressed into constrained clauses. The TPTP library contains 281 field problems,

Table 1. Numbers of solved non-trivial field, planning, and geometry problems.

Domain	SETHEO			C-SETHEO		
	$\leq$ 10 s	$\leq$ 60 s	$\leq$ 150 s	$\leq$ 10 s	$\leq$ 60 s	$\leq$ 150 s
Field	4	11	20	18	26	26
Planning	0	1	3	1	5	6
Geometry	0	2	5	4	9	10
total	4	14	28	23	40	42

30 planning problems, and 165 geometry problems. We consider a problem as *trivial* if it can be solved by SETHEO as well as by C-SETHEO in less than 10 seconds. Thus, 55 of the field problems, 23 of the planning problems, and 50 of the geometry problems are considered trivial. Table 1 shows how many of the remaining problems can be solved within 10, 60, or 150 seconds. More details on the results, e.g. run-times of single problems, are given in [4].

For each of the tested domains and in each investigated time interval, C-SETHEO solves more problems than SETHEO. C-SETHEO even solves in 10 seconds nearly as many field or geometry problems as SETHEO in 150 seconds. That is, the integration of disjunctive constraints has achieved an important speed-up.

6 Related Work

Data-base unification [2] has been developed as an improvement of the *connection method* [1] which is closely related to connection tableau calculi. In this approach, input clauses are compressed like in our approach. The arising conditions are expressed by specialized data-structures for the compact representation of substitutions, so-called *abstraction trees* [9]. In the terminology of connection tableau calculi, the approach can be described as follows: If a clause with a condition is attached to a connection tableau with a condition, the respective abstraction trees are merged by the operation *DB-join* [1]. Since abstraction trees *explicitly* represent the allowed instantiations of the variables occurring in a condition, all redundant or unsatisfiable parts of the resulting condition have to be eliminated. This means that all satisfiable sub-conditions are determined. Thus, in almost each inference the data-base unification determines all ordinary connection tableaux which are represented by the current connection tableau with its condition. In contrast to this, *only one* represented connection tableau is determined in the satisfiability test of our approach. Therefore, in our approach, considerably less connection tableaux are in general investigated during the search for a proof.

7 Conclusion

We have presented an approach to search space compression in connection tableau calculi which is based on the compression of structurally similar input clauses into clauses with constraints. The resulting constrained-connection-tableau calculi have been proven as complete. In order not to move the search space

which has been saved by the compression of clauses into the constraint handling, we have developed elaborate strategies for the simplification and the satisfiability testing of our constraints. Furthermore, we have extended the most important search pruning methods for connection tableau calculi (namely regularity, tautology-freeness, and local failure caching) to our new calculi, and we have presented two techniques for redundancy elimination in constrained-connection-tableau calculi. An experimental evaluation shows that our approach increases the performance of the connection tableau prover SETHEO in the investigated domains. In a discussion of related work, we have shown that our new approach overcomes the deficiencies of the most closely related previous approach.

The power of our approach mainly derives from the fact that in the satisfiability test only *one solution* of the constraint associated with the current tableau (representing one encoded ordinary tableau only) is computed explicitly. The fact that the generation of anti-lemmata usually requires obtaining a *generating set of solutions* of the respective constraint seems to nullify this advantage. However, firstly, anti-lemma generation usually occurs by far not as often as constraint satisfiability testing. Secondly, improvements of the anti-lemma generation which only need a generating set of solutions of a certain *sub-constraint* are possible. These improvements will be worked out in the near future.

Acknowledgments

I want to thank Marc Fuchs for lots of fruitful discussions and comments on drafts of this paper.

References

1. W. Bibel. *Deduction: Automated Logic*. Academic Press, London, 1993.
2. W. Bibel, S. Brüning, J. Otten, T. Rath, T. Schaub. Compressions and extensions. In *Applied Logic Series 8*, pages 133–179. Kluwer Academic Publishers, 1998.
3. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1996.
4. O. Ibens. *Connection Tableau Calculi with Disjunctive Constraints*. PhD thesis, Institut für Informatik, TU München, 1999.
5. O. Ibens, R. Letz. Subgoal Alternation in Model Elimination. In *TABLEAUX'97*, *LNAI 1227*, pages 201–215. Springer, 1997.
6. Reinhold Letz. *First-Order Calculi and Proof Procedures for Automated Deduction*. PhD thesis, Technische Hochschule Darmstadt, 1993.
7. R. Letz, K. Mayr, C. Goller. Controlled Integration of the Cut Rule into Connection Tableau Calculi. *JAR*, 13:297–337, 1994.
8. M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, K. Mayr. SETHEO and E-SETHEO – The CADE-13 Systems. *JAR*, 18:237–246, 1997.
9. H. J. Ohlbach. Abstraction Tree Indexing for Terms. In *Proc. ECAI-90*, pages 479–484, 1990.
10. G. Sutcliffe, C. B. Suttner, T. Yemenis. The TPTP problem library. In *Proc. CADE-12*, *LNAI 814*, pages 778–782. Springer, 1994.

Matrix-Based Inductive Theorem Proving

Christoph Kreitz[1] and Brigitte Pientka[2]

[1] Department of Computer Science, Cornell-University
Ithaca, NY 14853-7501, USA
kreitz@cs.cornell.edu
[2] Department of Computer Science, Carnegie Mellon University
Pittsburgh, PA, USA
bp@cs.cmu.edu

Abstract. We present an approach to inductive theorem proving that integrates rippling-based rewriting into matrix-based logical proof search. The selection of appropriate connections in a matrix proof is guided by the symmetries between induction hypothesis and induction conclusion while unification is extended by a rippling/reverse-rippling heuristic. Conditional substitutions are generated whenever a uniform substitution is impossible. We illustrate the combined approach by discussing several inductive proofs for the integer square root problem.

1 Introduction

Formal methods have become increasingly important for the verification and synthesis of software for safety-critical applications. Over the past years, a number of proof assistants such as NuPRL [12] or Coq [13] have been developed to support the logical reasoning process involved in these methods. But they suffer from the low degree of automation, which is due to their expressive underlying calculi. It is therefore desirable to extend the reasoning power of proof assistants by integrating well-understood techniques from automated theorem proving.

Logical proof search methods [4,22,28] have been successfully used for proving formulas in constructive first order logic. But these methods cannot deal with the inductive proofs obligations that typically occur when reasoning about recursive programs. Inductive theorem proving techniques, on the other hand, have been very successful in solving such proof obligations. *Rippling* [9,10] is a powerful annotated rewriting technique that guides the rewriting process from the induction conclusion towards the application of the induction hypothesis. However, little focus has been devoted to the logical proof search part and the automatic instantiation of existentially quantified variables, which is necessary for synthesizing programs from their specifications. In fact, many synthesis problems require both first-order and inductive proof methods to be applied simultaneously, as both techniques are not strong enough to solve the problem independently.

Example 1. Consider the formula $\forall x\,\exists y\; y^2{\leq}x \wedge x{<}(y{+}1)^2$, which specifies an algorithm for computing the integer square root of a natural number x. A straightforward inductive proof for this formula will lead to the step case

$$\exists y\; y^2{\leq}x \wedge x{<}(y{+}1)^2 \quad \vdash \quad \exists y\; y^2{\leq}x{+}1 \wedge x{+}1{<}(y{+}1)^2$$

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 294–308, 2000.

No single rippling sequence can rewrite the induction conclusion into the hypothesis, as the choice of y in the conclusion (Y_c) strongly depends on the properties of the y in the hypothesis (y_h). If $(y_h{+}1)^2{\leq}x{+}1$ then Y_c must be $y_h{+}1$ to make the first conjunct valid, while otherwise Y_c must be y_h to satisfy the second conjunct. Rippling would be able to rewrite the conclusion into the hypothesis in each of these two cases, but it can neither create the case analysis nor infer the value to be instantiated for Y_c.

Logical proof methods would not be able to detect the case distinction either, as they would have to prove the lemmata $x{<}(y_h{+}1)^2 \Rightarrow x{+}1{<}(y_h{+}1{+}1)^2$ for the first case, $y_h^2{\leq}x \Rightarrow y_h^2{\leq}x{+}1$ for the second, and also $(y_h{+}1)^2{\leq}x{+}1 \vee (y_h{+}1)^2{>}x{+}1$. Of course, it is easy to prove the induction step case if the crucial lemmata are already provided. But one would have to consult hundreds of lemmata about arithmetic to find the ones that complete the proof, which would make the proof search very inefficient.

To overcome these deficiencies we aim at combining logical proof search techniques with controlled rewrite techniques such as rippling in order to improve the degree of automation when reasoning inductively about program specifications. In [24] we have shown that a combination of a rippling-based extended matching procedure and simple sequent proof tactic can already be used to solve synthesis problems like the above automatically. The logical proof search technique in this approach, however, was not designed to be complete, as it focuses only on a top-down decomposition of formulae, and should therefore be replaced by a complete search procedure for constructive first-order logic.

Among the well-known proof procedures for intuitionistic first-order logic [22], [28] matrix-based proof techniques such as the *connection method* [4,20] can be understood as very compact representations of sequent proof search. This makes them not only much more efficient but also allows to convert their results back into sequent proofs [27], which means that (constructive) matrix methods can be used to guide proof and program development in interactive proof assistants [7,19]. These advantages suggest an integration of our extended rippling technique into matrix-based proof procedures.

In this paper we will present concepts for combining rippling techniques and matrix-based constructive theorem proving. In Section 2 we will review our extended rippling techniques while Section 3 summarizes the theoretical foundations of matrix-based logical proof search. In Section 4 we extend these theoretical foundations according to our extended rippling techniques. In Section 5 we will describe an inductive proof method that is based on these extensions.

2 Rippling and Inductive Theorem Proving

Rippling is an annotated rewriting technique specifically tailored for inductive theorem proving. Differences between the induction hypothesis and the induction conclusion are marked by meta-level annotations, called *wave annotations*. Expressions that appear in both are called *skeleton*. Expressions that appear only in the conclusion are called *wave fronts*. The induction variable that is surrounded by a wave front is called *wave hole*. $\lfloor$*Sinks*$\rfloor$ are parts of the goal that correspond to universally quantified variables in the hypothesis. We call

the annotated rewrite rules *wave rules*. To illustrate, consider a wave rule that is derived from the recursive definitions of $+$.

$$\overline{|\underline{U}+1|}^{\uparrow} + V \overset{R}{\longmapsto} \overline{|\underline{(U+V)}+1|}^{\uparrow}$$

In this rule, $\overline{|\ldots+1|}$ marks a wave front. The underlined parts $\underline{U}$ and $\underline{U+V}$ mark wave holes. Intuitively, the position and orientation of the wave fronts define the direction in which the wave front has to move within the term tree. An up-arrow $\uparrow$ indicates that the wave front has to move towards the root of the term tree (*rippling-out*). A down-arrow $\downarrow$ moves the wave front inwards or sideways towards the sink in the term tree (*rippling-in*). If the annotations can be moved either to the root of the term tree or to a sink, then rippling terminates successfully and the induction hypothesis matches the induction conclusion. Rippling terminates unsuccessfully if no wave rule is applicable anymore and the hypothesis does not match the conclusion.

Basin & Walsh [2] have developed a calculus for rippling and defined a well-founded *wave measure* under which rippling terminates if no meta-variables occur in the goal. The measure associates weights to the wave fronts to measure its *width* or its *size*. Rewriting is restricted to the application of wave rule that are skeleton preserving and measure decreasing according to the defined wave measure. For instantiating existentially quantified variables within the framework of rippling, mainly two approaches have been suggested. Bundy et al. [10] proposes special existential wave rules that can be derived from non-existential ones. This, however, increases the search problem in the presence of existential quantifiers. Other approaches [1,17,25] use meta-annotations, which requires higher-order unification and may lead to non-terminating rippling sequences.

The approach presented in [24] addresses these drawbacks by combining rippling with first-order proof techniques and extending the matching procedure for instantiating existentially quantified variables. It proceeds in three steps.

First, the step case formula of an inductive proof is decomposed into several subgoals and meta-variables are introduced in place of existentially quantified variables. A straightforward sequent proof tactic for constructive logic is used for this purpose, which allows to integrate the results into the NuPRL proof development system [12] and to extract programs from the inductive proof.

Next a *simultaneous match* is used to compute substitutions for the meta-variables incrementally. Given a substitution σ and a term C from the induction conclusion there are three possibilities. One can (1) prove the goal $\sigma(C)$ by a decision procedure, (2) unify $\sigma(C)$ and the corresponding hypothesis term H from the induction hypothesis, or (3) compute a rippling sequence and a substitution σ' such that $\sigma'(C)$ can be rippled to H. The key part of the third alternative is the *rippling/reverse rippling heuristic* illustrated below.

$$C \overset{R}{\longmapsto} C_0 \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} C_i \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} C_n \overset{R}{\longmapsto} H$$

rippling *reverse rippling*

The rippling sequence is generated in two phases: First the term C in the induction conclusion is rewritten to some formula C_i. If C_i does not match the

corresponding term H in the induction hypothesis then a rippling sequence $C_i \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} C_n \overset{R}{\longmapsto} H$ must be computed by reasoning backwards from the term H towards C_i. In this *reverse rippling* process the meta-variables in C are instantiated by a substitution σ'. If the generated rippling sequence is empty, then either H implies $\sigma(C)$ or $\sigma(C)$ and H can be unified.

The search for a rippling sequence is based on the *rippling-distance* strategy introduced in [21], which uses a new measure to ensure termination. In each rippling step the *distance* between the wave front and the sink must decrease. This allows a uniform treatment of the different rippling techniques, a more goal-oriented proof search, and avoids non-termination problems in the presence of unknown wave fronts. For a detailed analysis of this strategy we refer to [21,7].

After extended matching has generated a rippling sequence and a substitution σ for one subgoal it is checked whether all remaining subgoals are true under σ as well. To restrict the search space, all subgoals are required to be provable by standard arithmetic, rippling, and matching. If a subgoal cannot be proven by these techniques, it is selected as constraint for σ. The approach then proves all subgoals under the negated constraint and continues until it has found a complementary set of constrained substitutions that solve the induction step. The generated constraints form a case analysis in a sequent proof and lead to a conditional in the synthesized program. This simple heuristic already turned out to be sufficient for many induction problems.

Example 2. Consider again the integer square-root problem from example 1. To prove $\exists y\ y^2 \leq x \wedge x < (y+1)^2 \vdash \exists y\ y^2 \leq x+1 \wedge x+1 < (y+1)^2$ the induction hypothesis and the conclusion are decomposed, which also unfolds the abbreviation $y^2 \leq x$ to $\neg(x < y^2)$. The existentially quantified variable in the conclusion is replaced by a meta-variable Y_c. This results in two subgoals:

$$\underline{\neg(x < y_h^2)},\ x < (y_h+1)^2 \quad \vdash \quad \neg(|\underline{x+1}| < Y_c^2) \tag{1}$$

$$\neg(x < y_h^2),\ \underline{x < (y_h+1)^2} \quad \vdash \quad \overline{|\underline{x+1}|} < (Y_c+1)^2 \tag{2}$$

Next the conclusion is annotated such that its skeleton matches the corresponding part in the induction hypothesis. Corresponding parts are underlined. In the induction hypothesis y_h is marked as a sink variable because its mirror image in the conclusion is the meta-variable that we want to instantiate. The following wave rules are derived from the definitions of functions used in the specification:

$$\overline{|\underline{U+W)}|} < \overline{|\underline{V+W)}|} \quad \overset{R}{\longmapsto} \quad U < V \tag{3}$$

$$\overline{|\underline{s(U)}|} + V \quad \overset{R}{\longmapsto} \quad \overline{|\underline{s(U+V)}|} \tag{4}$$

$$(\overline{|\underline{s(A)}|})^2 \quad \overset{R}{\longmapsto} \quad \overline{|\underline{A^2+2A+1}|} \tag{5}$$

Starting with subgoal (2) the method then tries to match $C = \overline{|\underline{x+1}|} < (Y_c+1)^2$ and the induction hypothesis $= x < (y_h+1)^2$. The latter represents the final formula in the rippling sequence. As no wave rule is applicable, rippling leaves the subgoal unchanged and reverse rippling reasons backwards from to C. To determine which formula may precede , the right hand sides of the wave rules are inspected. Rule (3) suggests that the predecessor of has the form $\overline{|\underline{x+W}|} < \overline{|\underline{(y_h+1)^2+W)}|}$. This formula can then be rippled by wave rule (5), which instantiates W with $2(y_h+1)+1$.

Rippling towards the sink variable y_h is now straightforward. By rule (4) the wave front $\underline{|\ldots+1|}$ is moved to a position where it surrounds y_h, which leads to $C_0 = \underline{|\underline{x}+2(y_h+1)+1|} < (\lfloor \underline{|\underline{y_h}+1|} \rfloor +1)^2$. As rippling has terminated successfully, Y_c can be instantiated by the sink value, which results in $\sigma_1 := \{Y_c \backslash y_h+1\}$. To prove that C_0 implies $\sigma_1(C)$ a decision procedure for induction-free arithmetic [11] is used.

As subgoal (1) is not valid under σ_1, its instantiated conclusion $\neg(x+1<(y_h+1)^2)$ is added as constraint. Next both subgoals must be proved for the case $x+1<(y_h+1)^2$. Normal matching $x+1<(Y_c+1)^2$ from subgoal (2) and $x+1<(y_h+1)^2$ yields the substitution $\sigma_2 := \{Y_c \backslash y_h\}$, which also makes the subgoal (1) true under the given constraint.

Thus the step case of the induction is provable under the set of conditional substitutions $\{[\neg(x+1<(y_h+1)^2), \{Y_c \backslash y_h+1\}],\ [x+1<(y_h+1)^2, \{Y_c \backslash y_h\}]\}$.

In a final step the conditional substitutions and rippling sequences that prove the induction step are translated into a sequent proof. Following the *proofs-as-programs* principle [3] one can then extract a proof expression that describes an algorithm for the given specification.

The extended rippling technique, which is described in detail in [24], has been implemented as NuPRL-tactic and applied to several program specifications such as quotient remainder, append, last, $\log_2$, and unification. It has also been used for the instantiation of universally quantified variables in the hypothesis list, which is necessary for dealing with non-standard induction schemes that enable a synthesis of more efficient while loops. However, as the sequent proof tactic underlying this approach is neither complete nor very efficient, because it is driven only by a top-down decomposition of formulae. In the rest of this paper we will show how to integrate extended rippling into a complete, and more efficient proof procedure for constructive first-order logic.

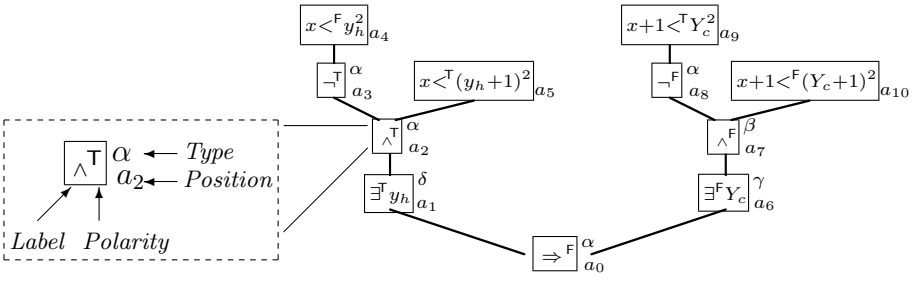
3 Matrix-Based Constructive Theorem Proving

Matrix-based proof search procedures [4,6] can be understood as compact representations of tableaux or sequent proof techniques. They avoid the usual redundancies contained in these calculi and are driven by *complementary connections*, i.e. pairs of atomic formulae that may become leaves in a sequent proof, instead of the logical connectives of a proof goal. Although originally developed for classical logic, the *connection method* has been extended to a variety of non-classical logics such as intuitionistic logic [22], modal logics [20], and fragments of linear logic [18]. In this section we will briefly summarize its essential concepts.

A *formula tree* is the tree-representation of a formula F. Each *position* u in the tree is marked with a unique name and a *label* that denotes the connective of the corresponding subformula or the subformula itself, if it is atomic. In the latter case, u is called an *atom*. The *tree ordering* $<$ of F is the partial ordering on the positions in the formula tree. As in the tableaux calculus each position is associated with a *polarity* (F or T) and a *principal type* (α, β, γ, or δ). Formulae of type γ can be used in a proof several times in different ways. A quantifier *multiplicity* μ encodes the number of distinct instances of γ-subformulae that need to be considered during the proof search. By F^μ we denote an *indexed formula*, i.e. a formula and its multiplicity. The formula tree for

$$F_0 \;\equiv\; \exists y\, \neg(x{<}y^2) \wedge x{<}(y{+}1)^2 \;\;\Rightarrow\;\; \exists y\, \neg(x{+}1{<}y^2) \wedge x{+}1{<}(y{+}1)^2$$

from the example 1 (after unfolding $\leq$), together with polarities and principal types is presented below. As usual we denote γ-variables by capital letters.[1]



The *matrix(-representation)* of a formula F is a two-dimensional representation of its atomic formulae without connectives and quantifiers. In it α-related positions appear side by side and β-related positions appear on top of each other, where two positions u and v are *α-related* (*β-related*) if the greatest common ancestor of u and v wrt. $<$ is of principal type α (β). The matrix-representation of F_0 is shown in Figure 1.

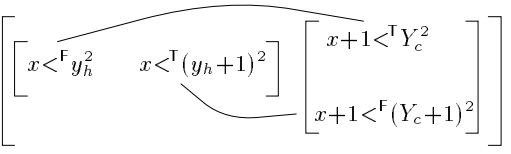


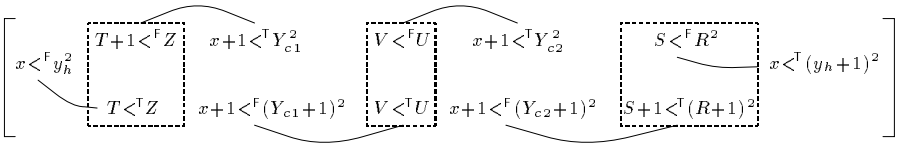
Fig. 1. Matrix of the formula F_0

The *matrix-characterization* of logical validity depends on the concepts of paths, connections, and complementarity. A *path* through a formula F is a horizontal path through its matrix-representation, i.e. a maximal set of mutually α-related atomic positions. A *connection* is a pair of atomic positions labelled with the same predicate symbol but with different polarities. In a matrix we indicate connections by arcs between two atoms. A connection is *complementary* if the connected atoms can be made equal by applying some *admissible* substitution σ to γ-variables, where admissibility encodes the eigenvariable condition on the substituted terms. With these definitions the following *matrix-characterization of logical validity* has been shown.

Theorem 1. *A formula F is valid iff there is a multiplicity μ, an admissible substitution σ, and a set of complementary connections such that every path through F^μ contains a connection from this set.*

Example 3. The formula F_0 from example 1 can be proven in first-order logic if we add to it the crucial lemmata $\forall z \forall t\; t{+}1{<}z \Rightarrow t{<}z$ and $\forall s \forall r\; s{<}r^2 \Rightarrow s{+}1{<}(r{+}1)^2$ as well as the case analysis $\forall u \forall v\; v{<}u \vee \neg(v{<}u)$, as the following matrix proof with $\mu{=}2$ in the conclusion shows.

[1] Note that extended rippling considers γ-variables to be meta-variables.



There are 32 paths through the extended formula, each containing one of the six connections depicted above. The terms of these connections can be unified by the substitution $\sigma = \{Z\backslash y_h^2, T\backslash x, Y_{c1}\backslash y_h, V\backslash x{+}1, U\backslash (y_h{+}1)^2, Y_{c2}\backslash y_h{+}1, S\backslash x, R\backslash y_h{+}1\}$. Thus the formula F_0 has become valid after being extended by the three arithmetical lemmata.

So far we have only characterized validity in classical logic. In *constructive* logic one not only has to check whether the terms of connected atoms can be unified by a *quantifier substitution* σ_Q but also that both atoms be reached by applying the same sequence of sequent rules, since only then they can form a leaf in a sequent proof. Technically, this can be done by computing an additional *intuitionistic substitution* σ_J, which unifies the prefixes of the connected atoms. A *prefix* is a string of positions between the root and the atom that are labelled with $\forall$, $\Rightarrow$, $\neg$, or atomic formulae, where positions of polarity F are considered as constants and the others as variables. Thus, a connection is constructively *complementary* if there is an admissible *combined substitution* $\sigma{:=}(\sigma_Q, \sigma_J)$ that unifies both the terms and the prefixes of the connected atoms. Admissibility now involves a few additional conditions (see [29]). With these modifications theorem 1 holds accordingly for constructive logic.

The *connection method* for constructive first-order logic describes an efficient, connection-driven technique for checking the complementarity of all paths through a formula F. Starting with all possible paths through F it eliminates step by step all paths that contain a given connection, provided it can be shown to be complementary. A detailed description of the algorithms for checking paths and unifying prefixes as well as their justifications can be found in [23,20].

4 Extending the Matrix-Characterization

In order to integrate our extended rippling techniques described in Section 2 into matrix-based proof procedures we have to modify the matrix-characterization of logical validity in several ways.

A first extension comes from the observation that a pair $\{A^{\mathsf{T}}, \bar{A}^{\mathsf{F}}\}$ of connected atoms does not necessarily have to be equal under a substitution σ. As complementary connections correspond to leaves in a sequent proof we only have to require that the left side of the leaf sequent, i.e. A, *implies* the right side $\bar{A}$ under σ, where the implication can be proven logically, by decision procedures, or by rewriting $\bar{A}$ into A. The corresponding extension of the matrix-characterization requires us to consider *directed connections* $(u^{\mathsf{T}}, v^{\mathsf{F}})$ and a notion of *implication with respect to a given theory* $\mathcal{T}$, written as $\Rightarrow_{\mathcal{T}}$. In principle, $\Rightarrow_{\mathcal{T}}$ denotes any implication that can be proven within the theory $\mathcal{T}$. Usually, however, we restrict ourselves to implications that can be proven by standard decision procedures or

by rippling sequences generated by extended matching. In particular, we consider *arithmetical implication* $A \Rightarrow_{\mathcal{A}} \bar{A}$ to denote that either $A \Rightarrow \bar{A}$ is provable by a decision procedure for some sub-theory of arithmetic[2] or that there is a rippling sequence $\bar{A} \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} A$ that rewrites $\bar{A}$ into A using arithmetical wave rules. With these notions the concept of complementarity is extended as follows:

Definition 1. A directed connection $(u^{\mathsf{T}}, v^{\mathsf{F}})$ is *σ-complementary with respect to a theory $\mathcal{T}$* iff $\sigma(A)=\sigma(\bar{A})$ or $\sigma(A) \Rightarrow_{\mathcal{T}} \sigma(\bar{A})$, where $A=label(u)$ and $\bar{A}=label(v)$.

Within a theory $\mathcal{T}$ equality and implication are not the only means to close a branch in a sequent proof. Sequents like $\cdot \vdash \bar{A}$ and $A \vdash \cdot$ are provable if $\bar{A}$ can be proven in $\mathcal{T}$ and A can be proven to be false in $\mathcal{T}$. For the sake of uniformity of notation we call the corresponding atomic positions *unary connections*.

Definition 2. Let u^{T} and v^{F} be unary connections, $A=label(u)$, and $\bar{A}=label(v)$.
u^{T} is *σ-complementary with respect to a theory $\mathcal{T}$* iff $\sigma(A) \Rightarrow_{\mathcal{T}} \mathsf{False}$.
v^{F} is *σ-complementary with respect to $\mathcal{T}$* iff $\mathsf{True} \Rightarrow_{\mathcal{T}} \sigma(\bar{A})$.

In a similar way the concept of complementary with respect to a theory $\mathcal{T}$, which resembles Bibel's theory-connections [6], could also be extended to n-ary connections w.r.t some theory $\mathcal{T}$. It enables us to formulate an extended matrix-characterization of logical validity exactly like Theorem 1 and to use the same general proof technique as for constructive first-order logic. The only modification is the test for complementarity: we can now use decision procedures to prove a negative literal A to be false or a positive literal $\bar{A}$ to be true in $\mathcal{T}$, unify two connected literals A and $\bar{A}$ via standard unification, prove the implication $A \Rightarrow \bar{A}$ with a decision procedure, or find a substitution σ and a rippling sequence $\sigma(\bar{A}) \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} \sigma(A)$ with our rippling / reverse rippling heuristic. Decision procedures can even be used to close the gap between forward and reverse rippling (c.f. example 2), which makes our method more flexible and enables it to find more proofs.

A second modification comes from the insight that there is a strong relation between individual subformulae of the induction hypothesis H and the conclusion C. As C is usually identical to $H[x\backslash\rho(x)]$, where x is the induction variable and ρ an inductive constructor like $\mathsf{s} = \lambda x.x{+}1$ or a destructor like the $\mathsf{p} = \lambda x.x{-}1$, a rippling proof usually links a subformula of H to the corresponding subformula of C, which means that proof-relevant connections run between corresponding subformulae of H and C. By checking such *orthogonal* connections first, we can drastically reduce the search space of an inductive matrix-proof.

Definition 3. A formula F is *orthogonal* (with respect to a variable x) if $F \equiv H \Rightarrow C$ and $C = H[x\backslash\rho(x)]$ for some substitution ρ. A connection $(u^{\mathsf{T}}, v^{\mathsf{F}})$ (or $(v^{\mathsf{T}}, u^{\mathsf{F}})$) where $u \in H$ and $v \in C$ is *orthogonal in F* if the relative position of u in H is identical to the relative position of v in C.

[2] A common decidable sub-theory of arithmetic is elementary arithmetic as defined in [11], which includes the common axioms of `+, -, *, /, =, <` but no induction.

Orthogonal connections are easy to identify, as they connect literals with the same relative positions in the two major subtrees of F. In an inductive proof it is sufficient to consider only orthogonal connections.

Theorem 2. *An orthogonal formula F is constructively valid if all orthogonal connections in F are complementary under some substitution σ.*

Proof. Using theorem 1 we show by induction on the structure of that every path through an orthogonal formula $F \equiv \Rightarrow C$ contains an orthogonal connection.
If is atomic the statement is trivially true. If has the form $_1\, op\ _2$ where op is of type α in F then $C = C_1\, \bar{op}\, C_2$ where $\bar{op}$ has type β in F. Consequently each path $\mathcal{P}$ through F goes through both $_1$ and $_2$ and either C_1 or C_2. By the induction hypothesis $\mathcal{P}$ must contain an orthogonal connection. The argument is the same if op is of type β in F, while γ- and δ-quantifiers do not affect the set of paths. □

In an orthogonal formula it is not necessary to check the unifiability of the prefixes of orthogonal connections, as a classical proof implies constructive validity.

Lemma 1. *In an orthogonal formula F the prefixes of all orthogonal connections are simultaneously unifiable.*

Proof. Let (u, v) be an orthogonal connection in F and $a_0 a_1 ... a_n$ be the prefix of u. Then the prefix of v is $a_0 b_1 ... b_n$ where b_i is a variable iff a_i is a constant. Substituting the variables by the corresponding constant unifies the two prefixes. Since this construction is the same for all orthogonal connections, it leads to a common unifier.

Checking the complementarity of orthogonal connections, which resembles Bibel's linear connection method [5] although used for a different purpose, is sufficient but not necessary for validity. If a proof attempt with orthogonal connections fails, the formula may be valid for reasons that have nothing to do with induction and a conventional (constructive!) matrix proof has to be tried.

A third extension of the matrix-characterization comes from the need for conditional substitutions in some inductive proofs. In a sequent proof the conditions generated by our extended matching procedure will lead to a case analysis as a first step, while each case can be proven by the proof techniques discussed so far. A justification of this case distinction is the observation that a formula F is valid if and only if there is a complete set $\{c_1,..,c_n\}$ of logical constraints (usually decidable formulae) such that each $F_i = c_i \Rightarrow F$ is logically valid. This observation, which in the context of sequent calculi can be proven easily by applying the cut rule, leads to the following extended version of theorem 1.

Theorem 3. *A formula F is valid iff there is a set $\{c_1,..,c_n\}$ of constraints such that $C = c_1 \vee .. \vee c_n$ is valid and for all i there is a multiplicity μ_i, a substitution σ_i, and a set $\mathcal{C}_i$ of σ_i-complementary connections such that every path through $c_i \Rightarrow F^{\mu_i}$ contains a connection from $\mathcal{C}_i$.*

Conditional substitutions also lead to an extension of theorem 2, as our proof method in [24] synthesizes constraints for orthogonal formulas whenever a connection cannot be shown to be complementary. In this case, all paths through the formula are closed either by an orthogonal connection or a connection bet-

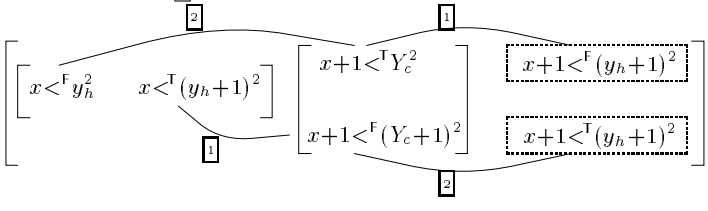
ween a constraint and one literal of an orthogonal connection. To formulate this insight, we call a connection (u, v) *complementary under the constraint* c if c and either u or v form a complementary connection.

Theorem 4. *A formula* $(c^1 \wedge \ldots \wedge c^k) \Rightarrow F$*, where* F *is orthogonal, is valid if all orthogonal connections* (u, v) *are* σ*-complementary or* σ*-complementary under one of the* c^i *for some substitution* σ*.*

Again, we do not have to check the unifiability of prefixes to prove σ-complementarity under c^i, since the prefixes always unify. Thus a classical proof is sufficient.

The above extensions of the matrix-characterization of logical validity enable us to prove inductive specification theorems with existential quantifiers without having to provide domain lemmata and case-splits beforehand. Instead we extend the notion of unification by rippling and other theory-based reasoning techniques, consider orthogonal connections first, and generate conditional substitutions if necessary. The following example shows the advantages of such a proof method.

Example 4. Consider $F_0 \equiv \exists y\, \neg(x{<}y^2) \wedge x{<}(y{+}1)^2 \Rightarrow \exists y\, \neg(x{+}1{<}y^2) \wedge x{+}1{<}(y{+}1)^2$ and its matrix-representation in Figure 1. The matrix contains two orthogonal connections $(x{<}^{\mathsf{T}}(y_h{+}1)^2,\ x{+}1{<}^{\mathsf{F}}(Y_c{+}1)^2)$ and $(Y_c^2{<}^{\mathsf{T}}x{+}1,\ y_h^2{<}^{\mathsf{F}}x)$. Applying extended matching to the first leads to the substitution $\sigma_1{=}\{Y_c \backslash y_h{+}1\}$, as described in example 2. Because the instantiated second connection is not complementary, we add $x{+}1{<}(y_h{+}1)^2$ as condition c_1 (i.e. with opposite polarity F) to the matrix. According to theorem 4 we have thus established the validity of F_0 under c_1, which completes the first subproof (marked by 1)



To complete the proof we now try to establish the validity of F_0 under the negated condition $c_2 = x{+}1{<}^{\mathsf{T}}(y_h{+}1)^2$, which we also add to the matrix. By connecting it to $x{+}1{<}^{\mathsf{F}}(Y_c{+}1)^2$ we get $\sigma_2{=}\{Y_c \backslash y_h\}$ through conventional unification. As the instantiated second connection $(y_h^2{<}^{\mathsf{T}}x{+}1,\ y_h^2{<}^{\mathsf{F}}x)$ is complementary in the sense of definition 1, we have completed the second subproof 2. F_0 is valid because of theorem 3.

5 A Matrix-Based Inductive Proof Method

Based on the concepts and characterizations introduced in the previous section we will now describe an inductive proof method that combines both matrix-based logical proof search and rippling techniques and is summarized in Figure 2. It uses the same basic steps as [24] but in a more refined and goal-directed way due to the use of a matrix-representation and an emphasis on connections.

In the first step *we decompose the step case formula* F by constructing its matrix-representation. This means that δ-quantified variables will become constants and γ-variables will become meta-variables. Since all existential quantifier

1. *Logical Decomposition* of F by constructing its orthogonal matrix-representation.
2. Simultaneously *check the complementarity of orthogonal connections* and generate a substitution σ for all γ-variables by (1) unifying the connected terms, (2) applying a decision procedure, or (3) applying the rippling / reverse rippling heuristic. If all orthogonal connections are complementary the formula is valid.
3. If the complementarity test fails, *synthesize a condition* c for the validity of F.
4. *Prove the validity of* $\neg c \Rightarrow F$ with the standard connection method. If the proof fails generate more constraints by continuing with step 2.

Fig. 2. Matrix-based proof method for an inductive step case formula F

in the conclusion and universal quantifiers in the induction hypothesis are represented by γ-variables, our proof method can be applied to inductive proof problems with arbitrarily nested quantifiers.

In the second step we try to *prove the complementarity of all orthogonal connections* and generate a substitution σ for all γ-variables in the process. As in the usual connection method [20] we investigate the connections in an order given by *active subgoals*, i.e. a set of β-related atoms that must be covered by complementary connections, and construct σ incrementally. Given a partial substitution σ' and a connection $(u^{\mathsf{T}}, v^{\mathsf{F}})$ with $A{=}label(u)$ and $\bar{A}{=}label(v)$ we (1) try to unify $\sigma'(A)$ and $\sigma'(\bar{A})$, (2) apply a decision procedure to prove $\sigma'(A) \Rightarrow_{\mathcal{T}} \sigma'(\bar{A})$, or (3) apply the rippling / reverse rippling heuristic to generate a substitution σ'' and a rippling sequence $\sigma''(\sigma'(\bar{A})) \overset{R}{\longmapsto} \ldots \overset{R}{\longmapsto} \sigma''(\sigma'(A))$. The complementarity of unary connections must be provable with a decision procedure. If all orthogonal connections are complementary, the formula F is valid according to Theorem 2 and the proof is complete.

Otherwise, we *synthesize a condition* c for the validity of F. For each connection that fails the complementarity test, we add the negation c^i of the current active subgoal to a set of constraints in order to *create* a complementary connection that covers this subgoal. As a result we get a condition $c \equiv c^1 \wedge \ldots \wedge c^k$, a substitution σ_1, and a matrix-proof for the validity of $c \Rightarrow F$ according to Theorem 4. Note that c is usually decidable as it is composed of atomic literals.

Using Theorem 3 we now have to *prove the validity of* $\neg c \Rightarrow F$ to complete the proof. We extend the matrix into a representation of $(c \vee \neg c) \Rightarrow F$ while increasing the multiplicity of F. This allows us to generate different substitutions for each condition and to reuse the matrix-proof for $c \Rightarrow F$. Only paths through $\neg c$ still need to be tested for complementarity. Since the condition $\neg c$ very likely characterizes a special case not to be proved by induction, we start with investigating $\neg c$ and use the standard connection method [20] with unification enhanced by decision procedures. No constraints will be introduced in this step.

If the conventional matrix proof for $\neg c \Rightarrow F$ fails, we attempt another inductive proof based on orthogonal connections, extended rippling, and possibly additional constraints by continuing with the second step again. In inductive proofs of program specifications each constraint will correspond to a case split in the program. The examples we tested so far required at most one such case split.

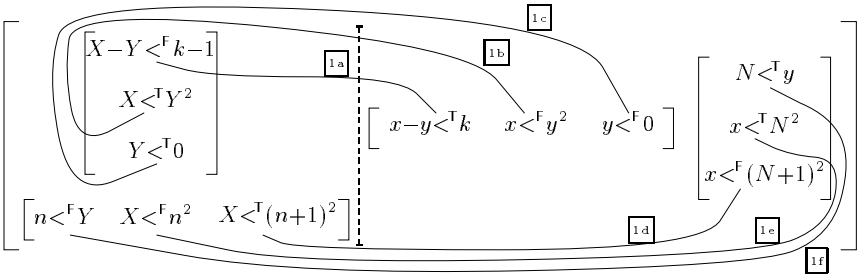


Fig. 3. Matrix of F_1 with orthogonal connections

Example 5. To illustrate our technique, we prove the integer square root specification $\forall x\,\exists y\;\; y^2{\leq}x \wedge x{<}(y{+}1)^2$ with a different induction scheme, which will result in a more efficient program. In each iteration we increment y instead of x and do so until $(y{+}1)^2{>}x$. The (destructor) induction has to proceed over an auxiliary variable k and yields the following step case formula.

$$\begin{aligned} F_1 \;\equiv\;\; & \forall x, y.\; x{-}y{<}k{-}1 \wedge y^2{\leq}x \wedge 0{\leq}y \;\Rightarrow\; \exists n.\, y{\leq}n \wedge n^2{\leq}x \wedge x{<}(n{+}1)^2 \\ \Rightarrow\; & \forall x, y.\; x{-}y{<}k \;\;\;\; \wedge y^2{\leq}x \wedge 0{\leq}y \;\Rightarrow\; \exists n.\, y{\leq}n \wedge n^2{\leq}x \wedge x{<}(n{+}1)^2 \end{aligned}$$

After unfolding $\leq$ we generate the matrix-representation of F_1 and the six orthogonal connections, as depicted in Figure 3. We separate the two orthogonal submatrices by a dashed line and denote γ-variables by capital letters. We select the induction hypothesis as starting point and try to prove the complementarity of the connections 1a ... 1f using a decision procedure Arith for arithmetic, unification, and the rippling / reverse rippling heuristic.

1a To prove the complementarity of the connection our extended matching procedure has to find a substitution σ and a rippling sequence from $\sigma(X{-}Y{<}k{-}1)$ to $\sigma(x{-}y{<}k)$. Using wave rules derived from monotonicity laws and the definition of $-$, reverse rippling succeeds and generates the substitution $\sigma = \{X\backslash x,\; Y\backslash y{+}1\}$

1b As $x{<}(y{+}1)^2 \Rightarrow_{\mathcal{A}} x{<}y^2$ is not valid in arithmetic, we add the inverse of the instantiated active subgoal, i.e. $c_1 \equiv x <^{\mathsf{F}} (y{+}1)^2$ as condition to the matrix.

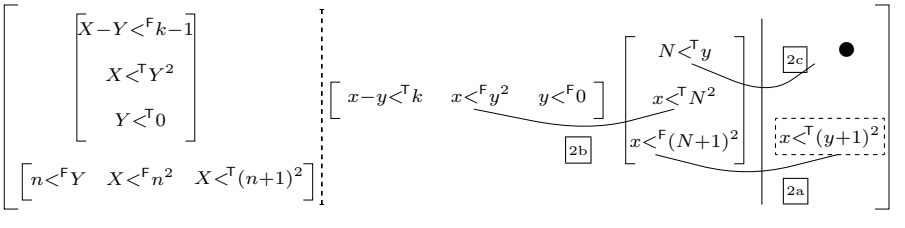
1c The instantiated third connection requires us to prove $y{+}1{<}0 \Rightarrow_{\mathcal{A}} y{<}0$, which is shown by Arith.

1d The instantiated fourth connection can be solved by unifying $x{<}(n{+}1)^2$ and $x{<}(N{+}1)^2$. This extends the substitution to $\sigma = \{X\backslash x,\; Y\backslash y{+}1,\; N\backslash n\}$.

1e The terms in the instantiated fifth connection are now equal.

1f The sixth connection requires us to prove $n{<}y \Rightarrow_{\mathcal{A}} n{<}y{+}1$, which again is shown by Arith.

This concludes the first subproof. We now have to prove the validity of F under the negation of c_1, i.e. $c_2 \equiv x <^{\mathsf{T}} (y{+}1)^2$. We add c_2 to the matrix and select this literal as starting point for the second sub-proof.



There are several $<^{\mathsf{F}}$-literals in the matrix that could be connected to c_2. Among those, all but $x <^{\mathsf{T}} (N+1)^2$ are immediately ruled out because of a constant mismatch. Thus we connect c_2 to $x <^{\mathsf{F}} (N+1)^2$ [2a] and get the substitution $\sigma = \{N \backslash y\}$. This leaves $x <^{\mathsf{T}} y^2$ and $y <^{\mathsf{T}} y$ as active subgoals. To solve the first, we connect to $x <^{\mathsf{F}} y^2$ [2b], which is already a complementary connection. The second open subgoal is solved by applying a decision procedure, which proves $y < y$ to be false. We indicate this unary connection by an arc to a bullet [2c].

This concludes the second subproof as well. F_1 is valid under the set of conditional substitutions $\{[x<(y+1)^2, \{X \backslash x, Y \backslash y+1, N \backslash n\}], [\neg(x<(y+1)^2), \{N \backslash y\}]\}$.

After a matrix-proof has been generated, it has to be converted into a sequent proof, which enables us to extract a verifiably correct algorithm if the proven formula represents a program specification. For this purpose one can combine the algorithms for transforming logical matrix proof into sequent proofs developed in [26,27] with the methods for transforming rippling sequences into sequent proofs described in [21,7].

Depending on the chosen induction and the matrix proof we will get different algorithms for the same specification. The matrix proof for the integer square root specification in Example 4, for instance, will result in an linear algorithm that iterates the input x, while the algorithm extracted from the proof in Example 5 will iterate the output y, which is much more efficient.

6 Conclusion

We have presented a method for integrating rippling-based rewriting into matrix-based constructive theorem proving as a means for generating inductive specification proofs. The proof search is guided by a connection-driven path-checking algorithm that gives preference to connections between the induction hypothesis and the induction conclusion. The complementarity of the connected atoms is checked by unification, decision procedures, and an extended matching procedure based on a rippling / reverse rippling heuristic. Constrained substitutions are generated if there is no unique substitution for the quantified variables. The resulting proof can be converted into a verifiably correct algorithm for a given inductive program specification.

There have been other approaches to automate the instantiation of quantified variables in inductive proofs. Biundo [8] replaces existentially quantified variables by Skolem functions, which represent programs to be synthesized, and proceeds by *clause-set translations* like rewriting and case splitting. The work of Kraan et al. [17] refines this idea by using rippling and middle-out reasoning [14] to control the search space. However, as both approaches are not integrated into a logical proof environment, they cannot guarantee that the synthesized program is correct and have to verify it afterwards.

Hutter [15] introduces a technique to synthesize induction orderings that is related to our reverse rippling heuristic [24]. To find an appropriate induction scheme, it looks at the function symbols surrounding existentially quantified variable and consults the applicable wave rules to guess an instantiation of this

variable. This localized approach, however, causes difficulties when function symbols are defined by two-step or even more complex recursions.

The INKA [16] and SPASS [30] provers integrate induction techniques into resolution-based theorem proving. However, they cannot synthesize case distinctions and resolution is generally non-constructive, which makes it difficult to use these tools for handling inductive program specifications.

The advantage of our approach is that it combines the strengths of inductive reasoning and constructive logical theorem proving in an elegant way. Matrix methods provide a uniform foundation for instantiating existentially and universally quantified variables, as they abstract from the notational peculiarities and focus on their type (i.e. γ or δ). Decision procedures and rippling-based rewriting are used on the level of the connections and thus extend the usual unification, while the systematic generation of case distinctions adds flexibility to the proof search procedure. The combination of these features allows us to deal with complex induction schemes, like the one used in example 5, which cannot be handled by either first-order and inductive proof methods individually.

Our approach is only a first step towards a full integration of first-order and inductive theorem proving. But most of the individual components used in our method, i.e. matrix-based proof search for constructive logic [20], the rippling / reverse rippling heuristic [24], and decision procedures for arithmetic [11] have already been implemented independently. We intend to combine these components as described in Section 5 but also aim at preserving the uniformity of the existing matrix-based theorem prover. We also intend to integrate transformation algorithms for converting matrix proofs and rippling sequences into sequent proofs [26,27,21,7] and use them to guide the development of proofs in interactive proof assistants and for the synthesis of conditional recursive programs.

References

1. A. Armando, A. Smaill & I. Green. Automatic synthesis of recursive programs: The proof-planning paradigm. *12^{th} IEEE International Automated Software Engineering Conference*, pp. 2–9, 1997.
2. D. Basin & T. Walsh. A calculus for and termination of rippling. *Journal of Automated Reasoning*, 16(2):147–180, 1996.
3. J. Bates & R. Constable. Proofs as programs. *ACM Transactions on Programming Languages and Systems*, 7(1):113–136, 1985.
4. W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28:633–645, 1981.
5. W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.
6. W. Bibel. *Automated Theorem Proving*. Vieweg, 1987.
7. W. Bibel, D. Korn, C. Kreitz, F. Kurucz, J. Otten, S. Schmitt & G. Stolpmann. A multi-level approach to program synthesis. *7^{th} International Workshop on Logic Program Synthesis and Transformation*, LNAI 1463, pp. 1–25, 1998.
8. S. Biundo. Automated synthesis of recursive algorithms as a theorem proving tool. *8^{th} European Conference on Artificial Intelligence*, 1988.

9. A. Bundy, F. van Harmelen, A. Smaill *et al.*. Extensions to the rippling–out tactic for guiding inductive proofs. *10^{th} Conference on Automated Deduction*, LNCS 449, pp. 132–146, 1990.
10. A. Bundy, A. Stevens, F. van Harmelen *et al.*. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62(2):185–253, 1993.
11. T. Chan. A decision procedure for checking PL/CV arithmetic inferences In *Introduction to the PL/CV2 Programming Logic*, LNCS 135, pp. 227–264, 1982.
12. R. Constable, S. Allen, M. Bromley, et. al. *Implementing Mathematics with the* NUPRL *proof development system.* Prentice Hall, 1986.
13. G. Dowek et. al. *The Coq proof assistant user's guide.* Institut National de Recherche en Informatique et en Automatique, Report RR 134, 1991.
14. J. Hesketh. *Using Middle-Out Reasoning to Guide Inductive Theorem Proving.* PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1991.
15. D. Hutter. *Synthesis of Induction Orderings for Existence Proofs. 12^{th} International Conference on Automated Deduction*, LNAI 814, 1994.
16. D. Hutter & C. Sengler. INKA, the next generation. *13^{th} Conference on Automated Deduction*, LNAI 1104, pp. 288–292, 1996.
17. I. Kraan, D. Basin & A. Bundy. Logic program synthesis via proof planning. *4^{th} International Workshop on Logic Program Synthesis and Transformation*, pp. 1–14, 1993.
18. C. Kreitz, H. Mantel, J. Otten & S. Schmitt. Connection-Based Proof Construction in Linear Logic. *14^{th} Conference on Automated Deduction*, LNAI 1249, pp. 207–221, 1997.
19. C. Kreitz, J. Otten & S. Schmitt. Guiding Program Development Systems by a Connection Based Proof Strategy. *Fifth International Workshop on Logic Program Synthesis and Transformation*, LNCS 1048, pp. 137–151, 1996.
20. C. Kreitz & J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal of Universal Computer Science*, 5(3):88–112, 1999.
21. F. Kurucz. Realisierung verschiedender Induktionsstrategien basierend auf dem Rippling-Kalkül. Diplomarbeit, Technische Universität Darmstadt, 1997.
22. J. Otten & C. Kreitz. A connection based proof method for intuitionistic logic. *4^{th} Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 918, pp. 122–137, 1995.
23. J. Otten & C. Kreitz. T-String-Unification: Unifying Prefixes in Non-Classical Proof Methods. *5^{th} Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 1071, pp. 244–260, 1996.
24. B. Pientka & C. Kreitz. Automating inductive specification proofs. *Fundamenta Informatica*, 39(1–2):189–209, 1999.
25. A. Smaill & I .Green. Automating the synthesis of functional programs. Research paper 777, Dept. of Artificial Intelligence, University of Edinburgh, 1995.
26. S. Schmitt & C. Kreitz. On transforming intuitionistic matrix proofs into standard-sequent proofs. *4^{th} Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, LNAI 918, pp. 106–121, 1995.
27. S. Schmitt & C. Kreitz. Converting non-classical matrix proofs into sequent-style systems. *13^{th} Conference on Automated Deduction*, LNAI 1104, pp. 418–432, 1996.
28. T. Tammet. A resolution theorem prover for intuitionistic logic. *13^{th} Conference on Automated Deduction*, LNAI 1104, pp. 2–16, Springer, 1996.
29. L. Wallen. *Automated deduction in nonclassical logic.* MIT Press, 1990.
30. C. Weidenbach. SPASS: Combining superposition, sorts and splitting. *Handbook of Automated Reasoning.* Elsevier, 1999.

Monotonic Preorders for Free Variable Tableaux

Pedro J. Martín and Antonio Gavilanes

Dep. de Sistemas Informáticos y Programación. Universidad Complutense de Madrid
E-mail: {pjmartin, agav}@sip.ucm.es**

Abstract. In this paper, we integrate monotonic preorders in tableaux using a simultaneous rigid unification calculus. For the case where monotonicity is not considered, we have defined a sound, complete and terminating calculus which has been improved by using rewrite techniques. Moreover we present a sound, complete but not terminating calculus to solve simultaneous rigid monotonic preordered problems.

1 Introduction

Rewrite techniques have been successfully applied to equational theorem provers as a way of improving their efficiency [11], [2], [9], [4]. When we study arbitrary (e.g. non-symmetric) relations, most of the notions of rewriting developed for the equational case can be extended to the so-called bi-rewrite systems [8]. That is, guided by well-founded syntactic orderings, we orient the inequalities of a set $\mathcal{I}$ to build two sets of rewrite rules: $R_{\leq}$ which allows the replacement of terms by bigger ones, and $R_{\geq}$, similarly by smaller ones. As for equality, we can use the pair $\langle R_{\leq}, R_{\geq} \rangle$ as a decision algorithm whenever certain commutation properties between $R_{\leq}$ and $R_{\geq}$ are satisfied. On the contrary, if $R_{\leq}$ and $R_{\geq}$ do not commute, we complete them by adding new rewrite rules until both rewrite systems capture all the information of $\mathcal{I}$.

In the context of automatic deduction systems, these rewrite techniques have been incorporated into resolution using the notion of *ordering chaining* [1]; but for free variable tableaux they only have been applied to equality [4]. If we integrate transitive relations into tableaux by means of a (simultaneous) unification calculus instead of introducing new inference rules, tableau systems present two advantages over resolution: we can build decision procedures that separate the difficulty of transitive relations away from first order logic and, second, free variables behave rigidly, so variable chaining is a not a critical problem.

In this paper, we incorporate rewrite techniques into preorders (transitive and reflexive relations) and define unification calculi for free variable tableaux. Section 3 presents a unification calculus for preorders that is extended to a simultaneous version and integrated into tableaux in Section 4. Although these calculi are already terminating, we improve them by means of rewrite techniques in Section 5.

More difficulties appear when dealing with monotonic preorders (cfr. [1], [8] for comments). Monotonicity introduces a new problem when non-linear variables are allowed because contexts have to be guessed. For example, $f(g(b), x, x) \leq$

** Research supported by the Spanish Project TIC98-00445-C03-02 "TREND".

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 309–323, 2000.

$f(y, y, g(a))$ is unifiable w.r.t. the theory $\{a \leq b\}$ if we use the substitution $\sigma = [g(a)/x, g(b)/y]$; nevertheless we cannot use standard unification nor a subterm chaining to solve the problem (apart from using any syntactic ordering, the unique possibilities —instancing x to a or y to b— do not build the proper context). In this case we need to design special rules for dealing with monotonicity. Section 6 presents a complete and sound unification calculus that lacks termination. This calculus is extended to a simultaneous version and integrated into tableaux in Section 7.

2 Preliminaries

Let Σ be a signature and X be a set of variables. $T(\Sigma, X)$ denotes the set of terms over the signature Σ with variables from X, while $T(\Sigma)$ stands for *ground* terms.

An *inequality* is an atomic formula $s \sqsubseteq t$ where $s, t \in T(\Sigma, X)$; the set $F(\Sigma, X)$ of Σ-formulas with variables from X is built from inequalities using $\neg, \wedge, \vee, \forall$ and $\exists$ as usual. $F(\Sigma)$ denotes the set of *sentences* and we use *disinequalities* $s \not\sqsubseteq t$, as shorthand of $\neg s \sqsubseteq t$. For simplicity, we assume that $\sqsubseteq$ is the only predicate symbol of our language; in first-order logic with equality $\simeq$, arbitrary predicate symbols P can be represented by introducing a sort *bool* and replacing $P(\bar{t})$ by $true \simeq P(\bar{t})$ (see e.g. [3], for details). The same idea applies to first-order logic with preorder $\sqsubseteq$, replacing $P(\bar{t})$ by $true \sqsubseteq P(\bar{t}) \wedge P(\bar{t}) \sqsubseteq true$; anyway this is out of the scope of this paper. Given a Σ-expression e (term or formula), $var(e)$ denotes the set of unbounded variables of e. An *inequality-theory*, or simply a *theory*, consists of a set of inequalities.

A Σ-*structure* $\mathcal{D}$ is composed of a pair $\langle D, \sqsubseteq^{\mathcal{D}} \rangle$, where $\sqsubseteq^{\mathcal{D}}$ is a preorder (i.e., a reflexive and transitive binary relation) over the non-empty domain D, and interpretations for any function symbol of Σ. A Σ-structure is *monotonic* if $\sqsubseteq^{\mathcal{D}}$ behaves monotonically, that is, if every function symbol f satisfies $f^{\mathcal{D}}(\ldots, d, \ldots) \sqsubseteq^{\mathcal{D}} f^{\mathcal{D}}(\ldots, d', \ldots)$, whenever $d \sqsubseteq^{\mathcal{D}} d'$. A $\mathcal{D}$-*valuation* is a mapping $\rho : X \to D$. A (monotonic) Σ-*interpretation* is a pair $\langle \mathcal{D}, \rho \rangle$ where $\mathcal{D}$ is a (monotonic) Σ-structure and ρ is a $\mathcal{D}$-valuation. Terms and formulas are interpreted in a (monotonic) Σ-interpretation as usual for first-order logic, but using $\sqsubseteq^{\mathcal{D}}$ as the semantic related to $\sqsubseteq$.

A substitution θ whose domain is $dom(\theta) = \{x_1, \ldots, x_n\}$ is represented by $[x_1\theta/x_1, \ldots, x_n\theta/x_n]$. Positions, subterms and replacements in terms are defined as usual, that is $t|_p$ denotes the subterm of t at position p, and $t[s]_p$ is the result of substituting s for $t|_p$ in t.

Rewrite techniques will be integrated by using reduction orderings, denoted by $\succ$, which are supposed to be total on ground terms.

Free variables behave rigidly in tableaux, that is they represent a unique element. In this sense, given the theory $\mathcal{I} = \{s_1 \sqsubseteq t_1, \ldots, s_n \sqsubseteq t_n\}$ and the inequality $s \sqsubseteq t$, the expression $\mathcal{I} \models_p s \sqsubseteq t$ (resp. $\mathcal{I} \models_m s \sqsubseteq t$) expresses that the formula $(s_1 \sqsubseteq t_1 \wedge \ldots \wedge s_n \sqsubseteq t_n) \to s \sqsubseteq t$ is true in every (resp. monotonic) Σ-interpretation.

In order to present unification calculi, we introduce the notion of *ordering constraints* (cfr. [9] [4]) which captures standard unification and orderings in a single framework.

Definition 1 *A (ordering) constraint is a set of equality constraints $s \simeq t$ and ordering constraints $s \succ t$. A substitution θ is a solution to a constraint $s \simeq t$ (resp. $s \succ t$) if θ is ground for $var(s) \cup var(t)$ and $s\theta$ coincides with $t\theta$ (resp. $s\theta \succ t\theta$). A substitution θ is a solution to a constraint C if θ is a solution to every element of C. A constraint C is satisfiable if it has a solution.*

In this paper, we assume the existence of effective procedures for checking constraint satisfiability: this assumption depends on the choice of the term ordering. For example, effective methods for lexicographic path orderings can be found in [9], [10].

3 Rigid Preordered Unification

In this section we present how to solve unification problems arising when closing a single branch of a tableau.

Definition 2 *A Rigid Preordered Unification (shortly RPU-)problem is an expression of the form $\mathcal{I} \vdash s \sqsubseteq t$, where $\mathcal{I}$, $s \sqsubseteq t$ are a finite theory and an inequality respectively. A ground substitution θ solves $\mathcal{I} \vdash s \sqsubseteq t$ if $\mathcal{I}\theta \models_p (s \sqsubseteq t)\theta$.*

Constraints are added to *RPU*-problems to build constraint *RPU*-problems; hence solving $\mathcal{I} \vdash s \sqsubseteq t$ plus the constraint C involves searching a substitution that solves both of them.

Definition 3 *A constraint RPU-problem is a pair composed of a RPU-problem Γ and a constraint C, and it is written $\Gamma \cdot C$.*

For solving *RPU*-problems, we present the following calculus $\mathcal{P}$ that transforms constraint *RPU*-problems by using the following two rules:

(left↓) Left Root Chaining
$$\frac{\{l \sqsubseteq r\} \cup \mathcal{I} \vdash s \sqsubseteq t \cdot C}{\mathcal{I} \vdash r \sqsubseteq t \cdot C \cup \{l \simeq s\}}$$

(end) Reflexive Closure
$$\frac{\mathcal{I} \vdash s \sqsubseteq t \cdot C}{\mathcal{I} \vdash s \sqsubseteq s \cdot C \cup \{s \simeq t\}}$$

Extra conditions restrict their application: *a*) the constraint at the conclusion of the rule has to be satisfiable, *b*) the inequality $s \sqsubseteq t$ at the premise is not *trivial* ($s \neq t$), and *c*) in rule *left*↓, $l \neq r$. The first condition will assure soundness of the calculus, the second one states that trivial inequalities represent the end of the search, while the third one is added as a form of optimization. Note also that only equality constraints are introduced, and that inequalities are removed from the theory once they have been used. This last remark plays an important role when proving termination. In the sequel, we represent $\mathcal{P}$-steps of the calculus by $\Gamma \cdot C \rightsquigarrow_{\mathcal{P}} \Gamma' \cdot C'$; $\rightsquigarrow_{\mathcal{P}}{}^*$ denotes its reflexive and transitive closure.

Definition 4 *The substitution θ is a $\mathcal{P}$-unifier for the RPU-problem $\mathcal{I} \vdash s \sqsubseteq t$ if there exists a constraint C such that:*

- $\mathcal{I} \vdash s \sqsubseteq t \cdot \emptyset \rightsquigarrow_{\mathcal{P}}^{*} \mathcal{I}' \vdash r \sqsubseteq r \cdot C$, *where $\mathcal{I}'$ is a theory and $r \in T(\Sigma, X)$*
- *θ is a solution to C.*

Given a *RPU*-problem Γ, we can see the computation of $\mathcal{P}$-unifiers as a search in the $\mathcal{P}$-derivation tree with $\Gamma \cdot \emptyset$ in the root. Observe that successful leaves contain a trivial inequality and a constraint that possibly represents several $\mathcal{P}$-unifiers.

3.1 Properties of the Calculus $\mathcal{P}$

The unification calculus $\mathcal{P}$ is terminating because it consumes inequalities from the theory.

Theorem 5 (Termination) *The P-derivation tree for every RPU-problem is finite.*

The calculus $\mathcal{P}$ is also sound in the sense that every $\mathcal{P}$-unifier solves the *RPU*-problem for which it is obtained.

Theorem 6 (Soundness) *Let $\Gamma = \mathcal{I} \vdash s \sqsubseteq t$ be a RPU-problem and θ a $\mathcal{P}$-unifier for Γ, then θ is a solution to Γ.*

Reciprocally, $\mathcal{P}$ is complete in the sense that every solution to a given *RPU*-problem can be obtained as a $\mathcal{P}$-unifier by applying the calculus to the problem. Before proving completeness, we characterize syntactically a solvable *RPU*-problem; we use the concept of *Rigid Preordered Theory* and prove an inequational variant of Birkhoff's Theorem.

Definition 7 *Let $\mathcal{I}$ be a theory, the Rigid Preordered Theory induced by $\mathcal{I}$, written $Th_p(\mathcal{I})$, is the minimum set of inequalities satisfying:*

1. *$\mathcal{I} \subseteq Th_p(\mathcal{I})$.*
2. *$s \sqsubseteq s \in Th_p(\mathcal{I})$, for every $s \in T(\Sigma, X)$.*
3. *If $t_1 \sqsubseteq t_2, t_2 \sqsubseteq t_3 \in Th_p(\mathcal{I})$ then $t_1 \sqsubseteq t_3 \in Th_p(\mathcal{I})$.*

Theorem 8 (Birkhoff's Theorem) *$\mathcal{I} \models_p s \sqsubseteq t$ iff $s \sqsubseteq t \in Th_p(\mathcal{I})$.*

We use *proofs* to deduce that an inequality belongs to a Rigid Preordered Theory. Inside these proofs we can avoid redundancies —such as trivial chainings or cycles— as the following lemma shows.

Lemma 9 *Let $\mathcal{I}$ be a theory and $s \sqsubseteq t \in Th_p(\mathcal{I})$ such that $s \neq t$. Then there exists a finite sequence of inequalities from $\mathcal{I}$, $l_1 \sqsubseteq r_1, \ldots, l_n \sqsubseteq r_n$, that satisfies: (a) $l_1 = s$, $r_n = t$, (b) $r_i = l_{i+1}$, for every $1 \leq i \leq n-1$, (c) $(l_i \sqsubseteq r_i) \neq (l_j \sqsubseteq r_j)$, for every $1 \leq i < j \leq n$, (d) $l_i \neq r_i$, for every $1 \leq i \leq n$.*

The completeness of $\mathcal{P}$ means that a $\mathcal{P}$-unifier for a RPU-problem Γ must exist whenever there is a substitution θ solving Γ; in fact, $\mathcal{P}$ can compute θ itself. The idea is to lift the proof related to $(s \sqsubseteq t)\theta \in Th_p(\mathcal{I}\theta)$, whose existence is ensured by Lemma 9.

Theorem 10 (Completeness) *Let θ be a solution to the RPU-problem $\Gamma = \mathcal{I} \vdash s \sqsubseteq t$, then θ is a $\mathcal{P}$-unifier for Γ.*

Observe that $\mathcal{P}$ can be used as a decision procedure for RPU-problems thanks to Theorems 5, 6 and 10.

Corollary 11 *The RPU-problem is decidable.*

4 Integrating Simultaneous Rigid Preordered Unification into Free Variable Tableaux

The closure of a tableau $\mathcal{T}$ involves the closure of every branch of $\mathcal{T}$. For this reason, we extend RPU-problems to *simultaneous* problems.

Definition 12 *A Simultaneous Rigid Preordered Unification (shortly SRPU-) problem consists of a finite sequence $\Gamma_1, \ldots, \Gamma_n$ of RPU-problems.*

In this section, we present the calculus $\mathcal{SP}$ as the natural extension of the calculus $\mathcal{P}$ for dealing with simultaneity. $\mathcal{SP}$ transforms sequences of RPU-problems by (local) applications of *left*↓ or *end* rules to some of its single RPU-problems. As variables behave rigidly, we gather the related constraint of each RPU-problem into a unique set. With a single (global) constraint, we avoid the construction of independent and (locally) satisfiable constraints that could not be globally compatible. Dealing with tableaux, we prune more effectively the search space if we close a tableau by simultaneously closing its branches, using a global constraint, instead of closing each branch separately. The rules of $\mathcal{SP}$ are:

(sleft↓) Simultaneous Left Root Chaining $$\frac{\Gamma_1, ..., \{l \sqsubseteq r\} \cup \mathcal{I}_i \vdash s_i \sqsubseteq t_i, ..., \Gamma_n \cdot C}{\Gamma_1, ..., \mathcal{I}_i \vdash r \sqsubseteq t_i, ..., \Gamma_n \cdot C \cup \{l \simeq s_i\}}$$

(send) Reflexive Closure $$\frac{\Gamma_1, \ldots, \mathcal{I}_i \vdash s_i \sqsubseteq t_i, \ldots, \Gamma_n \cdot C}{\Gamma_1, \ldots, \mathcal{I}_i \vdash s_i \sqsubseteq s_i, \ldots, \Gamma_n \cdot C \cup \{s_i \simeq t_i\}}$$

The extra conditions for the applicability of these rules coincide with those required for $\mathcal{P}$ above. We define the relation $\rightsquigarrow_{\mathcal{SP}}$ and the concept of $\mathcal{SP}$-unifier as for $\mathcal{P}$, so the computation of $\mathcal{SP}$-unifiers is the search in the $\mathcal{SP}$-derivation tree with $\Gamma_1, \ldots, \Gamma_n \cdot \emptyset$ in the root.

The $SRPU$-problem is also decidable because the calculus $\mathcal{SP}$ satisfies the same properties as $\mathcal{P}$.

Theorem 13 (Termination, Soundness and Completeness) *Let $\Gamma_1, \ldots, \Gamma_n$ be a SRPU-problem. Then:*

1. The $\mathcal{SP}$-derivation tree for $\Gamma_1, \ldots, \Gamma_n$ is finite.

2. If θ is a $\mathcal{SP}$-unifier for $\Gamma_1, \ldots, \Gamma_n$ then θ is a solution to Γ_i, $1 \leq i \leq n$.
3. If θ is a solution for the RPU-problem Γ_i $(1 \leq i \leq n)$ then θ is a $\mathcal{SP}$-unifier for $\Gamma_1, \ldots, \Gamma_n$.

We integrate simultaneous rigid preordered unification into free variable tableaux using the calculus $\mathcal{SP}$ as a closure rule. Let $\mathcal{SPT}$ be the free variable tableau system composed of the rules α, β, γ' and δ' [5] and the following closure rule:

(SRPU-Closure Rule) Let $\mathcal{T}$ be a free variable tableau with branches $B_1, \ldots, B_n$. Let $\mathcal{I}_i$ be the theory occurring in B_i and $s_i \not\sqsubseteq t_i$ be a dis-inequality appearing in B_i, $1 \leq i \leq n$. Then $\mathcal{T}$ is closed if there exists a $\mathcal{SP}$-unifier for the SRPU-problem $\mathcal{I}_1 \vdash s_1 \sqsubseteq t_1, \ldots, \mathcal{I}_n \vdash s_n \sqsubseteq t_n$.

We use the tableau system $\mathcal{SPT}$ as follows:

1. Expand non-deterministically the tableau applying rules α, β, γ' and δ'.
2. Select a dis-inequality from every branch and look for a $\mathcal{SP}$-unifier for the related $SRPU$-problem. If such a unifier exists then finish; otherwise, choose a different dis-inequality in some branch. Go back to 1 if no $SRPU$-problem admits a $\mathcal{SP}$-unifier.

Observe that preorders are only taken into account in step 2, which always finishes. Hence we have separated the complexity of preorders away from the undecidability of first-order logic.

The soundness of the tableau system $\mathcal{SPT}$ follows from Theorem 13. For proving completeness, we assume that the ground tableau system composed of α, β, γ and δ [5], the rules for preorders *Ref* and *Tran* below, and the atomic ground closure, is complete (see [6] for details).

$$\textit{Ref)} \quad \frac{}{t \sqsubseteq t} \qquad\qquad \textit{Tran)} \quad \frac{t_1 \sqsubseteq t_2 \quad t_2 \sqsubseteq t_3}{t_1 \sqsubseteq t_3}$$

Then we lift every ground closed tableau, by using Theorem 13, and we conclude that $\mathcal{SPT}$ is also complete.

Theorem 14 *A set of sentences Φ is not satisfiable iff there exists a closed $\mathcal{SPT}$-tableau for Φ.*

Example 15 *Let a, b, c, d constants. Then we can conclude $b \sqsubseteq c \vee d \sqsubseteq b$ in every total preorder satisfying $a \sqsubseteq c$ and $d \sqsubseteq a$. In effect, we can firstly build a $\mathcal{SPT}$-tableau as sketched in Figure 1 and then close it using the calculus $\mathcal{SP}$ as in Figure 2 (we have simplified the constraints).*

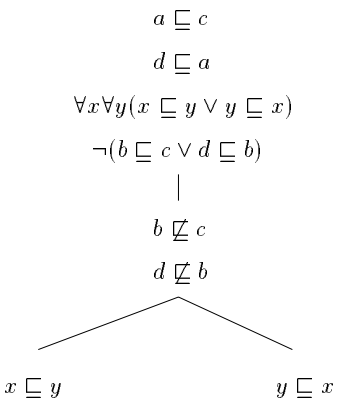


Fig. 1. Sketch of $\mathcal{SPT}$-tableau

$$
\begin{array}{lllll}
 & \{a \sqsubseteq c, d \sqsubseteq a, x \sqsubseteq y\} \vdash d \sqsubseteq b & \{a \sqsubseteq c, d \sqsubseteq a, y \sqsubseteq x\} \vdash b \sqsubseteq c & \cdot \emptyset & \\
\mathit{sleft}\downarrow & \hline
\{a \sqsubseteq c, x \sqsubseteq y\} \vdash a \sqsubseteq b & \{a \sqsubseteq c, d \sqsubseteq a, y \sqsubseteq x\} \vdash b \sqsubseteq c & \cdot \{d \simeq d\} & \\
\mathit{sleft}\downarrow & \hline
\{a \sqsubseteq c\} \vdash y \sqsubseteq b & \{a \sqsubseteq c, d \sqsubseteq a, y \sqsubseteq x\} \vdash b \sqsubseteq c & \cdot \{x \simeq a\} & \\
\mathit{send} & \hline
\{a \sqsubseteq c\} \vdash y \sqsubseteq y & \{a \sqsubseteq c, d \sqsubseteq a, y \sqsubseteq x\} \vdash b \sqsubseteq c & \cdot \{x \simeq a, y \simeq b\} & \\
\mathit{sleft}\downarrow & \hline
\{a \sqsubseteq c\} \vdash y \sqsubseteq y & \{a \sqsubseteq c, d \sqsubseteq a\} \vdash x \sqsubseteq c & \cdot \{x \simeq a, y \simeq b\} & \\
\mathit{sleft}\downarrow & \hline
\{a \sqsubseteq c\} \vdash y \sqsubseteq y & \{d \sqsubseteq a\} \vdash c \sqsubseteq c & \cdot \{x \simeq a, y \simeq b\} &
\end{array}
$$

Fig. 2. $\mathcal{SP}$-derivation

5 Rewrite Techniques for Preorders

We introduce rewrite strategies in order to reduce the $\mathcal{P}$-search space. When we solve a ground RPU-problem $\mathcal{I} \vdash s \sqsubseteq t$, the calculus $\mathcal{P}$ could compute the transitive closure of $\mathcal{I}$ with s as the minimum element, that is $\mathcal{P}$ could find every ground term t' such that $\mathcal{I} \models_p s \sqsubseteq t'$. For example, if $\Gamma = \{a \sqsubseteq b, a \sqsubseteq c, a \sqsubseteq d \vdash a \sqsubseteq b\}$ we can apply the rule *left*$\downarrow$ in three different ways to cover the terms b, c, d that are greater or equal than a. Then we prune drastically the search space if we restrict the application of rule *left*$\downarrow$ to those cases where the inequality $l \sqsubseteq r \in \mathcal{I}$ satisfies $l \succ r$. If $c \succ d \succ a \succ b$ in the example above, the three previous choices are reduced to the one corresponding to $a \sqsubseteq b$.

When using orderings, extra rules are needed in order to assure completeness. For example, if $b \succ a$ then we cannot solve the satisfiable RPU-problem $\{a \sqsubseteq b\} \vdash a \sqsubseteq b$ because $a \not\succ b$. So we must allow to rewrite b into a in the right hand side of the inequality $a \sqsubseteq b$. Even so, a symmetric version of the restricted rule *left*$\downarrow$ is still insufficient because *peaks* are not solved: if $b \succ a$ and $b \succ c$ then we cannot solve the satisfiable RPU-problem $\{a \sqsubseteq b, b \sqsubseteq c\} \vdash a \sqsubseteq c$. To solve this kind of problems we need to rewrite into the inequalities of the theory. With these ideas in mind, we define $\mathcal{P}1$ as the calculus composed of the following rules:

(left$_\succ$ $\downarrow$) Left Root Chaining $$\frac{\{l \sqsubseteq r\} \cup \mathcal{I} \vdash s \sqsubseteq t \cdot C}{\mathcal{I} \vdash r \sqsubseteq t \cdot C \cup \{l \simeq s, l \succ r\}}$$

(right$_\succ$ $\downarrow$) Right Root Chaining $$\frac{\{l \sqsubseteq r\} \cup \mathcal{I} \vdash s \sqsubseteq t \cdot C}{\mathcal{I} \vdash s \sqsubseteq l \cdot C \cup \{t \simeq r, r \succ l\}}$$

(peaks$_\succ$) Peak Root Chaining $$\frac{\{l_1 \sqsubseteq r_1, l_2 \sqsubseteq r_2\} \cup \mathcal{I} \vdash s \sqsubseteq t \cdot C}{\{l_1 \sqsubseteq r_2\} \cup \mathcal{I} \vdash s \sqsubseteq t \cdot C \cup \{r_1 \simeq l_2 \; r_1 \succ l_1, l_2 \succ r_2\}}$$

(end) Reflexive Closure $$\frac{\mathcal{I} \vdash s \sqsubseteq t \cdot C}{\mathcal{I} \vdash s \sqsubseteq s \cdot C \cup \{s \simeq t\}}$$

The extra conditions for the applicability of these rules coincides with those required for $\mathcal{P}$ previously. We define the relation $\rightsquigarrow_{\mathcal{P}1}$ and the concept of $\mathcal{P}1$-unifier as for $\mathcal{P}$. Observe that $\mathcal{P}1$ can be seen as a suitable restriction of $\mathcal{P}$, where ordering constraints are involved, because every $\mathcal{P}1$-unifier is a $\mathcal{P}$-unifier. In this sense, $\mathcal{P}1$ is also terminating and sound.

Theorem 16 (Termination) *The $\mathcal{P}1$-derivation tree for every RPU-problem is finite.*

Theorem 17 (Soundness) *Let Γ be a RPU-problem and θ a $\mathcal{P}1$-unifier for Γ, then θ is a solution to Γ.*

We use Theorem 8 and Lemma 9 to prove that $\mathcal{P}1$ is also complete when asking for solutions that groundly instance $var(\Gamma)$. Note that $\succ$ is total on ground terms, then we can orient every step of a ground proof of the form:

$$s\theta = l_1\theta \sqsubseteq r_1\theta = \ldots l_n\theta \sqsubseteq r_n\theta = t\theta$$

Theorem 18 (Completeness) *Let θ be a solution to the RPU-problem Γ which grounds $var(\Gamma)$, then θ is a $\mathcal{P}1$-unifier for Γ.*

As for the calculus $\mathcal{P}$, we extend $\mathcal{P}1$ to a simultaneous calculus $\mathcal{SP}1$ in order to incorporate it into tableaux. $\mathcal{SP}1$ transforms *SRPU*-problems by the local application of *left*$_\succ$ $\downarrow$, *right*$_\succ$ $\downarrow$, *peaks*$_\succ$ and *end* to any of its related *RPU*-problems. As for $\mathcal{SP}$, $\mathcal{SP}1$ only uses a unique constraint. $\mathcal{SP}1$ is terminating, sound and complete as consequence of Theorems 16, 17 and 18.

Let $\mathcal{SPT}1$ be a tableau system like $\mathcal{SPT}$, but using the simultaneous unification calculus $\mathcal{SP}1$ instead of $\mathcal{SP}$ in its closure rule. Then $\mathcal{SPT}1$ is sound and complete.

Theorem 19 *A set of sentences Φ is not satisfiable iff there exists a closed $\mathcal{SPT}1$-tableau for Φ.*

6 Rigid Monotonic Preordered Unification

In the sequel we only consider monotonic preorders. Then the unification problems closing single branches are *RPU*-problems but their solutions must consider monotonicity.

Definition 20 *A Rigid Monotonic Preordered Unification (RMPU-)problem is an expression of the form* $\mathcal{I} \vdash s \sqsubseteq t$ *where* $\mathcal{I}$, $s \sqsubseteq t$ *are a finite theory and an inequality respectively. A ground substitution* θ *solves* $\mathcal{I} \vdash s \sqsubseteq t$ *if* $\mathcal{I}\theta \models_m (s \sqsubseteq t)\theta$.

In order to solve *RMPU*-problems, one wonders if the natural generalization of the rule *left*↓ that allows replacements into internal positions —what we call *subterm chaining*— is enough to ensure completeness and termination. However monotonicity introduces new problems which are also present in rigid E-unification:

1. **Termination versus completeness**:
When monotonicity is not considered, the finite set of successful $\mathcal{P}$-constraints were enough to characterize all the solutions to a *RPU*-problem (Theorem 10). In contrast, we need an infinite set, and so a non-terminating calculus, if we are interested in every solution to a *RMPU*-problem. For example, we need an infinite number of sets of constraints ($\{x \simeq f^n(a)\}$, with $n \geq 0$) to obtain the complete set of solutions to the *RMPU*-problem $\{a \sqsubseteq f(a)\} \vdash a \sqsubseteq x$.

Two approaches have been followed to fix the incompatibility between termination and completeness when dealing with rigid E-unification:

- designing terminating unification calculus that are existentially complete (e.g. [7]), which means that they are able to find at least one unifier when the problem has a solution. This research line presents two problems in the case of monotonic preorders. On one hand, symmetry plays an important role here because it allows the comparison between solutions through the concept of E-generality ($\leq_E$). In fact the whole set of solutions can be captured with a finite set of E-unifiers (what is usually called *complete* set of E-unifiers). For example, in the equality version of the previous problem $\{a \simeq f(a)\} \vdash a \simeq x$, $\{[a/x]\}$ is a complete set of unifiers that covers all the solutions ($\{[f^n(a)/x]/n \geq 0\}$).
  On the other hand, it has been proved that simultaneous rigid E-unification is undecidable [4], hence this approach cannot be naturally extended to tableaux with equality. Therefore the simultaneous version of the *RMPU*-problem must be undecidable, because simultaneous rigid E-unification can be reduced to it.
- designing incomplete but terminating calculi that can be integrated into tableaux to build complete deduction systems [4]. The key for this approach involves restricting the ground solutions that can be lifted.

2. **Ensuring termination**:
The inequalities of the theory cannot be erased once they have been used. For example, when solving the *RMPU*-problem $\{a \sqsubseteq b\} \vdash f(a, a) \sqsubseteq f(b, b)$, we need the inequality $a \sqsubseteq b$ twice. Then the reason why the calculus $\mathcal{SP}$ is terminating is not available when monotonicity is considered. In this case rewrite techniques seem to be a useful tool to achieve termination, as for equality.

3. **Guessing contexts**:
Let θ be a ground substitution that solves the *RMPU*-problem $\Gamma = \mathcal{I} \vdash s \sqsubseteq t$.

Then it is obvious that the related ground $RMPU$-problem $\Gamma\theta = \mathcal{I}\theta \vdash (s \sqsubseteq t)\theta$ is also satisfiable. However the subterm chainings used to solve $\Gamma\theta$ that are placed in positions introduced by θ itself are not applicable to the problem Γ. For example, if $\Gamma = \{a \sqsubseteq b\} \vdash x \sqsubseteq y$ and $\theta = [f(a)/x, f(b)/y]$, we cannot lift the subterm chaining $f(a) \sqsubseteq f(b)$ used to solve $\Gamma\theta$. Even so, in this case we can solve Γ following other ways: using *end* to obtain the constraint $\{x \simeq y\}$ or using *left*↓ and *end* to obtain $\{x \simeq a, y \simeq b\}$, but none of these successful constraints covers θ because they do not build the proper context.

In the case of rigid E-unification, the solution to this problem has gradually developed. Initially contexts were explicitly built by using the *function reflexivity axioms* [5]. Latter such reconstruction was avoided by restricting the study to the so called *irreducible* substitutions (e.g. [7]) —those whose introduced terms do not admit any subterm rewriting. Nevertheless symmetry is still needed to define this concept, so this idea turns out to be useless for monotonic preorders in the presence of non-linear variables[1]. For example, the substitution $[g(a)/x, g(b)/y]$ solves the $RMPU$-problem $\{a \sqsubseteq b\} \vdash f(g(b), x, x) \sqsubseteq f(y, y, g(a))$ but surprisingly the calculus cannot solve it: *end* is not applicable and the possible subterm chainings —instancing x to a or y to b— do not succeed because they do not build the proper context. Therefore *end* and subterm chaining are not enough to ensure (existential) completeness. Observe that the same problem but using equality $\{a \simeq b\} \vdash f(g(b), x, x) \simeq f(y, y, g(a))$ can be easily overcome using a subterm replacement (a instead b —available because of symmetry) and then *end* to obtain the solution $[g(a)/x, g(a)/y]$.

In this section we define the unification calculus $\mathcal{MP}$ that is existentially complete for solving $RMPU$-problems. Then $\mathcal{MP}$ represents the basis for the integration of rewrite techniques that are needed for termination.

In order to present $\mathcal{MP}$, we deduce the rules that are needed for ensuring completeness. Then we firstly characterize syntactically when a $RMPU$-problem has a solution by using the concept of *Rigid Monotonic Preordered Theory* and proving a new variant of Birkhoff's Theorem.

Definition 21 *Let $\mathcal{I}$ be a theory, the Rigid Monotonic Preordered Theory induced by $\mathcal{I}$, written $Th_m(\mathcal{I})$, is the minimum set of inequalities that holds:*

1. $\mathcal{I} \subseteq Th_m(\mathcal{I})$.
2. $s \sqsubseteq s \in Th_m(\mathcal{I})$, *for every* $s \in T(\Sigma, X)$.
3. $t_1 \sqsubseteq t_2, t_2 \sqsubseteq t_3 \in Th_m(\mathcal{I})$ *then* $t_1 \sqsubseteq t_3 \in Th_m(\mathcal{I})$.
4. $s \sqsubseteq t \in Th_m(\mathcal{I})$ *then* $c[s]_p \sqsubseteq c[t]_p \in Th_m(\mathcal{I})$, *for every* $c \in T(\Sigma, X)$.

Theorem 22 (Birkhoff's Theorem) $\mathcal{I} \models_m s \sqsubseteq t$ *iff* $s \sqsubseteq t \in Th_m(\mathcal{I})$.

As in Section 3, we use *proofs* to show that an inequality belongs to a Rigid Monotonic Preordered Theory. In this case, we chain inequalities from $\mathcal{I}$ after they have been included in arbitrary contexts. Note that we cannot avoid the repeated use of an inequality in a proof.

[1] Variables occurring more than once in a term.

Lemma 23 *Let $\mathcal{I}$ be a theory and $s \sqsubseteq t \in Th_m(\mathcal{I})$ such that $s \neq t$. Then there exists a finite sequence, $l_1 \sqsubseteq r_1, \ldots, l_n \sqsubseteq r_n$, of inequalities from $\mathcal{I}$ and contexts $c_i \in T(\Sigma, X)$ with positions p_i, $1 \leq i \leq n$, holding: (a) $s = c_1[l_1]_{p_1}$, $c_n[r_n]_{p_n} = t$, (b) $c_i[r_i]_{p_i} = c_{i+1}[l_{i+1}]_{p_{i+1}}$, for every $1 \leq i \leq n-1$, (c) $l_i \neq r_i$, for every $1 \leq i \leq n-1$.*

Let us suppose that the substitution θ solves the $RMPU$-problem $\Gamma = \mathcal{I} \vdash s \sqsubseteq t$, that is $\mathcal{I}\theta \models_m (s \sqsubseteq t)\theta$. By Theorem 22 and Lemma 23, there exists a proof of the form:

$$s\theta = c_1[l_1\theta]_{p_1} \sqsubseteq c_1[r_1\theta]_{p_1} = c_2[l_2\theta]_{p_2} \sqsubseteq \ldots \sqsubseteq c_n[r_n\theta]_{p_n} = t\theta$$

where $(l_i \sqsubseteq r_i) \in \mathcal{I}$, $l_i\theta \neq r_i\theta$ and c_i are certain contexts with positions p_i, $1 \leq i \leq n$. The calculus $\mathcal{MP}$ has to be able to lift any such proof when solving Γ. Let us study the different cases in order to deduce the required rules:

1. If $n = 0$ then $s\theta = t\theta$, so we can apply reflexive closure (*end*).
2. If $n > 0$ and there are some steps occurring at the root. Let i be the smallest index with $p_i = \varepsilon$, then we can split the proof into two parts in such a way that the first one does not use root positions ($s\theta = c_1[l_1\theta]_{p_1} \ldots \sqsubseteq l_i\theta$). Then we need some kind of root chaining to solve this case.
3. If $n > 0$ and there are no steps occurring at the root.
   a) If neither s nor t are variables, then they have the same functional symbol at the root, that is $s = f(s_1, \ldots, s_q)$ and $t = f(t_1, \ldots, t_q)$. For this case we can use decomposition and split the proof into q proofs for $s_i\theta \sqsubseteq t_i\theta$, $1 \leq i \leq q$.
   b) If $s = x$ and $t = f(t_1, \ldots, t_q)$, then $x\theta = f(s_1, \ldots, s_q)$ and we can split the proof into q proofs for $s_i \sqsubseteq t_i\theta$. In this case we need some kind of root imitation in the sense that x must be replaced by $f(y_1, \ldots, y_q)$, where the variables y_i are new.
   c) If s, t are variables and $s\theta = f(s_1, \ldots, s_q)$, $t\theta = f(t_1, \ldots, t_q)$, then we can split the proof into q proofs for $s_i \sqsubseteq t_i$. We need some kind of root imitation to solve this case as in b).

Let us do some reflections about the previous analysis. First notice that a single $RMPU$-problem can be split into several ones; for example, in case 2 we would transform the initial $RMPU$-problem $\mathcal{I} \vdash s \sqsubseteq t$ into two new problems $\mathcal{I} \vdash s \sqsubseteq l$ and $\mathcal{I} \vdash r \sqsubseteq t$. In this sense we face several $RMPU$-problems simultaneously although we are considering a single branch. In order to use a suitable notation which does not duplicate theories, we extend the definition of $RMPU$-problem to deal with a sequence of inequalities to be proved. Then the expression $\mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, s_n \sqsubseteq t_n$ simplifies n $RMPU$-problems of the form $\mathcal{I} \vdash s_i \sqsubseteq t_i$. Constraints are also added to $RMPU$-problems to build *constraint $RMPU$-problems* as in Definition 3.

Observe that the required root imitation rules for cases 3 b) and 3 c) would work differently from an implementation point of view, in the sense that in case c) the function symbol has to be guessed. In fact, in case 3 c) such a rule seems to

operate like the inefficient function reflexivity axioms because it builds arbitrary contexts, instead of case b) where contexts are destroyed. For this reason the calculus $\mathcal{MP}$ does not cover case 3 c) explicitly, although it must build the contexts of such case in order to ensure completeness. This is achieved if we apply the mgu that is obtained in cases 1 and 3 b). This is the idea: first we can lift the other cases till all the inequalities to be proved are trivial or have the form $x \sqsubseteq y$, and second we finish applying a *general* reflexive closure (*end*). Note that the latter can always be done because the remaining variables in the $RMPU$-problem (e.g. x, y) do not belong to the domain of the mgu of its related constraint, whenever we apply the mgu in cases 1 and 3 b).

As consequence of the previous analysis, the unification calculus $\mathcal{MP}$ transforms constraint $RMPU$-problems by the application of the following five rules:

(root) Root Chaining $$\frac{\{l \sqsubseteq r\} \cup \mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, s_i \sqsubseteq t_i, \ldots, s_n \sqsubseteq t_n \cdot C}{\{l \sqsubseteq r\} \cup \mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, s_i \sqsubseteq l, r \sqsubseteq t_i, \ldots, s_n \sqsubseteq t_n \cdot C}$$

(dec) Decomposition $$\frac{\mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, f(u_1, ..., u_q) \sqsubseteq f(v_1, ..., v_q), \ldots, s_n \sqsubseteq t_n \cdot C}{\mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, u_1 \sqsubseteq v_1, \ldots, u_q \sqsubseteq v_q, \ldots, s_n \sqsubseteq t_n \cdot C}$$

(imit1) Root Imit 1 $$\frac{\mathcal{I} \vdash \ldots, x \sqsubseteq f(v_1, ..., v_q), \ldots \cdot C}{\mathcal{I}\tau \vdash (\ldots, y_1 \sqsubseteq v_1, \ldots, y_q \sqsubseteq v_q, \ldots)\tau \cdot C \cup \{x \simeq f(y_1, ..., y_q)\}}$$

where y_i are new variables and $\tau = [f(y_1, ..., y_q)/x]$.

(imit2) Root Imit 2: similar to *imit1* but applied to $f(v_1, \ldots, v_q) \sqsubseteq x$.

(end) Reflexive Closure $$\frac{\mathcal{I} \vdash s_1 \sqsubseteq t_1, \ldots, s_i \sqsubseteq t_i, \ldots, s_n \sqsubseteq t_n \cdot C}{\mathcal{I}\tau \vdash (s_1 \sqsubseteq t_1, \ldots, s_i \sqsubseteq s_i, \ldots, s_n \sqsubseteq t_n)\tau \cdot C \cup \{s_i \simeq t_i\}}$$

where τ is a most general unifier of s_i and t_i.

As for previous calculi, the extra conditions are: *a*) the constraint at the conclusion has to be satisfiable, *b*) the inequality $s \sqsubseteq t$ used at the premise is not trivial, and *c*) $l \neq r$ in *root*. Note also that only equality constraints are introduced and that constraints are only used to keep the solution because substitutions are applied to the current $RMPU$-problem in rules *end* and *imit1/2*. In the sequel, we write $\mathcal{MP}$-steps as $\Gamma \cdot C \rightsquigarrow_{\mathcal{MP}} \Gamma' \cdot C'$.

Definition 24 *The substitution θ is a $\mathcal{MP}$-unifier for the RMPU-problem $\mathcal{I} \vdash s \sqsubseteq t$ if there exists a constraint C such that:*

- $\mathcal{I} \vdash s \sqsubseteq t \cdot \emptyset \rightsquigarrow_{\mathcal{MP}}{}^* \mathcal{I} \vdash r_1 \sqsubseteq r_1, \ldots, r_n \sqsubseteq r_n \cdot C$, *where* $r_i \in T(\Sigma, X)$.
- *θ is a solution to C.*

Given a $RMPU$-problem Γ, we can see the computation of $\mathcal{MP}$-unifiers as a search in the $\mathcal{MP}$-derivation tree with $\Gamma \cdot \emptyset$ in the root. Observe that successful leaves contain a sequence of trivial inequalities and a constraint that possibly represents several $\mathcal{MP}$-unifiers.

6.1 Properties of the Calculus $\mathcal{MP}$

The calculus $\mathcal{MP}$ is sound and existentially complete in the following sense.

Theorem 25 (Soundness) *If θ is a $\mathcal{MP}$-unifier for a RMPU-problem Γ, then θ solves Γ.*

Theorem 26 (Completeness) *Let θ be a solution to the RMPU-problem Γ then there exists a $\mathcal{MP}$-unifier for Γ.*

Sketch of proof. The previous analysis states the different cases and how to prevent case 3 c). Then we have to prove that we can finitely apply the rules till all the inequalities to be proved are trivial or have the form $x \sqsubseteq y$. This is so because the complexity of the related $RMPU$-problem decreases when applying the rules: *root* decreases the length of the ground proofs, *decomposition* and *imit1/2* decrease its depth and *end* increases the number of trivial inequalities to be proved. ■

The calculus $\mathcal{MP}$ is efficient because it does not lift the naive solutions concerning case 3 c). Therefore we have stated the basis needed when requiring termination in the sense of the approach followed in [4] for rigid E-unification, where the set of solutions to be lifted is suitably restricted (see the previous comments on *termination versus completeness*). For example, the calculus $\mathcal{MP}$ cannot build the solution $[f(a)/x, f(b)/y]$ to the $RMPU$-problem $\{a \sqsubseteq b\} \vdash x \sqsubseteq y$. Note that we can solve it in other non-naive ways: using *end* to obtain the constraint $\{x \simeq y\}$ or using *root* and *end* to obtain $\{x \simeq a, y \simeq b\}$.

Nevertheless the calculus $\mathcal{MP}$ lacks of termination due to the rule *root* without control. For example, the following infinite sequence of $\mathcal{MP}$-steps is available:

$$\{b \sqsubseteq c\} \vdash a \sqsubseteq b \cdot \emptyset \rightsquigarrow_{\mathcal{MP}} \{b \sqsubseteq c\} \vdash a \sqsubseteq b, c \sqsubseteq b \cdot \emptyset \rightsquigarrow_{\mathcal{MP}} \{b \sqsubseteq c\} \vdash a \sqsubseteq b, c \sqsubseteq b, c \sqsubseteq b \cdot \emptyset \rightsquigarrow_{\mathcal{MP}} \ldots$$

In fact *imit1/2* can be also used to build infinite sequences of $\mathcal{MP}$-steps because we apply the related mgu:

$$\emptyset \vdash x \sqsubseteq f(x) \cdot \emptyset \rightsquigarrow_{\mathcal{MP}} \emptyset \vdash x_1 \sqsubseteq f(x_1) \cdot \{x \simeq f(x_1)\} \rightsquigarrow_{\mathcal{MP}} \emptyset \vdash x_2 \sqsubseteq f(x_2) \cdot \{x \simeq f(x_1), x_1 \simeq f(x_2)\} \rightsquigarrow_{\mathcal{MP}} \ldots$$

7 Integrating Rigid Monotonic Preordered Unification into Free Variable Tableaux

We extend $RMPU$-problems to simultaneous problems in order to close every branch of a tableau at once.

Definition 27 *A Simultaneous Rigid Monotonic Preordered Unification (SRMPU-)problem consists of a finite sequence of RMPU-problems.*

We present the calculus $\mathcal{SMP}$ as the natural extension of $\mathcal{MP}$ dealing with simultaneity. Then $\mathcal{SMP}$ transforms sequences of $RMPU$-problems by the (local) application of *root, dec, imit1, imit2* or *end* to any of its single $RMPU$-problems. Variables behave rigidly, hence we join all the constraint sets into a unique set and we globally apply the related mgu in rules *end* and *imit1/2*. For example for the simultaneous version of *imit1* we would have:

$$\frac{\Gamma_1, \ldots, \mathcal{I} \vdash s_1 \sqsubseteq t_1 \ldots, x \sqsubseteq f(v_1, ..., v_q), \ldots s_n \sqsubseteq t_n, \ldots, \Gamma_p \cdot C}{(..., \mathcal{I} \vdash s_1 \sqsubseteq t_1 \ldots, y_1 \sqsubseteq v_1, \ldots, y_q \sqsubseteq v_q, \ldots s_n \sqsubseteq t_n, ...)\tau \cdot C \cup \{x \simeq f(y_1, ..., y_q)\}}$$

where y_i are new variables and $\tau = [f(y_1, ..., y_q)/x]$.

The extra conditions for the applicability of these rules coincides with those required for $\mathcal{MP}$ previously. We define the relation $\rightsquigarrow_{\mathcal{SMP}}$ and the concept of $\mathcal{SMP}$-unifier as for $\mathcal{MP}$, that is we search in the $\mathcal{SMP}$-derivation tree with $\Gamma_1, \ldots, \Gamma_n \cdot \emptyset$ in the root.

The calculus $\mathcal{SMP}$ satisfies the same properties than $\mathcal{MP}$, then it is sound, existentially complete and non-terminating.

Theorem 28 (Soundness) *Let θ be a $\mathcal{SMP}$-unifier for the SRMPU-problem $\Gamma_1, \ldots, \Gamma_n$, then θ is a solution to Γ_i, $1 \leq i \leq n$.*

Theorem 29 (Completeness) *Let θ be a solution to every RMPU-problem Γ_i, $1 \leq i \leq n$, then there exists a $\mathcal{SMP}$-unifier for $\Gamma_1, \ldots, \Gamma_n$.*

We integrate the calculus $\mathcal{SMP}$ into free variable tableaux through the following closure rule:

(SRMPU-Closure Rule) Let $\mathcal{T}$ be a free variable with branches $B_1, \ldots, B_n$. Let $\mathcal{I}_i$ be the theory occurring in B_i and $s_i \not\sqsubseteq t_i$ be a dis-inequality from B_i, $1 \leq i \leq n$. Then $\mathcal{T}$ is closed if there exists a $\mathcal{SMP}$-unifier for the SRMPU-problem $\mathcal{I}_1 \vdash s_1 \sqsubseteq t_1, \ldots, \mathcal{I}_n \vdash s_n \sqsubseteq t_n$.

Let $\mathcal{SMPT}$ be the tableau system composed of the rules $\alpha, \beta, \gamma', \delta'$ [5] and the previous *SRMPU-Closure* rule. As when monotonicity was not considered, we use this system expanding the tableau till the *SRMPU-Closure* rule closes it.

The soundness of $\mathcal{SMPT}$ follows from Theorem 28. For proving completeness, we assume that the ground tableau system composed of a) α, β, γ and δ [5], b) the preordered rules *Tr*, *Ref* and *Mon* below, and c) atomic ground closure is complete (see [6] for details). In *Mon*, $c[]_p$ denotes any ground context.

$$\textit{Ref)} \quad \frac{}{t \sqsubseteq t} \qquad \textit{Tr)} \quad \frac{t_1 \sqsubseteq t_2 \quad t_2 \sqsubseteq t_3}{t_1 \sqsubseteq t_3} \qquad \textit{Mon)} \quad \frac{t_1 \sqsubseteq t_2}{c[t_1]_p \sqsubseteq c[t_2]_p}$$

Then we use any ground closed tableau and Theorem 29 to conclude that $\mathcal{SMPT}$ is also complete.

Theorem 30 *A set of sentences Φ is not satisfiable in monotonic structures iff there exists a closed $\mathcal{SMPT}$-tableau for Φ.*

Notice that the calculus $\mathcal{SMP}$ involved in the *SRMPU-Closure* rule is non-terminating, so it cannot be used as a decision procedure. Therefore such calculi may never terminate when solving a not satisfiable $SRMPU$-problem related to a certain tableau expansion. In this sense termination is necessary if we want a suitable integration of rigid monotonic preordered unification into tableaux.

8 Conclusions and Future Work

We have integrated monotonic preorders in tableaux using a simultaneous rigid unification calculus. When monotonicity is not considered, we have defined the sound and complete unification calculus $\mathcal{P}$ that is also terminating, because inequalities can be erased once they have been used. Then rewrite techniques have been introduced to build the calculus $\mathcal{P}1$ which prunes the search space efficiently. Such a calculus is complete because it includes new rules in order to ensure the commutation between the oriented inequalities.

We have discussed at length about the problems that arise when dealing with monotonicity. Then we have presented the sound and existentially complete calculus $\mathcal{SMP}$ to solve simultaneous rigid monotonic preordered problems. As a consequence of the undecidability of simultaneous rigid E-unification [4], the calculus $\mathcal{SMP}$ is not terminating, however it is efficient because it avoids certain naive solutions. In this sense we have stated the basis required to achieve termination which seems to be essential to obtain a suitable integration of any unification calculus into tableau methods.

At present we are searching for a terminating calculus. Moreover we are studying *a)* how to reduce the ground solutions that are necessary to be lifted and *b)* how to introduce rewrite techniques.

Acknowledgements. We are greatly indebted to Susana Nieva and Javier Leach for helpful discussions and comments.

References

1. L. Bachmair, H. Ganzinger. *Rewrite techniques for transitive relations*. Procs. LICS'94, 384–393. IEEE Computer Science Press, 1994.
2. L. Bachmair, H. Ganzinger, C. Lynch, W. Snyder. *Basic paramodulation and superposition*. Proc. CADE'11. LNAI 607, 462–476. Springer, 1992.
3. L. Bachmair, H. Ganzinger, C. Lynch, W. Snyder. *Basic paramodulation*. Information and Computation 121, 172–192, 1995.
4. A. Degtyarev, A. Voronkov. *What you always wanted to know about rigid E-unification*. Journal of Automated Reasoning 20(1), 47–80, 1998.
5. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Second edition. Springer, 1996.
6. A. Gavilanes, J. Leach, P.J. Martín, S. Nieva. *Semantic Tableaux for a Logic with Preorders and Dynamic Declarations*. Tableaux'97 (Position Paper) CRIN 97-R-030, 7–12, 1997.
7. J. Gallier, P. Narendran, D. Plaisted, W. Snyder. *Rigid E-unification: NP-completeness and applications to equational matings*. Information and Computation, 87(1/2), 129–195, 1990.
8. J. Levy, J. Agustí. *Bi-rewrite systems*. J. Symbolic Computation 22, 279–314, 1996.
9. R. Nieuwenhuis, A. Rubio. *Theorem Proving with Ordering and Equality Constrained Clauses*. J. Symbolic Computation 19, 321–351, 1995.
10. R. Nieuwenhuis. *Simple LPO constraint solving methods*. Information Processing Letters 47, 65–69, 1993.
11. M. Rusinowitch. *Theorem proving with resolution and superposition:* . J. Symbolic Computation 11, 21–49, 1991.

The Mosaic Method for Temporal Logics

Maarten Marx[1], Szabolcs Mikulás[2⋆], and Mark Reynolds[3]

[1] Institute for Logic, Language and Computation,
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands.
marx@wins.uva.nl
[2] Department of Computer Science, Birkbeck College,
Malet Street, London WC1E 7HX, UK.
szabolcs@dcs.bbk.ac.uk
[3] School of Information Technology, Murdoch University,
Perth, Western Australia 6150, Australia.
m.reynolds@murdoch.edu.au

Abstract. The aim of this paper is to apply the mosaic method for proving complexity, Hilbert-style and tableau completeness results for Prior's temporal logic over linear flows of time. We also show how to implement the mosaic idea for automated theorem-proving. Finally we indicate the modifications required to achieve similar results for special linear flows of time and for the more expressive logic of *until* and *since*.

1 Introduction

The aim of this paper is to apply the mosaic method for proving decidability, Hilbert-style (strong) completeness and tableau (weak) completeness of temporal logics over linear flows of time. We also consider implications for automated theorem-proving with temporal logics. Most of these results are known, but we feel that the mosaic approach serves as a general method and slight modifications should enable us to prove important new results for many different logics of linear flows of time. So the idea here is to illustrate the general method on the basic logics avoiding the detail needed for any specific new results.

The origin of the mosaic method is in algebraic logic to prove decidability of the equational theories of certain classes of algebras of relations, cf. [13], [14]. The main idea is to show that the existence of a model is equivalent to the existence of a finite set of partial models, called mosaics, satisfying certain coherency conditions. This gives us a decision procedure, and, intuitively, a systematic procedure to check the theoremhood of a certain formula. Recently the mosaic method has been applied to prove decidability, Hilbert-style axiomatizability and complexity results for various modal logics, cf., e.g., [9], [12], and [21].

The paper is organized as follows. First we give the definition of mosaics and saturated sets of mosaics, SSMs, for temporal logic with future and past operators for linear flows of time. We will prove a key lemma stating that satisfiability

⋆ Research supported by EPSRC grants No. GR/L82441 and No. GR/L26872.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 324–340, 2000.

is equivalent to the existence of an SSM. Then we show how to achieve Hilbert-style and tableau completeness, decidability and complexity results using the key lemma. Section 6 is devoted to an automated theorem-prover: we will see that the mosaic approach gives a reasonably efficient standard method for automated theorem-proving in which only minor modifications need be made to move from one temporal logic to another. Next we will sketch how to modify the basic definitions to cope with special linear flows of time. Finally, we indicate the necessary modification for the more expressive temporal language of *until* and *since*.

2 Mosaics for Temporal Logic

In this section we show how to define mosaics and saturated sets of mosaics, SSMs, for temporal logic with future F and past P over linear flows of time. We will show that the existence of an SSM for a formula φ is equivalent to the existence of a model for φ.

Let us start with the basic definitions for temporal logic. The language consists of (countably many) propositional variables (also called atoms), propositional connectives (the primitives $\neg$ and $\wedge$ and the usual defined connectives $\vee$, $\rightarrow$ and $\leftrightarrow$) and the temporal connectives F (future) and P (past). We will also use the propositional constants $\top$ (true) and $\bot$ (false); they can be defined as $p \vee \neg p$ and $p \wedge \neg p$, respectively, for an arbitrary atom p. A frame $\mathfrak{T}$ is a non-empty set T equipped with a strict linear ordering[1] $<$: $\mathfrak{T} = (T, <)$. A model is a frame together with an evaluation v of the propositional variables: $v(p) \subseteq T$ for every propositional variable p. Truth of formulas in a model $(T, <, v)$ is defined as follows: for $t \in T$, propositional variable p and formulas φ and ψ,

$$\begin{aligned} t \Vdash p &\iff t \in v(p) \\ t \Vdash \neg\varphi &\iff \text{not } t \Vdash \varphi \\ t \Vdash \varphi \wedge \psi &\iff t \Vdash \varphi \text{ and } t \Vdash \psi \\ t \Vdash \mathsf{F}\varphi &\iff t' \Vdash \varphi \text{ for some } t' > t \\ t \Vdash \mathsf{P}\varphi &\iff t' \Vdash \varphi \text{ for some } t' < t. \end{aligned}$$

We will use the defined temporal connectives G (always in the future) and H (always in the past) as well. Here are their definitions:

$$\mathsf{G}\varphi \equiv \neg\mathsf{F}\neg\varphi \qquad \mathsf{H}\varphi \equiv \neg\mathsf{P}\neg\varphi.$$

Note that we assumed that the linear order is irreflexive ($t \not< t$ for every $t \in T$); the "weak" version of future (using $\leq$ instead of $<$) can be expressed as $\varphi \vee \mathsf{F}\varphi$.

We assume familiarity with the basic notions of (temporal) logic such as subformulas, consistency, satisfiability, validity, maximal consistent sets, etc. Let X be a fixed set of formulas closed under subformulas and single negation, in the language of F and P containing at most countably many atoms.

[1] an irreflexive, transitive and connected ordering

Definition 1. *By a* mosaic μ *we mean a structure* $(\langle m_0, m_1\rangle, \ell)$ *satisfying the following. The* labelling function ℓ *associates a subset of* X *to each element of the* base $\{m_0, m_1\}$ *of* μ *such that, for every formula in* X *and index* $i < 2$,

1. $\varphi \in \ell(m_i) \iff \neg\varphi \notin \ell(m_i)$;
2. $\varphi \wedge \psi \in \ell(m_i) \iff \{\varphi, \psi\} \subseteq \ell(m_i)$;
3. $\mathsf{G}\varphi \in \ell(m_0) \Rightarrow \mathsf{G}\varphi \in \ell(m_1)$;
4. $\mathsf{G}\varphi \in \ell(m_0) \Rightarrow \varphi \in \ell(m_1)$;
5. $\mathsf{H}\varphi \in \ell(m_1) \Rightarrow \mathsf{H}\varphi \in \ell(m_0)$;
6. $\mathsf{H}\varphi \in \ell(m_1) \Rightarrow \varphi \in \ell(m_0)$.

We also allow mosaics with a singleton base ($m_0 = m_1$)*, but then we require conditions 1 and 2 only.*

An isomorphism *between mosaics is a label-preserving bijection between the two bases.*

We say that conditions 5 and 6 are the *mirror images* of conditions 3 and 4, respectively. In general, mirror images are defined by swapping future operators with the corresponding past operators and reversing the order.

In the Hilbert-style completeness proof, the labelling set X will be the set of all formulas and we will label with maximally consistent sets of formulas, while in the decidability and tableau completeness proofs, X can be chosen as the set of subformulas and their single negations of the formula ξ that we are interested in.

Definition 2. *A set* M *of mosaics is a* saturated set of mosaics*, an SSM for short, if it satisfies the following saturation conditions.*
For every $\mu = (\langle m_0, m_1\rangle, \ell) \in M$,

1. *if* $\mathsf{F}\varphi \in \ell(m_1)$*, then there is a mosaic* $\mu' = (\langle m'_0, m'_1\rangle, \ell') \in M$ *such that* $\ell(m_1) = \ell'(m'_0)$ *and* $\varphi \in \ell'(m'_1)$;
2. *if* $\mathsf{F}\varphi \in \ell(m_0)$*, then either* $\mathsf{F}\varphi \in \ell(m_1)$ *or there are mosaics* $\mu' = (\langle m'_0, m'_1\rangle, \ell')$ *and* $\mu'' = (\langle m''_0, m''_1\rangle, \ell'')$ *in* M *such that* $\ell(m_0) = \ell'(m'_0)$, $\ell(m_1) = \ell''(m''_1)$ *and* $\varphi \in \ell'(m'_1) = \ell''(m''_0)$;
3. *if* $\mathsf{P}\varphi \in \ell(m_0)$*, then there is a mosaic* $\mu' = (\langle m'_0, m'_1\rangle, \ell') \in M$ *such that* $\ell(m_0) = \ell'(m'_1)$ *and* $\varphi \in \ell'(m'_0)$;
4. *if* $\mathsf{P}\varphi \in \ell(m_1)$*, then either* $\mathsf{P}\varphi \in \ell(m_0)$ *or there are mosaics* $\mu' = (\langle m'_0, m'_1\rangle, \ell')$ *and* $\mu'' = (\langle m''_0, m''_1\rangle, \ell'')$ *in* M *such that* $\ell(m_0) = \ell'(m'_0)$, $\ell(m_1) = \ell''(m''_1)$ *and* $\varphi \in \ell'(m'_1) = \ell''(m''_0)$.

Given a set Γ *of formulas we say that there exists an SSM for* Γ *if there is an SSM such that* Γ *is contained in the label of one of the points of a mosaic. If* Γ *is a singleton set* $\{\gamma\}$*, then we will talk about* γ*-SSM.*

Again, 3 an 4 are the mirror images of 1 and 2, respectively. Later we will not bother with formulating the mirror images of conditions if no confusion is likely to occur.

We are ready to formulate our main lemma.

Lemma 1. *For any set Γ of formulas, Γ is satisfiable iff there exists an SSM for Γ.*

Proof. First we show the easy direction, viz., that for any satisfiable set Γ of formulas, there is an SSM for Γ. Let $(T, <, v)$ be a model and $t \in T$ such that Γ holds at t: $t \Vdash \Gamma$. Let X be a set containing Γ which is closed under subformulas and single negations. For any $s \in T$, we let $\ell(s) = \{\varphi : s \Vdash \varphi\} \cap X$. For any $u < v \in T$, we define the mosaic $\mu(u, v) = (\langle u, v\rangle, \ell(u), \ell(v))$. It is easy to check that $\mu(u, v)$ is indeed a mosaic. Furthermore, the set $\{\mu(u, v) : u < v \in T\}$ is an SSM for Γ, since it contains all the mosaics from the same model for Γ.

To prove the other direction, assume that M is an SSM for Γ. We have to show that there is a model satisfying Γ. We will construct such a model step by step using the elements of M as building blocks. The idea is to "cure the defects" of the participating mosaics by "gluing" mosaics together. The model we are going to build will be a substructure of the rational numbers. Let X be the labelling set of M (i.e., we label the points with subsets of X).

First we enumerate all the possible *defects* of labelled structures over subsets of the rational numbers $\mathbb{Q}$:

$$D = \{\langle q, \mathsf{F}\varphi\rangle, \langle q, \mathsf{P}\varphi\rangle : q \in \mathbb{Q} \text{ and } \mathsf{F}\varphi, \mathsf{P}\varphi \in X\}.$$

Note that we assume that the language contains at most countably many atoms, hence the set of formulas in X is at most countable, also the cardinality of $\mathbb{Q}$ is ω; thus $|D| \leq \omega$.

Let $\mathfrak{I} = (I, <, \ell)$ be a labelled structure such that $I \subseteq \mathbb{Q}$, $<$ is the ordering of rational numbers according to magnitude restricted to I, and $\ell : I \to \mathcal{P}(X)$. An element $\langle q, \mathsf{F}\varphi\rangle$ of D is a *future defect* of $\mathfrak{I}$ if

1. $q \in I$
2. $\mathsf{F}\varphi \in \ell(q)$ and
3. for every $p \in I$ such that $q < p$, we have $\varphi \notin \ell(p)$.

That is, a future defect is an "unfulfilled eventuality". Past defects are defined analogously.

We say that a labelled structure $\mathfrak{I} = (I, <, \ell)$ is *coherent* if it satisfies the coherency conditions as in Definition 1: the labels are propositionally consistent, $\mathsf{G}\varphi \in \ell(i)$ implies $\mathsf{G}\varphi, \varphi \in \ell(j)$ for every $j > i$, and $\mathsf{H}\varphi \in \ell(i)$ implies $\mathsf{H}\varphi, \varphi \in \ell(j)$ for every $j < i$. Note that a coherent structure may contain defects.

Below, we will define coherent labelled substructures $\mathfrak{I}_n = (I_n, <, \ell_n)$ of the rational numbers. Let $\sigma : \omega \to \omega$ be a map such that, for every $j \in \omega$, there are infinitely many $k \in \omega$ such that $\sigma(k) = j$. We will use σ as a scheduling function that tells us which defect $D(\sigma(n))$ to cure in the nth step of the construction.

0TH STEP. Take a mosaic $\mu \in M$ such that $\Gamma \subseteq \ell(m_i)$ for some $i < 2$. If $m_0 \neq m_1$, we define $\mathfrak{I}_0 = (\{0, 1\}, \langle 0, 1\rangle, \ell_0)$, where $\ell_0(j) = \ell(m_j)$ for $j < 2$. Otherwise, if the base of μ is a singleton, we define $\mathfrak{I}_0 = (\{0\}, \emptyset, \ell(m_0))$. Clearly, $\mathfrak{I}_0$ is coherent.

$n+1$ST STEP. Assume that we have already constructed a finite linear order of rational numbers $\langle i_0 < i_1 < \ldots < i_k \rangle$ with a coherent labelling ℓ_n such that, for every j, $(\langle i_j, i_{j+1} \rangle, \ell(i_j), \ell(i_{j+1}))$ is (isomorphic to) an element of M. Let us denote this structure by $\mathfrak{I}_n = (I_n, <, \ell_n)$. Let $\sigma(n+1) = l$. Consider the list D of potential defects and take its lth element $D(l)$. We assume that $D(l)$ has the form $\langle i, \mathsf{F}\varphi \rangle$; for past defects a straightforward modification of the construction below works. Next we check if $D(l)$ is an actual defect of $\mathfrak{I}_n$. If it is not (e.g., $i \notin I_n$ or there is $j > i$ in I_n such that $\varphi \in \ell_n(j)$), we define $\mathfrak{I}_{n+1} = \mathfrak{I}_n$.

So assume that $D(l)$ is in fact a defect of $\mathfrak{I}_n$. Let i_j be the greatest element of the order $i_0 < \ldots < i_k$ such that $\mathsf{F}\varphi \in \ell_n(i_j)$; since $D(l)$ is a defect of $\mathfrak{I}_n$, such i_j exists.

If i_j is the last point of I_n in the order ($j = k$), then there is a mosaic $(\langle m'_0, m'_1 \rangle, \ell'(m'_0), \ell'(m'_1))$ in M that we can glue to $(\langle i_{k-1}, i_k \rangle, \ell_n(i_{k-1}), \ell_n(i_k))$ witnessing $\mathsf{F}\varphi \in \ell_n(i_k)$; that is, the two mosaics satisfy Definition 2.1, and $\ell'(m'_0) = \ell_n(i_k)$ and $\varphi \in \ell'(m'_1)$. We add a new point i_{k+1} at the end of the linear order with label $\ell'(m'_1)$ to our structure: we define $\mathfrak{I}_{n+1}$ as the order $\langle i_0 < \ldots < i_k < i_{k+1} \rangle$ with labelling $\ell_{n+1}(i_{k+1}) = \ell'(m'_1)$ and $\ell_{n+1}(i_p) = \ell_n(i_p)$ for $0 \leq p \leq k$.

Now assume that $i_j < i_k$. Let us consider the mosaic $(\langle i_j, i_{j+1} \rangle, \ell_n(i_j), \ell_n(i_{j+1}))$. By the maximality condition on i_j, we have that $\neg\mathsf{F}\varphi \in \ell_n(i_{j+1})$. Then we can find two mosaics $(\langle m'_0, m'_1 \rangle, \ell'(m'_0), \ell'(m'_1))$ and $(\langle m''_0, m''_1 \rangle, \ell''(m''_0), \ell''(m''_1))$ such that $\ell'(m'_0) = \ell_n(i_j)$, $\varphi \in \ell'(m'_1) = \ell''(m''_0)$ and $\ell''(m''_1) = \ell_n(i_{j+1})$, see Definition 2.2. Then we insert a new point i' between i_j and i_{j+1} with a label $\ell'(m'_1)$: we define $\mathfrak{I}_{n+1}$ as the order $\langle i_0 < \ldots < i_j < i' < i_{j+1} < \ldots < i_k \rangle$ with labelling $\ell_{n+1}(i') = \ell'(m'_1)$ and $\ell_{n+1}(i_p) = \ell_n(i_p)$ for $0 \leq p \leq k$.

Thus, we can define a structure $\mathfrak{I}_{n+1} = (I_{n+1}, <, \ell_{n+1})$ where $D(l)$ is not a defect anymore.[2] Clearly $\mathfrak{I}_{n+1}$ is a union of copies of elements of M and it is coherent. Furthermore, for every $j \in I_n$, we have that $\ell_n(j) = \ell_{n+1}(j)$; that is the new labelling ℓ_{n+1} is compatible with the old one ℓ_n. Also note that once we cured a defect by providing a witness, the same defect cannot occur in any expansion of the structure.

ωTH STEP. Take the union $\mathfrak{I} = (I, <, \ell)$ of the labelled structures defined so far (the compatibility of the labellings makes this well defined). It is easy to see that we get a coherent linear structure that does not contain any defect — every potential defect has been dealt with infinitely many times, hence we can be sure that if it is an actual defect in some stage, then we repair it in a later step.

Let the valuation v be defined as $i \in v(p) \iff p \in \ell(i)$, for every point $i \in I$ and atom p (if p is not in our labelling set X, then let $v(p)$ be arbitrary).

Consider the model $(I, <, v)$. We claim that $i \Vdash \varphi \iff \varphi \in \ell(i)$, for every $i \in I$ and $\varphi \in X$. We proceed by induction on the complexity of φ. The case of atoms and the propositional connectives are easy. Now assume that $\mathsf{F}\varphi \in \ell(i)$. Since $(I, <, \ell)$ does not contain any defect, there is $j > i$ such that $\varphi \in \ell(j)$. By the induction hypothesis, $j \Vdash \varphi$, whence $i \Vdash \mathsf{F}\varphi$. If $i \Vdash \mathsf{F}\varphi$, then there is

[2] If $D(l)$ is a past defect $\langle i, \mathsf{P}\varphi \rangle$, then use conditions 3 and 4 in Definition 2.

$j > i$ such that $j \Vdash \varphi$. Then $\varphi \in \ell(j)$ by the induction hypothesis. Hence, by coherency, $\mathsf{F}\varphi \in \ell(i)$. The proof for $\mathsf{P}\varphi$ is completely analogous.

Since the mosaic in the 0th step was labelled by Γ, we get that $(I, <, v)$ satisfies Γ.

3 Hilbert-Style Completeness

In this section, we show how to prove strong Hilbert-style completeness (that is, we will prove that derivability in a Hilbert-style axiom system coincides with semantical consequence) using mosaics and Lemma 1. This has the advantage that we do not have to create a model for a consistent formula set, an SSM will do.

First let us recall from [5] that the minimal temporal logic K_t is the normal logic with conjugate modalities F and P interpreted over transitive frames. We define the calculus $\vdash$ as follows: its axioms include enough propositional axioms, the axioms for K_t

$$\begin{array}{ll} (K_F) & \mathsf{G}(\varphi \to \psi) \to (\mathsf{G}\varphi \to \mathsf{G}\psi) \\ (K_P) & \mathsf{H}(\varphi \to \psi) \to (\mathsf{H}\varphi \to \mathsf{H}\psi) \\ (C_F) & \varphi \to \mathsf{GP}\varphi \\ (C_P) & \varphi \to \mathsf{HF}\varphi \\ (4_F) & \mathsf{G}\varphi \to \mathsf{GG}\varphi \\ (4_P) & \mathsf{H}\varphi \to \mathsf{HH}\varphi \end{array}$$

plus the following linearity axioms

$$(\textit{Lin.1}) \qquad \Box\varphi \to \mathsf{HG}\varphi$$

and

$$(\textit{Lin.2}) \qquad \Box\varphi \to \mathsf{GH}\varphi$$

where $\Box\varphi$ stands for $\mathsf{H}\varphi \wedge \varphi \wedge \mathsf{G}\varphi$; the derivation rules are Modus Ponens and Universal Generalization (or Necessitation):

$$\text{from } \varphi \text{ infer } \mathsf{G}\varphi \text{ and } \mathsf{H}\varphi.$$

Our next aim is to show that $\vdash$ is a strongly complete inference system for linear temporal logic, i.e., for every set Γ of formulas,

$$\Gamma \text{ is consistent iff } \Gamma \text{ is satisfiable}^3.$$

In the light of Lemma 1, the strong completeness theorem is equivalent to the following.

Lemma 2. *For any set Γ of formulas, Γ is consistent iff there exists an SSM for Γ.*

[3] Recall that consistency of Γ means that it is impossible to derive contradictions from Γ by using the inference system, while Γ is satisfiable if it has a model.

Proof. By Lemma 1, if there exists an SSM for Γ, then Γ is satisfiable, and hence consistent by the soundness of $\vdash$ (we leave the easy task to check this to the reader).

For the other direction, let us assume that Γ is consistent. We have to show that there is an SSM for Γ.

Let our labelling set be the set of all formulas[4] (in the language of Γ). We define a set of mosaics as the collection of all $(\langle 0\rangle, \ell(0))$ and $(\langle 0,1\rangle, \ell(0), \ell(1))$ where $\ell(0)$ and $\ell(1)$ are maximal consistent sets, MCSs for short, satisfying

$$\{\varphi, \mathsf{G}\varphi : \mathsf{G}\varphi \in \ell(0)\} \subseteq \ell(1). \tag{1}$$

We will say that $\langle \ell(0), \ell(1)\rangle$ satisfies (1). Using the axioms for transitivity (4_F and 4_P) and for the connection between F and P (C_F and C_P), one can easily show that the conditions of Definition 1 are met. Also, (1) is equivalent to

$$\{\varphi, \mathsf{H}\varphi : \mathsf{H}\varphi \in \ell(1)\} \subseteq \ell(0). \tag{2}$$

Since we can extend Γ to a MCS, we have a mosaic witnessing Γ.

Let X, Y, Z denote MCSs. It suffices to show 1 and 2 below (and their mirror images) to prove that we have a saturated set of mosaics.

1. For every X and $\mathsf{F}\varphi \in X$, there is a Y such that $\varphi \in Y$ and $\langle X, Y\rangle$ satisfies (1).
2. Let X, Y and ψ be such that $\langle X, Y\rangle$ satisfies (1), $\mathsf{F}\psi \in X$ and both $\neg\mathsf{F}\psi$ and $\neg\psi$ are in Y. Then there is a Z such that $\psi \in Z$ and both $\langle X, Z\rangle$ and $\langle Z, Y\rangle$ satisfy (1).

1 holds by general $K4$-consideration (as in [5]).

2 holds because of the following. By 1, the set $\{\varphi, \mathsf{G}\varphi : \mathsf{G}\varphi \in X\} \cup \{\psi\}$ is consistent. Let Z be a MCS containing it. We show that Z satisfies the claim. It suffices to show that $\langle Z, Y\rangle$ satisfies (1), i.e., $\langle Y, Z\rangle$ satisfies (2). Let us assume that $\mathsf{H}\varphi \in Y$. Then we have $\mathsf{HH}\varphi \in Y$ by (4_P). Hence $\mathsf{H}(\varphi \wedge \mathsf{H}\varphi) \in Y$ and thus $\mathsf{H}(\psi \to (\varphi \wedge \mathsf{H}\varphi)) \in Y$. Recall that $\neg\psi \in Y$, whence $\psi \to (\varphi \wedge \mathsf{H}\varphi) \in Y$. As Y contains $\neg\mathsf{F}\psi$, we have that $\mathsf{G}(\psi \to (\varphi \wedge \mathsf{H}\varphi))$ as well. Putting these together we see that Y contains $\Box(\psi \to (\varphi \wedge \mathsf{H}\varphi))$. By ($Lin.1$) we can conclude that $\mathsf{HG}(\psi \to (\varphi \wedge \mathsf{H}\varphi))$ is in Y, whence $\mathsf{G}(\psi \to (\varphi \wedge \mathsf{H}\varphi))$ is in X (by (1)). Since $\langle X, Z\rangle$ satisfies (1) and $\psi \in Z$, we have that $\varphi \wedge \mathsf{H}\varphi$ is in Z as required.

The proof of the mirror images is similar.

Thus the set of all mosaics defined above forms an SSM.

4 Decidability and Complexity

By Lemma 1, the decidability proof amounts to proving that the existence of an SSM for a given formula is decidable.

[4] In this case, the labelling sets may be infinite sets of formulas. We allowed this in the definition of mosaics and SSM. For decidability, complexity and weak completeness purposes we will restrict the labelling sets to finite ones.

Lemma 3. *For any formula ξ, it is decidable whether there is an SSM for ξ.*

Proof. Use the set of subformulas of ξ and their negations as the labelling set. Then the usual argument works: given the complexity of ξ, we can compute the number of mosaics; since checking the saturation conditions is decidable, we can decide whether any subset of the set of all mosaics form a saturated set of mosaics for ξ.

Below we show how to use the above idea to prove complexity results. One can prove NP-completeness of linear temporal logic (due to Ono and Nakamura [15]) by considering non-standard models for ξ of size $|\xi|$. However, using the mosaic idea, we can avoid non-standard (not linear) models. Below we will show that if there is an SSM for a formula ξ with complexity n, then there is an SSM with at most $n^2 + 2n + 1$ elements. Given a set of mosaics checking the saturation conditions can be done in polynomial time. Thus we get a proof for the NP-completeness[5] of linear temporal logic by guessing an SSM of size $O(n^2)$ and checking in polynomial time whether it is indeed an SSM.

Lemma 4. *Let ξ be a formula with complexity n. If ξ is satisfiable, then there is an SSM for ξ with size $O(n^2)$.*

Proof. Let $(W, <, v)$ be a (linear) model such that $w_0 \Vdash \xi$ for some $w_0 \in W$. Let X be the set of subformulas and their negations of ξ and put $Y = \{\mathsf{F}\varphi \in X : \text{ there exists } w \in W \text{ such that } w \Vdash \varphi\} \cup \{\mathsf{P}\varphi \in X : \text{ there exists } w \in W \text{ such that } w \Vdash \varphi\}$. For every $w, w' \in W$, let $\ell(w) = \{\varphi \in X : w \Vdash \varphi\}$ and let $w \equiv w'$ iff $\ell(w) = \ell(w')$. Note that there are finitely many equivalence classes, since X is finite. For every formula $\mathsf{F}\varphi \in Y$, let $W(\varphi) = \{w \in W : w \Vdash \varphi\}$. Let $\overline{w}(\varphi)$ be a *maximal* element of $W(\varphi)$ in the following sense:

$$(\forall w' \in W(\varphi))(\exists w'' \in W(\varphi))w' \leq w'' \ \&\ w'' \equiv \overline{w}(\varphi).$$

The existence of a maximal element can be shown by an easy induction on the number of equivalence classes.

For every $\mathsf{F}\varphi \in Y$ choose a maximal element from $W(\varphi)$. Similarly, for every $\mathsf{P}\varphi \in Y$, choose a *minimal* element $\underline{w}(\varphi)$ from $W(\varphi)$ (we leave the details to the reader).

Let

$$W' = \{w_0\} \cup \{\overline{w}(\varphi) : \mathsf{F}\varphi \in Y\} \cup \{\underline{w}(\varphi) : \mathsf{P}\varphi \in Y\}.$$

Note that $|W'|$ is $O(n)$. Let M be

$$\{(\langle x, y\rangle, \ell(x), \ell(y)) : x, y \in W', (\exists x' \in W)(\exists y' \in W)x \equiv x' \ \&\ y \equiv y' \ \&\ x' < y'\}.$$

We claim that M is the desired SSM for ξ. Clearly, $|M|$ is $O(n^2)$, and M consists of mosaics.

It remains to show the saturation conditions. Let $(\langle x, y\rangle, \ell(x), \ell(y)) \in M$.

[5] The problem is NP-hard as the logic embeds classical propositional logic.

First assume that $\mathsf{F}\varphi \in \ell(y)$. Let $y' \in W$ such that $y \equiv y'$. Since $y' \Vdash \mathsf{F}\varphi$, there is $z \in W$ such that $z \Vdash \varphi$ and $y' < z$. Let $\overline{w}(\varphi)$ be the maximal element in $W(\varphi)$ that we put in W'. By the maximality of $\overline{w}(\varphi)$, there is $z' \in W$ such that $z \leq z'$ and $z' \equiv \overline{w}(\varphi)$. By this observation, we get that $(\langle y, \overline{w}(\varphi)\rangle, \ell(y), \ell(\overline{w}(\varphi))) \in M$.

Now assume that $\mathsf{F}\varphi \in \ell(x)$ and $\neg\mathsf{F}\varphi, \neg\varphi \in \ell(y)$. Let $x', y' \in W$ such that $x \equiv x'$, $y \equiv y'$ and $x' < y'$. Then there is $z \in W$ such that $x' < z < y'$ and $z \Vdash \varphi$. Let $\overline{w}(\varphi)$ be the maximal element of $W(\varphi)$ that we put in W'. Then there is $z' \in W$ such that $z \leq z'$ and $z' \equiv \overline{w}(\varphi)$. Hence $(\langle x, \overline{w}(\varphi)\rangle, \ell(x), \ell(\overline{w}(\varphi))) \in M$. Since $y' \Vdash \neg\mathsf{F}\varphi$, $y' \Vdash \neg\varphi$, and $z \Vdash \varphi$ we cannot have $y' \leq z'$. Thus $z' < y'$ and $(\langle \overline{w}(\varphi), y\rangle, \ell(\overline{w}(\varphi)\rangle, \ell(y)) \in M$ as well. Thus we can insert the appropriate point between x and y.

Checking the saturation conditions for past formulas is completely analogous.

5 Tableau Completeness

Now we show how to use the mosaic idea to give a complete tableau axiomatization for linear temporal logic. The tableau system below follows the mosaic idea and it shows that, in theory, the mosaic approach is suitable for achieving tableau completeness. We are aware that the tableau system below is not very attractive from the implementational point of view. In Section 6, we show how to design an automated theorem-prover directly from the mosaic idea without relying on the tableau formalism below.

Semantic tableaux are used in temporal logics, cf. [6,7,8,10,19,22,23]. [6] concentrates on the restricted language with future as the only temporal connective, and gives cut-free tableau systems for discrete and dense linear and branching time logics. Goré's systems are cut-free, but use formulas that are not subformulas of the original formula (though there is a bound on these formulas). Our system below uses a weak version of the cut rule, where the application of the rule is restricted to (perhaps negated) subformulas.[6] [10] introduces a cut-free sequent calculus for various temporal logics including linear (transitive and connected) temporal logic. A difference between Kashima's calculus and our system below is the way how linearity is handled. Intuitively, when one wants to build a model for a temporal formula, and in particular to create a witness for a future formula $\mathsf{F}\varphi$, one has to find out where to put the required φ-world into the linear order (e.g., it cannot come after a time point satisfying $\neg\mathsf{F}\varphi$). Kashima introduces a rule (called *TrCo*) with branching according to the possibilities of the location of the witnessing point. Our strategy below is firstly to decide if a future point satisfies $\mathsf{F}\varphi$ or $\neg\mathsf{F}\varphi$ and secondly to apply the corresponding rule to create a φ-witness with a suitable location. [19] gives another tableau system for linear temporal logic. Their main aim is to define a system which does not require loop-checking (cf. [22]). Their calculus uses formulas labelled with time

[6] We are currently working on a tableau system based on the mosaic idea in which the cut can be eliminated.

intervals. We also use labelled tableau, but our labels are time points reflecting the mosaic idea.[7] Finally we note that Zimmerman's tableau system [23,8] for minimal temporal logic might be extended to cover the linear case.

We will use labelled tableaux. In general, a tableau rule in our system has the form

$$\frac{l : A;\ \ m : B;\ \ r : C}{l : A';\ \ m : B';\ \ r : C' \mid l : A'';\ \ m : B'';\ \ r : C''}$$

where $A,\ldots,C''$ are finite sets of formulas and l, m and r stand for *l*eft, *m*iddle and *r*ight, respectively. We will also allow rules where some of the formula sets are empty, e.g., the numerator may be $l : A;\ \ m : \emptyset;\ \ r : C$ that we will abbreviate as $l : A;\ \ r : C$. A model $(N, <, v)$ satisfies the labelled triple $l : A_l;\ \ m : A_m;\ r : A_r$ if for every non-empty A_i $(i \in \{l, m, r\})$, there is $n_i \in N$ such that $n_i \Vdash A_i$ and $n_l < n_m$, $n_m < n_r$ and $n_l < n_r$. The intuition for a rule is that if the numerator $l : A;\ \ m : B;\ \ r : C$ is satisfiable in a model, then so is one of the denominators $l : A';\ \ m : B';\ \ r : C'$ or $l : A'';\ \ m : B'';\ \ r : C''$.

Now we recall the basic definitions for tableau systems, cf. [3]. A *tableau* for a set X of formulas is a finite tree with a root containing $m : X$. If a node carries $l : X;\ \ m : Y;\ \ r : Z$ and we apply the rule

$$\frac{l : A;\ \ m : B;\ \ r : C}{l : A';\ \ m : B';\ \ r : C' \mid l : A'';\ \ m : B'';\ \ r : C''}$$

then it has two successor nodes carrying $l : A';\ \ m : B';\ \ r : C'$ and $l : A'';\ m : B'';\ \ r : C''$. A branch is *finished* at a node if there is no applicable rule for this node or the node already occurred in the same branch. A branch is *closed* if its end node carries $m : \bot$. A tableau is closed if each of its branches is closed. A set X of formulas is *consistent* if there is no closed tableau with root $m : X$. Similarly, we will say that $l : X;\ \ m : Y;\ \ r : Z$ is consistent, if it cannot occur at the root of a closed tableau.

Let X be a given set of formulas and $Sf(X)$ be the smallest set containing the subformulas of the elements of X and their negations. Let $Y \subseteq Sf(X)$. Y is *maximal* in $Sf(X)$ if each element of $Sf(X)$ or its negation is in Y. We will need maximal sets in our tableau proofs, but note that we will work *inside* $Sf(\{\xi\})$, where ξ is the formula that we want to prove. Thus, although one may consider the need for maximal consistent sets an undesirable feature, our tableau system has the ±subformula property: every formula occurring in the tableau is a (perhaps negated) subformula of ξ.

So let us fix a formula ξ and define $\Xi = Sf(\{\xi\})$. We will assume that Ξ contains $\bot$; e.g., we might start with the equivalent formula $\xi \vee \bot$ instead of ξ. We have four types of rules. The first type of rules describes mosaics. These contain rules for propositional logic:

$$\frac{l : X;\ \ m : Y \cup \{\varphi, \neg\varphi\};\ \ r : Z}{m : \bot}$$

[7] It is not necessary to use labelled formulas in the tableau system below, but using labels makes the connection with the mosaic decidability proof more transparent.

$$\frac{l : X;\ \ m : Y \cup \{\varphi \wedge \psi\};\ \ r : Z}{l : X;\ \ m : Y \cup \{\varphi, \psi\};\ \ r : Z}$$

$$\frac{l : X;\ \ m : Y \cup \{\neg(\varphi \wedge \psi)\};\ \ r : Z}{l : X;\ \ m : Y \cup \{\neg\varphi\};\ \ r : Z \mid l : X;\ \ m : Y \cup \{\neg\psi\};\ \ r : Z}$$

$$\frac{l : X;\ \ m : Y \cup \{\neg\neg\varphi\};\ \ r : Z}{l : X;\ \ m : Y \cup \{\varphi\};\ \ r : Z}.$$

We have similar rules with $l : X$ and $r : Z$ instead of $m : Y$. By the coherency conditions 3 and 4 in Definition 1, if we have $l : X \cup \{\mathsf{G}\varphi\};\ \ r : Y$, then Y must not contain $\neg\mathsf{G}\varphi$ or $\neg\varphi$. The corresponding rules are

$$(C3)\quad \frac{l : X \cup \{\mathsf{G}\varphi\};\ \ r : Y \cup \{\neg\varphi\}}{m : \bot} \qquad (C4)\quad \frac{l : X \cup \{\mathsf{G}\varphi\};\ \ r : Y \cup \{\neg\mathsf{G}\varphi\}}{m : \bot}$$

We also have the "mirror" rules $(C5)$ and $(C6)$ corresponding to conditions 5 and 6 of Definition 1.

More interesting are the rules corresponding to the saturation conditions. We use the abbreviations $\mathsf{GG}^{-1}X = \{\mathsf{G}\varphi : \mathsf{G}\varphi \in X\}$ and $\mathsf{G}^{-1}X = \{\varphi : \mathsf{G}\varphi \in X\}$ and similarly for HH^{-1} and H^{-1}. Rules for saturation conditions 1 and 3 of Definition 2 are

$$(S1)\quad \frac{m : X \cup \{\mathsf{F}\varphi\}}{l : X \cup \{\mathsf{F}\varphi\};\ \ r : \mathsf{GG}^{-1}X \cup \mathsf{G}^{-1}X \cup \{\varphi\}}$$

and its mirror rule

$$(S3)\quad \frac{m : X \cup \{\mathsf{P}\varphi\}}{l : \mathsf{HH}^{-1}X \cup \mathsf{H}^{-1}X \cup \{\varphi\};\ \ r : X \cup \{\mathsf{P}\varphi\}}$$

and those corresponding to 2 and 4 of Definition 2 are $(S2)$

$$\frac{l : X \cup \{\mathsf{F}\varphi\};\ r : Y \cup \{\neg\mathsf{F}\varphi, \neg\varphi\}}{l : X \cup \{\mathsf{F}\varphi\};\ m : \mathsf{GG}^{-1}X \cup \mathsf{G}^{-1}X \cup \{\varphi\} \cup \mathsf{HH}^{-1}Y \cup \mathsf{H}^{-1}Y;\ r : Y \cup \{\neg\mathsf{F}\varphi, \neg\varphi\}}$$

and its mirror rule $(S4)$

$$\frac{l : Y \cup \{\neg\mathsf{P}\varphi, \neg\varphi\};\ r : X \cup \{\mathsf{P}\varphi\}}{l : Y \cup \{\neg\mathsf{P}\varphi, \neg\varphi\};\ m : \mathsf{GG}^{-1}Y \cup \mathsf{G}^{-1}Y \cup \{\varphi\} \cup \mathsf{HH}^{-1}X \cup \mathsf{H}^{-1}X;\ r : X \cup \{\mathsf{P}\varphi\}}.$$

The third type of rules are used to extend consistent sets of formulas to maximal ones in Ξ. The rules to construct maximal sets are: for every $\varphi \in \Xi$,

$$(Mm)\quad \frac{l : X;\ \ m : Y;\ \ r : Z}{l : X;\ \ m : Y \cup \{\varphi\};\ \ r : Z \mid l : X;\ \ m : Y \cup \{\neg\varphi\};\ \ r : Z}$$

and the similar rules (Ml) and (Mr) for $l : X$ and $r : Z$ instead of $m : Y$.

Finally, we have rules that allow us to discard some of the assumptions. The intuition is that if a triple of labelled formulas is satisfiable, then so is any subset of the triple.

$$\frac{l : X;\ \ m : Y;\ \ r : Z}{m : X} \qquad \frac{l : X;\ \ m : Y;\ \ r : Z}{m : Y} \qquad \frac{l : X;\ \ m : Y;\ \ r : Z}{m : Z}$$

$$\frac{l : X;\ \ m : Y;\ \ r : Z}{l : X;\ \ r : Y} \qquad \frac{l : X;\ \ m : Y;\ \ r : Z}{l : X;\ \ r : Z} \qquad \frac{l : X;\ \ m : Y;\ \ r : Z}{l : Y;\ \ r : Z}$$

For establishing the soundness one has to check that consistent sets are satisfiable. This amounts to proving that if the numerator of a rule is satisfiable, then so is one of the denominators — we leave this easy task to the reader. Completeness is guaranteed by the following lemma.

Lemma 5. *Every consistent formula is satisfiable.*

Proof. Let ξ be a consistent formula, i.e., assume that no tableau closes for $m : \{\xi\}$. As before, let $\Xi = Sf(\{\xi\})$.

Using the propositional rules and (Mm) (with empty left and right formula sets), let us extend $\{\xi\}$ to a consistent maximal set A in Ξ. If A does not contain any formula of the form $\mathsf{F}\varphi$ or $\mathsf{P}\varphi$, then the one element model with the obvious evaluation will do.

The interesting case is when A contains some defect. Instead of directly constructing a model, we will show that there is a saturated set of mosaics for A, where the labels of the mosaics are maximally consistent sets in Ξ.

We define M as the set of labelled pairs $(\langle 0, 1\rangle, X, Y)$ such that X and Y are maximal in Ξ and no tableau closes for $l : X;\ \ r : Y$.

Using the rules for the mosaic conditions, we get that $(\langle 0, 1\rangle, X, Y)$ is in fact a mosaic. Indeed, if one of the coherency conditions of Definition 1 were violated, then the rule corresponding to this condition would close the tableau.

By assumption, A contains some defect, say $\mathsf{F}\varphi \in A$. Then the application of $(S1)$ gives us a node carrying $l : A \cup \{\mathsf{F}\varphi\};\ \ r : \mathsf{G}\mathsf{G}^{-1}A \cup \mathsf{G}^{-1}A \cup \{\varphi\}$. Then using the rule (Mr) (with empty middle formula set) for creating maximal sets, we can extend $\mathsf{G}\mathsf{G}^{-1}A \cup \mathsf{G}^{-1}A \cup \{\varphi\}$ to a maximal B such that $l : A \cup \{\mathsf{F}\varphi\};\ \ r : B$ is consistent. Thus, ξ occurs in the label A of an element of M.

It remains to show that M is saturated. Let $(\langle 0, 1\rangle, X, Y)$ be any element of M.

If $\mathsf{F}\varphi \in Y$, then using $(S1)$ we have $l : Y \cup \{\mathsf{F}\varphi\};\ \ r : \mathsf{G}\mathsf{G}^{-1}Y \cup \mathsf{G}^{-1}Y \cup \{\varphi\}$. Again, we can extend the set on the right to a maximal Z such that $l : Y \cup \{\mathsf{F}\varphi\};\ \ r : Z$ is consistent. This provides the necessary mosaic curing the defect as in 1 of Definition 2.

If $\mathsf{F}\varphi \in X$ but $\neg\varphi, \neg\mathsf{F}\varphi \in Y$, then we apply $(S3)$. Thus we get that

$$l : X \cup \{\mathsf{F}\varphi\};\ m : \mathsf{G}\mathsf{G}^{-1}X \cup \mathsf{G}^{-1}X \cup \{\varphi\} \cup \mathsf{H}\mathsf{H}^{-1}Y \cup \mathsf{H}^{-1}Y;\ r : Y \cup \{\neg\mathsf{F}\varphi, \neg\varphi\}$$

is consistent. Next we extend the set in the middle to a maximal Z. Thus we have a consistent $l : X;\ \ m : Z;\ \ r : Y$. Then, by the last type of rules, we get that $l : X;\ \ r : Z$ and $l : Z;\ \ r : Y$ are consistent, i.e., $(\langle 0, 1\rangle, X, Z)$ and $(\langle 0, 1\rangle, Z, Y)$ are in M. Then we have the required two mosaics to insert into $(\langle 0, 1\rangle, X, Y)$, cf. 2 of Definition 2.

Checking conditions 2 and 4 of Definition 2 is the same, just use $(S2)$ and $(S4)$ instead of $(S1)$ and $(S3)$.

6 Automated Theorem-Proving

A draft version of a mosaic-based theorem-prover for the temporal logic with F and P over linear flows of time is available [18]. This Java applet checks the formula typed in by the user for satisfiability. The theorem-prover only considers mosaics with a non-singleton base and so first checks whether the formula has a one point model. If there is no one point model, the theorem-prover finds all non-singleton mosaics using the closure as in Lemma 3. Then it tries to find an SSM within the set of all mosaics; by Lemma 1, this procedure gives the correct answer. If it succeeds, then the SSM is displayed, otherwise the formula is unsatisfiable.

By systematically exploring all ways to complete a propositionally consistent subset of the labelling set, for a formula φ, it finds all MPCs, maximally propositionally consistent subsets of the labelling set. When the length of the formula is less than about 27, there usually is enough memory to store an array detailing the contents of each MPC. In any mosaic the labels at each end are MPCs. So we can easily (by checking coherency conditions, i.e., the conditions 1 to 6 of Definition 1) find and store an array of all mosaics indicating which MPC is at each end. To decide whether there is an SSM we consider a set M of mosaics which starts off as the set of all mosaics and is whittled down by the following iterative process. In each iteration check whether the formula φ appears in one end or the other of some mosaic in M: if not then we can stop and conclude that there is no SSM for φ. Then go through each mosaic in M and check that the conditions of Definition 2 are satisfied: if not then remove the mosaic from M. If we go through an iteration without removing any mosaics then we can stop and conclude that we have an SSM.

Note that the current implementation can, despite the worst case exponential time behaviour, quite quickly decide formulas to a length of about 27. There may be quicker ways of deciding formulas in this particular logic but the general mosaic-based approach of this theorem-prover can, as we will see, be used for many other temporal logics with only minor modifications.

7 Other Linear Flows of Time

In this section, we will describe how to modify the definition of SSM for special linear flows of time. We will concentrate on decidability and weak completeness, thus we can assume that the labelling set $Sf(\chi)$ is the set of subformulas of χ and their negations, where χ is the formula to be satisfied. It should be clear that the modifications below does not effect the decidability of the existence of an SSM.

Substructures of the whole numbers. We require that in item 2 of Definition 2 we have the additional condition that $\neg \mathsf{F}\varphi \in \ell'(m'_1)$ (and the corresponding condition in its mirror image 4).

It is easy to check that the proof of Lemma 1 goes through without any non-trivial modification (of course, we assume that in this case the model satisfying χ is based on a substructure of whole numbers).

We claim that the structure defined using the χ-SSM is in fact a substructure of the whole numbers. The key observation is that only once we can insert a φ-point to provide a witness for a future formula $\mathsf{F}\varphi$. Indeed, let us assume that we inserted a point k, say in step $n+1$, such that $\varphi, \neg\mathsf{F}\varphi \in \ell_{n+1}(k)$ between two points $i < j$ because we had $\mathsf{F}\varphi \in \ell_n(i)$ and $\neg\mathsf{F}\varphi, \neg\varphi \in \ell_n(j)$ (recall that according to the construction, if we can avoid inserting a point, then we create the witness at the end of the finite linear order). Towards a contradiction let us assume that in a later stage $m+1$ we have to insert another φ-point because of a a future defect $\langle l, \mathsf{F}\varphi\rangle$ in $\mathfrak{I}_m$. Where can this point l be compared to k in the linear order? First assume that $l < k$: if $\mathsf{F}\varphi \in \ell_m(l)$, then $\varphi \in \ell_m(k)$ is a good witness. On the other hand, if $l \geq k$, then $\neg\mathsf{F}\varphi \in \ell_m(l)$, since $\neg\mathsf{F}\varphi \in \ell_m(k)$ and $\mathfrak{I}_m$ is coherent. (Note that we might have to insert another φ-witness because of a past defect $\langle i', \mathsf{P}\varphi\rangle$, but the same argument shows that it can happen only once.)

Since we are using a finite label set $Sf(\chi)$, there are only finitely many $\mathsf{F}\varphi$ and $\mathsf{P}\varphi$ formulas in $Sf(\chi)$. Hence we are inserting only finitely many points into the linear order that we are constructing, and all the other points that we have to add, we glue to the beginning (for past defects) or to the end (for future defects) of the linear order we constructed so far. That is, for any two points i and j, we insert only finitely many points between i and j during the construction. Thus every non-startpoint in the linear order has an immediate predecessor and every non-endpoint has an immediate successor. As the construction terminates in at most ω steps, the linear order must be a substructure of the whole numbers.

Without endpoint(s). Add the formula $\mathsf{GF}\top \wedge \mathsf{F}\top$ (and $\mathsf{HP}\top \wedge \mathsf{P}\top$) to the label containing χ. Then, during the construction, we have to add points at the end of the linear orders infinitely often, since in each step we have a future defect $\mathsf{F}\top$ at the end of the linear order (or the past defect $\mathsf{P}\top$ at the start).

With endpoint(s). Add the formula $\mathsf{FG}\bot \vee \mathsf{G}\bot$ (and $\mathsf{PH}\bot \vee \mathsf{H}\bot$) to the label containing χ — this has the effect that we cannot add new points at the end of the structure once we created the witness for $\mathsf{FG}\bot$.

Dense. Require in the definition of SSM that, for every mosaic, there exist mosaics that can be inserted, like in Definition 2.2. Then, in the construction of the model from the SSM, in each step insert the provided points between all neighbouring points. Then, in the limit step, there will not be immediate successors and predecessors.

Natural numbers. Use the combination of the strategies for substructures of the whole numbers, linear flow with starting point and without (future) endpoint.

8 Mosaics for *Until* and *Since*

Many of the actual decidability and axiomatic completeness results above are well known. What is new is the method of establishing them.

However, we claim that the mosaic method is a powerful general method for establishing such results for many temporal logics. To justify this claim we need to move beyond Prior's language of F and P. Some important new results have already been gained by using the mosaic methods as demonstrated above on more expressive logics with the connectives *until*, U, and *since*, S, of [11].

The language, as seen in [4], for example, has two-place connectives U and S along with the classical $\neg$ and $\wedge$. The formula $\mathsf{U}(\alpha, \beta)$ is supposed to capture the idea of α being true at some time in the future and β holding at all points strictly in between now and that time. So the semantic clause for *until* over a linear flow of time $(T, <)$ is:

$$x \Vdash \mathsf{U}(\alpha, \beta) \text{ iff there is } y > x \text{ in } T \text{ such that } y \Vdash \alpha \text{ and for all } z \in T \text{ such that } x < z < y, \text{ we have } z \Vdash \beta.$$

The definition of *since* is the mirror image. Note that there are slightly different versions of these connectives which are sometimes used for discrete flows of time.

There has long been interest in the expressive temporal logics using U and S over general linear time or over specific dense flows such as the rationals and the real numbers. For example ([2] or [1]), these logics are useful for reasoning about concurrency and the behaviour of analogue devices. Despite this interest the problems of the complexity of decision procedures have remained open. (In contrast, in the case of discrete natural-numbers flows of time, it was shown in [20] that the decision problem is PSPACE-complete.)

The mosaic methods as described above are now being used to answer these questions and provide a basis for automated theorem-proving techniques in this important area. Recent work using mosaics establishes a PSPACE-complete complexity for the decision procedures for (1) the logic of U over general linear time [17] and (2) the logic of U and S over the reals [16].

The general idea of the mosaics and the decision procedures are slight modifications of the ideas for F and P. However, the details get quite complicated in specific ways when we try to fix tight complexity bounds.

In future work we will be trying to extend these results to logics such as the logic of U and S over general linear time. Here, for example, is our proposal for mosaics for this logic.

Given a formula φ, as the labelling set we use the set $Sf(\varphi)$ of all subformulas of φ and their negation. We will define a mosaic to be a triple (A, B, C) of sets of formulas. The intuition is that this corresponds to two points from a structure: A is the set of formulas (from $Sf(\varphi)$) true at the earlier point, C is the set true at the later point and B is the set of formulas which hold at all points strictly in between.

Definition 3. *Fix a formula φ. A* φ-mosaic *is a triple (A, B, C) of subsets of $Sf(\varphi)$ such that:*

0.1. A and C are maximally propositionally consistent, and
0.2. for all $\beta \in Sf(\varphi)$ with $\neg\neg\beta \in Sf(\varphi)$ we have $\neg\neg\beta \in B$ iff $\beta \in B$
and the following four coherency *conditions hold:*
C1. if $\neg \mathsf{U}(\alpha, \beta) \in A$ and $\beta \in B$ then we have both:
C1.1. $\neg\alpha \in C$ and either $\neg\beta \in C$ or $\neg \mathsf{U}(\alpha, \beta) \in C$; and
C1.2. $\neg\alpha \in B$ and $\neg \mathsf{U}(\alpha, \beta) \in B$;
C2. if $\mathsf{U}(\alpha, \beta) \in A$ and $\neg\alpha \in B$ then we have both:
C2.1 either $\alpha \in C$ or both $\beta \in C$ and $\mathsf{U}(\alpha, \beta) \in C$; and
C2.2. $\beta \in B$ and $\mathsf{U}(\alpha, \beta) \in B$;
C3-4. mirror images of C1-C2.

In order fit two mosaics together end-to-end to decompose another mosaic we need a notion of composition:

Definition 4. *We say that φ-mosaics (A', B', C') and (A'', B'', C'')* compose *iff $C' = A''$. In that case, their* composition *is $(A', B' \cap C' \cap B'', C'')$.*

It is straightforward to prove that this is a mosaic and that composition of mosaics is associative.

Now we have our analogue to Definition 2:

Definition 5. *A set M of mosaics is an SSM if it satisfies the following condition. For every $(A, B, C) \in M$,*

1. *for all $\mathsf{U}(\alpha, \beta) \in A$ we have*
   1.1. $\beta \in B$ and either ($\beta \in C$ and $\mathsf{U}(\alpha, \beta) \in C$) or $\alpha \in C$,
   1.2. or there is some $(A', B', C') \in M$ and some $(A'', B'', C'') \in M$, composing to (A, B, C) such that $\alpha \in C'$ and $\beta \in B'$;
2. *the mirror image of 1.; and*
3. *for each $\neg\beta \in Sf(\varphi)$ such that $\beta \notin B$ there is some $(A', B', C') \in M$ and some $(A'', B'', C'') \in M$, composing to (A, B, C) such that $\neg\beta \in C'$.*

Immediate future work for the mosaic-theorists will be establishing that a formula in the language of U and S is satisfiable over general linear time iff there is an SSM which contains a mosaic (A, B, C) with the formula contained in A or C. Further work should be able to show that this gives a PSPACE procedure for satisfiability/validity.

As we have mentioned, these logics are useful and it should be practical to modify the mosaic-based automated theorem-proving approach to handle them. Indeed, the modification required will be minor: we keep the overall procedure the same but we make a straightforward substitution of the coherency (cf. Definition 1) and saturation (cf. Definition 2) conditions. There is even a possibility that a general temporal logic theorem-prover generator could be implemented to accept user specification of particular coherency and saturation conditions.

References

1. H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the thirteenth ACM symposium on the principles of Programming Languages, St. Petersberg Beach, Florida.* ACM, January 1986.

2. J. P. Burgess and Y. Gurevich. The decision problem for linear temporal logic. *Notre Dame J. Formal Logic*, 26(2):115–128, 1985.
3. M. Fitting. *Proof methods for modal and intuitionistic logics*. Reidel, 1983.
4. D. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1*. Oxford University Press, 1994.
5. R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI. Lecture Notes*. The Chicago University Press, Chicago, 1987.
6. R. Goré. Cut-free sequent and tableau systems for propositional Diodorean modal logics. *Studia Logica*, 53:433–457, 1994.
7. R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.
8. A. Heuerding, M. Seyfried and H Zimmermann. Efficient loop-check for backward proof search in some non-classical logics. *Tableaux 96: Proceedings of Theorem Proving with Analytic Tableaux and Related Methods*, pages 210–225. Volume 1071 of *LNAI*, Springer, 1996.
9. R. Hirsch, I. Hodkinson, M. Marx, S. Mikulás, and M. Reynolds. Mosaics and step-by-step. Remarks on "A modal logic of relations". In E. Orlowska, editor, *Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa*, volume 24 of *Studies in Fuzziness and Soft Computing*, pages 158–167. Springer-Verlag, 1999.
10. R. Kashima. Cut-free sequent calculi for some tense logics. *Studia Logica*, 53:119–135, 1994.
11. H. Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.
12. S. Mikulás. Taming first-order logic. *Journal of the IGPL*, 6(2):305–316, 1998.
13. I. Németi. *Free Algebras and Decidability in Algebraic Logic*. PhD thesis, Hungarian Academy of Sciences, Budapest, 1986. In Hungarian.
14. I. Németi. Decidable versions of first order logic and cylindric-relativized set algebras. In L. Csirmaz, D. Gabbay, and M. de Rijke, editors, *Logic Colloquium '92*, pages 171–241. CSLI Publications, 1995.
15. H. Ono and A. Nakamura. On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica*, 39:325–333, 1980.
16. M. Reynolds. The complexity of the temporal logic over the reals, submitted. Version available at http://www.it.murdoch.edu.au/∼mark/research/online.
17. M. Reynolds. The complexity of the temporal logic with until over general linear time, submitted.
18. M. Reynolds. Online theorem-prover at http://www.it.murdoch.edu.au/∼mark/research/online/demos/tempmos/TempMosApplet.html
19. P. H. Schmitt and J. Goubault-Larrecq. A tableau system for linear-time temporal logic. In E. Brinksma, editor, *TACAS'97*, volume 1217 of *Lecture Notes in A.I.* Springer-Verlag, 1997.
20. A. Sistla and E. Clarke. Complexity of propositional linear temporal logics. *J. ACM*, 32:733–749, 1985.
21. Y. Venema and M. Marx. A modal logic of relations. In E. Orlowska, editor, *Logic at Work*. Springer-Verlag, 1999.
22. P. Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, (N.S.) 28 numbers 110–111, pages 119–136, 1985.
23. H. Zimmermann. *A Directed Tree Calculus for Minimal Tense Logic*. Master's Thesis, Institute for Applied Mathematics and Computer Science, University of Bern, Switzerland, 1995.

Sequent-Like Tableau Systems with the Analytic Superformula Property for the Modal Logics *KB*, *KDB*, *K5*, *KD5*

Linh Anh Nguyen

Institute of Informatics, Warsaw University, ul. Banacha 2, 02-097 Warsaw
nguyen@melkor.mimuw.edu.pl

Abstract. We give complete sequent-like tableau systems for the modal logics KB, KDB, $K5$, and $KD5$. Analytic cut rules are used to obtain the completeness. The systems have the analytic superformula property and can thus give a decision procedure.

1 Introduction

Tableau methods have been widely applied for modal logics: some of the best accounts of this are the works by Fitting [2] and Goré [3]. There are two kinds of tableau systems for modal logics: sequent-like, and labeled systems. In [8], Massacci successfully gives labeled tableau systems for all the basic normal modal logics obtainable from the logic K by the addition of any combination of the axioms T, D, 4, 5, and B in a modular way. There is a difficulty in developing sequent-like systems for symmetric modal logics (i.e. the ones containing the axiom B or/and 5) as in such logics "the future can affect the past", whereas in sequent-like systems "the past" cannot be changed. In [2], Fitting gives semi-analytic sequent-like tableau systems for the logics KB, KDB, B, and $S5$, but they do not have the analytic superformula property and thus cannot give a decision procedure. There are known sequent-like tableau systems with the analytic superformula property for the logics B, $KB4$, $K45$, $KD45$, and $S5$ (see [3] for the history), but such systems for the logics KB, KDB, $K5$, and $KD5$ are, as raised by Goré [3], open problems.

In this work, we present complete sequent-like tableau systems for the latter logics. These systems use analytic cuts and have the analytic superformula property. Our systems for the logics KB and KDB are based on the system $\mathcal{C}B$ of Rautenberg [10]. For the logics $K5$ and $KD5$, we use a special symbol to distinguish the "actual" world from the others. To obtain the analytic superformula property we use an extra connective as a "blocked" version of the modality $\Box$.

Our tableau formulation is based on the work by Goré [3]. We use a similar technique to prove completeness of the systems. To show completeness of $\mathcal{C}L$ we give an algorithm that, given a finite $\mathcal{C}L$-consistent formula set X, constructs a L-model graph that satisfies every one of its formulae at the corresponding world.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 341 351, 2000.

2 Preliminaries

2.1 Syntax and Semantics Definition for Modal Logics

A modal formula, hereafter simply called a *formula*, is defined by the following rules: any primitive proposition p is a formula, and if ϕ and ψ are formulae then so are $\neg\phi$, $\phi \wedge \psi$, and $\Box\phi$. We write $\phi \vee \psi$, $\phi \rightarrow \psi$, and $\Diamond\phi$ to denote shortened forms of $\neg(\neg\phi \wedge \neg\psi)$, $\neg(\phi \wedge \neg\psi)$, and $\neg\Box\neg\phi$, respectively.

We use small letters p, q to denote primitive propositions, Greek letters like ϕ, ψ to denote formulae, and block letters like X, Y, Z to denote formula sets. By $\mathcal{P}$ we denote the set of primitive propositions.

A Kripke *frame* is a triple $\langle W, \tau, R\rangle$, where W is a nonempty set of possible worlds, $\tau \in W$ is the actual world, and R is a binary relation on W called the accessibility relation. If $R(w, u)$ holds, then we say that the world u is accessible from the world w, or that u is reachable from w.

A Kripke *model* is a tuple $\langle W, \tau, R, h\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, $h : W \rightarrow P(\mathcal{P})$, and $h(w)$ is the set of primitive propositions which are "true" at the world w.

Given some Kripke model $M = \langle W, \tau, R, h\rangle$, and some $w \in W$, the satisfaction relation $M, w \vDash \phi$ is defined recursively as follows.

$$
\begin{array}{ll}
M, w \vDash p & \text{iff } p \in h(w);\\
M, w \vDash \neg\phi & \text{iff } M, w \nvDash \phi;\\
M, w \vDash \phi \wedge \psi & \text{iff } M, w \vDash \phi \text{ and } M, w \vDash \psi;\\
M, w \vDash \Box\phi & \text{iff for all } v \in W \text{ such that } R(w, v), M, v \vDash \phi.
\end{array}
$$

We say that M *satisfies* ϕ *at* w iff $M, w \vDash \phi$, and that M *satisfies* ϕ, or ϕ is *satisfied* in M, iff $M, \tau \vDash \phi$. If M satisfies ϕ then we also call M a *model of* ϕ.

2.2 Modal Logic Correspondences

The simplest normal modal logic, called K, is axiomatized by the standard axioms for the classical propositional logic, the *modus ponens* inference rule, the K*-axiom* $\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$, plus the *necessitation rule*

$$\frac{\vdash \phi}{\vdash \Box\phi}$$

It can be shown that a modal formula is provable in this axiomatization iff it is satisfied in every Kripke model (i.e. without any special R-properties) [7]. It is known that certain axiom schemata added to this axiomatization are mirrored by certain properties of the accessibility relation (see also [6,1]).

Different modal logics are distinguished by their respective additional axiom schemata. The modal logics KB, KDB, $K5$, $KD5$ together with their axiom schemata are listed in Tables 1 and 2. We refer to properties of the accessibility relation of a modal logic L as L*-frame restrictions*.

We call a model M a L*-model* if the accessibility relation of M satisfies all L-frame restrictions. We say that ϕ is L-satisfiable if there exists a L-model of ϕ. A formula ϕ is called *tautology* in a logic L if it is satisfied in every L-model.

Table 1. Axioms and corresponding first-order conditions on R

Axiom	Schemata	First-Order Formula
D	$\Box\Phi \rightarrow \Diamond\Phi$	$\forall x\ \exists y\ R(x,y)$
B	$\Phi \rightarrow \Box\Diamond\Phi$	$\forall x, y\ R(x,y) \rightarrow R(y,x)$
5	$\Diamond\Phi \rightarrow \Box\Diamond\Phi$	$\forall x, y, z\ R(x,y) \wedge R(x,z) \rightarrow R(y,z)$

Table 2. Modal logics and frame restriction

Logic	Axiom	Frame Restriction
KB	KB	symmetric
KDB	KDB	serial and symmetric
$K5$	K5	euclidean
$KD5$	KD5	serial and euclidean

2.3 Syntax, Soundness, and Completeness of Modal Tableau Systems

Our tableau formulation is adopted from the work by Goré [3], which in turn is related to the ones by Hintikka [4] and Rautenberg [10]. A number of terms and notations used in this work are borrowed from Goré [3].

A *tableau rule* δ consists of a numerator N above the line and a (finite) list of denominators D_1, D_2, ..., D_k (below the line) separated by vertical bars.

$$\frac{N}{D_1 \mid D_2 \mid \ \ldots\ \mid D_k}$$

The numerator is either a finite formula set or the special symbol $\bot$, and so is each denominator. As we shall see later, each rule is read downwards as "if the numerator is L-satisfiable, then so is one of the denominators". The numerator of each tableau rule contains one or more distinguished formulae called the *principal formulae*.

A *tableau system* (or *calculus*) $\mathcal{C}L$ is a finite set of tableau rules.

A $\mathcal{C}L$-tableau for X is a tree with root X whose nodes carry finite formula sets. A tableau rule with numerator N is applicable to a node carrying a set Y if Y is an instance of N. The steps for extending a tableau are:

- choose a leaf node n carrying Y where n is not an end node (defined below), and choose a rule δ which is applicable to n;
- if δ has k denominators then create k successors nodes for n, with successor i carrying an appropriate instance of denominator D_i;
- all with the proviso that if a successor s carries a set Z and Z has already appeared on the branch from the root to s then s is an *end node*.

Let Δ be a set of tableau rules. We say that Y is *obtainable from* X *by applications of rules from* Δ if there exists a tableau for X which uses only rules from Δ and has a node that carries Y.

A branch in a tableau is *closed* if it ends with $\bot$. A tableau is *closed* if every its branch is closed. A tableau is *open* if it is not closed. A finite formula set X is said to be $\mathcal{C}L$*-consistent* if every $\mathcal{C}L$-tableau for X is open. If there is a closed $\mathcal{C}L$-tableau for X then we say that X is $\mathcal{C}L$*-inconsistent*.

A tableau system $\mathcal{C}L$ is said to be *sound* if for any finite formula set X, if X is L-satisfiable then X is $\mathcal{C}L$-consistent. A tableau system $\mathcal{C}L$ is said to be *complete* if for any finite formula set X, if X is $\mathcal{C}L$-consistent then X is L-satisfiable.

Let δ be one of the rules of $\mathcal{C}L$. We say that δ is sound wrt. L if for any instance δ' of δ, if the numerator of δ' is L-satisfiable then so is one of the denominators of δ'. It is clear that if $\mathcal{C}L$ contains only rules sound wrt. L then $\mathcal{C}L$ is sound.

3 Tableau Systems for the Modal Logics *KB*, *KDB*, *K*5, and *KD*5

Tables 3 and 4 represent tableau rules and calculi for the modal logics KB, KDB, $K5$, and $KD5$. We write $X;Y$ for $X \bigcup Y$, and $X;\phi$ for $X \bigcup \{\phi\}$. The connective $\boxtimes$ has the same semantics as $\Box$, i.e. $M, w \vDash \boxtimes\phi$ iff $M, w \vDash \Box\phi$. Syntactically, formulae starting with $\boxtimes$ are blocked from being principal formulae. This "blocking" technique has previously been used in the work by Hudelmaier [5]. The symbol $*$ is a special formula with the following semantics: $M, w \vDash *$ iff there exists a world u such that $R(u, w)$ holds. By $\Box X$ we denote the set $\{\Box\phi \mid \phi \in X\}$. The sets $\boxtimes X$ and $\neg X$ are defined similarly.

Thinning is built into the rules of our systems, whereas in [3] Goré uses an explicit thinning rule. An explicit thinning rule is not desirable for our systems since the rule (5_*) is applicable only when the numerator contains $*$, and an explicit thinning rule would allow us to remove the $*$, leading to blocked proofs and wasted search.

Following Goré [3], we categorize each rule as either a *static rule* or as a *transitional rule*. The intuition behind this sorting is that in the static rules, the numerator and denominator represent the same world (in the same model), whereas in the transitional rules, the numerator and denominator represent different worlds (in the same model).

We write $Sf(\phi)$ to denote the set of all subformulae of ϕ and their negations. By $Sf(X)$ we denote the set $\bigcup_{\phi \in X} Sf(\phi)$. We say that X is *subformula-complete* if $Sf(X) = X$.

The rules $(B_\Box)$, $(B_\Diamond)$, $(sfc_\Box)$, $(sfc_\Diamond)$, $(sfc_\vee)$ are usually called *analytic cut* rules. They make $\mathcal{C}L$-saturations (defined in the next section) subformula-complete, which will be exploited to prove completeness of the given calculi.

A tableau system $\mathcal{C}L$ has the *analytic superformula* property iff to every finite set X we can assign a finite set $X^*_{\mathcal{C}L}$ such that $X^*_{\mathcal{C}L}$ contains all formulae that may appear in any tableau for X. Our systems have the analytic superformula property, with $X^*_{\mathcal{C}L} = Sf(\boxtimes Sf(X)) \bigcup \{\bot\}$.

Lemma 1. *The calculi $\mathcal{C}KB$, $\mathcal{C}KDB$, $\mathcal{C}K5$, and $\mathcal{C}KD5$ are sound.*

Table 3. Tableau rules for KB, KDB, $K5$, and $KD5$

$$(\bot)\ \frac{X;\phi;\neg\phi}{\bot} \qquad (\neg)\ \frac{X;\neg\neg\phi}{X;\phi} \qquad (\wedge)\ \frac{X;\phi\wedge\psi}{X;\phi;\psi}$$

$$(K)\ \frac{X;\Box Y;\boxtimes Z;\neg\Box\phi}{Y;Z;\neg\phi} \qquad (KD)\ \frac{X;\Box Y;\boxtimes Z}{Y;Z}$$

$$(B_\Box)\ \frac{X;\Box\phi}{X;\Box\phi;\phi \mid X;\Box\phi;\neg\phi;\boxtimes\neg\Box\phi} \qquad (B_\Diamond)\ \frac{X;\neg\Box\phi}{X;\neg\Box\phi;\phi \mid X;\neg\Box\phi;\neg\phi;\boxtimes\neg\Box\phi}$$

$$(5)\ \frac{X;\Box Y;\boxtimes Z;\neg\Box U;\neg\Box\phi}{Y;Z;\neg\Box U;\neg\Box\phi;\neg\phi;*} \qquad (KD_*)\ \frac{X;\Box Y;\boxtimes Z}{Y;Z;*}$$

$$(5_*)\ \frac{X;\Box\phi;*}{X;\Box\phi;\boxtimes\Box\phi;*} \qquad (5_\Box)\ \frac{X;\Box\phi}{X;\Box\phi;\boxtimes\Box\phi \mid X;\Box\phi;\boxtimes\neg\Box\phi}\ \text{where } * \notin X$$

$$(sfc_\Box)\ \frac{X;\Box\phi}{X;\Box\phi;\phi \mid X;\Box\phi;\neg\phi} \qquad (sfc_\Diamond)\ \frac{X;\neg\Box\phi}{X;\neg\Box\phi;\phi \mid X;\neg\Box\phi;\neg\phi}$$

$$(sfc_\vee)\ \frac{X;\neg(\phi\wedge\psi)}{X;\neg\phi;\neg\psi \mid X;\neg\phi;\psi \mid X;\phi;\neg\psi} \qquad (sfc)\ = \{(sfc_\Box),(sfc_\Diamond),(sfc_\vee)\}$$

Table 4. Tableau systems for KB, KDB, $K5$, and $KD5$

$\mathcal{C}L$	Static Rules	Transitional Rules
$\mathcal{C}KB$	$(\bot),(\neg),(\wedge),(sfc_\vee),(B_\Box),(B_\Diamond)$	(K)
$\mathcal{C}KDB$	$(\bot),(\neg),(\wedge),(sfc_\vee),(B_\Box),(B_\Diamond)$	$(K),(KD)$
$\mathcal{C}K5$	$(\bot),(\neg),(\wedge),(sfc),(5_*),(5_\Box)$	(5)
$\mathcal{C}KD5$	$(\bot),(\neg),(\wedge),(sfc),(5_*),(5_\Box)$	$(5),(KD_*)$

Proof. We show that $\mathcal{C}L$ contains only rules sound wrt. L, where L is KB, KDB, $K5$, or $KD5$. If the considered rule is static, then we show that if the numerator is satisfied at a world w, then so is one of the denominators. If the rule is transitional and its numerator is satisfied at w, then we show that the denominator is satisfied at some world reachable from w. Nontrivial cases are when the considered rule is one of $(B_\Box)$, $(B_\Diamond)$, (5), (5_*), $(5_\Box)$.

For $(B_\Box)$ and $(B_\Diamond)$, just note that $\neg\phi\rightarrow\Box\neg\Box\phi$ is a tautology in KB.

For (5), suppose that $M,w\vDash X;\Box Y;\boxtimes Z;\neg\Box U;\neg\Box\phi$, where $M=\langle W,\tau,R,h\rangle$ is a $K5$-model and $w\in W$. There exists u such that $R(w,u)$ holds and $M,u\vDash\neg\phi$. Since $\neg\Box\psi\rightarrow\Box\neg\Box\psi$ is a tautology in $K5$, we have $M,u\vDash Y;Z;\neg\Box U;\neg\Box\phi;\neg\phi;*$. Therefore (5) is sound wrt. $K5$.

For (5_*), suppose that $M,w\vDash X;\Box\phi;*$, where $M=\langle W,\tau,R,h\rangle$ is a $K5$-model and $w\in W$. We show that $M,w\vDash\boxtimes\Box\phi$. It suffices to show that for any $u,v\in W$ such that $R(w,u)$ and $R(u,v)$ hold, $R(w,v)$ also holds. Since $M,w\vDash *$, there exists w_0 such that $R(w_0,w)$ holds. From the frame restriction

$\forall x, y, z\ R(x,y) \wedge R(x,z) \rightarrow R(y,z)$, we derive that $R(w,w)$, $R(u,w)$, and $R(w,v)$ hold.

To show that $(5_\Box)$ is sound wrt. $K5$ and $KD5$, it suffices to show that $\neg(\Box\neg\Box\phi) \rightarrow \Box\Box\phi$ is a tautology in $K5$. This assertion holds because $\neg\Box\neg\Box\phi \equiv \Diamond\Box\phi$, and $\Diamond(\Box\phi) \rightarrow \Box\Diamond(\Box\phi)$, $\Diamond\Box\phi \rightarrow \Box\phi$, and $\Box(\Diamond\Box\phi) \rightarrow \Box(\Box\phi)$ are tautologies in $K5$.

4 Completeness of the Calculi

In this section, we use L to denote one of the logics KB, KDB, $K5$, $KD5$, and $\mathcal{CL}$ to denote the corresponding calculus. In order to prove completeness of the given calculi we first need some technical machinery.

4.1 Saturation

In the rules $(\neg)$, $(\wedge)$, $(sfc_\vee)$, the principal formula does not occur in the denominators. For δ being one of these rules, let δ' denote the rule obtained from δ by adding the principal formula to each of the denominators. Let $\mathcal{SCL}$ denote the set of static rules of $\mathcal{CL}$ with $(\neg)$, $(\wedge)$, $(sfc_\vee)$ replaced by $(\neg')$, $(\wedge')$, $(sfc'_\vee)$. Note that for any rule of $\mathcal{SCL}$ except $(\bot)$, the numerator is included in each of the denominators.

For X being a finite $\mathcal{CL}$-consistent formula set, a formula set Y is called a *$\mathcal{CL}$-saturation of X* if Y is a maximal $\mathcal{CL}$-consistent set obtainable from X by applications of the rules of $\mathcal{SCL}$.

A set X is *closed wrt. a tableau rule* if, whenever the rule is applicable to X, one of the corresponding instances of the denominators is equal to X.

As stated by the following lemma, $\mathcal{CL}$-saturations have the same nature as "downward saturated sets" defined in the works by Hintikka [4] and Goré [3].

Lemma 2. *Let X be a finite $\mathcal{CL}$-consistent formula set, and Y a $\mathcal{CL}$-saturation of X. Then $X \subseteq Y \subseteq X^*_{\mathcal{CL}}$, Y is closed wrt. the rules of $\mathcal{SCL}$, and Y is subformula-complete.*

Proof. It is easily seen that the first assertion holds.

If a rule of $\mathcal{SCL}$ is applicable to Y, then one of the corresponding instances of the denominators is $\mathcal{CL}$-consistent. Since Y is a $\mathcal{CL}$-saturation (of X), Y is closed wrt. the rules of $\mathcal{SCL}$.

It is straightforward to prove by induction on the construction of ϕ that if ϕ belongs to Y, then for any subformula ψ of ϕ not starting with $\neg$, either ψ or $\neg\psi$ belongs to Y. Therefore Y is subformula-complete.

Lemma 3. *There is an effective procedure that, given a finite $\mathcal{CL}$-consistent formula set X, constructs some $\mathcal{CL}$-saturation of X.*

Proof. We construct a $\mathcal{C}L$-saturation of X as follows: Let $Y = X$. While there is some rule δ of $\mathcal{SC}L$ applicable to Y with the property that one of the corresponding instances of the denominators, denoted by Z, is $\mathcal{C}L$-consistent and strictly contains Y, set $Y = Z$.

At each iteration, $Y \subset Z \subseteq X^*_{\mathcal{C}L}$. Hence the above process always terminates. It is clear that the resulting set Y is a $\mathcal{C}L$-saturation of X.

4.2 Proving Completeness Via Model Graphs

We prove completeness of our calculi via model graphs in a similar way as Rautenberg [10] and Goré [3] do for their systems.

By $\mathcal{F}$ we denote the set of formulae. A *model graph* is a tuple $\langle W, \tau, R, H\rangle$, where $\langle W, \tau, R\rangle$ is a Kripke frame, and $H : W \to P(\mathcal{F})$ ($H(w)$ is the set of formulae which should be "true" at the world w). We sometimes treat model graphs as models with h being H restricted to the set of primitive propositions. A model graph that satisfies all L-frame restrictions is called *L-model graph*.

Our definition of model graphs is not adequate with respect to Rautenberg's. A model graph $M = \langle W, \tau, R, H\rangle$ by his definition is accompanied by certain properties which guarantee that for any $\phi \in H(w)$, where $w \in W$, $M, w \vDash \phi$. Denote this condition by (*). We use the term "model graph" merely to denote a data structure, and leave the condition (*) as a criterion of *good* model graphs.

Given a finite $\mathcal{C}L$-consistent set X, as a L-model for X we construct a L-model graph $M = \langle W, \tau, R, H\rangle$ that satisfies the condition (*) and $X \subseteq H(\tau)$. We prove (*) for M by induction on the number of connectives occurring in ϕ. If for every $w \in W$, $H(w)$ is a $\mathcal{C}L$-saturation of some set, then (*) obviously holds (under inductive assumption) for the cases when ϕ is of the form p, $\neg p$, $\neg\neg\psi$, $\psi \wedge \zeta$, or $\neg(\psi \wedge \zeta)$. For the case when ϕ is of the form $\neg\Box\psi$, we show that there exists a world u reachable from w such that $\neg\psi \in H(u)$. For the case when ϕ is of the form $\Box\psi$ or $\boxtimes\psi$, in most of cases we show that for any world u reachable from w, $\psi \in H(u)$.

4.3 Completeness of $\mathcal{CKB}$ and $\mathcal{CKDB}$

In this subsection, let L denote one of the logics KB, KDB.

Algorithm 1
Input: A finite $\mathcal{C}L$-consistent set X of formulae not containing $\boxtimes$, $*$.
Output: A L-model graph $M = \langle W, \tau, R, H\rangle$ such that the corresponding model satisfies X.
(In this algorithm, we will mark the worlds of M either as *unresolved* or as *resolved*.)

1. Let $W = \{\tau\}$, $R_0 = \emptyset$, and $H(\tau)$ be a $\mathcal{C}L$-saturation of X. Mark τ as unresolved.
2. While there are unresolved worlds, take one, denoted by w, and do the following:

a) For every formula $\neg\Box\phi$ in $H(w)$:
   i. Let Y be the result of the application of the rule (K) to $H(w)$, i.e.
   $$Y = \{\neg\phi\} \bigcup \{\psi \mid \Box\psi \in H(w) \text{ or } \boxtimes\psi \in H(w)\},$$
   and let Z be a $\mathcal{C}L$-saturation of Y.
   ii. If there exists a world $v \in W$ such that $H(v) = Z$, then add the edge (w, v) to R_0. Otherwise, add a new world w_ϕ with content Z to W, mark it as unresolved, and add the edge (w, w_ϕ) to R_0.

b) If $L = KDB$ and there is no x such that $R(w, x)$ holds, then
   i. Let Y be the result of the application of the rule (KD) to $H(w)$, i.e.
   $$Y = \{\psi \mid \Box\psi \in H(w) \text{ or } \boxtimes\psi \in H(w)\},$$
   and let Z be a $\mathcal{C}L$-saturation of Y.
   ii. Do the same thing as the step 2(a)ii.

c) Mark w as resolved.

3. Let R be the symmetric closure of R_0.

This algorithm always terminates because H is one-to-one, and for any $w \in W$, $H(w) \subseteq Sf(\boxtimes Sf(X))$.

Lemma 4. *Let X be a finite $\mathcal{C}L$-consistent set of formulae not containing $\boxtimes$, $*$. Let $M = \langle W, \tau, R, H\rangle$ be the model graph constructed by the above algorithm for X. Then for any $w \in W$ and any $\phi \in H(w)$, $M, w \vDash \phi$.*

Proof. We prove this lemma by induction on the number of connectives occurring in ϕ. The only nontrivial case is when ϕ is of the form $\Box\psi$ or $\boxtimes\psi$. For this case it suffices to show that if $R_0(u, w)$ holds, and $\Box\psi \in H(w)$ or $\boxtimes\psi \in H(w)$, then $M, u \vDash \psi$. Assume that $R_0(u, w)$ holds.

Suppose that $\boxtimes\psi \in H(w)$. The formula $\boxtimes\psi$ can be introduced only by the rule $(B_\Box)$ or $(B_\Diamond)$, hence ψ is of the form $\neg\Box\zeta$, and we have $\neg\zeta \in H(w)$. By inductive assumption, $M, w \vDash \neg\zeta$. Hence $M, u \vDash \neg\Box\zeta$, and $M, u \vDash \psi$. (Note that it is not necessary that $\psi \in H(u)$.)

Now assume that $\Box\psi \in H(w)$. We show that $\psi \in H(u)$. Suppose oppositely that $\psi \notin H(u)$. Note that w is created from u, and for any formula ζ, if $\Box\zeta \in Sf(H(w))$ then $\Box\zeta \in Sf(H(u))$. Since $\Box\psi \in H(w)$, it follows that $\Box\psi \in Sf(H(u))$. Hence $\Box\psi \in H(u)$ or $\neg\Box\psi \in H(u)$, since $H(u)$ is subformula-complete. By the rules $(B_\Box)$ and $(B_\Diamond)$, from $\psi \notin H(u)$ we derive that $\boxtimes\neg\Box\psi \in H(u)$. It follows that $\neg\Box\psi \in H(w)$, which contradicts the assumption that $\Box\psi \in H(w)$. Therefore $\psi \in H(u)$, and by inductive assumption, we have $M, u \vDash \psi$. This completes our proof.

Corollary 1. *Let X be a finite $\mathcal{C}L$-consistent set of formulae not containing $\boxtimes$, $*$. Then X is L-satisfiable.*

Proof. Let $M = \langle W, \tau, R, H\rangle$ be a model graph constructed by the above algorithm for X. It is clear that R satisfies all L-frame restrictions. By the above lemma, we have $M, \tau \vDash H(\tau)$. Since $H(\tau)$ is a $\mathcal{C}L$-saturation of X, we also have $M, \tau \vDash X$. Therefore X is L-satisfiable.

The following theorem immediately follows from the above corollary and Lemma 1.

Theorem 2. *The calculi $\mathcal{C}KB$ and $\mathcal{C}KDB$ are sound and complete.*

4.4 Completeness of $\mathcal{C}K5$ and $\mathcal{C}KD5$

In this subsection, let L denote one of the logics $K5$, $KD5$.

Algorithm 3
Input: A finite $\mathcal{C}L$-consistent set X of formulae not containing $\boxtimes$, $*$.
Output: A L-model graph $M = \langle W, \tau, R, H\rangle$ such that the corresponding model satisfies X.

1. Let $W_1 = W_2 = \emptyset$, and $H(\tau)$ be a $\mathcal{C}L$-saturation of X.
2. For every formula $\neg\Box\phi \in H(\tau)$:
   - Let Y be the result of the application of the rule (5) to $H(\tau)$, i.e.
   $$Y = \{\neg\phi, *\} \bigcup \{\psi \mid \Box\psi \in H(\tau) \text{ or } \boxtimes\psi \in H(\tau)\} \bigcup \{\neg\Box\psi \mid \neg\Box\psi \in H(\tau)\}.$$
   - Add a new world w with $H(w)$ being a $\mathcal{C}L$-saturation of Y to W_1.
3. If $L = KD5$ and there is no $\neg\Box\phi \in H(\tau)$, then:
   - Let Y be the result of the application of the rule (KD_*) to $H(\tau)$, i.e.
   $$Y = \{*\} \bigcup \{\psi \mid \Box\psi \in H(\tau) \text{ or } \boxtimes\psi \in H(\tau)\}.$$
   - Add a new world w with $H(w)$ being a $\mathcal{C}L$-saturation of Y to W_1.
4. Let $R_0 = \{\tau\} \times W_1$.
5. For every $w \in W_1$, and for every formula $\neg\Box\phi \in H(w)$:
   - Let Y be the result of the application of the rule (5) to H(w), i.e.
   $$Y = \{\neg\phi, *\} \bigcup \{\psi \mid \Box\psi \in H(w) \text{ or } \boxtimes\psi \in H(w)\} \bigcup \{\neg\Box\psi \mid \neg\Box\psi \in H(w)\}.$$
   - Add a new world w_ϕ with $H(w_\phi)$ being a $\mathcal{C}L$-saturation of Y to W_2, and add the edge (w, w_ϕ) to R_0.
6. Let $W = \{\tau\} \bigcup W_1 \bigcup W_2$, and R be the euclidean closure of R_0.

It is clear that this algorithm always terminates. The following lemma has the same content as Lemma 4, but L now refers to $K5$, $KD5$.

Lemma 5. *Let X be a finite $\mathcal{C}L$-consistent set of formulae not containing $\boxtimes$, $*$. Let $M = \langle W, \tau, R, H\rangle$ be the model graph constructed by the above algorithm for X. Then for any $w \in W$ and any $\phi \in H(w)$, $M, w \vDash \phi$.*

Proof. We prove this lemma by induction on the number of connectives occurring in ϕ. It suffices to show that

1. For any $x \in W_2$ and any $\neg\Box\phi \in H(x)$, there exists $y \in W_2$ such that $\neg\phi \in H(y)$.
2. For any $x, y \in W_1 \bigcup W_2$ and any formula ϕ, if $\Box\phi$ or $\boxtimes\phi$ belongs to $H(x)$, then $\phi \in H(y)$.

Proof of 1:

Suppose that $x \in W_2$ and $\neg\Box\phi \in H(x)$. There exists $u \in W_1$ such that $R_0(u, x)$ holds. We have $* \in H(u)$. Since x is created from u and $\neg\Box\phi \in H(x)$, it follows that $\Box\phi \in Sf(H(u))$. Hence either $\Box\phi \in H(u)$ or $\neg\Box\phi \in H(u)$, since $H(u)$ is subformula-complete. If $\Box\phi \in H(u)$, then, by the rule (5_*), $\boxtimes\Box\phi \in H(u)$, and hence $\Box\phi \in H(x)$, which contradicts the fact that $\neg\Box\phi \in H(x)$. Therefore $\neg\Box\phi \in H(u)$, and there exists $y \in W_2$ such that $\neg\phi \in H(y)$.

Proof of 2:

Suppose that $x \in W_1$ and $\Box\phi \in H(x)$. Since x is created from τ and $\Box\phi \in H(x)$, we have $\Box\phi \in Sf(H(\tau))$. Hence either $\Box\phi \in H(\tau)$ or $\neg\Box\phi \in H(\tau)$. Since $\Box\phi \in H(x)$, $\neg\Box\phi$ and $\boxtimes\neg\Box\phi$ cannot belong to $H(\tau)$, otherwise, by the rule (5), we would have $\neg\Box\phi \in H(x)$ (a contradiction). Hence $\Box\phi \in H(\tau)$, and by the rule $(5_\Box)$, $\boxtimes\Box\phi \in H(\tau)$. It follows that for every $y \in W_1 \bigcup W_2$ we have $\phi \in H(y)$.

Now suppose that $x \in W_2$ and $\Box\phi \in H(x)$. Let $u \in W_1$ be the world such that $R_0(u, x)$ holds. Since x is created from u and $\Box\phi \in H(x)$, it follows that $\Box\phi \in Sf(H(u))$. Hence either $\Box\phi \in H(u)$ or $\neg\Box\phi \in H(u)$. If $\neg\Box\phi \in H(u)$, then, by the rule (5), $\neg\Box\phi \in H(x)$, which contradicts the assumption that $\Box\phi \in H(x)$. Hence $\Box\phi \in H(u)$. Reasoning similarly as for the above case, we derive that for every $y \in W_1 \bigcup W_2$ it holds that $\phi \in H(y)$.

For the last case, assume that $x \in W_1 \bigcup W_2$ and $\boxtimes\phi \in H(x)$. Since $* \in H(x)$, the formula $\boxtimes\phi$ in $H(x)$ must be introduced by the rule (5_*). Hence ϕ is of the form $\Box\psi$ and $\Box\psi \in H(x)$. Reasoning similarly as for the above cases, we derive that $\boxtimes\Box\psi \in H(\tau)$. It is clear that for every $y \in W_1$, $\Box\psi \in H(y)$. Suppose that $y \in W_2$. There exists $z \in W_1$ such that $R_0(z, y)$ holds. Since $\{\Box\psi, *\} \subseteq H(z)$, by the rule (5_*), it follows that $\boxtimes\Box\psi \in H(z)$, and hence $\Box\psi \in H(y)$. Therefore, for every $y \in W_1 \bigcup W_2$, $\phi \in H(y)$.

Corollary 2. *Let X be a finite $\mathcal{C}L$-consistent set of formulae not containing $\boxtimes$, $*$. Then X is L-satisfiable.*

The proof of this corollary is similar to the proof of Corollary 1. The following theorem immediately follows from this corollary and Lemma 1.

Theorem 4. *The calculi $\mathcal{C}K5$ and $\mathcal{C}KD5$ are sound and complete.*

Our proofs give refined L-models for these logics, as done in Goré's article [3]. That is, for any L-satisfiable formula ϕ, there exists a finite L-model of ϕ with a frame $\langle W, \tau, R\rangle$ such that:

- if $L = KD5$, then, for $U = W - \{\tau\}$, there exists $V \subseteq U$ such that $R = \{\tau\} \times V \bigcup U \times U$;
- if $L = K5$, then $W = \{\tau\}$ and $R = \emptyset$, or the frame is a $KD5$-frame.

5 Conclusions

We have given complete sequent-like tableau systems for the modal logics KB, KDB, $K5$, and $KD5$. Our systems have the analytic superformula property

and can thus give a decision procedure. Our presentation fulfills the picture of sequent-like tableau systems for the basic normal modal logics (i.e. the ones obtainable from the logic K by the addition of any combination of the axioms T, D, 4, 5, and B).

As suggested by one of the reviewers, as a further work we will study whether our tableau systems can be used to obtain interpolation theorems for the considered logics as done by Rautenberg.

Our systems are not efficient since they use cuts. In our recent work [9], efficient clausal tableau systems for the modal logic KB, KDB, B, among others, are presented. The systems are sequent-like, cut-free, and give a decision procedure that runs in $O(n^2)$-space. They require, however, inputs in clausal form.

Acknowledgements

The author would like to thank the anonymous reviewers for helpful comments.

References

1. B.F. Chellas. *Modal Logic: An Introduction*. Cambridge Univ. Press, 1980.
2. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. volume 169 of Synthese Library. D. Reidel, Dordrecht, Holland, 1983.
3. R. Goré. Tableau methods for modal and temporal logics. In D'Agostino, Gabbay, Hähnle, and Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.
4. K.J.J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:3–55, 1955.
5. J. Hudelmaier. Improved decision procedures for the modal logics K, T, S4. In *H. Kleine Büning, editor, Proceedings of CSL'95, LNCS 1092*, pages 320–334. Springer, 1996.
6. G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Methuen, 1968.
7. S.A. Kripke. Semantical analysis of modal logic, i. normal modal propositional calculus. *Z. Math. Logic Grundlag*, 9:67–96, 1963.
8. F. Massacci. Strongly analytic tableaux for normal modal logics. In *A. Bundy, editor, Proceedings of CADE-12, LNAI 814*, pages 723–737. Springer, 1994.
9. L.A. Nguyen. Clausal tableau systems and space bounds for the modal logics K, KD, T, KB, KDB, and B. *Technical Report TR 00-01(261), Warsaw University*, 2000.
10. W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12:403–423, 1983.

A Tableau Calculus for Equilibrium Entailment

David Pearce[1], Inmaculada P. de Guzmán[2], and Agustín Valverde[2]

[1] DFKI, Saarbrücken, Germany
[2] Dept. of Applied Mathematics
University of Málaga, Spain

Abstract. We apply tableau methods to the problem of computing entailment in the nonmonotonic system of *equilibrium* logic, a generalisation of the inference relation associated with the stable model and answer set semantics for logic programs. We describe tableau calculi for the nonclassical logics underlying equilibrium entailment, namely here-and-there with strong negation and its strengthening classical logic with strong negation. A further tableau calculus is then presented for computing equilibrium entailment. This makes use of a new method for reducing the complexity of the tableau expansion rules, which we call *signing-up*.

1 Introduction

Equilibrium logic is a formal system of nonmonotonic reasoning proposed and discussed by the first-named author in [28,29]. It is currently defined for propositional logic and can therefore be applied also to grounded (quantifier-free) theories in a first-order predicate language. One of the interesting features of equilibrium logic is that it generalises the stable model and answer set semantics for logic programs, as developed in [6,7,8].[1] In fact the equilibrium models of a theory coincide with its stable models or answer sets if the theory in question has the syntactic form of a logic program (for which the latter models are defined). It therefore offers a means to extend the reasoning mechanism associated with stable model semantics beyond the syntactic limitations of logic programs. The basic notions and properties of equilibrium logic are discussed in [28,29]; the present paper is devoted to developing a tableau calculus for the generation of equilibrium models and for checking entailment in equilibrium logic. Tableau methods seem to be the most promising here. In particular, since arbitrary formulas cannot in general be reduced to the syntactic form of logic programs, the algorithms developed for computing stable models cannot be applied in the general case.

Equilibrium logic is based on a 5-valued logic called *here-and-there with strong negation* and denoted here by $\boldsymbol{N}_5$. This logic can be viewed both as an axiomatic extension of Nelson's constructive logic with strong negation, $\boldsymbol{N}$ [21], and as a conservative extension of the Smetanich logic of here-and-there [32], obtained by adding the strong negation operator together with the well-known

[1] To our knowledge it is the first such generalisation that is defined for an arbitrary propositional language.

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 352–367, 2000.

axioms of Vorob'ev, [35,36]. Although tableau systems for strong negation logics can be found in the literature, [31], there has been relatively little work on the proof theory of logics extending $\boldsymbol{N}$ (for an algebraic treatment of $\boldsymbol{N}_5$ and related logics, see [16]). Similarly, although tableau calculi have been developed in the past for systems of nonmonotonic logic (eg. [26,22]), these have mainly been based on classical logic.

We are going to introduce three tableau systems: one to prove validity in $\boldsymbol{N}_5$, one to generate *total* models and lastly one to check the *equilibrium property* of a total model. All these systems use the technique introduce by R. Hähnle [13] for many-valued logics and take truth-value sets as *signs*, that is, as labels in the tableau nodes. However, in developing the tableau calculus for checking the equilibrium property, one novelty in our approach is introduced, the method of *signing-up*. The expansion rules in a many-valued tableau system can be regarded as a propagation of signs from the formula to its innermost subformulas, ie as a *signing-down* process. As a counterpart, the signing-up process evaluates partially the formula from its innermost subformulas; the objective is to reduce the complexity of the expansion rules.

Since equilibrium entailment is a conservative extension of the inference relation associated with stable model semantics, the techniques and results of this paper apply *a fortiori* to stable models for logic programs. However, our main aim is not that of providing a more efficient implementation of stable model semantics. The bases for such implementations already exist in the literature, eg. for normal logic programs in [24] and for disjunctive programs in [17]. These provide special purpose algorithms that are tailored to the specific syntax of logic programs and are therefore likely to be more efficient in this restricted setting. By contrast, the more general theorem proving techniques discussed here apply to the case of full propositional logic and are therefore likely to be of interest to those seeking to extend stable model reasoning beyond the language of logic programs, as for instance the approach of [18] which considers programs with nested expressions. We hope the methods of this paper may also be of some interest for the field of automated deduction for nonclassical logics; in particular, by illustrating the use of tableau methods in a particular nonmonotonic extension of a many-valued logic.

The remainder of the paper is laid out as follows. In order to describe equilibrium entailment we first consider the logic on which it is based, here-and-there with strong negation, $\boldsymbol{N}_5$. We present $\boldsymbol{N}_5$ first in Section 2 as the least strong negation extension of the logic of linear Kripke frames with two elements ('here-and-there'), secondly in Section 3 as a many-valued logic taking truth-values in the set $\mathbf{5} = \{-2, -1, 0, 1, 2\}$. In Section 4, equilibrium models are characterised by a simple minimality condition on the Kripke models for $\boldsymbol{N}_5$, and we then translate this condition into an equivalent one in terms of truth-assignments. This prepares the way for treating equilibrium entailment using tableau methods in 4.4. We also describe here a modification, called *signing-up*, which reduces the complexity of the tableau rules. The paper concludes with a discussion of related approaches and future work.

2 Strong Negation Logics

Although we shall later work in the framework of many-valued logics, it will help to motivate the logics concerned if we look briefly at the original characterisation of equilibrium inference and its underlying logic, $\boldsymbol{N}_5$. We assume the reader has some familiarity with Heyting's intuitionistic logic, which we denote by $\boldsymbol{H}$. Formulas of $\boldsymbol{H}$ are built-up in the usual way using the logical constants: $\wedge$, $\vee$, $\rightarrow$, $\neg$, standing respectively for conjunction, disjunction, implication and negation.

There are several types of semantics for intuitionistic logic. We mention here only the method of Kripke models, also known as *possible worlds semantics*. Formally, one starts with a so-called Kripke *frame* $\mathcal{F}$, where $\mathcal{F} = \langle W, \leq \rangle$ W is a set and $\leq$ is a partial-ordering on W. The elements of W are sometimes called possible worlds or stages. At each world or stage $w \in W$ some primitive propositions (atoms) are verified as *true*, and, once verified at some stage w, an atom α remains true at every 'later' stage, ie. at all w' such that $w \leq w'$. A Kripke *model* $\mathcal{M}$ can therefore be represented as a frame $\mathcal{F}$ together with an assignment i of sets of atoms to each element of W, such that if $w \leq w'$ then $i(w) \subseteq i(w')$. An assignment is then extended inductively to all formulas via the following rules:

$\varphi \wedge \psi \in i(w)$ iff $\varphi \in i(w)$ and $\psi \in i(w)$
$\varphi \vee \psi \in i(w)$ iff $\varphi \in i(w)$ or $\psi \in i(w)$
$\varphi \rightarrow \psi \in i(w)$ iff for all w' such that $w \leq w'$, $\varphi \in i(w')$ implies $\psi \in i(w')$
$\neg\varphi \in i(w)$ iff for all w' such that $w \leq w'$, $\varphi \notin i(w')$

An alternative approach to constructive reasoning was developed by Nelson [21] in 1949. Nelson's logic $\boldsymbol{N}$ is known as *constructive logic with strong negation*. It adds to Heyting's logic the insight that primitive propositions may not only be constructively *verified* but also constructively *falsified*. The language of intuitionistic logic is accordingly extended by adding a new, strong negation symbol, '$\sim$', with the interpretation that $\sim\varphi$ is true if φ is constructively false.

Terms and formulas are built-up using the logical constants of $\boldsymbol{H}$: $\wedge, \vee, \neg, \rightarrow$ and the additional negation '$\sim$'. Intuitionistic negation is actually definable in N by

$$\neg\varphi := \varphi \rightarrow \sim\varphi.$$

A semantics for $\boldsymbol{N}$ can be obtained through a simple modification of the Kripke-semantics for $\boldsymbol{H}$. As before a model comprises a Kripke-frame

$$\mathcal{F} = \langle W, \leq \rangle,$$

together with an assignment i. However i now assigns to each element of W a set of *literals*,[2] such that as before if $w \leq w'$ then $i(w) \subseteq i(w')$. An assignment is then extended inductively to all formulas via the previous rules for conjunction,

[2] We use the term 'literal' to denote an atom, or atom prefixed by strong negation.

disjunction, implication and (weak) negation together with the following rules governing strongly negated formulas:

$$\sim(\varphi \wedge \psi) \in i(w) \text{ iff } \sim\varphi \in i(w) \text{ or } \sim\psi \in i(w)$$
$$\sim(\varphi \vee \psi) \in i(w) \text{ iff } \sim\varphi \in i(w) \text{ and } \sim\psi \in i(w)$$
$$\sim(\varphi \to \psi) \in i(w) \text{ iff } \varphi \in i(w) \text{ and } \sim\psi \in i(w)$$
$$\sim\neg\varphi \in i(w) \text{ iff } \sim\sim\varphi \in i(w) \text{ iff } \varphi \in i(w)$$

A formula φ is true in a Kripke model $\mathcal{M} = \langle W, \leq, i\rangle$ at a world $w \in W$, in symbols $\mathcal{M}, w \models \varphi$, iff $\varphi \in i(w)$. φ is true in a Kripke model $\mathcal{M}$, in symbols $\mathcal{M} \models \varphi$, if it is true at all worlds in $\mathcal{M}$. A formula φ is said to be *valid*, in symbols, $\models_N \varphi$, if it is true in all Kripke models. Logical consequence for $\boldsymbol{N}$ is understood as follows: φ is said to be an $\boldsymbol{N}$-consequence of a set Π of formulas, written $\Pi \models_N \varphi$, iff for all models $\mathcal{M}$ and any world $w \in \mathcal{M}$, $\mathcal{M}, w \models \Pi$ implies $\mathcal{M}, w \models \varphi$. Equivalently this can be expressed by saying that φ is true in all models of Π. The consequence relation for logics strengthening $\boldsymbol{N}$ is defined in a similar way. The logic $\boldsymbol{N}$ can also be presented axiomatically. The above set of valid formulas can be captured via the axioms and rules of $\boldsymbol{H}$ (see eg. [5]) together with the following axiom schemata involving strong negation (where '$\alpha \leftrightarrow \beta$' abbreviates $(\alpha \to \beta) \wedge (\beta \to \alpha)$):

N1. $\sim(\alpha \to \beta) \iff \alpha \wedge \sim\beta$
N2. $\sim(\alpha \wedge \beta) \iff \sim\alpha \vee \sim\beta$
N3. $\sim\alpha \vee \beta) \iff \sim\alpha \wedge \sim\beta$
N4. $\alpha \iff \sim\sim\alpha$
N5. $\sim\neg\alpha \iff \alpha$
N6. (for atomic α) $\sim\alpha \to \neg\alpha$

These axioms are taken from the calculus of Vorob'ev [35,36], where $\boldsymbol{N}$ is treated as an extension of intuitionistic logic. $\boldsymbol{N}$ is in fact a conservative extension of $\boldsymbol{H}$ in the sense that any formula without strong negation is a theorem of $\boldsymbol{N}$ if and only if it is a theorem of $\boldsymbol{H}$. Notice that Nelson's negation '$\sim$' is termed 'strong', since in $\boldsymbol{N}$, $\sim\varphi \to \neg\varphi$ is a theorem, for all φ (not only atomic φ). (See eg. [10, 5]). The derivability relation for $\boldsymbol{N}$ is denoted by $\vdash_N$. The Kripke semantics is *sound* and *complete* for $\boldsymbol{N}$ in the sense that for all Π and φ

$$\Pi \vdash_N \varphi \text{ iff } \Pi \models_N \varphi.$$

2.1 Intermediate Logics

We now consider *intermediate* logics, obtained by adding additional axioms to $\boldsymbol{H}$; they are complete wrt. a generalised notion of Kripke frame. An intermediate logic is called *proper* if it is strictly contained in classical logic. In the lattice of intermediate propositional logics (extensively investigated in the literature) classical logic has a unique lower cover which is the supremum of all proper intermediate logics.[3] This greatest proper intermediate logic we denote by $\boldsymbol{J}$. It is

[3] An excellent general source of information on intermediate propositional logics is [4].

often referred to as the logic of 'here-and-there', since it is characterised by linear Kripke frames having precisely two elements or worlds: 'here' and 'there'. $\boldsymbol{J}$ is also characterised by the three element Heyting algebra, and is known by a variety of other names, including the Smetanich logic, and the logic of 'present and future'. Truth tables for $\boldsymbol{J}$ were already given by Heyting [14], and the logic was further used by Gödel in a paper of 1932, [9] (hence it is sometimes known as Gödel's 3-valued logic). However, it was apparently first axiomatised by Łukasiewicz [19]. He characterised $\boldsymbol{J}$ by adding to $\mathbf{H}$ the axiom schema

$$(\neg\alpha \to \beta) \to (((\beta \to \alpha) \to \beta) \to \beta).$$

He also showed that disjunction is definable in $\boldsymbol{J}$.[4]

Strong negation together with the Vorob'ev axioms N1-N6 can be added to any intermediate logic $\boldsymbol{L}$, to form a (least) strong negation extension of $\boldsymbol{L}$ (this is always a conservative extension). Moreover if an intermediate logic $\boldsymbol{L}$ is complete for a given class of Kripke frames, its least strong negation extension is complete for the same class of frames under the generalised notion of assignment. Likewise, many properties of an intermediate logic are inherited by its least strong negation extension, see [16]. Accordingly, we define the logic of *here-and-there with strong negation* to be the smallest logic that extends $\boldsymbol{J}$ by adding the operator $\sim$ satisfying axioms N1-N6. This logic is tabular and can be characterised using five truth-values; we denote it by $\boldsymbol{N}_5$. By the above remarks it follows that $\boldsymbol{N}_5$ is complete for the class of here-and-there frames. Since these contain only two worlds, say h and t with $h \leq t$, it is convenient to represent an $\boldsymbol{N}_5$-model as an ordered pair $\langle H, T\rangle$ of sets of literals, where $H = i(h)$ and $T = i(t)$ under a suitable assignment i. By $h \leq t$, it follows that $H \subseteq T$. Again, by extending i inductively we know what it means for an arbitrary formula φ to be true in a model $\langle H, T\rangle$. Logical consequence for $\boldsymbol{N}_5$ is defined as for $\boldsymbol{N}$, where $\boldsymbol{N}$-models are replaced by $\boldsymbol{N}_5$-models. If Π is a set of $\boldsymbol{N}_5$ formulas and φ is an $\boldsymbol{N}_5$ formula, we write simply $\Pi \models \varphi$ to denote that φ is an $\boldsymbol{N}_5$ consequence of Π.

3 Many-Valued Semantics for $\boldsymbol{N_5}$

The Kripke semantics for $\boldsymbol{N}_5$ logic can be easily characterised using a many-valued approach, specifically with a five-valued logic.[5] In this section we define this interpretation and then describe a five-valued tableau system to check inference.

[4] Smetanich studied the logic $\boldsymbol{J}$ in [32]. Important results about the logic can also be found in [20].

[5] In the case of the (3-valued) logic of here-and-there, if a formula φ takes the 'third' (intermediate) truth value, this corresponds to the idea that 'φ cannot be false, but it is not demonstrably true'. Adding strong negation increases (by two) the number of truth-values because one now distinguishes between φ being strongly false, ie. $\sim\varphi$ is true, and the weaker notion that $\sim\varphi$ is not false, but not demonstrably true. This is illustrated in the table in section 3.0.

We are going to work with the following set of truth values:

$$\mathbf{5} = \{-2, -1, 0, 1, 2\}$$

The subset $\{-2, 0, 2\}$ will be denoted by $\mathbf{3}$. Using *possible truth values functions* we are going to define a family of logics denoted by $\boldsymbol{N}_\omega$. Every logic is defined over the same language, $\mathcal{N}$, the propositional language over a set of propositional variables $\mathcal{V}$ and connectives $\wedge$, $\vee$, $\rightarrow$, $\neg$ and $\sim$. The semantic is defined using the same matrix, the Nelson algebra

$$\mathfrak{N} = (\{-2, -1, 0, 1, 2\}, \min, \max, \rightarrow, \neg, \sim)$$

where $\sim x = -x$,

$$x \rightarrow y = \begin{cases} 2 & \text{if either } x \leq 0 \text{ or } x \leq y \\ y & \text{otherwise} \end{cases} \qquad \text{and} \qquad \neg x = \begin{cases} 2 & \text{if } x \leq 0 \\ -x & \text{otherwise} \end{cases}$$

The truth tables for the connectives $\rightarrow$, $\neg$ and $\sim$ are as follows:

$\rightarrow$	-2	-1	0	1	2
-2	2	2	2	2	2
-1	2	2	2	2	2
0	2	2	2	2	2
1	-2	-1	0	2	2
2	-2	-1	0	1	2

$\neg$	
-2	2
-1	2
0	2
1	-1
2	-2

$\sim$	
-2	2
-1	1
0	0
1	-1
2	-2

Definition 1. *Let* $\omega\colon \mathcal{V} \rightarrow 2^{\mathbf{5}} \setminus \varnothing$ *a mapping, called a* possible truth values function*. The* Nelson logic restricted by ω*, denoted by* $\boldsymbol{N}_\omega$*, is a logic defined over the language* $\mathcal{N}$*, with the Nelson algebra* $\mathfrak{N}$ *as matrix; the assignments in* $\boldsymbol{N}_\omega$*, called* ω-assignments*, are the homomorphisms of algebras* $\sigma\colon \mathcal{N} \rightarrow \mathfrak{N}$ *defined as the unique extensions of mappings* $\varsigma\colon \mathcal{V} \rightarrow \mathbf{5}$ *verifying* $\varsigma(p) \in \omega(p)$*.*

Validity in the logics $\boldsymbol{N}_\omega$ is defined in the standard way. If Π is a set of formulas in $\mathcal{N}$ and ψ is a formula, then $\Pi \models_\omega \psi$ if the following property holds: for every ω-assignment σ, if $\sigma(\varphi) = 2$ for all $\varphi \in \Pi$, then $\sigma(\psi) = 2$.

If the function ω is just $\omega(p) = \mathbf{5}$ for all p, the corresponding logic will be denoted by $\boldsymbol{N}_5$ and is just the standard logic of here-and-there with strong negation, introduced in the previous section.

If the function ω is in fact $\omega(p) = \mathbf{3}$ for all p, the corresponding logic will be denoted by $\boldsymbol{N}_3$. This logic is actually a three-valued logic because, in this case, every function is evaluated in $\mathbf{3}$ (this property doesn't hold for all functions ω). Since it is also obtained by adding the Vorob'ev axioms to classical logic, it is known as *classical logic with strong negation*, and in [10] it is shown via an embedding to be equivalent to the 3-valued logic of Łukasiewicz, though the $\rightarrow$ connective in $\boldsymbol{N}_3$ is not the Łukasiewicz implication.

Example 1. Let us consider the formula $\varphi = (p \rightarrow \sim q) \rightarrow (\neg p \vee \neg q)$ in the language $\mathcal{N}$ and the possible truth values function given by:

$$\omega(p) = \{-2, 0, 2\}, \qquad \omega(q) = \{0, 1, 2\}$$

The assignment $\sigma(p) = 1, \sigma(q) = 0$ is an assignment for this formula in $\boldsymbol{N}_5$ but it is neither an assignment in $\boldsymbol{N}_\omega$ nor in $\boldsymbol{N}_3$. The assignment $\tau(p) = -2, \tau(q) = 2$ is an assignment in $\boldsymbol{N}_5$, $\boldsymbol{N}_\omega$ and $\boldsymbol{N}_3$; however, the value of the assignments over φ doesn't depend on the logic, $\sigma(\varphi) = \tau(\varphi) = 2$. Actually, the formula φ is valid in $\boldsymbol{N}_5$ and thus in every logic $\boldsymbol{N}_\omega$.

We will in fact work only in the logic $\boldsymbol{N}_5$; the general logics $\boldsymbol{N}_\omega$ are introduced to characterise the total and equilibrium properties of the models. Reduced logics, specifically *reduced signed logics* have been used in [11] and [34] to improve ATPs in many-valued logics.

If ω_1 and ω_2 are two possible truth values functions and $\omega_1(p) \subset \omega_2(p)$ for all variables p, then we say that ω_1 is a *reduction* of ω_2 and write $\omega_1 \subset \omega_2$. The relationship between the logics $\boldsymbol{N}_{\omega_1}$ and $\boldsymbol{N}_{\omega_2}$ in this case is explained in the following theorem.

Theorem 1. *Let ω_1 and ω_2 be two possible truth-values functions with $\omega_1 \subset \omega_2$. Then:*

1. *Every ω_1-assignment is an ω_2-assignment.*
2. *If σ is a model of φ in $\boldsymbol{N}_{\omega_1}$, then it is a model in $\boldsymbol{N}_{\omega_2}$.*
3. *If φ is valid in $\boldsymbol{N}_{\omega_2}$, then it is valid in $\boldsymbol{N}_{\omega_1}$.*

Any $\boldsymbol{N}_5$ model σ as a truth-value assignment can trivially be converted into a Kripke model $\langle H, T\rangle$, and *vice versa*. For example, if σ is an assignment and p is a propositional variable, then the corresponding Kripke model is determined by the equivalences:

$$\begin{array}{lll}
\sigma(p) = \ \ 2 & \text{iff} & p \in H \\
\sigma(p) = \ \ 1 & \text{iff} & p \in T, p \notin H \\
\sigma(p) = \ \ 0 & \text{iff} & p \notin T, {\sim}p \notin T \\
\sigma(p) = -1 & \text{iff} & {\sim}p \in T, {\sim}p \notin H \\
\sigma(p) = -2 & \text{iff} & {\sim}p \in H
\end{array}$$

Remark 1. The many-valued semantics and the Kripke semantics for $\boldsymbol{N}_5$ are equivalent. In other words, if Π is a set of formulas in $\boldsymbol{N}_5$ and ψ is a formula, then $\Pi \models \psi$ iff for every assignment σ in $\boldsymbol{N}_5$, if $\sigma(\varphi) = 2$ for every $\varphi \in \Pi$, then $\sigma(\psi) = 2$

3.1 Tableau System for $\boldsymbol{N_5}$

To study the validity of an inference in $\boldsymbol{N}_5$ we are going to use the following tableau system:

Initial tableau for $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$

$$\begin{array}{c}
\hline
\{2\}{:}\varphi_1 \\
\cdots \\
\{2\}{:}\varphi_n \\
\{-2,-1,0,1\}{:}\psi
\end{array}$$

Tableau expansion rules We show the rules for the connective $\rightarrow$ in Figure 1: the other four connectives, $\wedge$, $\vee$, $\sim$ and $\neg$ are *regular* connectives, and the standard expansion rules can be applied (see [13] for details).

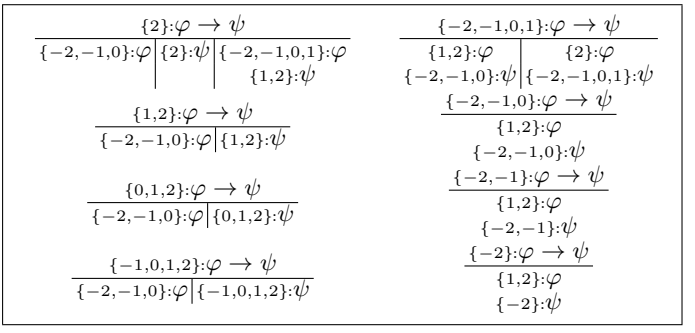


Fig. 1. Tableau expansion rules in $\boldsymbol{N}_5$ for $\rightarrow$

Definition 2. *Let $\Pi = \{\varphi_1, \ldots, \varphi_n\}$ be a set of formulas and ψ another formula:*

1. *If T is a tableau for (Π, ψ) and T' is the tree obtained from T applying one of the expansion rules, then T' is a tableau for (Π, ψ). As usual in tableau systems for propositional logics, if a formula can be used to expand the tableau, then the tableau is expanded in every branch below the formula using the corresponding rule; then the formula used to expand is marked and is no longer used.*
2. *A branch B in a tableau T is called* closed *if it contains a variable p with two signs, S:p, S':p, such that $S \cap S' = \varnothing$.*
3. *A branch B in a tableau T is called* finished *if it doesn't contain non-marked formulas; it is called* open *if it is non-closed and finished.*
4. *A tableau T is called* closed *if every branch is closed, it is* open *if it has an open branch, (ie. if it is non-closed), and it is* terminated *if every branch is either closed or open.*

The nodes in the tableaux are formulas in $\mathcal{N}$ labeled with a set of truth values, S:φ; this construction is called a *signed formula*; if p is a propositional variable, S:φ is called a *signed literal*.

The previous definition is common to any tableau system, the differences between them lie in the construction of the initial tableau and the specific family of tableau rules. The closed, open and terminated trees can be used in different ways; in the logic $\boldsymbol{N}_5$ we use closed tableaux to characterise validity.

Theorem 2 (Soundness and completeness of the tableau system). *The inference $\varphi_1, \ldots, \varphi_n \models \psi$ is valid if and only if there exists a closed tableau for $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$.*

4 Equilibrium Models and Equilibrium Inferences

For the time being we work in the language of $\boldsymbol{N}_5$ and revert to the earlier Kripke model semantics in order to give the original definition of equilibrium model. Equilibrium models are special kinds of minimal $\boldsymbol{N}_5$ Kripke models. We first define a partial ordering $\trianglelefteq$ on $\boldsymbol{N}_5$ models as follows.

Definition 3. *Given any two models* $\langle H, T\rangle$, $\langle H', T'\rangle$, *we set* $\langle H, T\rangle \trianglelefteq \langle H', T'\rangle$ *if* $T = T'$ *and* $H \subseteq H'$.

This leads to the following notion of equilibrium.

Definition 4. *Let* Π *be a set of* $\boldsymbol{N}_5$ *formulas and* $\langle H, T\rangle$ *a model of* Π.

1. $\langle H, T\rangle$ *is said total if* $H = T$.
2. $\langle H, T\rangle$ *is said an* equilibrium *model if it is minimal under* $\trianglelefteq$ *among models of* Π, *and it is total.*

As examples, consider the following sets of formulas, where p, q are distinct atoms: let $\Pi_1 = \{\sim p, \neg q \to p\}$; $\Pi_2 = \{\neg p \to q\}$; $\Pi_3 = \{\neg\neg p \to p\}$. It is easily verified that Π_1 has no equilibrium models since $\langle\{\sim p\}, \{q, \sim p\}\rangle$ is minimal, that Π_2 has a single equilibrium model $\langle\{q\}, \{q\}\rangle$, and that Π_3 has two equilibrium models: $\langle\{\}, \{\}\rangle$ and $\langle\{p\}, \{p\}\rangle$. The above notions are readily characterised using the many-valued semantics.

Remark 2. Let Π be a set of formulas in $\mathcal{N}$. In $\boldsymbol{N}_5$, the ordering $\sigma_1 \trianglelefteq \sigma_2$ among models σ_1 and σ_2 of Π holds iff for every propositional variable p occurring in Π the following properties hold:

1. $\sigma_1(p) = 0$ if and only if $\sigma_2(p) = 0$.
2. If $\sigma_1(p) \geq 1$, then $\sigma_1(p) \leq \sigma_2(p)$
3. If $\sigma_1(p) \leq -1$, then $\sigma_1(p) \geq \sigma_2(p)$

This yields characterisations of total model and equilibrium model, clearly equivalent to the original.

Remark 3. Let $\Pi = \{\varphi_1, \ldots, \varphi_n\}$ be a set of formulas in $\mathcal{N}$. A model σ of Π in $\boldsymbol{N}_5$ is a *total model* if $\sigma(p) \in \{-2, 0, 2\}$ for every propositional variable p in Π, that is, if it is a model of Π in $\boldsymbol{N}_3$.

4.1 Tableau System for Total Models

We now turn to a system for generating the total models of a set of formulas

$$\Pi = \{\varphi_1, \ldots, \varphi_n\}$$

Initial tableau for Π

$$\begin{array}{c} \overline{\{2\}{:}\varphi_1} \\ \ldots \\ \{2\}{:}\varphi_n \end{array}$$

Tableau expansion rules All the connectives in $\boldsymbol{N}_3$ are regular,[6] and therefore their expansion rules are described in the standard way; in Figure 2 we illustrate the rules for the connectives $\rightarrow$ an $\neg$.

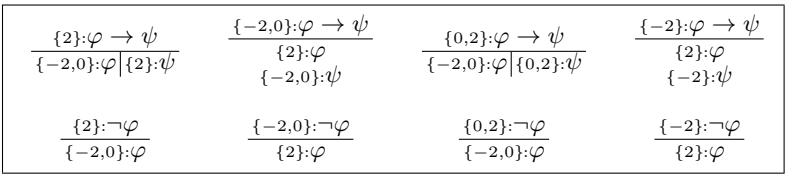


Fig. 2. Tableau expansion rules in $\boldsymbol{N}_3$ for the connectives $\rightarrow$ and $\neg$

The total models for a set Π are generated from any terminated tableau, as described in the following theorem.

Theorem 3. *Let T be a terminated tableau for $\Pi = \{\varphi_1, \ldots, \varphi_n\}$, and let $\{S_1{:}p_1, \ldots, S_n{:}p_n\}$ be the set of signed literals in an open branch. Then every assignment σ verifying $\sigma(p_i) \in S_i$, for all i, is a total model of Π. Moreover, all the total models of Π are generated from T in this way.*

Note that the variables in an open branch are not necessarily all the variables in Π; if a variable is missing in an open branch, then any value can be taken to complete the model.

4.2 Equilibrium Logic

Equilibrium logic is the logic determined by the equilibrium models of a theory; we define it formally below in terms of a nonmonotonic entailment relation. Part of the interest of equilibrium logic arises from the fact that on a syntactically restricted class of theories it coincides with a well-known nonmonotonic inference relation studied in logic programming. Formally, when a consistent theory has the syntactic shape of a (disjunctive, extended) logic program,[7] its equilibrium models coincide with its answer sets in the sense of [8].

Definition 5 (Equilibrium entailment). *Let $\varphi_1, \ldots, \varphi_n, \varphi$ be formulas in $\mathcal{N}$. We define the relation $\mid\sim$ called* equilibrium entailment*, as follows*

1. *If $\Pi = \{\varphi_1, \ldots, \varphi_n\}$ has equilibrium models, then $\varphi_1, \ldots, \varphi_n \mid\sim \varphi$ if every equilibrium model of Π is a model of φ in $\boldsymbol{N}_5$.*

[6] Actually, the connective $\rightarrow$ is not regular, but it is *ortho-regular*. The ortho-regular connectives were introduced in [34] to generalise regular connectives.

[7] This means that every formula of the theory can be expressed in the form of either a disjunction of literals, or an implication whose antecedent is a conjunction of literals and weakly negated literals, and whose consequent is a disjunction of literals.

2. *If either* $n = 0$ *or* Π *has no equilibrium models, then* $\varphi_1, \ldots, \varphi_n \mathrel{|\!\sim} \varphi$ *if* $\varphi_1, \ldots, \varphi_n \models \varphi$.

The process of checking equilibrium entailment, $\Pi \mathrel{|\!\sim} \psi$, will be understood as follows:

Step 1 Generate the total models of Π using the tableau system in section 4.1
Step 2 For every total model of Π, check the equilibrium property (using the tableau system in the next section). If there are equilibrium models, then go to step 3, else go to step 4.
Step 3 For every equilibrium model of Π, check if it is a model of ψ.
Step 4 If Π doesn't have equilibrium models, just check entailment in $\boldsymbol{N}_5$.

Therefore, in the next section we tackle the problem of checking the equilibrium property using tableau systems.

4.3 The Equilibrium Property

In this section we are going to work with a finite set of formulas Π and a total model σ of Π. We define the possible truth values function ω_σ as follows

$$\omega_\sigma(p) = \begin{cases} \{-2,-1\} & \text{if } \sigma(p) = -2 \\ \{0\} & \text{if } \sigma(p) = 0 \\ \{1,2\} & \text{if } \sigma(p) = 2 \end{cases}$$

Then, we can characterise the equilibrium property using the logic $\boldsymbol{N}_{\omega_\sigma}$.

Theorem 4. *The total model* σ *of* Π *is an equilibrium model iff* σ *is the unique model of* Π *in* $\boldsymbol{N}_{\omega_\sigma}$.

In this way, we can use a model generator in tableau style for $\boldsymbol{N}_{\omega_\sigma}$ to check the equilibrium property in $\boldsymbol{N}_5$.

4.4 A Tableau System for Checking the Equilibrium Property

Although the tableau system for $\boldsymbol{N}_5$ can be used, we shall introduce an improved system in which we can use restrictions in the logics $\boldsymbol{N}_{\omega_\sigma}$.

Let φ be a formula in a logic $\boldsymbol{N}_\omega$ and $S \subset \mathbf{5}$; we write φ_S if every assignment in $\boldsymbol{N}_\omega$ evaluates φ on S, ie. if φ is a *S-tautology.* We call *signing-up* the process which, applied to a formula φ, partially evaluates the function represented by this formula in $\mathfrak{N}$ to determine the smallest set S such that φ is an S-tautology; the expansion rule that one applies to a formula is selected according to the principle connective of the formula and the set that is calculated in the signing-up process. To introduce the rules we need to observe that (just looking at the truth tables) in $\boldsymbol{N}_{\omega_\sigma}$ every formula is either a $\{0\}$-tautology or a $\{-2,-1\}$-tautology or a $\{1,2\}$-tautology.

Lemma 1. *Let ω be a function such that, for every propositional variable p, either $\omega(p) = \{-2,-1\}$, or $\omega(p) = \{0\}$, or $\omega(p) = \{1,2\}$. Then, for every formula φ in $\boldsymbol{N}_\omega$, just one of the following properties holds.*

a) $\varphi_{\{-2,-1\}}$ b) $\varphi_{\{0\}}$ c) $\varphi_{\{1,2\}}$

Now we can describe the system to check if a total model σ of a set $\Pi = \{\varphi_1, \ldots, \varphi_n\}$ is an equilibrium model:

Initial tableau for (Π, σ)

$$\frac{}{\begin{array}{c}\{2\}{:}\varphi_{1S_1}\\ \ldots\\ \{2\}{:}\varphi_{nS_n}\end{array}}$$

Tableau expansion rules All the expansion rules are shown in Figure 3.

$$\frac{S_1{:}(\varphi_{S_2})}{\bot}\ (\text{If } S_1 \cap S_2 = \varnothing) \qquad \frac{S_1{:}(\varphi_{S_2})}{\top}\ (\text{If } S_2 \subset S_1) \qquad \frac{S_1{:}(\varphi_{S_2})}{(S_1 \cap S_2){:}\varphi}\ (\text{Otherwise})$$

$$\frac{\{2\}{:}(\varphi_{\{1,2\}} \to \psi_{\{1,2\}})}{\{1\}{:}\varphi \mid \{2\}{:}\psi} \qquad \frac{\{1\}{:}(\varphi_{\{1,2\}} \to \psi_{\{1,2\}})}{\begin{array}{c}\{2\}{:}\varphi\\ \{1\}{:}\psi\end{array}}$$

$$\frac{\{-2\}{:}(\varphi_{\{1,2\}} \to \psi_{\{-2,-1\}})}{\{-2\}{:}\psi} \qquad \frac{\{-1\}{:}(\varphi_{\{1,2\}} \to \psi_{\{-2,-1,0\}})}{\{-1\}{:}\psi}$$

$$\frac{\{-2\}{:}\neg(\varphi_{\{1,2\}})}{\{2\}{:}\varphi} \qquad \frac{\{-1\}{:}\neg(\varphi_{\{1,2\}})}{\{1\}{:}\varphi}$$

$$\frac{\{2\}{:}\sim(\varphi_{\{-2,-1\}})}{\{-2\}{:}\varphi} \qquad \frac{\{1\}{:}\sim(\varphi_{\{-2,-1\}})}{\{-1\}{:}\varphi}$$

$$\frac{\{-2\}{:}\sim(\varphi_{\{1,2\}})}{\{2\}{:}\varphi} \qquad \frac{\{-1\}{:}\sim(\varphi_{\{1,2\}})}{\{1\}{:}\varphi}$$

$$\frac{\{2\}{:}(\varphi_{\{1,2\}} \vee \psi_S)}{\{2\}{:}\varphi \mid \{2\}{:}\psi_S} \qquad \frac{\{1\}{:}(\varphi_{\{1,2\}} \vee \psi_S)}{\begin{array}{c}\{1\}{:}\varphi\\ \{-2,-1,0,1\}{:}\psi_S\end{array}}$$

$$\frac{\{-2\}{:}(\varphi_{\{-2,-1\}} \vee \psi_{\{-2,-1\}})}{\begin{array}{c}\{-2\}{:}\varphi\\ \{-2\}{:}\psi\end{array}} \qquad \frac{\{-1\}{:}(\varphi_{\{1,2\}} \vee \psi_S)}{\{-1\}{:}\varphi \mid \{-1\}{:}\psi_S}$$

$$\frac{\{2\}{:}(\varphi_{\{1,2\}} \wedge \psi_{\{1,2\}})}{\begin{array}{c}\{2\}{:}\varphi\\ \{2\}{:}\psi\end{array}} \qquad \frac{\{1\}{:}(\varphi_{\{-2,-1\}} \wedge \psi_{\{-2,-1\}})}{\{1\}{:}\varphi \mid \{1\}{:}\psi}$$

$$\frac{\{-2\}{:}(\varphi_{\{-2,-1\}} \wedge \psi_S)}{\{-2\}{:}\varphi \mid \{-2\}{:}\psi_S} \qquad \frac{\{-1\}{:}(\varphi_{\{-2,-1\}} \wedge \psi_S)}{\begin{array}{c}\{-1\}{:}\varphi\\ \{-1,0,1,2\}{:}\psi_S\end{array}}$$

Fig. 3. tableau expansion rules for $\boldsymbol{N}_\omega$

Some remarks on this may be useful. In the construction of the initial tree we add the set that is determined by the signing-up process. This set is used to decide which of the first three expansion rules is to be applied. Then, the same signing-up process applied to the subformulas allows one to select the subsequent expansion rule. The meaning of the first three rules is the following:

- The situation $S_1{:}(\varphi_{S_2})$ with $S_1 \cap S_2 = \varnothing$ cannot generate any model because the formula is always evaluated in S_2 and we need to evaluate it in the disjoint set S_1. So in this situation we can close the branch; this is represented by the $\bot$ symbol in the expansion rule.
- The situation $S_1{:}(\varphi_{S_2})$ with $S_2 \subset S_1$ doesn't give rise to any restriction because the formula is always evaluated in S_2 and then in S_1. So in this situation we just mark the signed formula without expanding the tableau; this is represented by the $\top$ symbol in the expansion rule.
- If we have the formula $S_1{:}(\varphi_{S_2})$ and the previous situations don't hold, then the sign can be propagated to the subformulas, actually we only need to propagate the sign $S_1 \cap S_2$.

To characterise the equilibrium property an extension of the closed branch notion is necessary.

Definition 6. *A branch B in a tableau T of Π is called* σ-closed *if one of the following conditions holds:*

1. *It is closed.*
2. *The logical constant $\bot$ appears in B.*
3. *The set of signed literals in the branch is just $\{^{\{\sigma(p_1)\}}{:}p_1, \ldots, ^{\{\sigma(p_n)\}}{:}p_n\}$ where $\{p_1, \ldots, p_n\}$ is the set of propositional variables in Π with $\sigma(p) \neq 0$.*

A tableau is called σ-closed *if every branch in it is σ-closed.*

Theorem 5. *Let σ be a total model of Π. σ is an equilibrium model of Π if and only if there exist a σ-closed tableau for (Π, σ).*

Example 2. Let us consider the following set $\Pi = \{\sim q, \neg p \to q, \neg r \to p\}$. First we generate the total models of Π; using the tableau system for $\boldsymbol{N}_3$ we obtain the terminated tableau in figure 4. Therefore, the total models are:

	p	q	r
σ_1	2	−2	−2

	p	q	r
σ_2	2	−2	0

	p	q	r
σ_3	2	−2	2

To check the equilibrium property for σ_1 we construct a tableau in the logic $\boldsymbol{N}_\omega$ where $\omega_1(p) = \{1, 2\}$, $\omega_1(q) = \{-2, -1\}$ and $\omega_1(r) = \{-2, -1\}$. This tableau is shown in the figure 5; it is open and therefore the model σ_1 is not an equilibrium model; from this open tableau we also obtain a model below σ_1 in the ordering $\trianglelefteq$, namely: $\tau(p) = 2$, $\tau(q) = -2$ and $\tau(r) = -1$.

We now proceed to check the equilibrium property for σ_2; we construct a tableau in the logic $\boldsymbol{N}_{\omega_2}$ where $\omega_2(p) = \{1, 2\}$, $\omega_2(q) = \{-2, -1\}$ and $\omega_2(r) = \{0\}$. This tableau is showed in the figure 5; the left branch is σ_2-closed because it contains the constant $\bot$, and the right branch is σ_2-closed because it contains the signed literals $^{\{2\}}{:}p$ and $^{\{-2\}}{:}q$ and $\sigma_2(r) = 0$. Therefore the tableau is σ_2-closed and σ_2 is an equilibrium model.

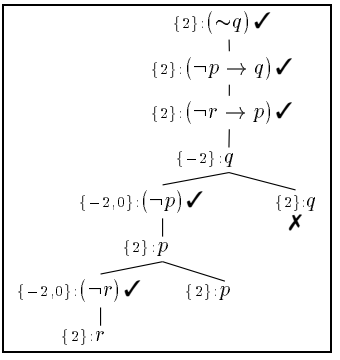


Fig. 4. Tableau for total models of the formula in the example 2.

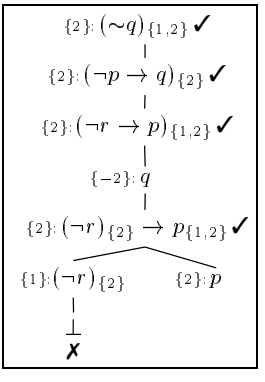


Fig. 5. Tableau for equilibrium property of the model σ_1 in example 2.

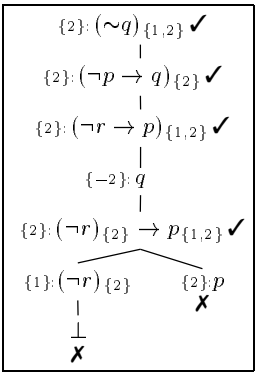


Fig. 6. Tableau for equilibrium property of the model σ_2 in example 2.

5 Related Work and Conclusions

There is now a growing body of literature devoted to applying tableau and related methods for computing entailment in systems of nonmonotonic reasoning. Recently, for example, Niemelä [22,23] has studied tableau calculi for reasoning based on minimal models and circumscription, while Bonatti and Olivetti [1, 2] have considered (sequent) calculi for default and autoepistemic logics; for an overview and additional references for the area, see Olivetti [27]. On the other hand, we do not know of previous studies of tableaux for nonmonotonic systems that are, like equilibrium entailment, based on a nonclassical underlying logic. Nor do we know of implementations of logic programming semantics, such as stable models, that make use of traditional forms of tableau calculi.[8]

[8] There is an approach of Stuber [33] that uses a series of transformation rules to simplify programs for computing stable models. And Inoue *et al.* [15] compute answer sets for programs with strong negation by applying a model generation theorem prover to a modal translation of the program.

We have presented here tableau calculi for the logic $\boldsymbol{N}_5$ of here-and-there with strong negation and for its strengthening $\boldsymbol{N}_3$, or classical logic with strong negation. Equilibrium models are special kinds of $\boldsymbol{N}_3$ models defined by a minimality condition over $\boldsymbol{N}_5$ models. They characterise the nonmonotonic equilibrium entailment relation. We have presented a tableau calculus for checking the equilibrium property and hence the nonmonotonic entailment relation. A novel feature of the calculus is its use of a generalisation of the signing method, called signing-up, which helps to simplify the complexity of the tableau rules. By the property that equilibrium logic generalises the answer set semantics of [8], it follows that our calculus can be used to test entailment in logic programs under the stable model or answer set semantics; though of course only an appropriate subset of the tableau rules would be needed in such a case, given the restricted syntax of logic programs. A topic for future work will be the analysis of the complexity of equilibrium entailment and model-checking.

References

1. Bonatti, P A, Sequent calculi for default and autoepistemic logics, in P Miglioli, U Moscato, D Mundici & M Ornaghi (eds), *Proc. Tableaux'96*, LNAI 1071, pages 107–121, Springer, 1996.
2. Bonatti, P A, & Olivetti, N, A sequent calculi for skeptical default logic, in D Galmiche (ed) *Proc. Tableaux'97*, LNAI 1227, pages 127–142, Springer, 1997.
3. Brass, S, & Dix, J, Characterizations of the Disjunctive Stable Semantics by Partial Evaluation, *J Logic Programming* 32(3) (1997), pages 207–228.
4. Chagrov, A, & Zakharyaschev, M, *Modal Logic*, Oxford: Clarendon Press, 1997.
5. van Dalen, D, Intuitionistic Logic, in D Gabbay, & F Guenthner (eds), *Handbook of Philosophical Logic, Vol. III*, pages 225–340, Kluwer, Dordrecht, 1986.
6. Gelfond, M, & Lifschitz, V, The Stable Model Semantics for Logic Programs, in K Bowen & R Kowalski (eds), *Proc 5th Int Conf on Logic Programming 2*, MIT Press, pages 1070–1080, 1988.
7. Gelfond, M & Lifschitz, V, Logic Programs with Classical Negation, in D Warren & P Szeredi (eds), *Proc ICLP-90*, MIT Press, pages 579–597, 1990.
8. Gelfond, M, & Lifschitz, V, Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing* (1991), pages 365–387.
9. Gödel, K, Zum intuitionistischen Aussagenkalkül, *Anzeiger der Akademie der Wissenschaften in Wien 69*, pages 65–66, 1932.
10. Gurevich, Y, Intuitionistic Logic with Strong Negation, *Studia Logica* 36 (1977), pages 49–59.
11. P. de Guzmán, I, Ojeda & Valverde, A. Multiple-Valued Tableaux with Δ-reductions. In *Proceedings of IC-AI'99*, pages 177–183. CSREA Press.
12. Hähnle, R Towards an efficient tableau proof procedure for multiple-valued logics. In Egon Börger, Hans Kleine Büning, Michael M. Richter, and Wolfgang Schönfeld, editors, *Selected Papers from Computer Science Logic, CSL'90, Heidelberg, Germany*, volume 533 of *Lectures Notes in Computer Science*, pages 248–260. Springer-Verlag, 1991.
13. Hähnle, R. *Automated Deduction in Multiple-valued Logics.* Oxford University Press, 1993.
14. Heyting, A, Die formalen Regeln der intuitionistischen Logik, *Sitzung Preuss. Akad. Wiss. Phys. Math. Kl*, pages 42–56, 1930.

15. Inoue, K, Koshimura, M, & Hasegawa, R, Embedding negation as failure into a model generation theorem prover, in D Kapur (ed), *Automated Deduction - Proc. CADE-11*, LNAI 607, pages 400–415. Springer, 1992.
16. Kracht, M, On Extensions of Intermediate Logics by Strong Negation, *J Philosophical Logic* 27, pages 49–73, 1998.
17. Leone, N, Rullo, P & Scarcello, F, Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics, and Computation, *Information and Computation*, 135 (2), 1997, pages 69–112.
18. Lifschitz, V, Tang, L R, and Turner, H, Nested expressions in logic programs, *Annals of Mathematics and Artificial Intelligence*, to appear.
19. Łukasiewicz, J, Die Logik und das Grundlagenproblem, in *Les Entreties de Zürich sur les Fondaments et la Méthode des Sciences Mathématiques* 6-9, 12 (1938), Zürich, pages 82–100, 1941.
20. Maksimova, L, Craig's interpolation theorem and amalgamable varieties, *Doklady Akademii Nauk SSSR*, 237, no. 6, (1977), pages 1281–1284 (translated as *Soviet Math. Doklady*).
21. Nelson, D, Constructible Falsity, *J Symbolic Logic* 14 (1949), pages 16–26.
22. Niemelä, I, A tableau calculus for minimal model reasoning, in P Miglioli *et al* (eds), *Theorem Proving with Analytic Tableaux and Related Methods*, pages 278–294. Springer LNAI 1071, 1996.
23. Niemelä, I, Implementing Circumscription Using a Tableau Method, in *Proc European Conference on Artificial Intelligence, ECAI 96*, pages 80–84. John Wiley, 1996.
24. Niemelä, I, & Simons, P, Efficient Implementation of the Well-founded and Stable Model Semantics, *Proc JICSLP 96*, pages 289–303. MIT Press, 1996.
25. Ojeda, M, P. de Guzmán, I, Aguilera, G, and Valverde, A. Reducing signed propositional formulas. *Soft Computing*, vol. 2(4), pages 157–166, 1998.
26. Olivetti, N, A tableau and sequent calculus for minimal entailment, *J Automated Reasoning* 9, pages 99–139, 1992.
27. Olivetti, N, Tableaux for nonmonotonic logics, in M D'Agostino, D M Gabbay, R Hähnle, J Posegga (eds), *Handbook of Tableau Methods*, Kluwer, 1999.
28. Pearce, D, A New Logical Characterisation of Stable Models and Answer Sets, in J Dix, L M Pereira, & T Przymusinski (eds), *Non-monotonic Extensions of Logic Programming. Proc NMELP 96.* Springer, LNAI 1216, pages 57–70, 1997.
29. Pearce, D, From Here to There: Stable Negation in Logic Programming, in D Gabbay & H Wansing (eds), *What is Negation?*. Kluwer, pages 161-181, 1999.
30. Przymusinski, T, Stable Semantics for Disjunctive Programs, *New Generation Computing* 9, pages 401–424, 1991.
31. Rautenberg, W, *Klassische und Nichtklassische Aussagenlogik*, Vieweg, 1979.
32. Smetanich, Ya, S, On Completeness of a Propositional Calculus with an additional Operation of One Variable (in Russian). *Trudy Moscovskogo matrmaticheskogo obshchestva*, 9, pages 357-372, 1960.
33. Stuber, J, Computing stable models by program transformation, in P Van Hentenryck (ed), *Proc. ICLP 94*, pages 58–73. MIT Press, 1994.
34. Valverde, A, *Δ-Árboles de implicantes e implicados y reducciones de lógicas signadas en ATPs*. PhD thesis, University of Málaga, Spain, July 1998.
35. Vorob'ev, N N, A Constructive Propositional Calculus with Strong Negation (in Russian), *Doklady Akademii Nauk SSR* 85 (1952), 465–468.
36. Vorob'ev, N N, The Problem of Deducibility in Constructive Propositional Calculus with Strong Negation (in Russian), *Doklady Akademii Nauk SSR* 85 (1952), 689–692.

Towards Tableau-Based Decision Procedures for Non-Well-Founded Fragments of Set Theory

Carla Piazza and Alberto Policriti

Dip. di Matematica e Informatica, Univ. di Udine
Via delle Scienze 206, 33100 Udine (Italy).
(piazza|policriti)@dimi.uniud.it

Abstract. We propose a tableau-like decision procedure for deciding the finite satisfiability of unquantified formulae with (a weak form of) *powerset* constructor. Our results apply to a rather large class of non-well-founded set theories. The decidability result presented can be seen as a first step towards the decidability of the class of formulae into which the $\Box$-as-$\mathcal{P}$ (box-as-powerset) translation maps modal formulae. The analysis we make clarifies some of the difficulties in deciding this class of formulae in the case without atoms. The procedure we define can be used as a subroutine to decide the same class of formulae in Set Theory without foundation.

1 Introduction

Decidability results for fragments of Set Theory have been studied since the late seventies [13] with the long-term aim of providing a collection of procedures capable of computing within the *decidable core* of Set Theory [9]. As a matter of fact, many issues are still not clearly stated in this area. Among the most important ones, in the opinion of the authors, are:

- a precise formulation of the above mentioned *decidable core*, and
- a sensible definition of the set theoretic assumptions (axioms) assumed to hold in the underlying theory of sets.

It is still rather unclear what kind of classification (*á la* Hilbert, cf. [12,15,6]) can be proposed in order to set the basis for the study of what could be called *The Classical Decision Problem for Set Theory*. For example, there are strong signals that a classification based on the quantificational prefix is neither suitable nor interesting. However, basing a classification on classes of unquantified formulae forces on a choice (that needs justification) on which operators to consider.

As for the second of the above two points, instead, it is clear that some non-standard set-theoretic principles are extremely useful and well-suited for applications. The most important and popular (especially among computer scientists) of those principles (cf. [2]) is the so-called *Anti-Foundation Axiom* (see [1,14]). In general, as far as the choice of the underlying axioms is concerned, when dealing with decidability problems the safest (and often most reasonable) choice is

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 368–382, 2000.

a minimal one. For example, again in the case of the foundation/anti-foundation principle, the stronger result one can try to obtain in term of decidability is a decision procedure that works for a set theory *without* foundation. Once such a decision procedure is discovered one can usually tune it in order to deal with (some form of) anti-foundation (cf. [16]). Moreover, the flexibility provided by the absence of some axioms allows also to play with sets in a rather peculiar way: in [11,3,4], exploiting the possibilities given by an unrestricted membership relation, a set-theoretic translation—called $\Box$-as-$\mathcal{P}$ (box-as-powerset) translation—rewriting any modal formula into a set-theoretic one, is proposed. Such a method shows that, with the amount of second order logic introduced by means of the axiomatic set theory driving the translation, a mild form of the deduction theorem holds for any modal logic specified by any set of Hilbert's axioms.

In order to provide decision procedures for the set-theoretic languages supporting the above mentioned translation technique, we propose here a tableau-based decision procedure for an unquantified set-theoretic class of formulae, in the non-well-founded and non-extensional case. Our decidability result can be seen as a first step towards the decidability of the class of formulae into which the $\Box$-as-$\mathcal{P}$ translation maps modal formulae.

Even though it is not entirely trivial to keep under control the technical details, the decidability result presented here allows to cope only with a non-functional form of the powerset operator, together with the usual operators of MLS, Multi-Level Syllogistic (already treated in conjunction with others in the well-founded case, with a tableau-based decision procedure in [7,8]), in models of set theories that are neither well-founded nor extensional. As we said before, the case in which some form of anti-foundation axiom is assumed should easily follow from these results. This does not seem to be the case for extensionality.

The paper is organized as follows: in Section 2 we present a tableau-like decision procedure for deciding the basic case of unquantified formulae with Boolean operators and membership with respect to a non-well-founded set theory; in Section 3 we generalize the previous result to decide formulae with powerset in presence of *atoms* (i.e. distinct empty sets). The conclusions are drawn in Section 4.

2 The Basic Case

2.1 The Language and the Axioms

Consider the set $\mathcal{F} = \{\cup, \cap, \setminus\}$ of function symbols, the set $\mathcal{P} = \{\in, \subseteq, =\}$ of predicate symbols, and a denumerable set $\mathcal{V}$ of variables. Let $\mathcal{L} = \langle \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$ be the first-order language built from these sets of symbols.

In order to introduce the anti-foundation axiom (AFA) (cf. [1,14]), we give the definition of *decoration*.

Definition 1. *Given a graph* $\mathcal{G} = \langle G, R\rangle$ *and an interpretation* $\mathcal{M} = \langle M, (.)^{\mathcal{M}}\rangle$ *of* $\mathcal{L}$*, a* decoration *of* $\mathcal{G}$ *over* $\mathcal{M}$ *is a function* $d : G \longrightarrow M$ *such that:*

$$\forall n \in G \forall y \in M(y \in^{\mathcal{M}} d(n) \leftrightarrow \exists m \in G(y = d(m) \wedge Rnm)).$$

The anti-foundation axiom (AFA) states:

Every graph has a unique decoration.

We use the notation $\mathcal{S}et^-$ to refer to the first-order theory whose axiomatization is presented in Figure 1, and $\mathcal{S}et$ to refer to $\mathcal{S}et^-$ together with the anti-foundation axiom (AFA) (cf. [1,14]). Notice that (E) is the extensionality axiom, (N) states the existence of the empty set, and (S) the definability of the *singleton* operator[1]. The letters φ and ψ will be used to refer to formulae of the language

$(\cup)$	$\forall x,\, y,\, z\,(x \in y \cup z \leftrightarrow x \in y \vee x \in z)$
$(\cap)$	$\forall x,\, y,\, z\,(x \in y \cap z \leftrightarrow x \in y \wedge x \in z)$
$(\setminus)$	$\forall x,\, y,\, z\,(x \in y \setminus z \leftrightarrow x \in y \wedge x \notin z)$
$(\subseteq)$	$\forall x,\, y\,(x \subseteq y \leftrightarrow \forall z\,(z \in x \rightarrow z \in y))$
(E)	$\forall x,\, y\,(x = y \leftrightarrow \forall z\,(z \in x \leftrightarrow z \in y))$
(N)	$\exists y \forall x\,(x \notin y)$
(S)	$\forall x \exists y \forall z\,(z \in y \leftrightarrow z = x)$

Fig. 1. The axiomatization of $\mathcal{S}et^-$

$\mathcal{L}$ and $FV(\varphi)$ denotes the set of free variables in φ.

2.2 The Decision Procedure for the Basic Case

Let φ_i be a conjunction of literals of the form:

$$x \in y,\, x \notin y,\, x_1 \circ x_2 = y \tag{1}$$

where x, x_1, x_2, y are variables and $\circ \in \{\cup, \cap, \setminus\}$. In this section we use φ to refer to a disjunction $\varphi_1 \vee \ldots \vee \varphi_n$ of such conjunctions. The class of all the disjunctions of conjunctions of literals of the form (1) will be called $\mathcal{C}_1$.

In Figure 2 we describe a procedure $\mathsf{de-Boole}$ which transforms a given formula φ of $\mathcal{C}_1$ in order to decide its satisfiability. The procedure makes use

$\mathsf{de-Boole}(\varphi)$:
$\varphi := \in\!\!-\mathsf{introduction}(\mathsf{Extensionalize}(\in\!\!-\mathsf{introduction}(\varphi)))$.

Fig. 2. The procedure $\mathsf{de-Boole}$

of the two functions, $\in\!\!-\mathsf{introduction}, \mathsf{Extensionalize} : \mathcal{C}_1 \longrightarrow \mathcal{C}_1$, described in Figure 3. The function $\in\!\!-\mathsf{introduction}$ maps a formula φ into a new formula

[1] The axioms (N) and (S) are not used in the decision procedure, but in the construction of a solution.

$\in-\mathsf{introduction}(\varphi)$ which is equivalent to φ with respect to the theory $\mathcal{S}et^-$. The rules applied to compute $\in-\mathsf{introduction}(\varphi)$ (rules of F) are presented in Figure 4. The aim of this function is to spell out the membership relation, according to the axioms which define the operators. Similarly, the function Extensionalize maps a formula ψ into an equivalent one, with respect to the theory $\mathcal{S}et^-$, making use of the rule S in Figure 5. The rule applied by the function Extensionalize to ψ is based on the extensionality axiom (E). This rule is applied only when it is necessary to check the satisfiability: it introduces a new variable and hence, if not controlled, it can lead to non-termination. We apply the rule in such a way that the number of new variables introduced is bounded by the number of variables of the formula ψ, see (Lemma 1). In the functions, the notation $\_ \overset{F}{\mapsto} \_$ (respectively $\_ \overset{S}{\mapsto} \_$) means that it is possible to apply a rule[2] of F (see Figure 4) (respectively S, see Figure 5) to literals of the l.h.s.: if this is the case, then the r.h.s. has to be added as a conjunct in the l.h.s.. In Example 1 we present some of the rules in tableau style. The function disj_form maps a formula to its disjunctive normal form.

```
∈−introduction(φ1 ∨ ... ∨ φn) :
    for i = 1 to n do {
        repeat {
            ψ := φi;
            if (φi ↦F θ) then
                φi := φi ∧ θ; }
        until (φi = ψ)}
    φ := disj_form(φ1 ∨ ... ∨ φn);
    return φ.
```

```
Extensionalize(φ1 ∨ ... ∨ φn) :
    for i = 1 to n do {
        ψ := true;
        repeat {
            σ := ψ;
            if (φi ↦S θ) then
                ψ := ψ ∧ θ; }
        until (ψ = σ);
        φi := φi ∧ ψ}
    φ := disj_form(φ1 ∨ ... ∨ φn);
    return φ.
```

Fig. 3. The functions $\in-\mathsf{introduction}$ and Extensionalize

Example 1. The first 2 rules of F written in tableau style are:

$$\frac{y \cup z = w \quad x \in y}{x \in w} \qquad \frac{y \cup z = w \quad x \in w}{x \in y \mid x \in z}$$

Lemma 1. *The procedure* $\mathsf{de}-\mathsf{Boole}(\varphi)$ *always terminates. The number of steps performed and the number of variables in the output formula are polynomial in the number of variables of* φ.

Lemma 2. *Let* $\psi \equiv \psi_1 \vee \ldots \vee \psi_m$ *be the output of* $\mathsf{de}-\mathsf{Boole}(\varphi)$. *The formula* ψ *is satisfiable in every model* $\mathcal{M}$ *of* $\mathcal{S}et$ *if and only if there exists* $j \in \{1, \ldots, m\}$ *such that in* ψ_j *there are no literals of the form* false.

Theorem 1. *Let* $\psi \equiv \psi_1 \vee \ldots \vee \psi_m$ *be the output formula of* $\mathsf{de}-\mathsf{Boole}(\varphi)$. $\mathcal{S}et \vdash \varphi$ *if and only if there exists* $j \in \{1, \ldots, m\}$ *such that in* ψ_j *there are no literals of the form* false.

[2] The first, third, fifth, and seventh rules are applied to both y and z.

$y \cup z = w \wedge x \in y \mapsto x \in w$
$y \cup z = w \wedge x \in w \mapsto \begin{cases} x \in y \vee \\ x \in z \end{cases}$
$y \cup z = w \wedge x \notin y \mapsto \begin{cases} (x \notin z \wedge x \notin w) \vee \\ (x \in z \wedge x \in w) \end{cases}$
$y \cup z = w \wedge x \notin w \mapsto x \notin y \wedge x \notin z$
$y \cap z = w \wedge x \in y \mapsto \begin{cases} (x \in z \wedge x \in w) \vee \\ (x \notin z \wedge x \notin w) \end{cases}$
$y \cap z = w \wedge x \in w \mapsto x \in y \wedge x \in z$
$y \cap z = w \wedge x \notin y \mapsto x \notin w$
$y \cap z = w \wedge x \notin w \mapsto \begin{cases} x \notin y \vee \\ x \notin z \end{cases}$
$y \setminus z = w \wedge x \in y \mapsto \begin{cases} (x \notin z \wedge x \in w) \vee \\ (x \in z \wedge x \notin w) \end{cases}$
$y \setminus z = w \wedge x \in z \mapsto x \notin w$
$y \setminus z = w \wedge x \in w \mapsto x \in y \wedge x \notin z$
$y \setminus z = w \wedge x \notin y \mapsto x \notin w$
$y \setminus z = w \wedge x \notin w \mapsto \begin{cases} x \notin y \vee \\ x \in z \end{cases}$
$x \in y \quad \wedge x \notin y \mapsto \mathsf{false}$

Fig. 4. The rules F for the function $\in-\mathsf{introduction}$

$x \in y \wedge z \notin y \mapsto \begin{cases} (n \in x \wedge n \notin z) \vee \\ (n \notin x \wedge n \in z) \end{cases}$

Fig. 5. The rule S for the function Extensionalize

It is easy to see that each purely existentially quantified formula θ of $\mathcal{L}$ is equivalent to a formula θ^{C_1} in the class $\mathcal{C}_1$. The procedure REDUCTION presented in Figure 6 allows, given a formula θ of $\mathcal{L}$, to obtain such an equivalent formula θ^{C_1}. Hence the procedure $\mathsf{de}-\mathsf{Boole}$, together with the procedure REDUCTION, can be used to decide the class of purely existentially quantified formulae of $\mathcal{L}$: given a purely existentially quantified formula θ of $\mathcal{L}$, in order to decide its satisfiability first we convert it into a formula θ^{C_1} in the class $\mathcal{C}_1$ using the procedure REDUCTION, then we apply $\mathsf{de}-\mathsf{Boole}$ to θ^{C_1}, and we check for a disjunct without occurrences of false. Notice that the empty set $\emptyset$, whose existence is stated from (N), can be expressed using a formula of $\mathcal{C}_1$: $x = x \setminus x$. It is easy to prove that $\mathcal{S}et^- \vdash \forall x\,(x = x \setminus x \leftrightarrow \forall y\,(y \notin x))$.

Example 2. Consider the formula $\exists x,\, y\,(x = x \cup y \wedge x \neq y)$. It is satisfied in a model of $\mathcal{S}et$ if and only if y is a proper subset of x. The procedure REDUCTION maps it into $(x = x \cup y \wedge w_{xy} \in y \wedge w_{xy} \notin x) \vee (x = x \cup y \wedge w_{xy} \notin y \wedge w_{xy} \in x)$. The procedure $\mathsf{de}-\mathsf{Boole}$ applied to the first disjunct returns false after two steps. It cannot be the case that there is an element of y which is not an element of x. The procedure $\mathsf{de}-\mathsf{Boole}$ applied to the second disjunct returns $x = x \cup y \wedge w_{xy} \notin y \wedge w_{xy} \in x$, which is clearly satisfiable.

$r \subseteq s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_r = x_r \cap x_s$
$r = s \circ t$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_t = t \wedge x_r = x_s \circ x_t$ $\{r, s, t\} \not\subseteq \mathcal{V}$ and $\circ \in \{\cup, \cap, \setminus\}$
$r \neq s$	$\mapsto$	$(x_r = r \wedge x_s = s \wedge w_{rs} \in x_s \wedge w_{rs} \notin x_r) \vee$ $(x_r = r \wedge x_s = s \wedge w_{rs} \notin x_s \wedge w_{rs} \in x_r)$
$r \in s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_r \in x_s$ $\{r, s\} \not\subseteq \mathcal{V}$
$r \notin s$	$\mapsto$	$x_r = r \wedge x_s = s \wedge x_r \notin x_s$ $\{r, s\} \not\subseteq \mathcal{V}$

REDUCTION(θ) :
repeat {
$\psi := \theta$;
if ℓ is a literal in θ and $\ell \overset{T}{\mapsto} p$ then
$\theta := \theta$ with ℓ replaced from p}
until ($\psi = \theta$)
$\theta :=$ disj_form(θ).

Fig. 6. The rules T and the procedure REDUCTION

We conclude this section by observing that a rather simple way to improve the procedure presented above consists in using a combinatorial result from [5,17] in order to reduce the number of variables introduced by Extensionalize.

3 The Case with Powerset

3.1 The Language and the Theory

In this section we refer to the first-order language $\mathcal{L} = \langle \mathcal{F}, \mathcal{C}, \mathcal{P}, \mathcal{V} \rangle$, with $\mathcal{F} = \emptyset$, $\mathcal{C} = \{a_i \mid i \in \mathbb{N}\}$ a denumerable set of constant symbols, $\mathcal{P} = \{\cup, \cap, \setminus, \wp, \in, \subseteq, =\}$, and $\mathcal{V}$ a denumerable set of variables. $\cup, \cap, \setminus$ are ternary predicate symbols, while $\wp, \in, \subseteq, =$ are binary predicate symbols.

In order to facilitate the readability, we introduce the unary predicate symbol at characterizing *atoms*, to be interpreted according to the following formula:

$$\forall x \, (\mathsf{at}(x) \leftrightarrow \forall y \, (y \notin x)).$$

An axiomatization for the first-order theory $\mathcal{A}Set^-$ (where $\mathcal{A}$ stands for 'atoms') is presented in Figure 7. The main differences between the theory $\mathcal{A}Set^-$ and the theory $\mathcal{S}et^-$ are the following:

- the axioms (N_a) and $(D_{(a,b)})$ ensure the existence of infinitely many atoms, i.e. sets without elements;
- the extensionality criterion for equality (clearly) does not apply to atoms.

As a consequence of these two facts, as it is explained in the following remark, the operators $\cup, \cap, \setminus, \wp$ are not functional any more.

Remark 1. The predicate symbol $\cup$ is not functional only in the case where its arguments are atoms, as follows from:
$\mathcal{A}Set^- \vdash \forall x, y, z, w \, ((\cup(x, y, z) \wedge \cup(x, y, w) \wedge z \neq w) \rightarrow (\mathsf{at}(x) \wedge \mathsf{at}(y)))$;

$(\cup)$	$\forall y,\, z,\, w\,(\cup(y,z,w) \leftrightarrow \forall x\,(x \in w \leftrightarrow x \in y \vee x \in z))$
$(\cap)$	$\forall y,\, z,\, w\,(\cap(y,z,w) \leftrightarrow \forall x\,(x \in w \leftrightarrow x \in y \wedge x \in z))$
$(\setminus)$	$\forall y,\, z,\, w\,(\setminus(y,z,w) \leftrightarrow \forall x\,(x \in w \leftrightarrow x \in y \wedge x \notin z))$
$(\subseteq)$	$\forall x,\, y\,(x \subseteq y \leftrightarrow \forall z\,(z \in x \rightarrow z \in y))$
$(\wp)$	$\forall y\, z\,(\wp(y,z) \leftrightarrow \forall x\,((x \subseteq y \wedge \neg\mathsf{at}(x) \rightarrow x \in z)) \wedge (x \in z \rightarrow x \subseteq y))$
(E)	$\forall x,\, y\,(\neg\mathsf{at}(x) \rightarrow (x = y \leftrightarrow \forall z\,(z \in x \leftrightarrow z \in y)))$
(N_a)	$\mathsf{at}(a)$ for all $a \in \mathcal{C}$
$(D_{(a,b)})$	$a \neq b$ for all $a,\, b \in \mathcal{C}$ with $a \not\equiv b$
(S)	$\forall x\, \exists y \forall z\,(z \in y \leftrightarrow z = x)$

Fig. 7. The axiomatization of $\mathcal{A}Set^{-}$

$\mathcal{A}Set^{-} \vdash \forall x, y, z\,(\mathsf{at}(x) \wedge \mathsf{at}(y) \wedge \mathsf{at}(z)) \rightarrow \cup(x,y,z))$.
The predicate symbol $\cap$ is not functional if its first argument does not share any element with its second argument. The predicate symbol $\setminus$ is not functional if its first argument is a subset of its second argument. The predicate symbol $\wp$ always admits infinitely many solutions, as follows from:
$\mathcal{A}Set^{-} \vdash \forall x, y, z\,((\wp(x,y) \wedge \wp(x,z)) \rightarrow \forall w\,(w \in z \wedge w \notin y \rightarrow \mathsf{at}(w)))$;
$\mathcal{A}Set^{-} \vdash \forall x, y, z((\wp(x,y) \wedge y \subseteq z \wedge \forall w\,(w \in z \wedge w \notin y \rightarrow \mathsf{at}(w))) \rightarrow \wp(x,z))$.

Definition 2. *A* system S *of* $\mathcal{L}$ *is a finite set of equations of the form*

$$x = \{y_1, \ldots, y_n\} \qquad or \qquad x = a$$

where $x, y_1, \ldots, y_n \in \mathcal{V}$*, and* $a \in \mathcal{C}$*, such that for each variable* x *in* S*, there is exactly one equation in* S *in which* x *occurs as the l.h.s..*

In our construction we will also make use of the following adaptation (AFA_A) of the anti-foundation axiom:

Every system of $\mathcal{L}$ *has a unique solution.*

We use $\mathcal{A}Set$ to refer to the theory $\mathcal{A}Set^{-}$ together with[3] the axiom (AFA_A).

3.2 A Model of $\mathcal{A}Set$

A rather natural question at this point is whether or not $\mathcal{A}Set$ has a model: the following construction provides one such model.

Let α be an infinite ordinal, let $V_{<\alpha}$ be the set of all well-founded sets of rank less than α, let $V_{=\alpha}$ the set of all well-founded sets of rank α, and let $\mathcal{U}'_\alpha = \mathcal{U} \setminus V_{<\alpha}$ be the universe of all non-well-founded sets (see [1]) not belonging to $V_{<\alpha}$. We use $\mathcal{W}_\alpha$ to denote the quotient $\mathcal{U}'_\alpha / \cong$, where:

$$\forall x, y \in \mathcal{U}'_\alpha (x \cong y \text{ iff } x \notin V_{=\alpha} \wedge y \notin V_{=\alpha} \wedge \forall z \in \mathcal{U}'_\alpha (z \in x \leftrightarrow z \in y))$$

[3] (AFA_A) is not a generalization of (AFA): each system corresponds to a finite graph.

We consider the structure $\mathcal{M}_\alpha$ for the language $\mathcal{L}$ with support $\mathcal{W}_\alpha$ and interpretation function $()'$ defined as follows:

$$
\begin{array}{lll}
x \in' y & \text{iff} & x \in y \\
\cup'(x,y,z) & \text{iff} & \begin{cases} z = x \cup y & \text{if } \exists w\,(w \in' x \vee w \in' y) \\ z \in V_{=\alpha} & \text{otherwise} \end{cases} \\
\cap'(x,y,z) & \text{iff} & \begin{cases} z = x \cap y & \text{if } \exists w\,(w \in' x \wedge w \in' y) \\ z \in V_{=\alpha} & \text{otherwise} \end{cases} \\
\setminus'(x,y,z) & \text{iff} & \begin{cases} z = x \setminus y & \text{if } \exists w\,(w \in' x \wedge w \notin' y) \\ z \in V_{=\alpha} & \text{otherwise} \end{cases} \\
x \subseteq' y & \text{iff} & \begin{cases} x \subseteq y & \text{if } \exists w\,(w \in' x) \\ x \in V_{=\alpha} & \text{otherwise} \end{cases} \\
\wp'(x,y) & \text{iff} & \forall w\,(w \in' y \rightarrow w \in \wp(x) \vee w \in V_{=\alpha}) \wedge \\
 & & \forall w\,((w \in \wp(x) \wedge w \notin V_{=\alpha}) \rightarrow w \in' y) \\
a_i' = w_i & & \text{for all } a_i \in \mathcal{C}, \text{ with } w_i \in V_{=\alpha}, w_i \not\equiv w_j
\end{array}
$$

Theorem 2. *If α is an infinite ordinal, then $\mathcal{M}_\alpha$ is a model of $\mathcal{ASet}$.*

3.3 A Decision Procedure for $\mathcal{C}_2$

In the following we present a decision procedure for the satisfiability problem for the class of purely existentially quantified formulae of the language $\mathcal{L}$, with respect to the theory $\mathcal{ASet}$. First we show how to deal with a restricted class $\mathcal{C}_2$ and then we prove that the whole class of purely existentially quantified formulae can be reduced to disjunctions of formulae in the class $\mathcal{C}_2$. While in Section 2.2 we have presented a deterministic procedure, here, for the sake of simplicity, we define a non-deterministic procedure: instead of considering all the possible disjuncts, each time we make a choice and we add only one of them.

Let φ be a conjunction of literals of the form:

$$x \in y,\ x \notin y,\ \circ(x_1, x_2, y),\ \wp(x, y) \tag{2}$$

where x, x_1, x_2, y are variables and $\circ \in \{\cup, \cap, \setminus\}$. The class $\mathcal{C}_2$ is the class of all conjunctions of literals of the form (2).

The procedure $\mathsf{de}-\wp\mathsf{Boole}$ (Figure 8), allows to decide the finite satisfiability of a formula φ of the class $\mathcal{C}_2$. Two procedures, $\in\!-\wp\mathsf{introduction}$ and

$\mathsf{de}-\wp\mathsf{Boole}(\varphi)$:
$S := FV(\varphi)$;
$\in\!-\wp\mathsf{introduction}(\varphi)$;
$\wp\mathsf{Extensionalize}(\varphi, S)$;
$\in\!-\wp\mathsf{introduction}(\varphi)$.

Fig. 8. The procedure $\mathsf{de}-\wp\mathsf{Boole}$

$\wp\mathsf{Extensionalize}$, are used in $\mathsf{de}-\wp\mathsf{Boole}$; these procedures take a formula as global variable, hence, for instance, the formula φ in the third line is not necessarily the same as the one which occurs in the second line.

The main differences between the function $\in-$introduction and the procedure $\in-\wp$introduction are the eight new rules in $\in-\wp$introduction to deal with the $\wp$ relation symbol and the fact that the rules of $\in-\wp$introduction have been split into *1P* and *2P* in order to control the termination.

The procedure $\in-\wp$introduction calls, inside a repeat-loop, two auxiliary procedures, $\in-$Boolean and $\in-$Powerset, which transform the formula φ into a formula φ' equivalent to φ with respect to the theory $\mathcal{A}Set^{-}$. At the end of each iteration of this loop three sets of variables are taken into consideration to enforce termination. The purpose of the procedures $\in-$Boolean and $\in-$Powerset

```
∈−℘introduction(φ) :
    FV := FV(φ);
    B := false;
    r := 0;
    repeat {
        ψ := φ;
        r := r + 1;
        ∈−Boolean(φ);
        ∈−Powerset(φ);
        New := {(x, y) | x, y ∈ S and (x ∘ y) in φ \ ψ with ∘ =∈, ∉};
        New := New ∪ {k_xy | k_xy ∈ FV(φ) \ FV(ψ)};
        if New ≠ ∅ then r := 0;
        N_n[r] := {V_n | n ∈ FV(φ) \ FV(ψ)};
        N_h[r] := {V_hxn | h_xn ∈ FV(φ) \ FV(ψ)};
        Inc[r] := {(x, y) | x, y ∈ S and if(z ∈ x) in φ, then (z ∈ y) in φ};
        Pict[r] := ⟨N_n[r], N_h[r], Inc[r]⟩;
        if (∃s < r Pict[s] = Pict[r]) then
            B := true;
    until (φ = ψ ∨ B});
    if B then φ := false.
```

Fig. 9. The procedure $\in-\wp$introduction

is to apply the rules presented in Figures 11 and 12. The rules applied by the procedure $\in-$Boolean are almost the same as the ones applied by the function[4] $\in-$introduction presented in Section 2.2. Two new rules have been introduced to deal with literals of the form $y = \{x_1, \ldots, x_h\}$ which, even though not literals of the language, are useful to explain the procedures. Moreover here the first, third, fifth, and seventh rules of *1P* (see Figure 11) need to be applied to both y and z. The rules *2P* applied by $\in-$Powerset deal with literals of the form $\wp(x, y)$. These rules cannot be applied without control, since they can lead to non-termination. The sets $N_n[r]$, $N_h[r]$, and $Inc[r]$ in the procedure $\in-\wp$introduction are used to check that there are no loops in the rules applied by $\in-$Powerset: the set $N_n[r]$ collects the sets of variables into which a new variable n (a variable introduced by the second rule of $2P$) has been added; the set $N_h[r]$ has the same purpose as $N_r[n]$ w.r.t. the variables h introduced by the sixth rule of $2P$; the set $Inc[r]$

[4] Here we use procedures instead of functions because of non-determinism.

collects the pairs (x, y) such that, at step r, x is still a subset of y. As follows from Lemma 3 the condition we impose over these sets ensures the termination, while Lemma 4 proves that $\exists s < r\, Pict[s] = Pict[r]$ leads only to infinite solutions. The second rule of *2P*, see Figure 12, must be applied once for each subset of literals $z_i \in y$ of the formula. For each new variable ℓ introduced by the second or the sixth rule of *2P*, the last two rules of *2P* decide to which variables of φ the variable ℓ belongs. The procedure $\wp$Extensionalize is the analogue of

$\in-$Boolean(φ) :
  repeat {
    $\psi := \varphi$;
    if ($\varphi \overset{1P}{\mapsto} \theta$) then
      $\varphi := \varphi \wedge \theta$; }
    until ($\varphi = \psi$)

$\in-$Powerset(φ) :
  $V := FV(\varphi)$;
  repeat {
    $\psi := \varphi$;
    if ($\varphi(x, y, z, z_i, w, t) \overset{2P}{\mapsto} \theta \wedge x, y, z, z_i, w, t \in V$) then
      $\varphi := \varphi \wedge \theta$; }
  until ($\varphi = \psi$)

Fig. 10. The procedures $\in-$Boolean and $\in-$Powerset

the function Extensionalize of Section 2.2. The new variables introduced by this procedure can always be instantiated to atoms.

Example 3. Each atom $a_i \in \mathcal{C}$ is a solution of the formula $\wp(x, x)$. Also the formula $\wp(x, x) \wedge y \in x$ admits finite solutions. In this case one of the output formulae of our procedure is $\wp(x, x) \wedge y \in x \wedge y = \{y\}$, which is satisfied by the solution of the system $x = \{y\} \wedge y = \{y\}$. The fact that $\wp(x, x)$ admits solutions is a consequence of the fact that it can be the case that some atom belongs to the second argument of a $\wp$-literal. On the other hand, if we consider the formula $\wp(x, x) \wedge y \in x \wedge y \notin y$, there are no models of $\mathcal{A}Set$ in which this formula has finite solutions: $\{y\}^i$ belongs to x, for all $i \in \mathbb{N}$, and $\{y\}^i \neq \{y\}^j$, if $i \neq j$. If we consider the set of rules $1P \cup 2P$ and we apply them to the formula without concern for the newly introduced variables, then we obtain: $y \in x$, hence $n_1 = \{y\} \wedge n_1 \in x$, hence $n_2 = \{n_1\} \wedge n_2 \in x$, and so on. Our procedure detects such loops and it terminates with output false.

Lemma 3. *The procedure* de $- \wp$Boole(φ) *always terminates.*

Proof. We prove that the two procedures $\in-\wp$introduction and $\wp$Extensionalize always terminate.
At the end of each iteration of the repeat-loop in $\in-\wp$introduction, one of the following holds:
1. $\varphi = \psi$; 2. $\exists s < r\, Pict[s] = Pict[r]$; 3. $New \neq \emptyset$; 4. $\forall s < r\, Pict[s] \neq Pict[r]$.

$\cup(y,z,w) \wedge x \in y$	$\mapsto x \in w$
$\cup(y,z,w) \wedge x \in w$	$\mapsto \begin{cases} x \in y & \text{or} \\ x \in z \end{cases}$
$\cup(y,z,w) \wedge x \notin y$	$\mapsto \begin{cases} x \notin z \wedge x \notin w & \text{or} \\ x \in z \wedge x \in w \end{cases}$
$\cup(y,z,w) \wedge x \notin w$	$\mapsto x \notin y \wedge x \notin z$
$\cap(y,z,w) \wedge x \in y$	$\mapsto \begin{cases} x \in z \wedge x \in w & \text{or} \\ x \notin z \wedge x \notin w \end{cases}$
$\cap(y,z,w) \wedge x \in w$	$\mapsto x \in y \wedge x \in z$
$\cap(y,z,w) \wedge x \notin y$	$\mapsto x \notin w$
$\cap(y,z,w) \wedge x \notin w$	$\mapsto \begin{cases} x \notin y & \text{or} \\ x \notin z \end{cases}$
$\setminus(y,z,w) \wedge x \in y$	$\mapsto \begin{cases} x \notin z \wedge x \in w & \text{or} \\ x \in z \wedge x \notin w \end{cases}$
$\setminus(y,z,w) \wedge x \in z$	$\mapsto x \notin w$
$\setminus(y,z,w) \wedge x \in w$	$\mapsto x \in y \wedge x \notin z$
$\setminus(y,z,w) \wedge x \notin y$	$\mapsto x \notin w$
$\setminus(y,z,w) \wedge x \notin w$	$\mapsto \begin{cases} x \notin y & \text{or} \\ x \in z \end{cases}$
$x \in y \quad \wedge x \notin y$	$\mapsto \mathsf{false}$
$y = \{x_1,\ldots,x_h\} \wedge x_i \notin y$	$\mapsto \mathsf{false}$
$\varphi \wedge y = \{x_1,\ldots,x_h\} \wedge x \in y$	$\mapsto \begin{cases} \varphi[x_1/x] & \text{or} \\ \ldots \\ \varphi[x_h/x] \end{cases}$

Fig. 11. The rules *1P* for the procedure $\in-\mathsf{Boolean}$

If either 1. or 2. holds, then $\in-\wp\mathsf{introduction}$ terminates. Condition 3. holds only a finite number of times, since there are at most $2|FV(\varphi)|^2$ literals of the form $x \in y$ or $x \notin y$ and at most $|FV(\varphi)|(|FV(\varphi)|-1)$ variables of the form k_{xy} which can be added to φ. So, there exists m such that for all $k \geq m$ condition 3. does not hold at the kth iteration of the repeat-loop. Condition 4. cannot hold infinitely many times after the m-th iteration of the repeat-loop: $N_n[r], N_h[r] \in \wp(\wp(|FV(\varphi)|))$ and $Inc[r] \in \wp(|FV(\varphi)|^2)$, hence condition 4. does not hold for more than $4^{2^{|FV(\varphi)|}}2^{|FV(\varphi)|^2}$ consecutive iterations of the loop. Hence, after at most $m+4^{2^{|FV(\varphi)|}}2^{|FV(\varphi)|^2}$ iterations, $\in-\wp\mathsf{introduction}$ terminates and $m \leq [2|FV(\varphi)|^2 + |FV(\varphi)|(|FV(\varphi)|-1))][4^{2^{|FV(\varphi)|}}2^{|FV(\varphi)|^2}]$.
The procedure $\wp\mathsf{Extensionalize}(\varphi)$ introduces new variables, but it makes use only of the literals in φ. Hence it must be the case that $x, y, z \in FV(\varphi)$ and $x \not\equiv z$. Therefore, the maximum number of steps and variables introduced by the procedure $\wp\mathsf{Extensionalize}(\varphi)$ is $|FV(\varphi)|^2(|FV(\varphi)|-1)$. □

Example 4. The number of new variables introduced by $\mathsf{de}-\wp\mathsf{Boole}$ blows up. As a matter of fact there are formulae with n variables whose minimal solution involves sets with $\underbrace{2^{2^{2^{\cdots}}}}_{n}$ elements: $\wp(x_0,x_1)\wedge\ldots\wedge\wp(x_{n-1},x_n)\wedge y \in x_0 \wedge \setminus(y,y,y)$.

$\wp(y,x) \wedge z \in x \wedge w \in z$	$\mapsto w \in y$
$\wp(y,x) \wedge (z_i \in y)_{i=1}^{h}$	$\mapsto n = \{z_1,\ldots,z_h\} \wedge n \in x$
$\wp(y,x) \wedge z \notin x \wedge w \in z$	$\mapsto k_{zy} \in z \wedge k_{zy} \notin y$ with $z \not\equiv \{\ldots\}$
$\wp(y,x) \wedge n \notin x \wedge n = \{z_1,\ldots,z_h\}$	$\mapsto \begin{cases} z_1 \notin y \ \text{ or} \\ \ldots \\ z_h \notin y \end{cases}$
$\wp(y,x) \wedge z \notin y \wedge w \in x$	$\mapsto z \notin w$
$t = \{z_1,\ldots,z_h\} \wedge \varphi[x]$	$\mapsto \begin{cases} \varphi[t/x] \ \text{ or} \\ h_{xt} \in x \wedge h_{xt} \notin t \end{cases}$
$\bigwedge_{y \in V_n} n \in y \wedge \bigwedge_{y \in S \setminus V_n} n \notin y$ with $V_n \subseteq S$	
$\bigwedge_{y \in V_{h_{xn}}} h_{xn} \in y \wedge \bigwedge_{y \in S \setminus V_{h_{xn}}} h_{xn} \notin y$ with $V_{h_{xn}} \subseteq S$	

Fig. 12. The rules *2P* for the procedure $\in-$Powerset

```
℘Extensionalize(φ, S) :
    ψ := true;
    repeat {
        σ := ψ;
        if (φ(x,y,z) ↦^SP θ ∧ x,y,z ∈ S) then
            ψ := ψ ∧ θ; }
    until (ψ = σ);
    φ := φ ∧ ψ
```

Fig. 13. The procedure ℘Extensionalize

Lemma 4. *If* false *is the output formula of* $\in-\wp$introduction(φ)*, then* φ *does not have finite solutions.*

Proof. If false has been returned using a rule of the procedure $\in-$Boolean, then φ is not satisfiable.

If false has been returned by the *if* condition of $\in-\wp$introduction, then there is a loop where new variables of type n and h_{xy} have been introduced. This means that there are two steps s and r such that $Pict[r] = Pict[s]$, with:

$n_1^s, \ldots, n_m^s, h_1^s, \ldots, h_\ell^s$ introduced at step s;
$n_1^{s+1}, \ldots, n_o^{s+1}, h_1^{s+1}, \ldots, h_v^{s+1}$ introduced at step $s+1$;
...
$n_1^r, \ldots, n_q^r, h_1^r, \ldots, h_t^r$ introduced at step r.

Moreover, neither new literals of the form $x \in y$ or $x \notin y$ with $x, y \in FV(\varphi)$, nor new variables k_{xy} are introduced in the steps between step s and step r.

We prove that we can reproduce the situation of step $s+1$ at step $r+1$.

Since neither new literals $x \in y$ or $x \notin y$, nor new variables k_{xy} (with $x, y \in FV(\varphi)$) are added, all the variables n's introduced at step $s+1$ have at least one element which has been introduced at step s. If $n_j^{s+1} = \{\ldots, t^s, \ldots\}$ with t^s introduced at step s, then we consider $n_j^{r+1} = \{\ldots, t^r, \ldots\}$, where t^r is an element introduced at step r which has the same characterization V_{t^r} of

$$\left\| x \in y \wedge z \notin y \mapsto \begin{cases} m_{xz} \in x \wedge m_{xz} \notin z & \text{or} \\ m_{zx} \notin x \wedge m_{zx} \in z \end{cases} \right\|$$

Fig. 14. The rule *SP* for the procedure $\wp$Extensionalize

t^s, and we choose for n^{r+1} the set $V_{n_j^{r+1}} = V_{n_j^{s+1}}$. If $h_{xq^s}^{s+1}$ has been introduced during step $s+1$, then q^s has been introduced during step s using a literal of the form $\wp(z, w)$, moreover the literal $q^s \in w$ has been introduced at step s. The introduction of $h_{xq^s}^{s+1}$ means that all the elements of x were elements of q^s at step s, i.e. $(x, z) \in Inc[s]$, since all the elements of q^s were elements of z. From our hypothesis we have $Inc[s] = Inc[r]$, and hence there is an element q^r introduced at step r using the same rule used to introduce q^s which has exactly all the elements of x. Therefore at step $r+1$ we can introduce $h_{xq^r}^{r+1}$ and choose $V_{h_{xq^r}^{r+1}} = V_{h_{xq^s}^{s+1}}$. If we have to introduce other elements at step $r+1$ we can introduce them using one of the characterizations used at step $s+1$. Hence we have that $New_n[r+1] = New_n[s+1]$ and $New_h[r+1] = New_h[s+1]$, since we have introduced enough new variables to cover $New_n[s+1]$ and $New_h[h+1]$, and all the choices have been performed inside these sets. Moreover $Inc[r+1] = Inc[s+1]$, since this follows from the fact that all the new variables introduced are added to the same subsets of variables used at step $s+1$. Hence, if the procedure does not fail, it loops forever and builds a solution which is not finite: all the new variables introduced are necessary to satisfy the formula and at each step we try to identify the new variables n's with the old variables. □

Lemma 5. *Let ψ be the output formula of* $\mathsf{de} - \wp\mathsf{Boole}(\varphi)$. *If in ψ there are no literals of the form* false, *then in every model of $\mathcal{A}Set$ ψ is finitely satisfiable.*

Proof. Let us consider the following sets of equations:

- $x = \{y_1, \ldots, y_h\}$ for each variable x in ψ such that in ψ there is at least one literal of the form $y_j \in x$, where $y_1, \ldots, y_h$ are all the variables which appear in $y_i \in x$ in ψ;
- $x = a_x$ for each variable x in ψ which does not appear in any literal of the form $y \in x$ the equation, where a_x is an atom and $a_x \neq a_y$, if $x \not\equiv y$.

(AFA_A) ensures that the system containing all the above equations has always a (unique) solution $\sigma = [x_1/\bar{x}_1, \ldots, x_n/\bar{x}_n]$. This σ satisfies φ. □

From this lemma we have:

Theorem 3. *Let $\psi_1, \ldots, \psi_m$ be the non-deterministic outputs of the procedure* $\mathsf{de} - \wp\mathsf{Boole}(\varphi)$. *In every model of $\mathcal{A}Set$, the formula φ has a finite solution if and only if there exists $j \in \{1, \ldots, m\}$ such that in ψ_j there are no literals of the form* false.

Each purely existentially quantified formula θ of $\mathcal{L}$ is equivalent to a disjunction of formulae $\theta_1^{C_2} \vee \ldots \vee \theta_n^{C_2}$ in the class $\mathcal{C}_2$. Given a formula θ, let $\theta_1 \vee \ldots \vee \theta_n$ be its disjunctive normal form. The procedure $\wp$REDUCTION presented in Figure 15 allows, given a formula θ_j which is a conjunction of literals of $\mathcal{L}$, to obtain a formula $\theta_j^{C_2}$ in $\mathcal{C}_2$. Hence the procedure $\mathsf{de} - \wp\mathsf{Boole}$, together with the

$x \subseteq y \mapsto \cap(x, y, x)$

$x \neq y \mapsto x \in z \wedge y \notin z$ with $z \in \mathcal{V} \setminus FV(\theta)$

REDUCTION(θ) :
  $S := FV(\theta)$;
  repeat {
    $\psi := \theta$;
    if ($a \in \mathcal{C}$ and a in θ then
      let $x \in \mathcal{V} \setminus FV(\theta)$ in $\theta := \theta[a/x]$; }
  until ($\psi = \theta$);
  for each $\{x_1, x_2\} \subseteq FV(\theta) \setminus S$ do
    $\theta := \theta \wedge x_1 \neq x_2$;
  repeat {
    $\psi := \theta$;
    if ℓ is a literal in θ and $\ell \overset{2T}{\mapsto} p$ then
      $\theta := \theta$ with ℓ replaced from p}
  until ($\psi = \theta$)
  repeat {
    $\psi := \theta$;
    if $x = y$ is a literal in θ then
      $\theta := (\theta \setminus (x = y))[x/y]$}
  until ($\psi = \theta$).

Fig. 15. The rules 2T and the procedure $\wp$REDUCTION

procedure $\wp$REDUCTION, can be used to decide the class of purely existentially quantified formulae of $\mathcal{L}$.

Corollary 1. *If θ is a conjunction of literals of $\mathcal{L}$ and ψ is the output formula of the procedure $\wp$*REDUCTION*, then ψ is in $\mathcal{C}_2$ and θ is satisfiable in a model of $\mathcal{A}Set$ if and only if ψ is satisfiable in a model of $\mathcal{A}Set$.*

4 Conclusion and Future Developments

The procedure we have presented to decide the finite satisfiability of purely existentially quantified formulae with a powerset-like predicate, makes use of rules which hold also in stronger theories, i.e. set theories without atoms, such as $\mathcal{S}et \cup \{(\wp)\}$. It would be interesting to analyze if it is possible to apply the procedure for deciding the finite satisfiability in the context of set theories with extensionality:

- if the output of the procedure is false, then the formula is not finitely satisfiable, even without atoms;
- if the output is different from false and σ is a solution of the output formula in a model of $\mathcal{A}Set$, then it would be interesting to find conditions which ensure that from this solution we can move to a solution without atoms.

When we are working in the context of a well-founded set theory we also have to check that the membership relation is not circular. In this context the decision result presented in [10] ensures that a formula is satisfiable if and only if it is

finitely satisfiable and the result also gives a bound on the rank of a 'minimal' solution.

A possible line of investigation at this point is the analysis of which conditions ensure that formulae with powerset have the finite model property, i.e. they are satisfiable if and only if they are finitely satisfiable.

The formulae with powerset presented in this work can be seen as translations of modal formulae. Hence, we intend to study the relationships between the solutions with atoms that we have presented here, and the modal models of the formulae.

References

1. P. Aczel, *Non-Well-Founded Sets*, CSLI Lecture Notes, vol. 14, Stanford, 1988.
2. J. Barwise and L. Moss, *Vicious circles*, CSLI Public. Notes, vol. 60, Stanford, 1996.
3. J. F. A. K. van Benthem, G. D'Agostino, A. Montanari, and A. Policriti, *Modal deduction in second-order logic and set theory-I*, Journal of Logic and Computation **7** (1997), no. 2, 251–265.
4. ———, *Modal deduction in second-order logic and set theory-II*, Studia Logica **60** (1998), no. 3, 387–420.
5. B. Bollobás, *Combinatorics*, Cambridge University Press, 1986.
6. E. Börger, E. Grädel, and Y. Gurevich, *The classical decision problem*, Springer-Verlag, 1997.
7. D. Cantone, *A fast saturation strategy for set-theoretic tableaux*, Proceedings of TABLEAUX'97 (D. Galmiche, ed.), Springer-Verlag LNAI, vol. 1227, 1997, pp. 122–137.
8. D. Cantone and Zarba C., *A new fast saturation tableau-based decision procedure for an unquantified fragment of set theory*, Proc. of the International Workshop in First-Order Theorem Proving, FTP'98, 1998.
9. D. Cantone, A. Ferro, and E. G. Omodeo, *Computable set theory. vol. 1*, Oxford University Press, 1989, Int. Series of Monographs on Computer Science.
10. D. Cantone, E. G. Omodeo, and P. Ursino, *Transitive venn diagrams with applications to the decision problem in set theory*, Proc. APPIA GULP PRODE '99 Joint Conference on Declarative Programming (M.C. Meo, ed.), 1999.
11. G. D'Agostino, A. Montanari, and A. Policriti, *A set-theoretic translation method for polymodal logics*, Journal of Automated Reasoning **15** (1996), 317–337.
12. B. Dreben and W. D. Goldfarb, *The decision problem. solvable classes of quantificational formulas.*, Addison-Wesley, 1979.
13. A. Ferro, E. G. Omodeo, and J. T. Schwartz, *Decision Procedures for Elementary Sublanguages of Set Theory I. Multilevel Syllogistic and Some Extensions.*, Comm. Pure App. Math. **33** (1980), 599–608.
14. M. Forti and F. Honsell, *Set theory with free construction principles*, Annali Scuola Normale Superiore di Pisa, Cl. Sc. **IV** (1983), no. 10, 493–522.
15. H. R. Lewis, *Unsolvable classes of quantificational formulas.*, Addison-Wesley, 1979.
16. E. G. Omodeo and A. Policriti, *Solvable set/hyperset contexts: I. Some decision procedures for the pure, finite case*, Comm. Pure App. Math. **48** (1995), no. 9-10, 1123–1155, Special Issue in honor of J.T. Schwartz.
17. F. Parlamento, A. Policriti, and K. P. S. B. Rao, *Witnessing Differences Without Redundancies*, Tech. Report 23/93, Università di Udine, 1993.

Tableau Calculus for Only Knowing and Knowing at Most

Riccardo Rosati

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
email: rosati@dis.uniroma1.it

Abstract. We present a tableau method for Levesque's logic of only knowing $\mathcal{OL}$. More precisely, we define a tableau calculus for the logic of only knowing and knowing at most ($\mathcal{ONL}$), which is an extension of $\mathcal{OL}$. The method is based on the possible-world semantics of the logic $\mathcal{ONL}$, and can be considered as an extension of known tableau calculi for modal logic K45. From the technical viewpoint, the main features of such an extension are the explicit representation of "unreachable" worlds in the tableau, and an additional branch closure condition implementing the property that each world must be either reachable or unreachable. Such a calculus allows for establishing the computational complexity of reasoning about only knowing and knowing at most. Moreover, we prove that the method matches the worst-case complexity lower bound of the satisfiability problem in both $\mathcal{ONL}$ and $\mathcal{OL}$.

1 Introduction

Epistemic logics for commonsense reasoning are generally based on the idea of providing systems (agents) with the ability of introspecting on their own knowledge and ignorance [19,23,15]. To this aim, an *epistemic closure assumption* is generally adopted, which informally can be stated as follows: "the logical theory formalizing the agent is a *complete* specification of the agent's knowledge". As a consequence, any fact that is not entailed by such a theory, according to a given semantics, is assumed to be *not known* by the agent.

This paradigm underlies the vast majority of the logical formalizations of nonmonotonic reasoning [15]. Roughly speaking, there exist two different ways to embed such a principle into a logic: (i) by considering a *nonmonotonic* formalism, whose semantics *implicitly* realizes such a "closed" interpretation of the logical theory representing the agent's knowledge; (ii) by representing the closure assumption *explicitly* in the framework of a monotonic logic, suitably extending its syntax and semantics. The first approach has been pursued in the definition of several *modal* formalizations of nonmonotonic reasoning, e.g. McDermott and Doyle's nonmonotonic modal logics [17], Halpern and Moses' logic of minimal epistemic states [9] and Lifschitz's logic of minimal belief and negation as failure [16]. On the other hand, the second approach has been followed by Levesque

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 383–397, 2000.

[15] in the definition of the logic of *only knowing* ($\mathcal{OL}$), obtained by adding a second modal operator, denoted as O and interpreted in terms of "all-I-know", to modal logic K45. Informally, such an interpretation of the modality O is obtained through a maximization of the set of successors of each world satisfying O-formulas.

In a nutshell, the logic of only knowing is a monotonic formalism, in which the modality O allows for an explicit representation of the epistemic closure assumption at the object level (i.e. in the language of the logic), whereas in nonmonotonic formalisms the closure assumption is a meta-level notion. E.g., let φ be a modal formula specifying the knowledge of the agent: in the logic of only knowing, satisfiability of the formula $O\varphi$ in a world w requires maximization of the possible worlds connected to w and satisfying φ; an analogous kind of maximization is generally realized by the preference semantics of nonmonotonic modal logics, by choosing, among the models for φ, only the models having a "maximal" set of possible worlds, where such a notion of maximality changes according to the different proposals.

While the problem of finding a complete axiomatization for the logic $\mathcal{OL}$ has been extensively analyzed [15,7,8], few studies have analyzed the computational aspects of (and/or reasoning methods for) reasoning about only knowing. Indeed, the computational complexity of reasoning about only knowing in the propositional case has been only recently established [24]. The other related studies appearing in the literature concern a fragment of $\mathcal{OL}$ built upon a very restricted subclass of propositional formulas, for which satisfiability is tractable [14], and a computational study of a framework in which only knowing is added to a formal model of *limited* reasoning [12].

In this paper we present a tableau method for the modal propositional fragment of Levesque's logic of only knowing $\mathcal{OL}$. More precisely, we define a tableau calculus for the logic of only knowing and knowing at most ($\mathcal{ONL}$) [15,8], which extends $\mathcal{OL}$ with a third modality N interpreted in terms of "knowing at most". Informally, the meaning of a formula $N\varphi$ in $\mathcal{ONL}$ is "I know at most $\neg\varphi$", which is realized, in terms of a Kripke-style semantics, by imposing that $N\varphi$ is satisfied in a world w if and only if all the worlds satisfying $\neg\varphi$ are connected to w.

The method is strictly based on the possible-world semantics of the logic $\mathcal{ONL}$, and can be considered as an extension of known tableau calculi for the modal logic K45. From the technical viewpoint, the main feature of such an extension is the explicit representation of "unreachable" worlds in the tableau, which allows for a proper handling of N-subformulas in the tableau. However, the explicit representation of unreachable worlds requires an additional branch closure condition in the calculus, which implements the restriction that *each* possible world must be either reachable or unreachable from the initial world. Such a condition is decided in our calculus by means of a second level tableau, which looks for the existence of a world that can be neither reachable nor unreachable from the initial world of any model consistent with the branch of the main tableau. Therefore, if such a world exists (i.e., the second level tableau is open), then the branch of the main tableau is closed.

Our tableau calculus allows for establishing the computational complexity of reasoning about only knowing and knowing at most: in particular, we prove that satisfiability in the modal propositional fragment of $\mathcal{ONL}$ is Σ_2^p-complete, while validity is Π_2^p-complete. We also prove that our method matches the worst-case complexity lower bound of the satisfiability problem in both $\mathcal{ONL}$ and $\mathcal{OL}$, and in this sense it can be considered as "optimal" for such two logics.

We remark that, due to its powerful expressive capabilities, the logic of only knowing is generally considered as a very general formalism for nonmonotonic reasoning. In particular, it has been proven [3] that such a logic is able to naturally embed several well-known nonmonotonic formalisms, i.e., autoepistemic logic, prerequisite-free default logic, disjunctive logic programming under stable model semantics, and circumscription. Due to such embeddings, our method can be also seen as a general, semantic-based calculus for nonmonotonic reasoning.

In the following, we first briefly introduce the modal logic of only knowing and knowing at most $\mathcal{ONL}$. Then, in Section 3 we present the tableau calculus for $\mathcal{ONL}$, and in Section 4 we sketch the correctness proof of the method. In Section 5 we analyze the computational properties of our method, and establish the complexity of reasoning in $\mathcal{ONL}$. Finally, we discuss related work and draw some conclusions in Section 6.

2 The Logic $\mathcal{ONL}$

In this section we briefly recall the formalization of only knowing [15]. We assume that the reader is familiar with the basic notions of modal logic. We recall that K45 denotes the modal logic interpreted on Kripke structures whose accessibility relation among worlds is transitive and euclidean, while modal logic S5 imposes in addition that the relation be serial and reflexive (see e.g. [10] for more details).

We use $\mathcal{L}$ to denote a fixed propositional language with propositional connectives $\wedge, \neg$ (the symbols $\vee, \supset, \equiv$ are used as abbreviations), and whose generic atoms are elements of a countably infinite alphabet $\mathcal{A}$ of propositional symbols. We assume that $\mathcal{A}$ contains the symbols true, false. An interpretation (also called *world*) over $\mathcal{L}$ is a function that assigns a truth value to every atom of $\mathcal{L}$. For each interpretation w, $w(\mathsf{true}) = \mathit{TRUE}$ and $w(\mathsf{false}) = \mathit{FALSE}$. The interpretation of a propositional formula in an interpretation is defined in the usual way. We say that a formula $\varphi \in \mathcal{L}$ is *satisfiable* if there exists an interpretation w such that $w(\varphi) = \mathit{TRUE}$ (which we also denote as $w \models \varphi$).

We use $\mathcal{L}_O$ to denote the modal extension of $\mathcal{L}$ with the modalities K, N and O. We also use $\mathcal{L}_K$ to denote the modal extension of $\mathcal{L}$ with the only modality K, and $\mathcal{L}_N$ to denote the modal extension of $\mathcal{L}$ with the only modality N. We call *O-formula* a formula from $\mathcal{L}_O$ of the form $O\varphi$. Notice that, with respect to [15], we slightly change the language, using the modality K instead of B.

In the following, we call *modal atom* a sentence of the form $K\psi$, $N\psi$ or $O\psi$, with $\psi \in \mathcal{L}_O$. Given $\varphi \in \mathcal{L}_O$, we call *modal atoms of* φ (and denote as $MA(\varphi)$) the set of modal atoms occurring in φ.

The semantics of a formula $\varphi \in \mathcal{L}_O$ is defined in terms of satisfiability in a structure (w, M) where w is an interpretation (called *initial world*) and M is a set of interpretations.

Definition 1. *Let w be an interpretation on $\mathcal{L}$, and let M be a set of such interpretations. We say that a formula $\varphi \in \mathcal{L}_O$ is* satisfied *in (w, M), and write $(w, M) \models \varphi$, iff the following conditions hold:*

1. *if $\varphi \in \mathcal{L}$, then $(w, M) \models \varphi$ iff $w(\varphi) = TRUE$;*
2. *if $\varphi = \neg\varphi_1$, then $(w, M) \models \varphi$ iff $(w, M) \not\models \varphi_1$;*
3. *if $\varphi = \varphi_1 \wedge \varphi_2$, then $(w, M) \models \varphi$ iff $(w, M) \models \varphi_1$ and $(w, M) \models \varphi_2$;*
4. *if $\varphi = K\varphi_1$, then $(w, M) \models \varphi$ iff for every $w' \in M$, $(w', M) \models \varphi_1$;*
5. *if $\varphi = N\varphi_1$, then $(w, M) \models \varphi$ iff for every $w' \notin M$, $(w', M) \models \varphi_1$;*
6. *if $\varphi = O\varphi_1$, then $(w, M) \models \varphi$ iff for every w', $w' \in M$ iff $(w', M) \models \varphi_1$.*

From the above definition, it follows that the modality O can be expressed by means of the modality K and N: precisely, for each $\varphi \in \mathcal{L}_O$ and for each (w, M), $(w, M) \models O\varphi$ if and only if $(w, M) \models K\varphi \wedge N\neg\varphi$.

We say that $\varphi \in \mathcal{L}_O$ is *weakly $\mathcal{ONL}$*-satisfiable if there exists (w, M) such that $(w, M) \models \varphi$. The above semantics is not actually the one originally proposed in [15]: in addition to the above rules, a pair (w, M) must satisfy a maximality condition for the set M, as described below. Such a requirement is due to technical reasons, however, as mentioned in [7], the above, weaker notion of satisfiability is also meaningful.

In the following, $Th(M)$ denotes the set of formulas $K\varphi$ such that $\varphi \in \mathcal{L}_K$ and, for each $w \in M$, $(w, M) \models K\varphi$. Given two sets of interpretations M_1, M_2, we say that M_1 is *equivalent* to M_2 iff $Th(M_1) = Th(M_2)$. A set of interpretations M is *maximal* iff, for each set of interpretations M', if M' is equivalent to M then $M' \subseteq M$. A formula $\varphi \in \mathcal{L}_O$ is *$\mathcal{ONL}$-satisfiable* iff there exists a pair (w, M) such that $(w, M) \models \varphi$ and M is maximal.

We say that a formula $\varphi \in \mathcal{L}_O$ is *$\mathcal{ONL}$-valid* iff $\neg\varphi$ is not $\mathcal{ONL}$-satisfiable. It can be proven [24] that the notions of $\mathcal{ONL}$-satisfiability and weak $\mathcal{ONL}$-satisfiability of a formula from $\mathcal{L}_O$ coincide. We recall, however, that $\mathcal{ONL}$-satisfiability and weak $\mathcal{ONL}$-satisfiability for *infinite* theories are, in general, different (see [15, Section 2.4]).

As for reasoning in $\mathcal{ONL}$, we give the following definition, which allows us to immediately reduce reasoning to unsatisfiability in $\mathcal{ONL}$.

Definition 2. *A formula $\varphi \in \mathcal{L}_O$* entails *a formula $\psi \in \mathcal{L}_O$ in $\mathcal{ONL}$ (denoted as $\varphi \models_{\mathcal{ONL}} \psi$) iff $\varphi \supset \psi$ is $\mathcal{ONL}$-valid.*

Remark. An alternative definition of entailment is given in several studies on epistemic and nonmonotonic modal logics (see e.g. [17, Definition 7.9]). Such a notion is based on the following notion of *validity* of a modal formula in a model: a formula φ is valid in a Kripke model $\mathcal{M}$ iff, for each world w in $\mathcal{M}$, $(w, \mathcal{M}) \models \varphi$. The notion of entailment is then expressed as follows: "ψ is entailed by φ iff ψ is valid in every model in which φ is valid". The two notions are in general

different, and such a difference also holds for the logic $\mathcal{ONL}$. However, since in $\mathcal{ONL}$ the accessibility relation of each interpretation structure is transitive, it can immediately be shown [17, Remark 7.11] that the last notion of entailment can be reduced to the one given in Definition 2, and hence to validity in $\mathcal{ONL}$: ψ is entailed by φ according to the last notion iff $(\varphi \wedge K\varphi) \supset \psi$ is $\mathcal{ONL}$-valid. □

Notice that the above semantics strictly relates the logic $\mathcal{ONL}$ with modal logic $\mathsf{K45}$, since there is a precise correspondence between the pairs (w, M) used in the above definition and $\mathsf{K45}$ models. We recall that, with respect to the satisfiability problem, a $\mathsf{K45}$ model can be considered without loss of generality as a pair $(w, \mathcal{M})$, where w is a world, $\mathcal{M}$ is a set of worlds (possibly empty), w is connected to all the worlds in $\mathcal{M}$, the worlds in $\mathcal{M}$ are connected with each other (i.e. $\mathcal{M}$ is a *cluster*, namely a universal $\mathsf{S5}$ model) and no world in $\mathcal{M}$ is connected to w [17]. Thus, in the following we will refer to a pair (w, M) as a $\mathsf{K45}$ model whose $\mathsf{S5}$ component is M. Notice also that, if $\varphi \in \mathcal{L}_K$, then φ is $\mathcal{ONL}$-satisfiable if and only if it is $\mathsf{K45}$-satisfiable, which is shown by the fact that, if a $\mathsf{K45}$ model (w, M) satisfies such a φ, then there exists a maximal set M' equivalent to M, hence (w, M') satisfies φ. The same property holds for $\varphi \in \mathcal{L}_N$: precisely, it can be immediately proven that φ is $\mathcal{ONL}$-satisfiable if and only if $\varphi[N/K]$ is $\mathsf{K45}$-satisfiable, where $\varphi[N/K]$ is the formula obtained from φ by replacing each modality N with K.

The logic of only knowing $\mathcal{OL}$ [15] simply corresponds to the fragment of $\mathcal{ONL}$ obtained by restricting the language to the subset of $\mathcal{L}_O$ not containing the modality N, i.e., built upon the modalities K and O.

As a combination of the modalities K and N, the interpretation of the O modality is obtained through the maximization of the set of successors of each world satisfying an O-formula. As pointed out e.g. in [13], the meaning of an O-formula $O\varphi$ such that φ is non-modal is intuitive, whereas it is more difficult to understand the semantics of an O-formula with nested modalities.

Example 1. Suppose $\varphi \in \mathcal{L}$. Then, (w, M) is a model for $O\varphi$ iff $M = \{w : w \models \varphi\}$. Hence, prefixing φ with the modality O corresponds to maximizing the set of possible worlds in M, which contains all interpretations consistent with φ. □

Example 2. Suppose $\varphi \in \mathcal{L}$ and φ is not a tautology. Then, the formula $OK\varphi$ is not $\mathcal{ONL}$-satisfiable. Suppose $OK\varphi$ is $\mathcal{ONL}$-satisfiable: then, there exists (w, M) such that $(w, M) \models OK\varphi$. Now, it is easy to see that, by Definition 1, M cannot contain any interpretation w' such that $w' \not\models \varphi$. On the other hand, since φ is not a tautology, there exists such an interpretation w'; moreover, $(w', M) \models K\varphi$, since the interpretation of $K\varphi$ in (w', M) does not depend on the initial world, hence by Definition 1 it follows that $w' \in M$. Contradiction. Hence, $OK\varphi$ is not $\mathcal{ONL}$-satisfiable. On the contrary, $O(K\varphi \wedge \varphi)$ is $\mathcal{ONL}$-satisfiable, under the assumption that φ is satisfiable. □

3 The Tableau Method

In this section we define a tableau calculus for $\mathcal{ONL}$. In a nutshell, the calculus is based on tableau expansion rules which include the standard rules of a tableau for

and-rule: if $\sigma : \psi_1 \wedge \psi_2 \in \mathcal{B}$, and either $\sigma : \psi_1 \notin \mathcal{B}$ or $\sigma : \psi_2 \notin \mathcal{B}$, then add $\sigma : \psi_1$ and $\sigma : \psi_2$ to $\mathcal{B}$;
or-rule: if $\sigma : \psi_1 \vee \psi_2 \in \mathcal{B}$, and neither $\sigma : \psi_1 \in \mathcal{B}$ nor $\sigma : \psi_2 \in \mathcal{B}$, then add either $\sigma : \psi_1$ or $\sigma : \psi_2$ to $\mathcal{B}$;
not-and-rule: if $\sigma : \neg(\psi_1 \wedge \psi_2) \in \mathcal{B}$, and neither $\sigma : \neg\psi_1 \in \mathcal{B}$ nor $\sigma : \neg\psi_2 \in \mathcal{B}$, then add either $\sigma : \neg\psi_1$ or $\sigma : \neg\psi_2$ to $\mathcal{B}$;
not-or-rule: if $\sigma : \neg(\psi_1 \vee \psi_2) \in \mathcal{B}$, and either $\sigma : \neg\psi_1 \notin \mathcal{B}$ or $\sigma : \neg\psi_2 \notin \mathcal{B}$, then add $\sigma : \neg\psi_1$ and $\sigma : \neg\psi_2$ to $\mathcal{B}$;
not-not-rule: if $\sigma : \neg(\neg\psi) \in \mathcal{B}$ and $\sigma : \psi \notin \mathcal{B}$, then add $\sigma : \psi$ to $\mathcal{B}$.

Fig. 1. Propositional tableau expansion rules.

the logic K45 [5,18,6] for handling propositional connectives and the modalities K and N: in particular, K-formulas and N-formulas in each tableau branch are handled by two separate clusters: the first cluster represents the worlds that must be connected to the initial world, i.e., the set of *reachable* worlds, while the second one represents the set of *unreachable* worlds, i.e., the worlds that must be disconnected from the initial world. In addition, a restricted cut rule enforces the presence of each modal subformula of the initial formula in each branch. Finally, a special branch closure condition enforces the restriction that each possible world must be either reachable or unreachable, i.e., must belong either to the first or to the second cluster: such a condition is decided by means of an auxiliary tableau.

The tableau expansion rules are reported in Figure 1 and Figure 2. The tableau calculus deals with *prefixed* formulas of the form $\sigma : \psi$, where σ is a *prefix*, i.e. either the number 0 or a pair of the form $(1, n)$ or $(2, n)$, where n is an integer greater than 0. Given a formula $\varphi \in \mathcal{L}_O$, a *branch* of the tableau for φ is a set of prefixed formulas containing the prefixed formula $0 : \varphi$ and obtained by applying the expansion rules reported in Figure 1 and Figure 2.

Let us now briefly describe the tableau rules. First, the rules reported in Figure 1 are analogous to the usual rules for handling propositional connectives in tableau methods. As for the rules reported in Figure 2 for handling modalities, the K-rule and $\neg K$-rule are standard expansion rules for the K45 modality [5]: in particular, the $\neg K$-rule adds the representation of a new world in the branch, by introducing a new prefix. The N-rule and $\neg N$-rule are exactly the same as the K-rule and $\neg K$-rule, but they affect the second cluster (i.e., the prefixes of the form $(2, n)$) instead of the first one. The $\mathcal{M}$-rule propagates each formula prefixed by a modal operator in the initial world, which simplifies the treatment of such kind of formulas by the other expansion rules.

The O-rule and $\neg O$-rule simply convert O-formulas in terms of the modalities K and N, based on the fact that $O\psi$ is equivalent to $K\psi \wedge N\neg\psi$; therefore, the presence of a formula $O\psi$ in the branch causes the addition of the formulas $K\psi$, $N\neg\psi$ in the branch, while a formula $\neg O\psi$ causes the addition of either $\neg K\psi$ or $\neg N\neg\psi$ in the branch. We remark that the presence of the O-rule and the $\neg O$-rule

cut-rule: if $\psi \in MA(\varphi)$ and neither $0 : \psi \in \mathcal{B}$ nor $0 : \neg\psi \in \mathcal{B}$, then add either $0 : \psi$ or $0 : \neg\psi$ to $\mathcal{B}$;
$\mathcal{M}$-rule: if $\sigma : \mathcal{M}\psi \in \mathcal{B}$, where $\mathcal{M} \in \{K, \neg K, N, \neg N, O, \neg O\}$, and $0 : \mathcal{M}\psi \notin \mathcal{B}$, then add $0 : \mathcal{M}\psi$ to $\mathcal{B}$;
K-rule: if $0 : K\psi \in \mathcal{B}$, and there exists a prefix $(1, n)$ in $\mathcal{B}$ such that $(1, n) : \psi \notin \mathcal{B}$, then add $(1, n) : \psi$ to $\mathcal{B}$;
N-rule: if $0 : N\psi \in \mathcal{B}$, and there exists a prefix $(2, n)$ in $\mathcal{B}$ such that $(2, n) : \psi \notin \mathcal{B}$, then add $(2, n) : \psi$ to $\mathcal{B}$;
$\neg K$-rule: if $0 : \neg K\psi \in \mathcal{B}$, and there is no prefix of the form $(1, n)$ in $\mathcal{B}$ such that $(1, n) : \neg\psi \in \mathcal{B}$, then add $(1, m) : \neg\psi$ to $\mathcal{B}$, where $(1, m)$ is a new prefix in $\mathcal{B}$;
$\neg N$-rule: if $0 : \neg N\psi \in \mathcal{B}$, and there is no prefix of the form $(2, n)$ in $\mathcal{B}$ such that $(2, n) : \neg\psi \in \mathcal{B}$, then add $(2, m) : \neg\psi$ to $\mathcal{B}$, where $(2, m)$ is a new prefix in $\mathcal{B}$;
O-rule: if $0 : O\psi \in \mathcal{B}$, and either $0 : K\psi \notin \mathcal{B}$ or $0 : N\neg\psi \notin \mathcal{B}$, then add $0 : K\psi$ and $0 : N\neg\psi$ to $\mathcal{B}$;
$\neg O$-rule: if $0 : \neg O\psi \in \mathcal{B}$, and neither $0 : \neg K\psi \in \mathcal{B}$ nor $0 : \neg N\neg\psi \in \mathcal{B}$, then add either $0 : \neg K\psi$ or $0 : \neg N\neg\psi$ to $\mathcal{B}$.

Fig. 2. Modal tableau expansion rules.

in our calculus is due to complexity reasons. Indeed, we could just pre-process the input formula, replacing each O-subformula by its definition in terms of K and N: however, the formula thus obtained has in general size exponential in the size of the initial formula.

Finally, the cut-rule implements a restricted form of cut. Specifically, it enforces the presence in $\mathcal{B}$ of each modal subformula occurring in φ or its negation. As we shall see, such a rule is required in order to easily verify (through an auxiliary tableau) whether the set of reachable and unreachable worlds from the initial world is the set of all possible worlds, which corresponds to check whether each world satisfies the constraints, contained in $\mathcal{B}$, concerning either the first or the second cluster.

We now define the notions of closure and completeness of a branch. We say that a branch $\mathcal{B}$ is *completed* if no expansion rule is applicable to $\mathcal{B}$.

As for the notion of closure of a branch, we remark that a completed branch identifies a part of a model for ψ, in the sense that formulas of the form $0 : \psi$ in $\mathcal{B}$ are constraints for the initial world (0) of the model, while a formula of the form $(1, n) : \psi$ is a constraint for a world (n) which can be reached from the initial world, and a formula of the form $(2, m) : \psi$ is a constraint for a world (m) which *cannot* be reached from the initial world. Therefore, since the set of reachable and unreachable worlds from the initial world must be the set of all possible worlds, we have to verify that such constraints allow any world to be either reachable or unreachable. As the following example shows, this is not always the case.

Example 3. Let

$$\varphi = (c \vee K(a \vee b)) \wedge (Na \vee K\neg b)$$

It is easy to verify that the following is a completed branch of the tableau for φ:

$$\mathcal{B} = \{0 : K(a \vee b),\ \ 0 : Na,\ \ 0 : \neg K\neg b,\ \ (1,1) : b,\ \ (1,1) : a \vee b,\ \ (1,1) : a\}$$

Due to the presence of $K(a \vee b)$ in $\mathcal{B}$, any model satisfying $\mathcal{B}$ is such that each world which can be reached from the initial world satisfies either a or b, hence all worlds that satisfy both $\neg a$ and $\neg b$ are not reachable; on the other hand, the presence of Na implies that each unreachable world satisfies a. Consequently, each interpretation satisfying both $\neg a$ and $\neg b$ is neither reachable nor unreachable according to the formulas in $\mathcal{B}$. □

In order to verify that a branch $\mathcal{B}$ allows any world to be either reachable or unreachable from the initial world, we define an auxiliary tableau for $\mathcal{B}$. The auxiliary tableau uses as expansion rules only the propositional expansion rules reported in Figure 1.

Definition 3. *A branch $\mathcal{B}'$ of the auxiliary tableau for $\mathcal{B}$ is a collection of formulas (without prefix) containing the following formulas:*

1. *each formula ψ such that $0 : \psi \in \mathcal{B}$ and $\psi \in MA(\varphi)$;*
2. *$\bigvee_{0:K\psi\in\mathcal{B}} \neg\psi$ if there is at least one formula of the form $0 : K\psi$ in $\mathcal{B}$,* false *otherwise;*
3. *$\bigvee_{0:N\psi\in\mathcal{B}} \neg\psi$ if there is at least one formula of the form $0 : N\psi$ in $\mathcal{B}$,* false *otherwise,*

and obtained by applying the expansion rules reported in Figure 1.

Notice that any such branch contains modal formulas, however they are treated as propositional atoms, i.e., they are not further analyzed.

A branch $\mathcal{B}'$ of the auxiliary tableau is completed if no propositional expansion rule is applicable to $\mathcal{B}'$, and is open if and only if (i) there is no pair of formulas in $\mathcal{B}'$ of the form ψ and $\neg\psi$; (ii) the formula false does not belong to $\mathcal{B}'$. The auxiliary tableau for $\mathcal{B}$ is *open* if it has at least one open completed branch, otherwise it is *closed*.

Informally, the auxiliary tableau for a branch $\mathcal{B}$ of the initial tableau tries to identify a world which cannot be neither reachable nor unreachable from the initial world of any model consistent with the branch $\mathcal{B}$. If no such world exists (i.e., the tableau is closed), then the branch $\mathcal{B}$ is open, since it identifies a model for the initial formula φ.

Example 4. We now present an auxiliary tableau for the tableau of the formula φ of Example 3. Let us start from the following branch $\mathcal{B}$ (reported in Example 3) of the tableau for φ:

$$\mathcal{B} = \{0 : K(a \vee b),\ \ 0 : Na,\ \ 0 : \neg K\neg b,\ \ (1,1) : b,\ \ (1,1) : a \vee b,\ \ (1,1) : a\}$$

According to Definition 3, the auxiliary tableau for $\mathcal{B}$ starts with the following formulas:

$$\{K(a \vee b),\ \ Na,\ \ \neg K\neg b,\ \ \neg(a \vee b),\ \ \neg a\}$$

By applying the expansion rules of Figure 1, we obtain the following (unique) completed branch $\mathcal{B}'$:

$$\mathcal{B}' = \{K(a \vee b),\ \ Na,\ \ \neg K\neg b,\ \ \neg(a \vee b),\ \ \neg a,\ \ \neg b\}$$

Such a branch is open, and identifies a world (containing both $\neg a$ and $\neg b$) which is neither reachable nor unreachable from the initial world of any model consistent with the prefixed formulas in the branch $\mathcal{B}$ of the initial tableau for φ. We thus conclude that the branch $\mathcal{B}$ is not consistent with the condition that each world must be either reachable or unreachable from the initial world. □

We can now state the closure conditions for a branch $\mathcal{B}$ of the initial tableau.

Definition 4. *A completed branch $\mathcal{B}$ for $\varphi \in \mathcal{L}_O$ is* open *if and only if each of the following conditions holds:*

1. *there is no pair of prefixed formulas in $\mathcal{B}$ of the form $\sigma : \psi$ and $\sigma : \neg\psi$;*
2. *there exists no prefix σ such that the formula $\sigma :$ false belongs to $\mathcal{B}$;*
3. *the auxiliary tableau for $\mathcal{B}$ is closed, i.e., no world is neither reachable nor unreachable.*

The tableau for φ is *open* if it has at least one open completed branch, otherwise it is *closed*.

Example 5. We now apply the above defined tableau method to the formula

$$\varphi = c \wedge (K(a \vee b) \wedge (Oa \vee N\neg b))$$

It is immediate to see that each branch of the tableau for φ contains the following set of signed formulas S, obtained by two applications of the and-rule of Figure 1 to the initial formula $0 : c \wedge (K(a \vee b) \wedge (Oa \vee N\neg b))$:

$$S = \{0 : c \wedge (K(a \vee b) \wedge (Oa \vee N\neg b)),\ \ 0 : c,\ \ 0 : K(a \vee b),\ \ 0 : Oa \vee N\neg b\}$$

By applying the rules reported in Figure 1 and Figure 2, we obtain the following completed branches of the tableau for φ:

$$\mathcal{B}_1 = S \cup \{0 : Oa,\ \ 0 : Ka,\ \ 0 : N\neg a,\ \ 0 : N\neg b\}$$
$$\mathcal{B}_2 = S \cup \{0 : Oa,\ \ 0 : Ka,\ \ 0 : N\neg a,\ \ 0 : \neg N\neg b,\ \ (2,1) : b,\ \ (2,1) : \neg a\}$$
$$\mathcal{B}_3 = S \cup \{0 : N\neg b,\ \ 0 : \neg Oa,\ \ 0 : \neg Ka,\ \ (1,1) : \neg a,\ \ (1,1) : a \vee b,\ \ (1,1) : a\}$$
$$\mathcal{B}_4 = S \cup \{0 : N\neg b,\ \ 0 : \neg Oa,\ \ 0 : \neg Ka,\ \ (1,1) : \neg a,\ \ (1,1) : a \vee b,\ \ (1,1) : b\}$$
$$\mathcal{B}_5 = S \cup \{0 : N\neg b,\ \ 0 : \neg Oa,\ \ 0 : \neg N\neg a,\ \ (2,1) : a,\ \ (2,1) : \neg b\}$$

E.g., branch $\mathcal{B}_1$ is obtained by applying the or-rule to $0 : Oa \vee N\neg b$ and choosing Oa, then applying the O-rule, thus adding $0 : Ka,\ \ 0 : N\neg a$, and finally applying the cut-rule to $N\neg b$ choosing $0 : N\neg b$.

It is immediate to see that branch $\mathcal{B}_3$ is closed, since both $(1,1) : \neg a$ and $(1,1) : a$ belong to $\mathcal{B}_3$. For each remaining branch, we have to construct its auxiliary tableau in order to determine whether such a branch is open or closed. We obtain the following starting set of formulas $Aux(\mathcal{B}_i)$ for the auxiliary tableau of each branch $\mathcal{B}_i$:

$$Aux(\mathcal{B}_1) = \{Oa,\ \ Ka,\ \ N\neg a,\ \ N\neg b,\ \ \neg a \vee (\neg(a \vee b)),\ \ a \vee b\}$$
$$Aux(\mathcal{B}_2) = \{Oa,\ \ Ka,\ \ N\neg a,\ \ \neg N\neg b,\ \ \neg a \vee (\neg(a \vee b)),\ \ a\}$$
$$Aux(\mathcal{B}_4) = \{N\neg b,\ \ \neg Oa,\ \ \neg(a \vee b),\ \ b\}$$
$$Aux(\mathcal{B}_5) = \{N\neg b,\ \ \neg Oa,\ \ \neg(a \vee b),\ \ b\}$$

By applying the propositional expansion rules reported in Figure 1 we immediately obtain that:

1. the auxiliary tableau for $\mathcal{B}_1$ is open, therefore $\mathcal{B}_1$ is closed;
2. the auxiliary tableau for $\mathcal{B}_2$ is closed, due to the presence of the formula a and to the fact that each possible expansion of the formula $\neg a \vee (\neg(a \vee b))$ causes the addition of the formula $\neg a$. Therefore, $\mathcal{B}_2$ is open;
3. the auxiliary tableaux for $\mathcal{B}_4$ and $\mathcal{B}_5$ are closed, due to the presence of the formula b and to the fact that the expansion of the formula $\neg(a \vee b)$ causes the addition of the formula $\neg b$. Therefore, $\mathcal{B}_4$ and $\mathcal{B}_5$ are open.

Consequently, the tableau for φ is open. □

4 Soundness and Completeness

Due to space limitations, in this section we just sketch the soundness and completeness proof of the tableau calculus defined in the previous section.

First of all, we relate satisfiability in $\mathcal{ON}\mathcal{L}$ with the problem of finding Kripke structures constituted by an initial world and two clusters. Such structures correspond to the "extended situations" defined in [8].

Let w be a world and M, M' be sets of worlds. Then, we define satisfiability of a formula $\varphi \in \mathcal{L}_O$ in the structure (w, M, M') as follows:

1. if $\varphi \in \mathcal{L}$, then $(w, M, M') \models \varphi$ iff $w(\varphi) = TRUE$;
2. if $\varphi = \neg\varphi_1$, then $(w, M, M') \models \varphi$ iff $(w, M, M') \not\models \varphi_1$;
3. if $\varphi = \varphi_1 \wedge \varphi_2$, then $(w, M, M') \models \varphi$ iff $(w, M, M') \models \varphi_1$ and $(w, M, M') \models \varphi_2$;
4. if $\varphi = K\varphi_1$, then $(w, M, M') \models \varphi$ iff for every $w' \in M$, $(w', M, M') \models \varphi_1$;
5. if $\varphi = N\varphi_1$, then $(w, M, M') \models \varphi$ iff for every $w' \in M'$, $(w', M, M') \models \varphi_1$;
6. if $\varphi = O\varphi_1$, then $(w, M, M') \models \varphi$ iff for every $w' \in M$, $(w', M, M') \models \varphi_1$, and for every $w' \in M'$, $(w', M, M') \models \neg\varphi_1$.

From the above definition and Definition 1, it immediately follows that a formula $\varphi \in \mathcal{L}_O$ is $\mathcal{ON}\mathcal{L}$-satisfiable iff there exists a structure (w, M, M') such

that: (i) $(w, M, M') \models \varphi$; (ii) $M \cup M'$ is the set of all possible worlds; (iii) $M \cap M' = \emptyset$.

Notably, in [8] it has been proven that the last condition is not necessary, hence the existence of a structure (w, M, M') which satisfies φ and such that $M \cup M'$ is the set of all possible worlds is sufficient to establish $\mathcal{ONL}$-satisfiability of φ. Roughly speaking, this is due to the fact that the alphabet of propositions $\mathcal{A}$ is infinite, which guarantees that a finite formula cannot identify a world, i.e., no formula can impose the presence of a given world w in one of the two clusters M, M'. Therefore, the following property holds.

Lemma 1. *Let $\varphi \in \mathcal{L}_O$. Then, φ is $\mathcal{ONL}$-satisfiable iff there exists a structure (w, M, M') such that the following conditions hold:*

1. $(w, M, M') \models \varphi$;
2. $M \cup M'$ *is the set of all possible worlds.*

In the following, we call a branch $\mathcal{B}$ *weakly open* if and only if there is no pair of prefixed formulas in $\mathcal{B}$ of the form $\sigma : \psi$ and $\sigma : \neg\psi$, and there exists no prefix σ such that the formula $\sigma : \mathsf{false}$ belongs to $\mathcal{B}$. That is, we weaken the notion of open branch by discarding condition 3 in Definition 4.

The following property can be proven by easily extending the soundness and completeness proof of known tableau methods for the logic $\mathsf{K45}$ [18].

Lemma 2. *Let $\varphi \in \mathcal{L}_O$. Then, there exists a completed, weakly open branch of the tableau for φ iff there exists a structure (w, M, M') such that $(w, M, M') \models \varphi$.*

Therefore, the existence of a weakly open branch implies the existence of a structure (w, M, M') satisfying φ. However, this is not enough to imply that φ is $\mathcal{ONL}$-satisfiable. From Lemma 1, we have to verify that the structure is such that $M \cup M'$ is the set of all possible worlds.

In the following, given a branch $\mathcal{B}$ and a formula $\psi \in \mathcal{L}_O$, we denote as $\psi(\mathcal{B})$ the formula obtained from ψ as follows:

- replace each subformula of the form $K\xi$ with true if $0 : K\xi \in \mathcal{B}$ and with false if $0 : \neg K\xi \in \mathcal{B}$;
- replace each subformula of the form $N\xi$ with true if $0 : N\xi \in \mathcal{B}$ and with false if $0 : \neg N\xi \in \mathcal{B}$;
- replace each subformula of the form $O\xi$ with true if $0 : O\xi \in \mathcal{B}$ and with false if $0 : \neg O\xi \in \mathcal{B}$.

Definition 5. *Let $\varphi \in \mathcal{L}_O$ and let $\mathcal{B}$ be a completed, weakly open branch of the tableau for φ. We denote as $f_K(\mathcal{B})$ and $f_N(\mathcal{B})$ the following formulas:*

$$f_K(\mathcal{B}) = \bigwedge_{0:K\psi \in \mathcal{B}} \psi(\mathcal{B}) \qquad f_N(\mathcal{B}) = \bigwedge_{0:N\psi \in \mathcal{B}} \psi(\mathcal{B})$$

It is immediate to verify that, due to the cut-rule, $f_K(\mathcal{B})$ and $f_N(\mathcal{B})$ are both propositional formulas. In the following, we say that a structure (w, M, M') is *consistent* with a completed, weakly open branch $\mathcal{B}$ iff the following conditions hold:

1. for each formula ψ such that $0 : \psi \in \mathcal{B}$, $(w, M, M') \models \psi$;
2. for each prefix of the form $(1, n)$ in $\mathcal{B}$ there exists a world w' in M such that, for each formula $(1, n) : \psi \in \mathcal{B}$, $(w', M, M') \models \psi$;
3. for each prefix of the form $(2, n)$ in $\mathcal{B}$ there exists a world w' in M' such that, for each formula $(2, n) : \psi \in \mathcal{B}$, $(w', M, M') \models \psi$.

Lemma 3. *Let $\varphi \in \mathcal{L}_O$ and suppose $\mathcal{B}$ is a completed, weakly open branch of the tableau for φ. Then, each structure (w, M, M') consistent with $\mathcal{B}$ is such that if $w \in M$, then $w \models f_K(\mathcal{B})$, and if $w \in M'$, then $w \models f_N(\mathcal{B})$.*

The following property follows immediately from the definition of auxiliary tableau and soundness and completeness (with respect to propositional satisfiability) of the auxiliary tableau calculus.

Lemma 4. *Let $\varphi \in \mathcal{L}_O$ and let $\mathcal{B}$ be a completed, weakly open branch of the tableau for φ. Then, there exists an open branch $\mathcal{B}'$ of the auxiliary tableau for $\mathcal{B}$ iff there exists a world w such that $w \not\models f_K(\mathcal{B})$ and $w \not\models f_N(\mathcal{B})$.*

The last two lemmas imply the following property.

Lemma 5. *Let $\varphi \in \mathcal{L}_O$. If there exists a completed, weakly open branch of the tableau for φ then there exists a structure (w, M, M') such that $(w, M, M') \models \varphi$ and $M \cup M'$ is the set of all worlds.*

Therefore, from Lemma 1 we are able to prove completeness of the tableau calculus.

Lemma 6. *Let $\varphi \in \mathcal{L}_O$. If there exists an open completed branch of the tableau for φ, then φ is $\mathcal{ONL}$-satisfiable.*

Moreover, soundness follows immediately from Lemma 2, Lemma 3, and Lemma 4. Therefore, the following theorem holds.

Theorem 1. *Let $\varphi \in \mathcal{L}_O$. Then, φ is $\mathcal{ONL}$-satisfiable iff there exists an open completed branch of the tableau for φ.*

5 Complexity Analysis

In this section we analyze the computational aspects of reasoning in $\mathcal{ONL}$, based on our tableau calculus. We start by briefly introducing the complexity classes mentioned in the following (refer e.g. to [11] for further details). All the classes we use reside in the *polynomial hierarchy*. In particular, the complexity class Σ_2^p is the class of problems that are solved in polynomial time by a nondeterministic Turing machine that uses an NP-oracle (i.e., that solves in constant time any problem in NP), and Π_2^p is the class of problems that are complement of a problem in Σ_2^p.

We now analyze the complexity of our tableau method. First of all, it is immediate to verify that:

- any (prefixed) formula appearing either in a branch of the initial tableau for φ or in a branch of an auxiliary tableau has size linear in the size of φ;
- any completed branch of the initial tableau for φ contains a polynomial number (in the size of φ) of prefixed formulas;
- any completed branch of the auxiliary tableau for a branch $\mathcal{B}$ contains a number of formulas which is polynomial in the size of $\mathcal{B}$;
- for each tableau expansion rule, both deciding whether the rule can be applied and applying the rule can be done in polynomial time.

Therefore, each completed branch of the auxiliary tableau for a branch $\mathcal{B}$ can be constructed in time polynomial in the size of φ. Consequently, the method is able to check whether a completed branch $\mathcal{B}$ of the tableau for φ is open in nondeterministic polynomial time. Moreover, each completed branch of the tableau for φ can also be constructed in polynomial time. Thus, our method can be seen as a nondeterministic procedure which is able to decide in polynomial time whether there exists an open branch of the tableau for φ, using an NP-oracle for deciding whether a completed branch is open. Hence, the following theorem holds.

Theorem 2. *The satisfiability problem for a formula in $\mathcal{ONL}$ is in Σ_2^p.*

Moreover, since deciding satisfiability of a formula in $\mathcal{OL}$ is a Σ_2^p-complete problem [24], it immediately follows that satisfiability of a formula in $\mathcal{ONL}$ is Σ_2^p-hard. Consequently, the following property holds.

Theorem 3. *The satisfiability problem for a formula in $\mathcal{ONL}$ is Σ_2^p-complete.*

Therefore, the above theorem states that adding a "knowing at least" modality N to the logic of only knowing $\mathcal{OL}$ does not increase the computational complexity of reasoning.

As immediate corollaries of the above property, we obtain that both validity and entailment in $\mathcal{ONL}$ are Π_2^p-complete problems (see Definition 2).

Notice that, since the satisfiability problem in $\mathcal{ONL}$ is Σ_2^p-hard, a *single* tableau for such a logic should have branches of exponential length (unless $\Sigma_2^p =$ NP). Instead, as shown above, by using two distinct tableaux we are able to decide satisfiability using polynomial space.

Finally, the analysis of our method also points out that satisfiability in $\mathcal{ONL}$ can be decided in nondeterministic polynomial time, if the construction of auxiliary tableaux can be avoided. As an immediate consequence of Definition 3, it follows that the construction of the auxiliary tableau for a branch $\mathcal{B}$ is only needed when $\mathcal{B}$ contains at least one formula of the form $0 : K\psi$ and at least one formula of the form $0 : N\psi$ (otherwise the auxiliary tableau is closed, due to the presence of the formula false in each branch of the auxiliary tableau). Therefore, those branches in which there are either no formulas of the form $0 : K\psi$ or no formulas of the form $0 : N\psi$ should be generated first, which can be realized by avoiding, whenever possible, the addition of modal subformulas of the form $K\psi$, $N\psi$, $O\psi$ in the branch, by a suitable application of branching rules, in particular

the or-rule, not-and-rule, and cut-rule. Observe that a branch can contain either no formulas of the form $0 : K\psi$ or no formulas of the form $0 : N\psi$ even if the initial formula contains occurrences of both K-subformulas and N-subformulas.

6 Related Work and Conclusions

The tableau calculus presented in this paper allows for establishing the computational complexity of reasoning about only knowing and knowing at most: in particular, we have proven that satisfiability in the modal propositional fragment of $\mathcal{ONL}$ is Σ_2^p-complete, while validity is Π_2^p-complete. Therefore, our tableau calculus turns out to be the first "optimal" method for reasoning about only knowing and knowing at most.

Several studies have recently proposed tableau (or related) calculi for nonmonotonic reasoning. E.g., tableaux [22,21] and sequent calculi [1] have been proposed for circumscription and minimal model reasoning, and analogous methods have been defined for default logic [25,2]. Moreover, tableaux for nonmonotonic modal logics have been presented, in particular for autoepistemic logic [20] and for both McDermott and Doyle's and ground nonmonotonic modal logics [4]. None of such methods is able to deal with reasoning about only knowing and knowing at most, since no embedding is known of the logic $\mathcal{ONL}$ (or even the logic $\mathcal{OL}$) into another nonmonotonic formalism.

On the other hand, it is well-known [3] that the logic $\mathcal{ONL}$ is able to naturally embed the major nonmonotonic formalisms, i.e., autoepistemic logic, prerequisite-free default logic, (disjunctive) logic programming under the stable model semantics, and circumscription. Due to such embeddings, our method can be also seen as a general, semantic-based tableau calculus which is able to uniformly cover a large family of nonmonotonic formalisms.

One further development of the present work is towards the analysis of reasoning about only knowing and knowing at most in a first-order setting: in particular, it should be interesting to see whether it is possible to extend the tableau calculus presented in this paper for the modal propositional case to a first-order modal language.

Acknowledgments

This research has been supported by CNR grant 203.15.10.

References

1. P. A. Bonatti and N. Olivetti. A sequent calculus for circumscription. In *Proc. of CSL'97*, vol. 1414 of *LNAI*, pp. 98–114. Springer-Verlag, 1997.
2. P. A. Bonatti and N. Olivetti. A sequent calculus for skeptical default logic. In *Proc. of TABLEAUX'97*, vol. 1227 of *LNAI*, pp. 107–121. Springer-Verlag, 1997.
3. J. Chen. The logic of only knowing as a unified framework for non-monotonic reasoning. *Fundamenta Informaticae*, 21:205–220, 1994.

4. F. M. Donini, F. Massacci, D. Nardi, and R. Rosati. A uniform tableaux method for nonmonotonic modal logics. In *Proc. of JELIA'96*, vol. 1126 of *LNAI*, pp. 87–103. Springer-Verlag, 1996.
5. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Reidel, 1983.
6. R. Goré. Semi-analytic tableaux for modal logics with application to nonmonotonicity. *Logique et Analyse*, 133-134, 1991.
7. J. Y. Halpern and G. Lakemeyer. Levesque's axiomatization of only knowing is incomplete. *Artificial Intelligence*, 74(2):381–387, 1995.
8. J. Y. Halpern and G. Lakemeyer. Multi-agent only knowing. In *Proc. of TARK'96*. Morgan Kaufmann, 1996.
9. J. Y. Halpern and Y. Moses. Towards a theory of knowledge and ignorance: Preliminary report. In K. Apt, ed., *Logic and models of concurrent systems*. Springer-Verlag, 1985.
10. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
11. D. S. Johnson. A catalog of complexity classes. In J. van Leuven, ed., *Handbook of Theoretical Computer Science*, vol. A, chapter 2. Elsevier, 1990.
12. G. Lakemeyer. Limited reasoning in first-order knowledge bases with full introspection. *Artificial Intelligence*, 84:209–255, 1996.
13. G. Lakemeyer. Only knowing in the situation calculus. In *Proc. of KR'96*, pp. 14–25. Morgan Kaufmann, 1996.
14. G. Lakemeyer and H. J. Levesque. A tractable knowledge representation service with full introspection. In *Proc. of TARK'88*, pp. 145–159. Morgan Kaufmann, Los Altos, 1988.
15. H. J. Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42:263–310, 1990.
16. V. Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.
17. W. Marek and M. Truszczyński. *Nonmonotonic Logics – Context-Dependent Reasoning*. Springer-Verlag, 1993.
18. F. Massacci. Strongly analytic tableaux for normal modal logics. In *Proc. of CADE'94*, vol. 814 of *LNAI*, pp. 723–737. Springer-Verlag, 1994.
19. R. C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.
20. I. Niemelä. Decision procedure for autoepistemic logic. In *Proc. of CADE'88*, vol. 310 of *LNCS*, pp. 675–684. Springer-Verlag, 1988.
21. I. Niemelä. A tableau calculus for minimal model reasoning. In *Proc. of TABLEAUX'96*, vol. 1071 of *LNAI*, pp. 278–294. Springer-Verlag, 1996.
22. N. Olivetti. Tableaux and sequent calculus for minimal entailment. *J. of Automated Reasoning*, 9:99–139, 1992.
23. R. Reiter. What should a database know? *J. of Logic Programming*, 14:127–153, 1990.
24. R. Rosati. On the decidability and complexity of reasoning about only knowing. *Artificial Intelligence*, 116:193–215, 2000.
25. C. B. Schwind. A tableaux-based theorem prover for a decidable subset of default logic. In *Proc. of CADE'90*, vol. 449 of LNCS, pp. 528–542. Springer-Verlag, 1990.

A Tableau-Like Representation Framework for Efficient Proof Reconstruction

Stephan Schmitt

Department of Computer Science, Cornell University
Ithaca, NY 14853, USA
steph@cs.cornell.edu

Abstract. We present a tableau-like framework, so-called β-proofs, for a uniform representation of analytic tableaux or matrix proofs. β-proofs suggest split structures for sequent proofs but still violate non-permutabilities of sequent rules. Two operations on β-proofs stepwisely solve these violations while removing all redundant inference steps from the β-proofs. This process provides a general basis for a search-free reconstruction of sequent proofs, independent from concrete proof calculi. Our framework is is uniformly applicable to classical logic and to all the non-classical logics that have a matrix characterization of validity.

1 Introduction

Automated theorem proving in non-classical logics has become important in many branches of Artificial Intelligence and Computer Science. As a result, the resolution principle [25] and the connection method [3], which both have led to efficient theorem provers for classical logic [31,16,4], have been extended to characterizations of logical validity in modal logics, intuitionistic logic, and fragments of linear logic [21,30,12,18]. On this basis, uniform and efficient proof search procedures have been developed for all these logics [22,12,13,19].

In many applications of theorem proving it is not sufficient to show *that* a theorem is valid; for instance, when integrating theorem provers as inference engines into interactive program development systems [14,5] or other problem-oriented applications [6]. The need for further processing, e.g. generating programs from proofs, or a deeper understanding of the proof requires that proof details can be presented in a comprehensible form. Since the efficiency of automated proof methods strongly depends on a machine-oriented and compact characterization of logical validity, the reconstruction of sequent proofs or natural deduction proofs from automatically generated machine proofs becomes necessary.

Proof reconstruction in classical [20,23,24] and non-classical logics [27,28,12, 15] provides uniform procedures for constructing sequent proofs from machine-generated matrix proofs, i.e. within matrix calculi or analytic tableaux. The sequent proof for the input formula is created by traversing its formula tree in an order which respects a *reduction ordering* induced by the matrix proof. It selects an appropriate sequent rule for each visited node by consulting tables

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 398–414, 2000.

that represent the peculiarities of the different logics. At nodes which cause the sequent proof to branch, the reduction ordering has to be divided appropriately and certain redundancies need to be eliminated in order to ensure completeness.

As discussed in [29,15], redundancy elimination is the most crucial aspect during proof reconstruction: performing a complete redundancy deletion ensures a reconstruction process without additional search. This avoids redundant proof steps in the sequent proofs, which are impossible to identify when search is involved. Thus, we obtain an efficient method for a goal oriented construction of the sequent proofs. In order to explore all kinds of redundancy, the *inference steps* of a matrix proof need to be represented, encoding additional information about its internal structure. Usually, the inference steps are determined by the required proof steps for proving the input formula valid.

In this paper we present a uniform framework for the representation of matrix proofs, by generalizing the concept of inference steps. The framework is independent from the selected logic and, in addition, from the underlying proof calculus. Thus, it will be uniformly applicable to various proof methods for classical and non-classical logics, based on matrix or tableau calculi. We will introduce *β-proofs* as a tableau-like representation formalism, and investigate its correspondence to sequent proofs: β-proofs determine possible "split structures" for sequent proofs, but may still violate non-permutabilities of sequent rules. We will develop two elegant operations, which stepwisely solve these violations while removing all redundant inference steps from the β-proofs. For this, an intrinsic relation between β-proofs and the partially constructed sequent proofs will be established. More precisely, the inference steps in a β-proof encode the smallest set of proof steps that are required for all possible completions of the partial sequent proof. We show how this process "approximates" β-proofs towards sequent proofs in order to guide a search-free proof reconstruction.

Using β-proofs within the proof reconstruction approach provides us with a new dimension of uniformity. In addition to the dimension of logic-independent uniformity, our representation framework does neither depend on a particular proof search strategy nor on the proof reconstruction method itself. Thus, β-proofs realize a general interface for building efficient proof reconstruction components and combine them with a variety of concrete proof search procedures.

Section 3 gives a brief summary of matrix characterizations and proof reconstruction in non-classical logics. Section 4 summarizes the requirements for redundancy deletion during proof reconstruction. In Section 5, β-proofs as representation framework are presented. We develop the "approximation" operations and illustrate the integration into the reconstruction process. Section 6 discusses complexity issues and completeness of the refined proof reconstruction method.

2 Preliminaries

Matrix characterizations of logical validity were introduced for classical logic [3] and later extended to intuitionistic and modal logics [30], and fragments of linear logic [12,19]. For the purpose of this paper, it suffices to consider the class $\mathcal{L}$ of logics consisting of classical, intuitionistic, and modal logics as presented in [30].

2.1 Matrix Characterizations for Non-classical Logics

In matrix proofs a formula F is represented by its formula tree $\ll$ whose nodes are called *positions*. Each position x of $\ll$ refers to a unique subformula F_x of F. The root b_0 of $\ll$ represents F itself while its leaves (or *atomic positions*) refer to the atoms of F. Because of the corresponding subformula relation $\ll$ is called the *tree ordering* of F. Each position u is associated with a *polarity* $pol(u) \in \{0,1\}$, a *principal type* $Ptype(u)$, and its operator $op(u)$. The polarity determines whether F_u will appear in the succedent of a sequent proof ($pol(u)$=0) or in its antecedent. $F_u^{pol(u)}$ denotes the *signed formula* at position u. The principal type $Ptype(u)$ is the formula type of F_u according to the tableau classification in [30,11]. In the following, we will only consider the types α, β, and *atom*. Two atomic positions u and v are *β-related* ($v \sim_\beta u$) if their greatest common predecessor in $\ll$ has principal type β; otherwise they are *α-related* ($u \sim_\alpha v$). A *path* p through F is a *maximal* subset of atomic positions, which are pairwisely α-related. A *connection* is a subset $\{c_1, c_2\}$ of a path p such that the atoms F_{c_1} and F_{c_2} have the same predicate symbol but different polarities. It is *complementary* if F_{c_1} and F_{c_2} can be unified by some *combined substitution* $\sigma = \langle \sigma_Q, \sigma_L \rangle$. σ_Q is the usual quantifier substitution while σ_L, encoding non-permutabilities of sequent rules in a non-classical logic $\mathcal{L}$, unifies the prefixes of the connected atoms. The *prefix* of an atom F_u is a string consisting of special positions in $\ll$ between the root b_0 and u. A set of connections $\mathcal{C}$ *spans* a formula F (or is a *spanning mating*) if each path contains a complementary connection $c \in \mathcal{C}$. The substitution σ induces a relation $\sqsubset$ on the positions of $\ll$ such that $(a,u) \in \sqsubset$ iff a occurs in $\sigma(u)$ and $\sigma(u)$ is not a variable. σ is *admissible* if the *reduction ordering* $\lhd = (\ll \cup \sqsubset)^+$ is irreflexive and some global conditions hold (see [30]). The multiple uses of subformulae in a matrix proof are realized by a *combined multiplicity* $\mu = \langle \mu_Q, \mu_L \rangle$ of the positions u in $\ll$. This leads to a uniform characterization of logical validity:

Theorem 1. *A formula F is* valid *wrt. a logic $\mathcal{L}$ iff there exists a multiplicity μ, an admissible substitution σ, and a set of connections $\mathcal{C}$ which spans F.*

See [30] for a proof. For proof reconstruction, we shall use the reduction ordering $\propto^\star$, a slight technical modification of $\lhd$ containing a new root w (see [28]).

Example 1. Consider $F \equiv \Box((P \wedge Q) \vee Q) \wedge (\Box P \vee \Box R) \to \Box(P \wedge Q) \vee \Box R$ in modal logic T. The reduction ordering $\propto^\star$ of a matrix proof for F is shown in Figure 1, left hand side, where the main operator and polarity are written beside each position. The connections $\mathcal{C} = \{\{a_5, a_{16}\}, \{a_{17}, a_6\}, \{a_7, a_{17}\}, \{a_{16}, \bar{a}_{10}\}, \{\bar{a}_{12}, a_{19}\}\}$ span (the 8 paths of) F wrt. some admissible substitution σ, which induces the relation $\sqsubset = \{(a_{15}, \bar{a}_3), (a_{15}, \bar{a}_{10}), (a_{19}, \bar{a}_{12})\}$ (indicated by curved arrows).

2.2 Proof Reconstruction in Non-classical Logics

An algorithm for reconstructing a sequent proof from a matrix proof of a formula F essentially has to traverse the reduction ordering $\propto^\star$ while constructing a sequent rule at each visited position u. We focus on conversion into multiple

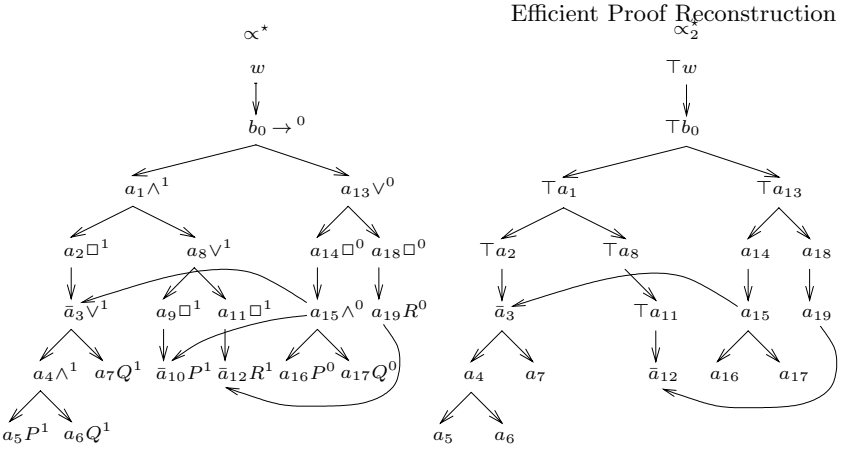


Fig. 1. Reduction orderings $\propto^\star, \propto_2^\star$ for the T-formula F from Examples 1 and 2

conclusion sequent calculi (cf. [11,30]), where a sequent $\Gamma \vdash \Delta$ is described by *associated sets* of signed formulae: $\mathcal{S}_\Delta = \{F_u^0 \mid F_u \in \Delta\}$, $\mathcal{S}_\Gamma = \{F_u^1 \mid F_u \in \Gamma\}$, and $\mathcal{S} = \mathcal{S}_\Delta \cup \mathcal{S}_\Gamma$. The main operator $op(u)$ and polarity $pol(u)$ uniquely describe the sequent rule necessary to reduce the sequent formula $F_u^{pol(u)} \in \mathcal{S}$. The subformulae resulting from this application are determined by the set $succ(u)$ of immediate successors of u in $\ll$. The relation $\sqsubset$ encodes the non-permutabilities of sequent rules in a logic $\mathcal{L}$ and "blocks" certain positions u: rule construction for u will be delayed until all its predecessors wrt. $\sqsubset$ have been visited first.

Traversal of $\propto^\star$. Each position in $\ll$ has to be visited and marked as *solved* if it is not blocked. A position u is *open* (i.e. elegible to be visited next) if its immediate predecessor $pred(u)$ is already solved but u is not. After u has been solved, the set P_o of all open positions is updated to $P_o' = (P_o \setminus \{u\}) \cup succ(u)$. At a β-position u, a sequent proof branches into two independent subproofs. Accordingly, the reduction ordering $\propto^\star$ must be split into suborderings $\propto_1^\star, \propto_2^\star$ and traversal continues separately on each of them. If two solved positions form a complementary connection $c \in \mathcal{C}$, then the conversion of the actual $\propto_i^\star$ terminates with an *axiom*-rule. Proof reconstruction terminates when all branches of the sequent proof have been closed by converting a connection from the matrix proof. P_o is initialized as $P_o = \{b_o\}$ marking the new root w of $\propto^\star$ as solved. Because of the uniformity of the reconstruction procedure the technical details are subtle. We refer to [28,15,12,26] for a complete and algorithmic presentation.

Splitting at β-positions. The main modification of the reduction ordering during the reconstruction process occurs, when reaching a β-position u and $\propto^\star$ has to be divided into $\propto_1^\star$ and $\propto_2^\star$. If $succ(u) = \{u_1, u_2\}$ in $\ll$, then $F_{u_1}^{pol(u_1)}$ will move to the left branch in the sequent proof and $F_{u_2}^{pol(u_2)}$ to the right one. Since the set of open positions P_o encodes the actual sequent, we update $P_o^i = (P_o \setminus \{u\}) \cup \{u_i\}$. Formally, splitting is based on the following definitions:

Definition 1. *For a position u of $\propto^\star$, let $\ll^u$ be the subtree ordering with root u and position set $pos(u)$, including $(pred(u), u) \in \ll^u$. The* restriction *of $\propto^\star$ is defined as $t^u := \ll^u \cup \sqsubset^u$, where $\sqsubset^u := \{(u_1, u_2) \in \sqsubset \mid u_1 \in pos(u) \vee u_2 \in pos(u)\}$. If $\mathcal{C}$ is the connection set of $\propto^\star$, then $\mathcal{C}^u := \{\{c_1, c_2\} \in \mathcal{C} \mid c_1 \in pos(u)\}$.*

Definition 2. *Let $Ptype(u) = \beta$ and $succ(u) = \{u_1, u_2\}$. The* β-split *of $\propto^\star$ at u is defined by β-split$(\propto^\star, u) := [\propto_1^\star, \propto_2^\star]$, where $\propto_1^\star = \propto^\star \setminus t^{u_2}$ and $\propto_2^\star = \propto^\star \setminus t^{u_1}$. For $i \neq j \in \{1, 2\}$, we have $\sqsubset_i := \sqsubset \setminus \sqsubset^{u_j}$, $\ll_i := \ll \setminus \ll^{u_j}$, and $\mathcal{C}_i := \mathcal{C} \setminus \mathcal{C}^{u_j}$.*

The operation *split*$(\propto^\star, u)$ developed in [28] first divides $\propto^\star$ using β*-split* and second, deletes components from the $\propto_i^\star$ that are no longer relevant in the corresponding sequent subproofs. This improves the efficiency of the reconstruction process and is necessary to ensure its completeness in non-classical logics $\mathcal{L}$.

3 Redundancy Deletion During Proof Reconstruction

The problem of redundancy deletion during proof reconstruction was extensively discussed in [28,29,15]. We will briefly summarize the main concepts and results from this discussion in order to motivate the framework introduced in Section 4.

The reduction ordering $\propto^\star$ is *not complete* wrt. rule non-permutabilities of sequent rules, due to the deletion of sequent formulae in sequent calculi for non-classical logics. In [28], we have introduced the concept of *wait*-labels, which are dynamically assigned to special positions during traversal and thus, complete $\propto^\star$ for all logics $\mathcal{L}$. In order to guarantee completeness of proof reconstruction, "outdated" *wait*-labels need to be removed by performing a *redundancy deletion* after β-splits, i.e. the elimination of redundant subrelations from the $\propto_i^\star$.

The purity principle. The central deletion operation in $\propto^\star$ is a generalized *purity reduction* (cf. [3]): an atomic position u of $\propto^\star$ is called *pure* if it is not connected. The extension of purity to subtrees in $\propto^\star$ is defined as follows:

Definition 3. *A position k with $|succ(k)| \geq 2$ is a* β-node *if $Ptype(k) = \beta$ and otherwise a* Θ-node. *The greatest predecessor k of a position u in $\ll$ with $|succ(k)| \geq 2$ is called the* associated node *of u. We write $k \ll^\beta u$ if k is a β-node and $k \ll^\Theta u$ otherwise.*

Definition 4. *Let $P_r = \{\, b \mid succ(b) = \emptyset \wedge \forall c \in \mathcal{C}.\, b \notin c \,\}$ be the set of pure leaf positions in $\propto^\star$. Let $succ_j^+(u) := \{succ_j(u)\} \cup succ^+(succ_j(u))$ where $succ^+(u)$ is the set of* all *successors of u in $\ll$ and $succ_j(u)$ is a selection function with $succ_j(u) = u_j$ if $succ(u) = \{u_1, .., u_n\}$. The (β, Θ)-purity reduction is defined as:*

(β, Θ)*-purity*$(\propto^\star, \mathcal{C})$ =
  while $P_r \neq \emptyset$
    select $b \in P_r$; $P_r := P_r \setminus \{b\}$; *let* k *be the associated node of* b
    if $k \ll^\beta b$ *with* $succ(k) = \{k_1, k_2\}$ **then**
      $\propto^\star := \propto^\star \setminus (t^{k_1} \cup t^{k_2})$; $\mathcal{C} := \mathcal{C} \setminus (\mathcal{C}^{k_1} \cup \mathcal{C}^{k_2})$; % β-purity
      recompute $P_r := \{b \mid succ(b) = \emptyset \wedge \forall c \in \mathcal{C}.\, b \notin c\}$
    else compute s *such that* $b \in succ_s^+(k)$; $\propto^\star := \propto^\star \setminus t^{succ_s(k)}$ % Θ-bpurity

The decomposition problem. After *β-split* and (β,Θ)*-purity* one of the resulting $\propto_i^\star$ may consist of several "isolated" subrelations, which do not have connections *between* each other. Then, only *one* of these subrelations is proof relevant. But the purity principle does not apply, since all leaves in $\propto_i^\star$ are connected and hence, assumed to be relevant. Combined with assigned *wait*-labels in $\propto_i^\star$, this situation may lead to a deadlock of the whole reconstruction process.

Definition 5. *A reduction ordering* $\propto^\star$ *is called a* deadlock *iff* $P_r = \emptyset$ *and for all* $u \in P_o$*, either* $(v,u) \in \sqsubset$ *for some unsolved position* v*, or* $wait[u] = \top$ *holds.*

Example 2. Consider the T-formula F and its matrix proof $\propto^\star$ from Example 1. We start proof reconstruction by solving the positions w, b_0, a_1, a_{13}, a_2, and split at the β-position a_8. The resulting subrelation $\propto_2^\star$ is shown in Figure 1, right hand side, where a_{11} has additionally been solved (marked with $\top$). We obtain the open positions $P_o^2 = \{\bar{a}_3, \bar{a}_{12}, a_{14}, a_{18}\}$ with $wait[a_{14}] = wait[a_{18}] = \top$ in logic T, and $\bar{a}_3, \bar{a}_{12}$ are blocked by $\sqsubset$. Since (β,Θ)*-purity* is not applicable in $\propto_2^\star$, proof reconstruction has run into a deadlock. But $\propto_2^\star$ has also been *decomposed* into two "isolated" subrelations $t_1 = \{t^{\bar{a}_3}, t^{a_{14}}\}$ and $t_2 = \{t^{\bar{a}_{12}}, t^{a_{18}}\}$. Since the relations t_1 and t_2 are α-related, only t_2 is relevant and should be selected in order to solve the deadlock and hence, to complete proof reconstruction.

Deadlocks occur in intuitionistic logic and in all modal logics considered in [28,15]. Selecting the appropriate isolated subrelation from $\propto^\star$ is called the *decomposition problem in* $\propto^\star$, which has been formalized in [29,26]. The following lemma characterizes completeness of proof reconstruction for all logics $\mathcal{L}$:

Lemma 1. *If* $\propto^\star$ *is a deadlock then there is a decomposition problem in* $\propto^\star$*.*

See [26] for a proof. Completeness of proof reconstruction can only be guaranteed if the decomposition problem can be either avoided or solved. A solution of the decomposition problem consists of establishing a selection function f_d, which chooses the only relevant subrelation from $\propto^\star$. Such a solution is called *adequate* if f_d can be realized without any additional search.

4 Proof Representations for Efficient Proof Reconstruction

Adequate solutions for the decomposition problem require to explore deeper forms of redundancy during proof reconstruction. For this, additional information about the internal structure of a matrix proof has to be provided, which is reflected by its *inference steps*. With inference steps we mean the steps that are necessary to prove a mating to be spanning for $\propto^\star$. Unlike a mating, inference steps additionally represent the multiple use of connections from the set $\mathcal{C}$. This enables us to refine the *split* operation such that decomposition problems, deadlocks, and hence, search during the reconstruction process will be avoided.

4.1 β-Proofs: A Representation Framework for Matrix Proofs

We present *β-proofs*, a tableau-like framework for representing the inference steps of matrix proofs. A β-proof for $\propto^\star$ determines an ordering on β-splits

for traversing the reduction ordering $\propto^\star$. Thus, it encodes a *possible* split structure for a sequent proof. However, the rule non-permutabilities of sequent rules (encoded into $\propto^\star$) are not respected, and the deletion of subformulae (as in sequent calculi) is suppressed. β-proofs are uniformly defined for all logics $\mathcal{L}$ under consideration and are *independent* from any concrete proof search strategy. All (sub)formulae are represented by their positions from the matrix proof $\propto^\star$.

Definition 6. *Let $\propto^\star$ be a matrix proof with connection set $\mathcal{C}$ and let u be a position in $\propto^\star$. We define the* α-layer $\alpha(u)$ *of u as:*

$$\alpha(u) = \begin{cases} \{u\}, & \text{if } \mathit{Ptype}(u) = \beta \text{ or } u \text{ is atomic} \\ \bigcup\{\alpha(u_i) \mid u_i \in \mathit{succ}(u)\}, & \text{otherwise} \end{cases}$$

By a sequence *we mean a concatenation of elements $y_1 \cdot y_2 \cdots y_m$, where the y_i are either positions or α-layers.*

Definition 7. *Let $\propto^\star$ be a matrix proof in a logic $\mathcal{L}$ with connection set $\mathcal{C}$ and let b_0 be the original root of $\ll$. A* β-structure $B_{\propto^\star}$ *for $\propto^\star$ is defined as follows:*

(i) $\alpha(b_0)$ *is a* β-structure*, consisting of the branch* $\alpha(b_0)$*.*
(ii) Let $B_{\propto^\star}$ *the be actual* β-structure*. Let b be a branch in $B_{\propto^\star}$, containing the α-layers $\alpha(u_1), \ldots, \alpha(u_n)$.*
  (a) β-expansion*: Let $\alpha(u_n)$ be the leaf of b. Select a β-position $v \in \alpha(u_i)$, $i \in \{1, \ldots, n\}$ which has not been selected before on b. Replace b by the two new branches $b \cdot v \cdot \alpha(v_1)$ and $b \cdot v \cdot \alpha(v_2)$, where $\mathit{succ}(v) = \{v_1, v_2\}$. The result of this replacement is again a* β-structure*.*
  (b) atom-expansion*: Select an atom $v \in \alpha(u_i)$, $i \in \{1, \ldots, n\}$. Replace b by $b \cdot v$. The result of this replacement is again a* β-structure*.*

A branch b in a β-structure $B_{\propto^\star}$ *is called* leaf-connected with (connections) C *iff there exists a non-empty set $C \subseteq \mathcal{C}$ such that:*

(a) $b = b' \cdot \alpha(u_n) \cdot b''$, where $\alpha(u_n)$ is the lowmost α-layer on b and b'' consists of atom expansions only.
(b) For every $\{c_1, c_2\} \in C$ it holds: $c_1 \in \alpha(u_i)$ on b and either $c_2 \in \alpha(u_n)$, or c_2 is an atom-expansion on b''

A branch b is called closed *iff it is leaf-connected with C. A* β-structure $B_{\propto^\star}$ *is called* proof representation *or* β-proof *of $\propto^\star$ iff all its branches are closed.*

A branch b in a β-structure $B_{\propto^\star}$ can be extended by β-expansions as long as its leaf is still an α-layer. The concept of leaf-connected branches with connections $C \subseteq \mathcal{C}$ allows to close a branch b in $B_{\propto^\star}$ with multiple connections. This reflects redundancies in matrix proofs, when several connections are used to make a single path complementary. Leaf-connections $c \in C$ ensure that *only* the actual branch b will be closed. Atom-expansions become necessary to preserve this property when permuting β-expansions in $B_{\propto^\star}$ (see Section 4.2).

Example 3. Reconsider the formula F from Example 1, which uses the connections $\mathcal{C} = \{\{a_5, a_{16}\}, \{a_{17}, a_6\}, \{a_7, a_{17}\}, \{a_{16}, \bar{a}_{10}\}, \{\bar{a}_{12}, a_{19}\}\}$ for its matrix proof $\propto^\star$ (see Figure 1). A β-structure $B_{\propto^\star}$ is shown in Figure 2: We start with

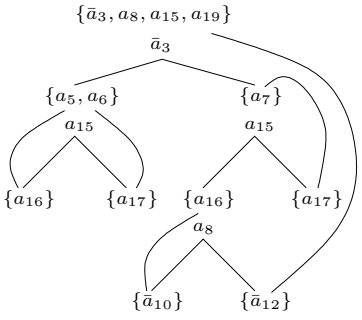


Fig. 2. β-proof $B_{\propto^\star}$ for the formula F from Example 3

$\alpha(b_0) = \{\bar{a}_3, a_8, a_{15}, a_{19}\}$ and the β-expansion at $\bar{a}_3$, obtaining the two α-layers $\{a_5, a_6\}$ and $\{a_7\}$. The β-expansion at a_{15} is needed on both resulting branches, whereas the β-expansion at a_8 is performed on the right branch only. All branches are closed with a singleton set $\{c\} \subseteq \mathcal{C}$. Thus, $B_{\propto^\star}$ is a β-proof for $\propto^\star$.

Definition 8. *Let $B_{\propto^\star}$ be a* β-structure *and b be a branch in $B_{\propto^\star}$.*

- *Every (possibly empty) sequence b' on b is called a* partial branch *of b. If $b = b' \cdot b''$ for two sequences b' and b'', then b' is called a* subbranch *of b.*
- *Let b' be a sequence. The set $S_{b'} = \{b'' \mid b' \cdot b'' \text{ is a branch in } B_{\propto^\star}\}$ of partial branches is called a* substructure *in $B_{\propto^\star}$. We write $b' \cdot B'$ to denote the substructure B', together with its common subbranch b'. We also refer to $b' \cdot B'$ as substructure when it is clear from the context.*
- *Let $b' \cdot \alpha(v) \cdot B''$ be a substructure in $B_{\propto^\star}$. Then, purity is defined as follows:*
  - *(i) An atom $u \in \alpha(v)$ is* pure *if for all $b'' \in S_{b' \cdot \alpha(v)}$ it holds: $u \notin \{c_1, c_2\}$ for every $\{c_1, c_2\} \in C$, where $C \subseteq \mathcal{C}$ closes the branch $b' \cdot \alpha(v) \cdot b''$ in $B_{\propto^\star}$.*
  - *(ii) A β-position $u \in \alpha(v)$ is* pure *if there exists no β-expansion at u in B''.*
  - *(iii) The α-layer $\alpha(v)$ is* pure *if every $u \in \alpha(v)$ is pure.*
  - *(iv) Let $b' \cdot v \cdot \alpha(v_1) \cdot B_1''$ and $b' \cdot v \cdot \alpha(v_2) \cdot B_2''$ be two substructures in $B_{\propto^\star}$, where v denotes a β-expansion and $succ(v) = \{v_1, v_2\}$. The β-expansion at v is called* pure *if $\alpha(v_1)$ or $\alpha(v_2)$ is pure.*

Every substructure $b \cdot B$ of a β-proof $B_{\propto^\star}$ is called a *β-subproof* since all branches in $b \cdot B$ are closed. $B_{\propto^\star}$ can be expressed as the β-subproof $\alpha(b_0) \cdot B$.

Definition 9. *Let $b \cdot B$ be a β-subproof of a β-proof $B_{\propto^\star}$. Let $\#\beta(b \cdot B)$ denote the number of β-expansions in B and $n = \#\beta(b \cdot B)+1$ be the number of branches in $b \cdot B$. Let branch i be leaf-connected with connections $C_i \subseteq \mathcal{C}$, $i = 1, \ldots, n$, and $\#Ax(b \cdot B) = \sum_{i=1}^{n} |C_i|$ denote the number of* axioms *in $b \cdot B$. Then,*

- *$b \cdot B$ is called* minimal *iff $\#Ax(b \cdot B) = \#\beta(b \cdot B) + 1$;*
- *$b \cdot B$ is called* optimal *iff it is minimal and no pure β-expansions occur in B.*

In minimal β-subproofs $b \cdot B$ it holds $|C_i| = 1$, for all $i = 1, \ldots, n$. The following theorem states the fundamental connection between β-proofs as representation framework and decomposition problems in $\propto^\star$ (see [26] for the proof).

Theorem 2. *Let $b \cdot B$ be the β-subproof representing the actual reduction ordering $\propto^\star$ during proof reconstruction. If $b \cdot B$ is optimal, then no decomposition problem occurs in $\propto^\star$.*

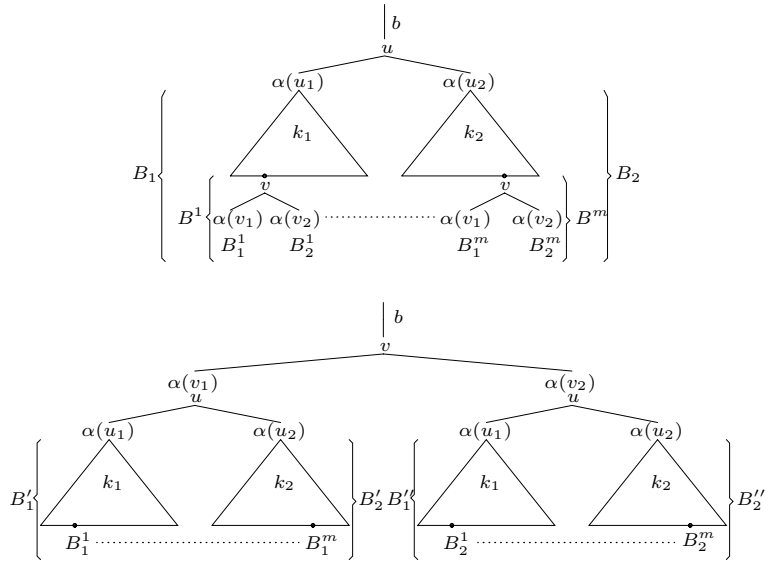


Fig. 3. Permutation of m β-expansions at v above the β-expansion at u

4.2 Approximating β-Proofs Towards Reduction Orderings

Usually, the order of β-expansions in a β-proof is not admissible for a sequent proof, since the reduction ordering $\propto^\star$ may be violated. Proof reconstruction can thus be redefined as the continuous "approximation" of a β-proof $B_{\propto^\star}$ towards an admissible sequent proof, respecting $\propto^\star$. This will be realized by rearranging the β-expansions in an optimal β-subproof $b\cdot B$ that represents the actual $\propto^\star$. We formalize two operations for this purpose, which, in addition, remove *all* redundant inference steps from the β-subproof. The latter will be essential for guiding a maximal redundancy deletion during proof reconstruction (see Section 4.3).

Split permutations in β-proofs. In the following, we represent a substructure B in a β-subproof $b\cdot B$ as $B = u\cdot\alpha(u_1)\cdot B_1 \mid u\cdot\alpha(u_2)\cdot B_2$, where u is the topmost β-expansion in B with $succ(u) = \{u_1, u_2\}$ and B_1, B_2 are substructures. This representation is intuitively extendable to multiple consecutive β-expansions.

The first operation is a split permutation in a β-subproof $b\cdot B$ representing $\propto^\star$. Let v be the β-position to be solved next during traversal of $\propto^\star$. If the topmost β-expansion in B is applied to a position $u \neq v$, then *all* occurrences of β-expansions at v in B have to be permuted *over* the β-expansion at u, i.e. the root of B. The split permutation is defined as follows:

Definition 10. *Let $b\cdot B$ be a β-subproof containing a topmost β-expansion at u with $succ(u) = \{u_1, u_2\}$ and $B = u\cdot\alpha(u_1)\cdot B_1 \mid u\cdot\alpha(u_2)\cdot B_2$. Let v be a β-position such that m β-expansions at v occur in B, i.e. in the two substructures B_1 and B_2. For $k = 1, \ldots, m$, let B^k denote the substructures*

$$B^k = v\cdot\alpha(v_1)\cdot B_1^k \mid v\cdot\alpha(v_2)\cdot B_2^k,$$

in B with $succ(v) = \{v_1, v_2\}$ (see Figure 3, upper part). Then the split permutation *of all occurrences of β-expansions at v above the β-expansion at u in B is defined by the following steps (see Figure 3, lower part):*

1. Permutation step: *Replace B with the substructure $\widehat{B}$ where:*

$$\widehat{B} = \begin{cases} v \cdot \alpha(v_1) \cdot u \cdot \alpha(u_1) \cdot B_1' \mid v \cdot \alpha(v_1) \cdot u \cdot \alpha(u_2) \cdot B_2' \mid \\ v \cdot \alpha(v_2) \cdot u \cdot \alpha(u_1) \cdot B_1'' \mid v \cdot \alpha(v_2) \cdot u \cdot \alpha(u_2) \cdot B_2''. \end{cases}$$

   B_1' and B_2' result from B_1 and B_2 by replacing every occurrence of B^k with its B_1^k; and B_1'' and B_2'' result from B_1 and B_2 by replacing every occurrence of B^k with its B_2^k, for all $k = 1, \ldots, m$.
2. Atom-expansion step: *Let the partial branch b_i^k, $i \in \{1,2\}$, $k \in \{1, \ldots, m\}$, denote a substructure B_i^k consisting of a (possibly empty) sequence of atom-expansions only. Let c_i^k denote all partial branches in B for which:*
   (a) *c_i^k is of the form $c_i^k = u \cdot \alpha(u_j) \cdot d_j^k \cdot v \cdot \alpha(v_i) \cdot b_i^k$, where $i, j \in \{1,2\}$, $k \in \{1, \ldots, m\}$, and d_j^k is a (possibly empty) partial branch.*
   (b) *The branch $b \cdot c_i^k$ is leaf-connected with connections C_i^k such that there exists $\{c_1, c_2\} \in C_i^k$ for which $c_1 \in \alpha(v_i)$ and c_2 does* not *occur on b_i^k.*

   For every c_i^k, let $\hat{c}_i^k = v \cdot \alpha(v_i) \cdot u \cdot \alpha(u_j) \cdot d_j^k \cdot b_i^k$ be the partial branch after *permutation in $\widehat{B}$. Then, for every $\{c_1, c_2\} \in C_i^k$, which has satisfied condition (b)* before *permutation, extend $\hat{c}_i^k$ by an atom-expansion at c_1.*

The atom-expansion step ensures that every leaf connection $\{c_1, c_2\}$ on a branch $b \cdot c_i^k$ with $c_1 \in \alpha(v_i)$ and c_2 *not* on b_i^k before permutation, will appear as a leaf connection on $b \cdot \hat{c}_i^k$ after permutation. Since the layers $\alpha(v_1)$ and $\alpha(v_2)$ are permuted towards the root of B, several atom-expansions have to performed on $\hat{c}_i^k$ in order to preserve *all* leaf connections in $\widehat{B}$. If $b \cdot B$ was minimal, this step guarantees that the resulting branches in $b \cdot \widehat{B}$ remain leaf-connected at all.

The split permutation is correct since all branches in $b \cdot \widehat{B}$ remain closed. The following lemma justifies the use of β-proofs for a complete redundancy deletion.

Lemma 2. *Let $b \cdot B$ be a minimal β-subproof. Then every β-subproof $b \cdot \widehat{B}$ resulting from a split permutation according to Definition 10 is minimal.*

Proof. Consider the subproof $b \cdot B$ shown in Figure 3, upper part. Let n^i be the number of β-expansions in B^i. We have $n^i = n_1^i + n_2^i + 1$ where n_1^i and n_2^i denote the number of β-expansions in B_1^i and B_2^i, for $i = 1, \ldots, m$. Let k_1 and k_2 be the number of β-expansions in B_1 and B_2, different from the B^i. We obtain

$$\#\beta(b \cdot B) = 1 + k_1 + k_2 + \sum_{i=1}^{m} 1 + n_1^i + n_2^i = 1 + k_1 + k_2 + m + \sum_{i=1}^{m} n_1^i + n_2^i.$$

Obviously, every β-subproof of $b \cdot B$ is also minimal. Hence, $\#Ax(b \cdot B) > (k_1 + 1) + (k_2 + 1) - m$, where the m subbranches $b \cdot b_i \cdot v$ in B_1 and B_2 are closed with the m substructures B^i, $i = 1, \ldots, m$. For these subbranches we obtain $\#Ax(b \cdot b_i \cdot v \cdot \alpha(v_j) \cdot B_j^i) = n_j^i + 1$, for $j = 1, 2$ and $i = 1, \ldots, m$. Thus,

$$\#Ax(b \cdot B) = (k_1 + 1) + (k_2 + 1) - m + \sum_{i=1}^{m} (n_1^i + 1) + (n_2^i + 1) = 1 + \#\beta(b \cdot B),$$

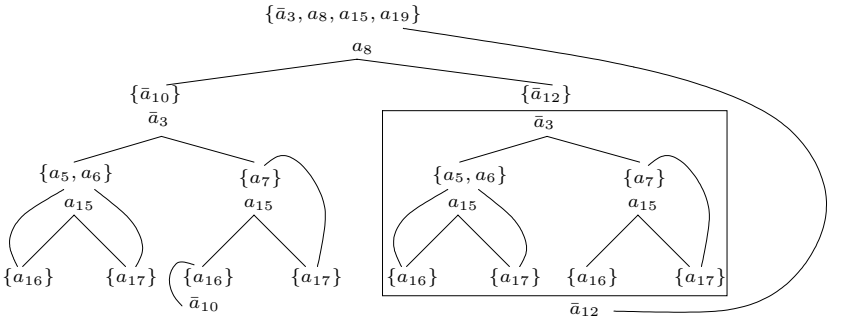


Fig. 4. Split permutation in the optimal β-proof from Example 4

which provides minimality of $b \cdot B$. The substructure $b \cdot \widehat{B}$ after permutation and atom-expansions is shown in Figure 3, lower part. We have

$$\#\beta(b \cdot \widehat{B}) = 1+(1+k_1+k_2+\sum_{i=1}^{m} n_1^i)+(1+k_1+k_2+\sum_{i=1}^{m} n_2^i) = 3+2(k_1+k_2)+\sum_{i=1}^{m} n_1^i+n_2^i.$$

Again, the substructures B_1^i close m subbranches in B_1' and B_2'; and the B_2^i close m subbranches in B_1'' and B_2'', for all $i = 1, \ldots, m$. This yields

$$\#Ax(b \cdot \widehat{B}) = ((k_1+1)+(k_2+1)-m+\sum_{i=1}^{m}(n_1^i+1))+$$
$$((k_1+1)+(k_2+1)-m+\sum_{i=1}^{m}(n_2^i+1)) = 1+\#\beta(b \cdot \widehat{B}).$$

Hence, $b \cdot \widehat{B}$ is minimal. (q.e.d.)

Example 4. Consider the β-proof $B_{\propto^\star}$ of the formula F from Example 3 (see Figure 2). Obviously, $B_{\propto^\star}$ is optimal. The original reduction ordering $\propto^\star$ requires the permutation of the β-expansion at a_8 above the β-expansion at $\bar{a}_3$. The following steps are performed according to Definition 10 (see Figure 4):

1. We have one ($m = 1$) substructure B^1 beginning with the β-expansion at v to be permuted. It occurs below the α-layer $\{a_{16}\}$ in the right substructure of $B_{\propto^\star}$. Moreover, $B_1^1 = B_2^1 = \varepsilon$ holds, for the two substructures in B^1.
2. Permutation step: Permute the β-expansion at a_8 towards the root and duplicate the substructure which starts with the β-expansion at $\bar{a}_3$. Replace B^1 below $\{a_{16}\}$ in the left substructure with $B_1^1 = \varepsilon$; replace B^1 below $\{a_{16}\}$ in the right substructure with $B_2^1 = \varepsilon$.
3. Atom-expansion step: We have $B_1^1 = b_1^1 = \varepsilon$ and $B_2^1 = b_2^1 = \varepsilon$. Consider the following partial branches before permutation:
   $c_1^1 = \bar{a}_3 \cdot \{a_7\} \cdot a_{15} \cdot \{a_{16}\} \cdot a_8 \cdot \bar{a}_{10} \cdot b_1^1$ and $c_2^1 = \bar{a}_3 \cdot \{a_7\} \cdot a_{15} \cdot \{a_{16}\} \cdot a_8 \cdot \bar{a}_{12} \cdot b_2^1$.
   For c_1^1 the connection $\{a_{16}, \bar{a}_{10}\}$ and for c_2^1 the connection $\{a_{19}, \bar{a}_{12}\}$ satisfies condition 2-(b) of Definition 10. After permutation we obtain:
   $\hat{c}_1^1 = a_8 \cdot \bar{a}_{10} \cdot \bar{a}_3 \cdot \{a_7\} \cdot a_{15} \cdot \{a_{16}\} \cdot b_1^1$ and $\hat{c}_2^1 = a_8 \cdot \bar{a}_{12} \cdot \bar{a}_3 \cdot \{a_7\} \cdot a_{15} \cdot \{a_{16}\} \cdot b_2^1$.
   We extend $\hat{c}_1^1$ with an atom-expansion at $\bar{a}_{10}$ and $\hat{c}_2^1$ with an atom-expansion at $\bar{a}_{12}$. Thus, all branches remain leaf-connected in the permuted β-proof.

The resulting β-proof (Figure 4) is minimal, but not optimal: The α-layer $\{a_{16}\}$ of the rightmost β-expansion at a_{15} and hence, this β-expansion itself is pure.

Purity reductions in β-proofs. A split permutation in an optimal β-subproof $b \cdot B$ results in a minimal but usually *not* optimal β-subproof $b \cdot \widehat{B}$. But this property is crucial in order avoid decomposition problems in $\propto^\star$ during the reconstruction process (Theorem 2). For this, we formalize the second operation, i.e. a purity reduction, in β-subproofs. A repeated application of this reduction to $b \cdot \widehat{B}$ leads to an optimal β-subproof $b \cdot \widehat{B'}$, which can be used for a complete redundancy deletion in $\propto^\star$. The purity reduction is defined as follows:

Definition 11. *(purity reduction) Let $b{\cdot}B$ be a β-subproof containing a subproof $b \cdot b' \cdot B'$, such that $succ(u) = \{u_1, u_2\}$ and $B' = u \cdot \alpha(u_1) \cdot B'_1 \mid u \cdot \alpha(u_2) \cdot B'_2$. Let the β-expansion at u be pure, containing a pure α-layer $\alpha(u_i)$, $i \in \{1, 2\}$. Then, the* purity reduction *at u is defined by replacing B' with B'_i in $b \cdot B$.*

The purity reduction at u is correct since all branches, which are closed in the β-subproof $b \cdot b' \cdot u \cdot \alpha(u_i) \cdot B'_i$, where $\alpha(u_i)$ is pure, are also closed in $b \cdot b' \cdot B'_i$. The following lemma shows minimality of the reduction (see [26] for a proof).

Lemma 3. *Let $b \cdot B$ be a minimal β-subproof, which contains a subproof $b \cdot b' \cdot B'$ with a topmost pure β-expansion at u in B' according to Definition 11. Then, the β-subproof $b \cdot b' \cdot B'_i$ resulting from the purity reduction at u is minimal.*

Example 5. Reconsider the minimal β-proof $B_{\propto^\star}$ after the split permutation from Example 4 (Figure 4). It is not optimal since the rightmost β-expansion at a_{15} is pure. We apply purity reductions according to Definition 11:

1. Replace this β-expansion at a_{15} with $B'_1 = \bar{a}_{12}$ below its pure α-layer $\{a_{16}\}$.
2. Then, the right β-expansion at $\bar{a}_3$ becomes pure since its α-layer $\{a_7\}$ is pure. Replace this β-expansion with $B'_2 = \bar{a}_{12}$, occurring below $\{a_7\}$ now.

After every purity reduction, the intermediate β-proofs are minimal. The resulting optimal β-proof is depicted in Figure 4 after removing the boxed part.

4.3 Guiding Proof Reconstruction by β-Proofs

Proof reconstruction is guided by the traversal of the reduction ordering $\propto^\star$ (Section 2.2). Integrating β-proofs into this process requires an optimal β-subproof $b \cdot B$ that represents the actual $\propto^\star$. Before starting traversal, an *initial* optimal β-proof $B_{\propto^\star}$ has to be computed from the given matrix proof $\propto^\star$ and its connections $\mathcal{C}$. For every (sub)branch b of a β-proof $B_{\propto^\star}$ we define the sets

$$\beta_b = \{v \mid v \text{ is } \beta\text{-expansion on } b\} \quad \text{and} \quad \alpha_b = \bigcup \{\alpha(v) \mid \alpha(v) \text{ is } \alpha\text{-layer on } b\}.$$

Then, $B_{\propto^\star}$ can be extracted from $\propto^\star$ and $\mathcal{C}$ using the algorithm in Figure 5.

Observe that the β-proof $B_{\propto^\star}$ of $\propto^\star$, i.e. the inference steps, is computed independently from a concrete proof calculus. The construction process terminates with a β-proof, since all paths in $\propto^\star$ contain a complementary connection $c \in \mathcal{C}$. Every branch $b \in \mathcal{B}$ becomes eventually leaf-connected with a singleton set $\{c\}$

1. Construct a minimal β-proof $B_{\propto^\star}$ from $\propto^\star$ and $\mathcal{C}$ in a logic $\mathcal{L}$.
   1.1 *Initialization:* Let b_0 be the original root of $\ll$. Start with the root branch $\alpha(b_0)$ and the leaf-connections $\mathcal{C}_{B_{\propto^\star}} = \emptyset$. Set $\mathcal{B} = \{\alpha(b_0)\}$.
   1.2 *Construction:* Select a branch $b \in \mathcal{B}$.
   - If there exists a connection $c \in \mathcal{C}$ such that $c \subseteq \alpha_b$ and b is leaf-connected with c, then set $\mathcal{B} := \mathcal{B} \setminus \{b\}$ and $\mathcal{C}_{B_{\propto^\star}} := \mathcal{C}_{B_{\propto^\star}} \cup \{c\}$.
   - Otherwise, select a β-position $a \in \alpha_b \setminus \beta_b$ and perform the β-expansion at a on b. Set $b_1 = b \cdot a \cdot \alpha(succ_1(a))$, $b_2 = b \cdot a \cdot \alpha(succ_2(a))$, and $\mathcal{B} := (\mathcal{B} \setminus \{b\}) \cup \{b_1, b_2\}$.

   1.3 *Recursion:* If $\mathcal{B} \neq \emptyset$, then proceed with step 1.2.
   1.4 *Termination:* If $\mathcal{B} = \emptyset$, then the result $B_{\propto^\star}$ is a minimal β-proof with leaf-connections $\mathcal{C}_{B_{\propto^\star}}$ of $\propto^\star$ in $\mathcal{L}$.
2. Apply the purity reduction to $B_{\propto^\star}$, resulting in an optimal β-proof of $\propto^\star$.

Fig. 5. Algorithm for computing an optimal β-proof $B_{\propto^\star}$ of $\propto^\star$

and thus, $B_{\propto^\star}$ is minimal. But $B_{\propto^\star}$ is not necessarily optimal, since redundant β-expansions might have been applied when selecting β-positions in step 1.2. The purity reduction in step 2. removes these redundancies from $B_{\propto^\star}$ and $\mathcal{C}_{B_{\propto^\star}}$.

During the reconstruction process, the optimal β-subproof $b \cdot B$ representing the actual $\propto^\star$ needs to be modified, whenever a β-split is performed in $\propto^\star$. For refining redundancy deletion in $\propto^\star$ wrt. $b \cdot B$, the following operation is needed:

Definition 12. *(connection deletion) Let $\propto^\star$ be the actual reduction ordering with its connections $\mathcal{C}$ during traversal, represented by an optimal β-subproof $b \cdot B$. Let $\mathcal{C}_{b \cdot B} \subseteq \mathcal{C}$ denote the set of all leaf connections in $b \cdot B$, i.e. used for branch closure. Then, $\mathcal{C}' := \mathcal{C}_{b \cdot B}$ is called the* connection deletion *in $\mathcal{C}$ wrt. $b \cdot B$.*

Lemma 4. *Let $b \cdot B$ the optimal β-subproof of $\propto^\star$ and let $\mathcal{C}' := \mathcal{C}_{b \cdot B}$ be the connection deletion. Then, every connection in $\mathcal{C}'$ is relevant for $\propto^\star$ wrt. $b \cdot B$.*

Lemma 4 guarantees that *all* redundant connections will be removed from $\mathcal{C}$ via connection deletion (see [26] for the proof). β-proofs will be integrated into proof reconstruction by extending the *split* operation at β-positions in $\propto^\star$. Let $[\propto^\star_1, \propto^\star_{2'}] := \beta\text{-}split(\propto^\star, u)$ according to Definition 2. Then, we obtain:

(1) If the topmost β-expansion in $b \cdot B$ is also performed at u, then $b \cdot B$ is compatible with the traversal order of $\propto^\star$. Thus, $b \cdot B$ is divided into two β-subproofs $b \cdot u \cdot \alpha(u_1) \cdot B_1$ and $b \cdot u \cdot \alpha(u_2) \cdot B_2$, representing $\propto^\star_{1'}$ and $\propto^\star_{2'}$.
(2) If the topmost β-expansion in $b \cdot B$ is performed at some v with $v \neq u$, then $b \cdot B$ does not agree with traversal order of $\propto^\star$. A split permutation of u above v yields $b \cdot \widehat{B}$. Applications of purity reductions result in an optimal β-subproof $b \cdot \widehat{B}'$, which will be divided into two β-subproofs $b \cdot u \cdot \alpha(u_1) \cdot B_1$ and $b \cdot u \cdot \alpha(u_2) \cdot B_2$, representing $\propto^\star_{1'}$ and $\propto^\star_{2'}$.

After this, we perform connection deletion by $\mathcal{C}'_i := \mathcal{C}_{b \cdot u \cdot \alpha(u_i) \cdot B_i}$, and finally apply $\propto^\star_i := (\beta, \Theta)\text{-}purity(\propto^\star_{i'}, \mathcal{C}'_i)$ (Definition 4) to delete pure subrelations from the $\propto^\star_{i'}$, for $i = 1, 2$. Recall that no further connections will be deleted from the $\mathcal{C}'_i$ since all of them are relevant for $\propto^\star_{i'}$ (Lemma 4). The extended *split* operation returns $[\langle \propto^\star_1, B_1 \rangle, \langle \propto^\star_2, B_2 \rangle]$, which is formalized by the following definition:

Definition 13. *Let u be the β-position in $\propto^\star$ to be solved next, and let $b \cdot B$ the optimal β-subproof representing $\propto^\star$, where $B = v \cdot \alpha(v_1) \cdot B_{v_1} \mid v \cdot \alpha(v_2) \cdot B_{v_2}$. Then, $split_2(\propto^\star, b \cdot B, u) := [\langle \propto_1^\star, B_1\rangle, \langle \propto_2^\star, B_2\rangle]$ is defined by the following steps:*

(a) $[\propto_{1'}^\star, \propto_{2'}^\star] := \beta\text{-}split(\propto^\star, u)$,

(b) for $i = 1, 2$, *set* $B_i = B_{v_i}$ *if* $v = u$, *or* $B_i = B_i'$ *if* $v \neq u$, *where* B_i' *is computed as follows: (i)* $b \cdot \widehat{B}$ *is the split permutation of* u *above* v *in* $b \cdot B$, *and (ii)* $b \cdot \widehat{B}'$ *is the optimal* β*-subproof after purity reductions in* $b \cdot \widehat{B}$ *with*

$$\widehat{B}' = u \cdot \alpha(u_1) \cdot B_1' \mid u \cdot \alpha(u_2) \cdot B_2',$$

(c) for $i = 1, 2$, *set* $\mathcal{C}_i' := \mathcal{C}_{b \cdot u \cdot \alpha(u_i) \cdot B_i}$ *(connection deletion),*

(d) for $i = 1, 2$, *set* $\propto_i^\star := (\beta, \Theta)\text{-}purity(\propto_{i'}^\star, \mathcal{C}_i')$.

Example 6. Reconsider the optimal β-proof from Example 5 (Figure 4) and the decomposition problem in $\propto_2^\star$ of Example 2 (see Figure 1). The optimal β-subproof representing $\propto_2^\star$ is the single branch $\{\bar{a}_3, a_8, a_{15}, a_{19}\} \cdot a_8 \cdot \{\bar{a}_{12}\} \cdot \bar{a}_{12}$, closed with the connection $\{a_{19}, \bar{a}_{12}\}$. Application of connection deletion yields $\mathcal{C}_2' := \{\{a_{19}, \bar{a}_{12}\}\}$ as single connection in $\propto_2^\star$. This avoids the decomposition problem and hence, the deadlock in $\propto_2^\star$ during further proof reconstruction.

5 Complexity and Adequate Completeness

We assume *standard matrix calculi* based on the matrix characterizations for the logics $\mathcal{L}$, such as the extension procedure [3,13] or analytic tableaux with unification [2,22]. We measure the complexity of proof calculi by the length of their proofs. The *length* of a matrix proof is defined by the number $N_\mathcal{C}$ of inference steps in a standard matrix calculus, for testing a mating $\mathcal{C}$ to be spanning. The *length* of a cut-free sequent proof is defined by the number of its *axiom*-rules. We have analyzed the complexity of matrix and sequent calculi by comparing the length of shortest matrix and sequent proofs for a particular class of formulae.

We were able to show that cut-free sequent calculi cannot polynomially simulate standard matrix calculi for all logics $\mathcal{L}$ (see [26] for details). This means, that there exist formulae for which every sequent proof has exponential length wrt. the length of a matrix proof for these formulae. The intrinsic complexity result is independent from a particular matrix calculus and thus, generally determines the worst case complexity of *every* proof reconstruction procedure.

Efficiency of β-proofs. Although we cannot overcome this general worst-case complexity, the integration of β-proofs eliminates the additional complexity of search behavior during the reconstruction process. As a result, redundant proof steps are avoided in the reconstructed sequent proofs. In a search-based approach, the selection function f_d has to consider all isolated subrelations of a decomposition problem in $\propto^\star$, until the proof relevant subrelation is found (see Section 3). The occurrence of a decomposition problem itself indicates redundancies in $\propto^\star$, which cannot be eliminated with naive methods. This leads to redundant proof steps in the sequent proofs, even before the search of f_d starts.

β-proofs provide a technically elegant elimination of search behavior, independent from the considered logics and particular proof calculi. For this reason,

they realize the high-level interface for efficient and goal-oriented proof reconstruction, while avoiding redundant proof steps in the resulting sequent proofs.

The length of a β-proof $B_{\propto^\star}$ is measured by the number $\#Ax(B_{\propto^\star})$ of its inference steps (see algorithm in Figure 5). This corresponds to the length $N_{\mathcal{C}}$ of a matrix proof in a particular matrix calculus, i.e. $N_{\mathcal{C}} = \#Ax(B_{\propto^\star})$. Due to the nature of proof calculi, $N_{\mathcal{C}}$ can in the worst case grow exponentially in the size of its mating $\mathcal{C}$, i.e. multiply used connections appear as different inference steps in $N_{\mathcal{C}}$ (see [26]). Since proof search returns only the inference steps wrt. the used matrix calculus, we either have to extract a mating from these inference steps and proceed with search-based proof reconstruction, or we have to recompute generalized inference steps in a β-proof $B_{\propto^\star}$ for a search-free reconstruction process. Hence, the use of β-proofs in every respect improves the efficiency of proof reconstruction due to the above mentioned elimination of search.

Correctness and adequate completeness. Proof reconstruction guided by β-proofs is correct, since the *split*$_2$ operation cuts subrelations from $\propto^\star$ without violating the relation $\sqsubset$ in $\propto^\star$. In order to prove completeness, we need to show that the *split*$_2$ operation deletes all redundancies from the $\propto_i^\star$ after splitting. According to Lemma 4, no redundant connections will survive after connection deletion. Since the β-subproofs for the $\propto_i^\star$ remain optimal, no decomposition problems can occur in the $\propto_i^\star$ due to Theorem 2. Hence, deadlocks will be avoided due to Lemma 1, leading to a search-free proof reconstruction process:

Theorem 3. *Proof reconstruction guided by β-proofs is correct and adequately complete for all logics $\mathcal{L}$ under consideration.*

6 Related Work

Transformations from classical matrix proofs into natural deduction proofs have been developed independently by Andrews and Bibel [1,3]. Andrews' approach has been refined and extended to higher order logic, using so-called *expansion tree proofs* in [20,23,24]. Proof generation in natural language has been developed with the ILF system [8]. Proof reconstruction from resolution-based proofs has been considered in [23,17,10,9]. In [10,9], the resulting natural deduction proofs are structured into macro steps for further processing into natural language.

Most of these approaches gain their advantages from a certain pre-structuring of the input proofs and thus, are restricted to classical logic only. As an exception, the expansion tree proofs as used in [23] correspond to the data structure of a reduction ordering used in our work: i.e. the formula tree of the input formula, its expansion by multiply used subformulae, ordering constraints on certain subformulae, and a mating. However, expansion tree proofs will not provide a sufficient data structure for a uniform extension to non-classical logics. This is due to additional non-permutabilities and deletion of sequent formulae during rule applications. Then, additional search would be involved, producing redundant proof steps in the resulting proofs, even when deleting redundancies with so-called *focused matings*, as presented in [20]. Our proof reconstruction

approach works for a class of logics without restricting the input proofs, while integrating the inference steps to avoid search. In earlier work [29], we have followed this approach by focusing on a single matrix calculus, the *extension procedure* [3,13]. The β-proofs in this paper abstract from particular proof calculi.

7 Conclusion and Future Work

We have presented β-proofs as a uniform, tableau-like representation framework for the inference steps of matrix proofs. Two operations on β-proofs have been developed, providing an elegant method for a complete redundancy deletion during proof reconstruction. As a main result, the framework yields a general relation between matrix and sequent calculi that makes it independent from particular proof calculi and proof reconstruction methods. Thus, β-proofs enable us to add a new dimension of uniformity to our proof reconstruction approach, which shows the most important and significant generalization to previous work [29].

We have illustrated the integration of β-proofs into our proof reconstruction procedure [28,12,15], resulting in an adequately complete, i.e. search free, reconstruction process for classical, intuitionistic, and the modal logics $K, K4, D, D4$, $T, S4$, $S5$, as well as for fragments of linear logic (see [26] for details).

As a further main contribution, we have discussed the improvement of efficiency when using β-proofs during proof reconstruction: β-proofs are indispensable in order to eliminate search behavior, while avoiding redundant proof steps in the reconstructed sequent proofs. We have shown that the representation overhead of β-proofs does not influence the gained efficiency during proof reconstruction. Thus, the established framework serves as a general and independent interface for building efficient and uniform proof reconstruction components.

In the future we will combine our uniform approach with existing proof procedures in order to guide derivations in interactive proof assistants. Currently, we work on an implementation for intuitionistic logic within the NuPRL system [7]. We will also consider extensions to larger fragments of linear logic [18,19].

Acknowledgements

I would like to thank Christoph Kreitz and the anonymous referees for many valuable comments concerning this paper and the work reported in it.

References

1. P. B. Andrews. Transforming matings into natural deduction proofs. *CADE–5*, LNCS 87, pp. 281–292. Springer, 1980.
2. B. Beckert, J. Posegga. leanTAP: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.
3. W. Bibel. *Automated theorem proving.* Vieweg, Braunschweig, 1982.
4. W. Bibel, S. Brüning, U. Egly, T. Rath. Komet. *CADE–12*, LNAI 814, pp. 783–787. Springer, 1994.
5. W. Bibel, D. Korn, C. Kreitz, F. Kurucz, J. Otten, S. Schmitt, G. Stolpmann. A multi-level approach to program synthesis. *7th LOPSTR Workshop*, LNAI 1463, pp. 1-25. Springer, 1998.

6. W. Bibel, D. Korn, C. Kreitz, S. Schmitt. Problem-oriented applications of automated theorem proving. *DISCO-96*, LNCS 1128, pp. 1–21. Springer, 1996.
7. R. Constable et. al. *Implementing mathematics with the* NuPRL *proof development system.* Prentice Hall, 1986.
8. B. I. Dahn et. al. Integrating logical functions with ILF. Preprint 94-10, Humboldt Universität zu Berlin, 1994.
9. H. Horacek. Presenting proofs in a human-oriented way. *CADE–16*, LNAI 1632, pp. 142–156. Springer, 1999.
10. X. Huang. Translating machine-generated resolution proofs into ND-proofs at the assertion level. 4^{th} *Pacific Rim International Conference on Artificial Intelligence*, LNCS 1114, pp. 399–410. Springer, 1996.
11. M. C. Fitting. *Proof methods for modal and intuitionistic logic.* D. Reidel, 1983.
12. C. Kreitz, H. Mantel, J. Otten, S. Schmitt. Connection-based proof construction in linear logic. *CADE–14*, LNAI 1249, pp. 207–221. Springer, 1997.
13. C. Kreitz, J. Otten. Connection-based theorem proving in classical and non-classical logics. *Journal for Universal Computer Science*, 5(3):88–112, 1999.
14. C. Kreitz, J. Otten, S. Schmitt. Guiding program development systems by a connection based proof strategy. 5^{th} *LOPSTR Workshop*, LNCS 1048, pp. 137–151. Springer, 1996.
15. C. Kreitz, S. Schmitt. A uniform procedure for converting matrix proofs into sequent-style systems. *Journal of Information and Computation*, 2000 (to appear).
16. R. Letz, J. Schumann, S. Bayerl, W. Bibel. Setheo: A high-performance theorem prover. *Journal of Automated Reasoning*, 8:183–212, 1992.
17. C. Lingenfelder. Structuring computer generated proofs. *IJCAI-89*. Morgan Kaufmann, 1989.
18. H. Mantel, C. Kreitz. A matrix characterization for *MELL*. *JELIA Workshop*, LNCS 1489, pp. 169–183. Springer, 1998.
19. H. Mantel, J. Otten. linTAP: A tableau prover for linear logic. *TABLEAUX Conference*, LNAI 1617, pp. 217–231. Springer, 1999.
20. D. A. Miller. Expansion tree proofs and their conversion to natural deduction proofs. *CADE–7*, LNCS 170, pp. 375–393. Springer, 1984.
21. H. J. Ohlbach. *A resolution calculus for modal logics.* PhD Thesis, Univ. Kaiserslautern, 1988.
22. J. Otten. leanTAP: An intuitionistic theorem prover. *TABLEAUX Conference*, LNAI 1227, pp. 307–386. Springer, 1997.
23. F. Pfenning. Analytic and non-analytic proofs. *CADE–7*, LNCS 170, pp. 394–413. Springer, 1984.
24. F. Pfenning. *Proof transformations in higher-order logic.* Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, 1987.
25. J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.
26. S. Schmitt. *Proof reconstruction in classical and non-classical logics.* PhD Thesis, TU-Darmstadt, 1999.
27. S. Schmitt, C. Kreitz. On transforming intuitionistic matrix proofs into standard-sequent proofs. 4^{th} *TABLEAUX Workshop*, LNAI 918, pp. 106–121. Springer, 1995.
28. S. Schmitt, C. Kreitz. Converting non-classical matrix proofs into sequent-style systems. *CADE–13*, LNAI 1104, pp. 418–432. Springer, 1996.
29. S. Schmitt, C. Kreitz. Deleting Redundancy in proof reconstruction. TABLEAUX Conference, LNAI 1397, pp. 262–276. Springer, 1998.
30. L. Wallen. *Automated deduction in non-classical logics.* MIT Press, 1990.
31. L. Wos et. al. Automated reasoning contributes to mathematics and logic. *CADE–10*, LNCS 449, p. 485–499. Springer, 1990.

The Semantic Tableaux Version of the Second Incompleteness Theorem Extends Almost to Robinson's Arithmetic Q

Dan E. Willard

Dept. of Computer Science, University at Albany, dew@cs.albany.edu. **

Abstract. We will generalize the Second Incompleteness Theorem *almost to the level* of Robinson's System Q. We will prove there exists a Π_1 sentence V, such that if α is *any* finite consistent extension of $Q+V$ then α will be unable to prove its Semantic Tableaux consistency.

1 Introduction

As originally formulated by Gödel, the Second Incompleteness Theorem discussed the inability of any possible extension of Peano Arithmetic (PA) to verify its self-consistency when it examined a proof-encoding that used a Hilbert-style methodology [5] as the underlying deductive calculi. Many generalizations of the Second Incompleteness Theorem were developed subsequently. For instance, let us recall that Robinson's Arithmetic system Q differs from Peano Arithmetic by containing no Induction axioms [10,16]. In 1985, Pudlák strengthened a theorem by Bezboruah and Shepherdson [3] to show that Q was subject to many limitations similar to Peano Arithmetic. In particular, Pudlák established that if α is *any extension* of Q then α must be unable to prove the non-existence of a proof of 0=1 when the proof formalism uses *simultaneously* a Hilbert-style calculi for deduction [5] and α represents its set of proper axioms. Moreover, Robert Solovay (private communications, [18]) combined several formalisms of Pudlák, Nelson [11] and Wilkie-Paris [22] to observe that Pudlák's theorem generalized to systems weaker than Q in that they would recognize Successor (but not Addition and Multiplication) as total functions.

The main gap left by the prior literature was that it had not resolved whether or not the Second Incompleteness Theorem was also valid for weak systems with respect to cut-free forms of deductive calculi, such as Semantic Tableaux. In particular, let us say an axiom system α has an ability to verify its **tableaux-based self-consistency** iff α can formally prove the non-existence of a proof of 0=1 in a deduction system that uses **simultaneously** α as the set of proper axioms and Fitting's precise rules for Semantic Tableaux [5] as the employed deductive calculi. The prior literature had not clarified fully to what extent the Second Incompleteness Theorem prohibits weak axiom systems α from proving this particular notion of their self-consistency.

** Supported by NSF Grant CCR 99-02726

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 415–430, 2000.

For example, consider an axiom system that uses a special atomic symbol $M(x, y, z)$ to represent that the multiplicative product of x times y equals z. Such a system would not necessarily recognize that Multiplication is a **"total function"** (i.e. it need not recognize that "$\forall x \, \forall y \, \exists z \, : \, M(x, y, z)$"). We devised in [23,24] a consistent axiom system of this type, called IS(A), that could

1. recognize its tableaux-based self-consistency (in the particular sense that was defined on the preceding page).
2. recognize Addition as a total function, and
3. prove all the Π_1-like theorems of Peano Arithmetic in a slightly modified language that replaces the the Multiplication Function symbol with a $M(x, y, z)-$atomic predicate symbol.

A very different type of partial caveat for the Semantic Tableaux version of the Second Incompleteness Theorem has been described by the literature [8,9,11,14,20,22]. The strongest version of this caveat has been described by Pudlák, and we will therefore use [6,14]'s notation. Say a formula $\Upsilon(v)$ is a **definable cut** for an an axiom system α **if** α **can prove both :** $\Upsilon(0)$ and $\forall v \, \{ \, \Upsilon(v) \, \supset \, \Upsilon(v+1) \, \}$. Then [8,9,11,14,20] have illustrated several examples of different formulae $\Upsilon(v)$, which are Definable Cuts for α, such that α can prove its *Semantic Tableaux* consistency local to these Cuts. Thus if the notation " $\text{SemPrf}_{\alpha}(x, y)$ " designates that y is a semantic tableaux proof of the theorem x and if $\ulcorner \Psi \urcorner$ designates Ψ's Gödel number, the prior literature has shown how several quite different axiom systems α can prove the **cut-localized statement of their** semantic tableaux consistency indicated below:

$$\forall y \quad \{ \quad \Upsilon(y) \quad \supset \quad \neg \, \text{SemPrf}_{\alpha} \, (\, \ulcorner 0 = 1 \urcorner \, , \, y \,) \quad \} \tag{1}$$

In light of these examples about how the Semantic Tableaux version of the Second Incompleteness Theorem can be partially evaded by certain types of formalisms, it is noteworthy that Adamowicz [1] has proven that the very-frequently-studied [6,22] axiom system $I\Sigma_0 + \Omega_2$ satisfies the Second Incompleteness property for semantic tableaux. Since $I\Sigma_0 + \Omega_2$ has an ability to prove the totality of functions growing substantially faster than Multiplication, Adamowicz's theorem raises the following two questions:

1. Where between the axiom system IS(A)'s evasion of the Second Incompleteness Theorem and $I\Sigma_0 + \Omega_2$'s obeying of it, does the Second Incompleteness effect become valid for Semantic Tableaux deduction?
2. Does the Second Incompleteness Effect for Semantic Tableaux proofs become valid for Robinson's System Q, or other axiom systems very near it?

We will offer a partial (albeit not complete) answer to these questions. We will prove there exists a Π_1 sentence, henceforth called V , such that if α is *any* consistent and finite extension of $Q + V$ then α will be unable to formally prove a theorem asserting its own tableaux-based self-consistency. Moreover, this result will generalize for axiom systems α with infinite cardinality (although we will have insufficient space to prove this generalization here.)

2 Notation and Intuition

For simplicity, our base language will be the language of Robinson's Q. Thus only Addition and Multiplication function symbols will appear in its terms t, and the existential and universal quantifiers in "$\exists\, x \leq t\ \phi(x)$" and "$\forall\, x \leq t\ \phi(x)$" will be called *bounded quantifiers*. A formula will be called Δ_0 if all its quantifiers are bounded. Similarly if $\Psi(x_1, x_2, ...x_k)$ is Δ_0, then we will use the term Π_1 to describe the normalized sentence $\forall x_1,\ \forall x_2\ ...\ \forall x_k\ \Psi(x_1, x_2, ...x_k)$.

In our discussion, $\text{SemPrf}_\alpha(x, y)$ will denote a Δ_0 formula indicating that y is a semantic tableaux proof of the theorem x from the axiom system α. For simplicity, we shall use a definition of Semantic Tableaux proof similar to that in Chapters 3.1 and 6.1 in Fitting's textbook [5]. However with one minor caveat, our version of the Second Incompleteness Theorem will actually generalize to all the other major definitions of semantic tableaux. The caveat is that each different definition of semantic tableaux deduction will be associated with a different Π_1 sentence V such that $Q + V$ is a threshold for the Second Incompleteness Theorem. (This is essentially because the Π_1 sentence V, constructed in the next section, will contain a Δ_0 subformula "SemPrf" that is dependent on the particular definition of Semantic Tableaux proof used.)

Given a sequence of integers $i_1, i_2, i_3, ...i_m$, we will say a Δ_0 formula $\Phi(g, j, x)$ makes g an encoding of this sequence iff $\Phi(g, j, x)$ is satisfied only when x represents the $j-$th element in the sequence $i_1, i_2, i_3, ...i_m$. We will say g is a **linear compressed encoding** of this sequence if $\text{Log}(g)$ has a magnitude proportional to the size of $\sum_{j=1}^{m} \text{Log}(i_j + 2)$. Buss, Hájek, Paris, Pudlák and Wilkie [4,6,22] have all given examples of such Δ_0 encodings, and we will therefore not also do so here. The Linear Compressed Encodings are considered to be the most efficient possible method to do a formal Gödel encoding of a sentence or proof, and we will therefore study it in this paper.

Let the symbol $\perp$ denote the Gödel number of the sentence $0 = 1$. The *weakest possible definition* of α's Semantic Tableaux Consistency is:

$$\forall p\ \neg\, \text{SemPrf}_\alpha(\ \perp\ ,\ p\ ) \tag{2}$$

If $\text{Def}(\alpha)$ denotes the above definition of α's tableaux consistency then there are also other available definitions, denoted as say $\text{Def}^*(\alpha)$, such that the two definitions are equivalent in adequately strong fragments of arithmetic, *but not in sufficiently weak fragments* (see for instance [24,25,26]). It is preferable to employ the *weakest available definition* when generalizing the Second Incompleteness Theorems. This is why our present article uses Equation (2)'s definition.

An alternate definition of a "semantic tableaux proof" appears below:

Definition 1. *Let Log(x) denote Base-2 Logarithm, with downwards rounding to the lowest integer. Let* $\text{Log}(x, k)$ *denote* $\text{Log}(\text{Log}(\text{Log}...(\text{Log}(x))))$ *– where there are* k *iterations of logarithm. For any fixed constant* $K > 1$*, the symbol* $\text{SemPrf}_\alpha^K(\ x\ ,\ y\ ,\ z\ )$ *will denote a* Δ_0 *formula indicating that* $\text{SemPrf}_\alpha(\ x\ ,\ y)$ *is valid* **and** *that* $y < \text{Log}(\ z\ ,\ K\ )$.

Oddly, our final theorem will be couched **solely** in terms of Equation (2)'s more desirable SemPrf(x , y) formalism, but **the intermediate steps** of our proof will be much simplified by using as well Definition 1's "$\text{SemPrf}_\alpha^K(x,y,z)$" formalism to shorten their analysis.

In particular, let $D(\alpha)$ denote the following diagonalization sentence:

* There is no Semantic Tableaux proof of **this sentence.**

Also, let $D^K(\alpha)$ denote the following $\text{SemPrf}_\alpha^K(x,y,z)$ modification of this diagonalizing sentence:

** In a context where one employs the slightly modified "$\text{SemPrf}_\alpha^K(x,y,z)$" proof-notation, there exists no code (y,z) that "proves" **this sentence.**

It is very easy to formally encode $D^K(\alpha)$ as a Π_1 sentence following the example of the prior literature on diagonalization. Thus, let $\text{Subst}(g,h)$ denote Gödel's classic Δ_0 substitution relation, defined below:

$\text{Subst}(g,h)$ = The integer g is an encoding of a formula, and h encodes a sentence identical to g, except that all free variables in g are replaced with a constant, whose value equals g.

Then $D^K(\alpha)$ can be defined as being the Π_1 sentence $\Gamma(\bar{n})$, where $\Gamma(g)$ denotes the formula (3) and $\bar{n}$ denotes $\Gamma(g)$'s Gödel number.

$$\forall h\ \forall y\ \forall z \quad \{ \quad \text{Subst}(g,h) \quad \supset \quad \neg\ \text{SemPrf}_\alpha^K\ (\ h\ ,\ y\ ,\ z\ ) \quad \} \tag{3}$$

In essence, our version of a proof of the Second Incompleteness Theorem will be similar to the classic diagonalization proofs, except that many intermediate steps will use $D^K(\alpha)$ instead of $D(\alpha)$.

Theorem 1. *Let $\alpha \vdash_S \Lambda$ denote that there is a semantic tableaux proof of the theorem Λ from the axiom system α. Then the combination of $\alpha \vdash_S \Lambda$, $\alpha \vdash_S \Theta$, and $\alpha \vdash_S \Lambda \wedge \Theta \supset \Xi$ implies $\alpha \vdash_S \Xi$.*

Proof. Immediate from Gentzen's Cut Elimination Theorem [19,21]. □

Theorem 2. *Suppose α is a finite extension of Robinson's system Q that (for some constant K) proves the three theorems below. Then α is inconsistent.*

A) $\quad \forall p\ \neg\ \mathit{SemPrf}_\alpha(\ \perp\ ,\ p\ )$
B) $\quad \{\ \exists y\ \exists z\ \mathit{SemPrf}_\alpha^K(\ \lceil D^K(\alpha)\rceil\ ,\ y,\ z\ )\ \}\ \supset\ \exists x\ \ \mathit{SemPrf}_\alpha(\ \perp\ ,\ x\ )$
C) $\quad \forall g\ \forall h\ \forall h^*\ \{\ \mathit{Subst}(\,g\,,\,h\,)\wedge \mathit{Subst}(\,g\,,\,h^*\,)\ \}\ \supset\ h = h^*$

Proof. Let D^* denote the Gödel-like "diagonalizing" sentence below:

$$D^* \quad =_{\text{df}} \quad \{\ \forall y\ \forall z\ \ \neg\ \text{SemPrf}_\alpha^K(\ \lceil D^K(\alpha)\rceil\ ,\ y,\ z\ )\ \} \tag{4}$$

We will apply Theorem 1 several times to help shorten the proof of Theorem 2. First, let us apply Theorem 1 with Λ and Θ denoting the sentences from (A)

and (B) in Theorem 2's hypothesis and with Ξ representing the sentence D^* (from Equation (4)). For these three particular values for Λ, Θ and Ξ, it is immediately apparent that $\alpha \vdash_S \Lambda \wedge \Theta \supset \Xi$. Hence, Theorem 1 implies

$$\alpha \quad \vdash_S \quad D^* \tag{5}$$

For any fixed K, $D^K(\alpha)$ and D^* are certainly equivalent sentences in sufficiently strong models of Arithmetic. However, we need more than this fact to duplicate Gödel's diagonalization proof in the present setting. We need that *the weak axiom system* α *is yet strong enough* to also recognize this equivalence !

To establish this fact, we begin by recalling that $\bar{n}$ denotes (3)'s Gödel number and that $D^K(\alpha)$'s definition implies Subst($\bar{n}$, $\ulcorner D^K(\alpha) \urcorner$) is true. Since Subst($\bar{n}$, $\ulcorner D^K(\alpha) \urcorner$) is a valid Δ_0 sentence and since Robinson's System Q can prove all valid Δ_0 sentences, it follows that Q can prove this sentence. Thus since Theorem 2's hypothesis indicates that α is an extension of Q, we get:

$$\alpha \quad \vdash_S \quad \text{Subst}(\ \bar{n}\ ,\ \ulcorner D^K(\alpha) \urcorner\ ) \tag{6}$$

We will now use Equation (6) to infer the validity of (7) below. In particular, we do so by applying Theorem 1 with Λ representing the sentence Subst($\bar{n}$ $\ulcorner D^K(\alpha) \urcorner$), with Θ representing the sentence (C) in Theorem 2's hypothesis, and with Ξ being the identity $D^K(\alpha) \equiv D^*$. For these three values for Λ, Θ and Ξ, we can infer that $\alpha \vdash_S \Lambda \wedge \Theta \supset \Xi$ (because $\Lambda \wedge \Theta$ enables α to prove that the only value for h satisfying the left side of Equation (3) is the quantity $\ulcorner D^K(\alpha) \urcorner$). Hence, Theorem 1 implies:

$$\alpha \quad \vdash_S \quad D^K(\alpha) \ \equiv \ D^* \tag{7}$$

Our final application of Theorem 1 is quite easy. We set Λ and Θ to be the sentences D^* and $D^K(\alpha) \equiv D^*$ and use Equations (5) and (7) to infer

$$\alpha \quad \vdash_S \quad D^K(\alpha) \tag{8}$$

We will now finish Theorem 2's proof by following Gödel's paradigm about proving a sentence that states roughly "*There is no proof of me*". Thus, let p denote (8)'s proof of the theorem $D^K(\alpha)$, q be a second integer satisfying $\text{Log}(q, K) > p$, and r denote $D^K(\alpha)$'s Gödel number. Let us also recall that if $\bar{n}$ denotes (3)'s Gödel number, then $D^K(\alpha)$ is the sentence:

$$\forall h\ \forall y\ \forall z \quad \{ \quad \text{Subst}(\ \bar{n}\ ,\ h\ ) \quad \supset \quad \neg\ \text{SemPrf}_{\alpha}^{K}\ (\ h\ ,\ y\ ,\ z\ ) \quad \} \tag{9}$$

We will follow Gödel's example by observing that (9) must be false because if we replace its three variables y, z, and h with the three constants p, q, and r then (9)'s formal statement is negated. Moreover, Robinson's System Q is known to have the capacity to formally disprove any Π_1 sentence that is invalid in the Standard Model. Hence, since α is an extension of Q, we get

$$\alpha \quad \vdash_S \quad \neg\ D^K(\alpha) \tag{10}$$

The combination of (8) and (10) shows that α is inconsistent. □

3 Main Theorems

We will assume that $x - y = 0$ when $x < y$ (so that Subtraction can be viewed as a total function). Also, $\mathrm{Log}(x, u)$ was defined by Definition 1.

Lemma 1. *There exists two Δ_0 formulae $S(x, y, z)$ and $P(x, u, z)$ such that:*

1. *The graphs of $S(x, y, z)$ and $P(x, u, z)$ represent the set of ordered triples satisfying respectively the Subtraction and Logarithm functions*
2. *A Π_1 sentence,* **henceforth denoted as V_1 ,** *will indicate that the two Δ_0 formulae $S(x, y, z)$ and $P(x, k, z)$ represent total functions assigning Subtraction and Logarithm their usual properties.*

Proof Sketch. Item 1's claims about Subtraction are obviously true. The unabridged version of this paper (which the author can mail to any interested readers) showed how $P(x, k, z)$ can encode $\mathrm{Log}(x, k) = z$ as a Δ_0 formula. (Its proof essentially uses Benett's dissertation [2] and the theory of LinH functions [6,7,27]) in a routine manner.) The further claims of Item 2 require no proof because they are an immediate consequence of Item 1 and of the fact that Subtraction and Logarithm are non-growth functions.

Lemma 2. *Using the predicates $S(x, y, z)$ and $P(x, u, z)$ (from Lemma 1), one can easily encode four Π_1 sentences, henceforth denoted as A_1, A_2, A_3 and A_4, that indicate that the functions Subtraction and Log(x, k) have the following four well-known properties:*

1. $\forall x\ \forall y\ \forall z \quad xy \le z \ \supset\ Log(z, 1) \ge Log(x, 1) + Log(y, 1)$
2. $\forall x\ \forall y \quad x^2 \le y \ \supset\ Log(y, 2) \ge Log(x, 2) + 1$
3. $\forall x\ \forall y \quad Log(x, y + 1) = Log\ (\ Log(x, y)\ ,\ 1\ )$
4. $\forall x\ \forall y \quad y \le x \ \supset\ [\ y = x \vee y \le x - 1\ ]$

Proof. A trivial consequence of Lemma 1. □

Let $\mathrm{FinAx}(\alpha)$ denote a Δ_0 formula, which will return the Boolean value of TRUE when the integer α is a Gödel number that represents a finite-length list of logical sentences. The formula $\mathrm{FinAx}(\alpha)$ will serve as a mechanism for recognizing when α represents a finite list of axioms. It thus allows us to treat " $\mathrm{SemPrf}_\alpha(x, y)$ " as a Δ_0 formula with three input variables x, y, α (rather than just x and y). Likewise, "$\mathrm{SemPrf}^k_\alpha(x, y, z)$" will now denote (in this section) a Δ_0 formula free in five variables x, y, z, α and k.

Lemma 3. *There exists a Δ_0 formula, henceforth denoted as $Map(\alpha, k, d)$, which has the property that the triple (α, k, d) satisfies this formula if and only if d equals the Gödel number of Section 2's diagonalization sentence $D^k(\alpha)$. (This implies that Paradox(y, z, α, k) , defined below, also has a Δ_0 encoding.)*

$$\mathrm{Paradox}(y, z, \alpha, k) \quad =_{\mathrm{df}} \quad \exists\, d < z \quad \mathrm{Map}(\alpha, k, d) \ \wedge\ \mathrm{SemPrf}^k_\alpha(\, d\,,\, y\,,\, z\,) \quad (11)$$

Justification. We will omit giving Lemma 3's proof here because its underlying structure is similar to Section 4 of our earlier paper [26]. It thus relies on

[6,7,27]'s theory of LinH functions. It uses the fact that every LinH procedure can be transformed into an analogous Δ_0 formula, in a context where the the tuples satisfying $\text{Paradox}(y, z, \alpha, k)$ and $\text{Map}(\alpha, k, d)$ can be identified by LinH decision procedures.

We are now ready to define the "V" used by our axiom system Q+V. It can be thought of as either one long Π_1 sentence, representing a conjunction of five Π_1 clauses $V_1 \wedge V_2 \wedge V_3 \wedge V_4 \wedge V_5$ or as a list of five different Π_1 axiom-sentences. Its sentence V_1 was defined by Lemma 1. In our discussion, $\text{FinAx5}(\alpha)$ will denote a Δ_0 formula indicating α is a Gödel number which encodes some finite list of axioms that includes all the axioms of $Q + V$. Also, $\text{FinAx4}(\alpha)$ will have an identical definition as $\text{FinAx5}(\alpha)$, except it will require α's list of axioms include only $Q + V_1 + V_2 + V_3 + V_4$. Below are defined V_2, V_3, V_4 and V_5.

$$V_2 \quad =_{\text{df}} \quad A_1 \wedge A_2 \wedge A_3 \wedge A_4 \text{ where Lemma 2 defines these } A_i \tag{12}$$

$$V_3 \quad =_{\text{df}} \quad \{ \ \forall g \ \forall h \ \forall h^* \ \{ \text{Subst}(g, h) \wedge \text{Subst}(g, h^*) \} \supset h = h^* \ \} \tag{13}$$

$$\begin{aligned} V_4 \quad =_{\text{df}} \quad & \{ \ \forall \alpha \ \forall k \ \forall g \ \forall h \ \forall y \ \forall z \ [\ \Upsilon(\alpha, k, g, h, y, z) \quad \supset \\ & \exists h^* \leq h \ \exists y^* \leq y \ \exists z^* \leq z \ \ \Upsilon(\alpha, k, g, h^*, y^*, z^*) \ ] \ \} \end{aligned} \tag{14}$$

$$\text{WHERE } \Upsilon(\alpha, k, g, h, y, z) \ =_{\text{df}} \ \{ \ \text{Subst}(g, h) \ \wedge \ \text{SemPrf}^k_\alpha(h, y, z) \ \}$$

Comment: Since V_4 is provable from Q, some readers may wonder why it is needed? The answer is a redundant axiom can super-exponentially shorten the length of some cut-free proofs (a fact that we will use when we prove Lemma 9).

$$\begin{aligned} V_5 \ =_{\text{df}} \ & \{ \ \forall y \forall z \forall \alpha \forall k \ \{ \ [\ \text{FinAx4}(\alpha) \wedge k \geq \alpha \wedge \text{Paradox}(y, z, \alpha, k) \ ] \ \supset \\ & \exists x < z \ \ \text{SemPrf}_\alpha (\ \bot \ , x) \ \} \} \end{aligned} \tag{15}$$

Below are listed our two main theorems about Q+V.

Theorem 3. *The axiom system $Q+V$ is consistent.*

Theorem 4. *Suppose α is a consistent axiom system that is a finite extension of $Q+V$. Then α cannot prove a theorem asserting its Semantic Tableaux consistency (i.e. it cannot verify $\forall x \ \neg \ \ SemPrf_\alpha(\bot, x)$).*

Comment : Before presenting the proofs of Theorems 3 and 4, it is desirable to briefly reflect upon our overall goals and strategies. The chief goal is, of course, to establish the existence of some Π_1 sentence V where $Q+V$ satisfies simultaneously Theorem 3's consistency property and Theorem 4's Incompleteness property. It turns out that there are many different Π_1 sentences V

that possess both these properties. Some of these V will oddly cause Theorem 3's half of the proof to be shorter, while others will cause Theorem 4's portion of the proof to be shorter. After some experimentation with this subject, we discovered that the paper's overall presentation would be best abbreviated if we chose to work with a particular axiom-sentence V where Theorem 4's portion of the proof would become extremely short, but Theorem 3's part of the proof would become much longer. The reason we are informing our readers about this trade-off is that some readers might otherwise find Theorem 4's proof (below) to be so alarmingly abbreviated to make them initially troubled by it. The best way to soothe such concerns is to remind our readers that it is possible to find some sentences V , where $Q + V$'s Incompleteness property has an astonishingly short proof, provided Theorem 3's proof of $Q + V$'s consistency property then becomes correspondingly longer.

Proof of Theorem 4. Suppose for the sake of contradiction the theorem was false. Then there would exist some fixed integer constant, denoted as say $\bar{\alpha}$, such that FinAx5($\bar{\alpha}$) is true and where $\bar{\alpha}$, viewed as an axiom system, is consistent and verifies its own Semantic-Tableaux-Consistency.

Hence, we may begin our contradiction proof by letting $\bar{\alpha}$ represent a consistent axiom system satisfying

$$\bar{\alpha} \quad \vdash \quad \forall\, x \;\; \neg\, \mathrm{SemPrf}_{\bar{\alpha}}\,(\,\bot\,,\, x\,) \tag{16}$$

Our goal is to show how Equation (16) will lead to a contradiction.

Let us now introduce a second constant $\bar{k} \;=\; \bar{\alpha} + 1$. We claim that the ordered pair ($\bar{\alpha}$, $\bar{k}$) will satisfy the hypothesis of Theorem 2. To establish this fact, we must show that $\bar{\alpha}$ can prove the three theorems (A), (B) and (C) required by Theorem 2's hypothesis. Below is the justification for this claim:

1. Equation (16) shows that $\bar{\alpha}$ can prove Item (A) from Theorem 2's hypothesis.
2. We will next show that $\bar{\alpha}$ also proves Theorem 2's needed sentence (B). Let Λ and Θ denote the following two sentences:

$$\mathrm{FinAx4}(\bar{\alpha}) \,\wedge\, \bar{k} \geq \bar{\alpha} \wedge \mathrm{Map}(\bar{\alpha}, \bar{k}, \lceil\, D^{\bar{k}}(\,\bar{\alpha}\,)\rceil\,)$$

$$\forall y\, \forall z\, \{\,[\,\mathrm{FinAx4}(\bar{\alpha}) \,\wedge\, \bar{k} \geq \bar{\alpha} \,\wedge\, \mathrm{Paradox}(y, z, \bar{\alpha}, \bar{k})\,] \;\supset\; \exists\, x < z \;\; \mathrm{SemPrf}_{\bar{\alpha}}\,(\,\bot\,,\, x)\}$$

   Both these sentences are provable from $\bar{\alpha}$. In particular, $\bar{\alpha} \vdash \Lambda$ holds because $\bar{\alpha}$ can prove every Δ_0 sentence that is valid in the Standard Model of the Natural Numbers, and $\bar{\alpha} \vdash \Theta$ holds because Θ's formal statement is identical to $\bar{\alpha}$'s V_5 axiom except that V_5's universally quantified variables α and k are replaced by the constants $\bar{\alpha}$ and $\bar{k}$. Moreover, if we let Ξ denote the sentence (B) from Theorem 2's hypothesis, it is straightforward to obtain $\bar{\alpha} \vdash \Lambda \wedge \Theta \supset \Xi$ from the underlying structure of these three sentences. Hence from Theorem 1, we immediately obtain that $\bar{\alpha}$ can prove (B) from Theorem 2's hypothesis.
3. Since Theorem 2's sentence (C) and $\bar{\alpha}$'s axiom V_3 are the same sentence, $\bar{\alpha}$ can trivially prove (C).

Hence since $\bar{\alpha}$ satisfies Theorem 2's three requirements, the theorem implies $\bar{\alpha}$ is inconsistent. This observation completes our proof-by-contradiction because its first paragraph assumed $\bar{\alpha}$ was consistent. □

4 The Consistency of Q+V

The axiom system Q+V and the preceding Theorem 4 would obviously both be entirely useless if Q+V was inconsistent. Therefore to establish their significance, we must prove Q+V's consistency. Our proof of Theorem 3 will require introducing several preliminary lemmas.

Lemma 4. *The first four axioms of* V *are valid in the Standard Model of the Natural Numbers, and these axioms can be written as* Π_1 *sentences.*

Proof. Lemmas 1 and 2 indicate the axioms V_1 and V_2 are valid Π_1 sentences in the Standard Model of the Natural Numbers. It is trivial that V_3 is also a valid Π_1 sentence. Finally for any Δ_0 formula $\phi(x, y)$, the sentence " $\forall d \forall e\, [\, \phi(d, e) \supset \exists f \leq e\, \phi(d, f)]$ " is clearly a valid Π_1 sentence. This implies that V_4 is also a valid Π_1 sentence, since its Equation (14) has a form identical to the preceding sentence except that it replaces d with (α, k, g), e with (h, y, z) and f with (h^*, y^*, z^*). □

The remainder of this section will finish the proof of Theorem 3 by showing that the axiom V_5 satisfies a logical validity property similar to V_1 through V_4.

Lemma 5. *If the clause on the left side of the axiom* V_5*'s implication symbol is true then* $y < Log(\, z\, ,\, 2^{3,000}\,)$.

Proof. The formula on the left side of the axiom V_5's $\supset$ symbol is:

$$\text{FinAx4}(\alpha) \wedge k \geq \alpha \wedge \text{Paradox}(y, z, \alpha, k) \tag{17}$$

Clearly $k \geq 2^{3,000}$ because (17) indicates that $k \geq \alpha$ and α must require more than 3,000 bits to encode an axiom system that includes $Q + V_1 + V_2 + V_3 + V_4$. Moreover since (17) indicates that $\text{Paradox}(y, z, \alpha, k)$ is satisfied, Equation (11) and Definition 1 imply $y < \text{Log}(z, k))$. □

Lemma 6. *Let* $\phi(\, x_1\, ,\, x_2 \ldots x_m\,)$ *denote a* Δ_0 *formula. For any m-tuple of constants* $(\, \bar{c}_1\, ,\, \bar{c}_2 \ldots \bar{c}_m\,)$ *with each* $c_i \leq n$ *, assume that the semantic tableaux proof (or disproof) of* $\phi(\, \bar{c}_1\, ,\, \bar{c}_2 \ldots \bar{c}_m\,)$ *from the axiom system* α *requires a tree with no more than* s *nodes. Let the symbol "* $Q_i(\, x_i \leq \bar{k}_i\,)$ *" be an abbreviation for either a bounded existential or universal quantifier. (In other words, "* $Q_i(\, x_i \leq \bar{k}_i\,)$ *" is an abbreviation for either "* $\exists\, x_i \leq \bar{k}_i$ *" or for "*$\forall\, x_i \leq \bar{k}_i$ *".) Let* Ψ *denote a canonical* Δ_0 *sentence of the following form:*

$$Q_1(\, x_1 \leq \bar{k}_1\,),\ Q_2(\, x_2 \leq \bar{k}_2\,) \ldots\ Q_m(\, x_m \leq \bar{k}_m\,)\ \{\ \phi(\, x_1\, ,\, x_2 \ldots x_m\,)\} \tag{18}$$

Assume α *is an extension of* $Q + V_2$ *and* $n \geq MAX(\, \bar{k}_1\, ,\, \bar{k}_2\, ,\, \ldots \bar{k}_m\,)$ *. Then it follows that:*

1. *If the sentence* Ψ *(formally defined by Equation (18)) is TRUE then there will exist a semantic tableaux proof of it of approximate size* $O(\ s \cdot n^m\ )$.
2. *And if the sentence* Ψ *is FALSE then there will analogously exist a semantic tableaux disproof of it of approximate size* $O(\ s \cdot n^m\ )$.

Proof Sketch. Since all the quantifiers in Equation (18) are bounded quantifiers, it is immediately apparent that a brute-force algorithm can span the cross-product space of approximate size $O(n^m)$ and determine via a trivial procedure whether or not the sentence (18) is true or false. The fundamental point is that whenever an axiom system α is rich enough to include all four clauses of Equation (12)'s V_2 axiom, a corresponding a semantic tableaux proof from it can simulate the exhaustive search paradigm and produce a similar-sized proof or disproof of (18) having an approximate $O(s \cdot n^m)$ length. (There is insufficient page space to provide more details about Lemma 6's proof here, but the preceding proof-sketch should be adequate to explain the main intuition.)

Definition 2. *Given an axiom system* α *, say* q *is a* **closed subtree rooted in** Ψ *iff* q *has a structure identical to a semantic tableaux proof, except that the root of* q *consists of "* Ψ *" (rather than "* $\neg\ \Psi$ *").*

Lemma 7. *Let* $\bar{\alpha}$, $\bar{k}$ *and* $\bar{g}$ *denote three constants and consider the following sentence:*

$$\forall h\, \forall y\, \forall z \quad \{\ Subst(\bar{g}, h) \quad \supset \quad \neg\ SemPrf_{\bar{\alpha}}^{\bar{k}}(\ h\ ,\ y\ ,\ z\ )\ \} \tag{19}$$

Suppose p *is a Semantic Tableaux proof from* $\bar{\alpha}$ *of the theorem (19). Let* Υ *denote the formula from Equation (14). Then for any triple* $(\bar{h}, \bar{y}, \bar{z})$ *, it is possible to map* p *onto a "closed" tableaux subtree* q *where:*

a. *The root of* q *is "* $\exists h^* \leq \bar{h}\ \exists y^* \leq \bar{y}\ \exists z^* \leq \bar{z}\ \ \Upsilon(\ \bar{\alpha}\ ,\ \bar{k}\ ,\ \bar{g}\ , h^*, y^*, z^*\ )$ *"*
b. *There will be a natural correspondence between the paths in* p *and the corresponding paths in* q *. Under this correspondence, the resulting path in* q *will have a length exceeding the path in* p *by no more than some constant* C *whose value is independent of* $\bar{\alpha}$, $\bar{k}$, $\bar{g}$ $\bar{h}$, $\bar{y}$ *and* $\bar{z}$. *(Different authors use slightly different definitions of a "semantic tableaux deduction", and these minor variations will produce slightly different values of Lemma 7's constant* C. *Under a typical definition, such as say that in Fitting's textbook [5],* C *can represent a constant roughly* $\cong$ 12.)

Proof. It is obvious that (19) is provable only if the root of q's subtree is false (and refutable). Thus, one would intuitively expect to be able to map the proof p onto an analogous tree-proof q of roughly the same height. The formal nature of this transformation is trivial, and omitted. □

Definition 3. *Given a natural number* N *, its* **canonical binary representation**, *denoted formally as "* $\underbrace{N}$ *", will be a term of length* $O(\ Log\ N\ )$ *that defines the value* N *using only the constant symbols for the numbers 0, 1 and 2. In particular, let* $b_0, b_1, ...b_m$ *denote a sequence of bits where* $N = \sum_{i=0}^{m} b_i\ 2^i$ *and* $b_m = 1$ *. Then* $\underbrace{N}$ *will be the term:*

$$(b_0 \ + \ 2 \cdot (\ b_1 \ + \ 2 \cdot (\ b_2 \ + \ 2 \cdot (\ ... \ (b_{m-1} \ + \ 2 \cdot b_m \ ))))) $$

Lemma 8. *Suppose* n, z *and* $e > 2$ *satisfy* $Log(z, e) > n$. *Let* Ψ *denote a theorem which states "* $\exists r \ \ Log(\ r \ , \ \underbrace{e}\ \ ) > \underbrace{n}$ *" (where "Log" is obviously encoded using Lemma 1's notation). Then there exists a constant* C *(whose value is independent of* n, z *and* e *) such that a semantic tableaux proof* t *of the theorem* Ψ *from the axiom system* $Q + V$ *can have its proof-length bounded by* $O\{ \ [\ LogLog(z) \ ]^C \ \}$.

Proof. Fitting's definition of semantic tableaux [5] indicates that a proof of the theorem " $\exists r$ Log(r , $\underbrace{e}$) > $\underbrace{n}$ " will store the negation of this sentence in its root. Thus, the root will be essentially:

$$\forall r \ \ \text{Log}(\ r \ , \ \underbrace{e}\ \ ) \leq \underbrace{n} \tag{20}$$

The remainder of t's proof will consist of two fragments, which we shall denote as x_1 and x_0 . The substring x_1 will be the topmost section of the proof-tree t . It will use the fact that $Q+V$ recognizes multiplication as a total function to create a series of newly-created constant symbols u_0 , u_1 , u_2 , $...u_n$, satisfying $u_0 = 2$, $\ u_{i+1} = (u_i)^2$ and having its last term u_n satisfy $z < u_n \leq z^2$.

The second portion of the proof tree t will be called x_0. It will use the last paragraph's $u_n > z$ inequality to formally contradict the root's sentence (stated in (20)). It is fairly straightforward to use the four clauses of the axiom V_2 to construct a formal semantic tableaux proof such that the proof substring x_1 will contain $O \ \{ \ \text{LogLog}(z) \ \}$ nodes, and x_0 will contain $O \ \{ \ [\ \text{LogLog}(z) \ ]^C \ \}$ nodes (for some constant C). □

Lemma 9. *Suppose that the 4-tuple* (y , z , α , k) *satisfies the formula on the left side of the axiom* V_5*'s* $\supset$ *symbol. This formula is duplicated below:*

$$FinAx4(\alpha) \ \wedge \ k \geq \alpha \ \wedge \ Paradox(y, z, \alpha, k) \tag{21}$$

Then there will exist some constant C *whose value is independent of* y, z, α, k *such that there exists a semantic tableaux proof* x *from* α *of the theorem 0=1 , where* x *'s bit-length* $\leq O \{ \ [\ LogLog(z) \ ]^C \ \}$.

Proof. Although the statements of Lemmas 8 and 9 are very different, it turns out that Lemma 9's proof tree x has some very similar components to Lemma 8's tree t . In particular, x will be divided into five fragments, denoted as $x_0 , x_1 , x_2 , x_3 , x_4$. The first two of these five components will be identical to the x_0 and x_1 fragments of Lemma 8's proof tree t .

Some notation will clarify the differences between these two proof trees. If x_i denotes any one of $x_0 , x_1 , x_2 , x_3 , x_4$, let us use the following terminology:

1. The fragment x_i , viewed as an integer encoding a string of bits, will be said to be **z-tiny** iff $\text{Log}(x_i) \leq \text{O}(\ \text{LogLogLog}(z) \ )$ (there are *three* iterations of "Log" attached here to z).

2. The bit-string x_i will be said to be **z-adequately small** when x_i satisfies the inequality $\text{Log}(x_i) \leq O\{ [\text{LogLog}(z)]^C \}$.

Our proof-tree x will be essentially the concatenation of two "z-adequately small" parts, x_0 and x_1 , with three "z-tiny" subtrees, x_2 , x_3 and x_4 . As we already noted, x_0 and x_1 will be identical to their counterparts from Lemma 8's tree t . (Thus, it will turn out that the sole differences between the proof-trees, t and x , will be three very minuscule-sized "z-tiny" fragments.)

We will now formally describe the proof-tree x . Since x is a proof of the theorem 0=1, the root of its proof-tree will obviously be " $0 \neq 1$ ". Immediately below this root will be the fragment x_1 from Lemma 8's proof. It will thus consist of an iterated series of newly-created constant symbols u_0 , u_1 , u_2 , ...u_n , satisfying $u_0 = 2$, $u_{i+1} = (u_i)^2$ and having its last term u_n satisfy

$$z < u_n \leq z^2 \tag{22}$$

At the bottom of the proof fragment x_1 will appear two further sentences. The first will be Equation (14)'s axiom V_4 . (It is permissible to include this axiom in the proof x because Equation (21)'s FinAx4(α) clause indicates that the axiom system α includes this axiom.) The last sentence at the bottom of the segment x_1 is formally defined by Equation (23) at the end of this paragraph. This last sentence will be identical to the axiom V_4 , except that V_4 's six universally quantified variables will now be replaced by six terms, denoted as $\underbrace{\alpha}$, $\underbrace{k}$, $\underbrace{g}$, $\underbrace{h}$, $\underbrace{y}$ and u_n. The following rules will define these terms:

1. Let us recall that Lemma 9's hypothesis indicated that (y , z , α , k) would satisfy Equation (21). The values of $\underbrace{\alpha}$, $\underbrace{k}$ and $\underbrace{y}$ will represent the corresponding quantities in the tuple (y , z , α , k) . Formally, these three terms will be encoded using the "Canonical Binary" form of Definition 3. (The last term u_n will obviously satisfy Equation (22).)
2. Let us recall that Equation (21)'s Paradox(y, z, α, k) formula indicates that y is a proof of the theorem $D^k(\alpha)$. Let h denote $D^k(\alpha)$'s Gödel number. We saw previously in Equation (3) of Section 2 how h was constructed. (It basically was constructed using methods similar to Gödel's construction of a sentence that states *"There is no proof of me"*. In particular, h was constructed by taking a "masking formula" g , that was free in one variable, and letting h denote the unique integer that satisfies the Gödel-like formula of Subst(g, h).) Our formal definition of the terms $\underbrace{g}$ and $\underbrace{h}$ is that they will be canonical binary terms that represent the particular integer values associated with these two numbers g and h .

Thus in a context Items (1) and (2) define the terms $\underbrace{\alpha}$, $\underbrace{k}$, $\underbrace{g}$, $\underbrace{h}$, $\underbrace{y}$ and u_n, the node immediately below V_4 in our proof tree will be identical to V_4 , except that V_4 's six universally quantified variables of α , k , g , h , y and z will be replaced by these corresponding terms. This sentence is shown below.

$$\Upsilon(\underbrace{\alpha}, \underbrace{k}, \underbrace{g}, \underbrace{h}, \underbrace{y}, u_n) \supset \exists h^* \leq \underbrace{h}\ \exists y^* \leq \underbrace{y}\ \exists z^* \leq u_n$$
$$\Upsilon(\underbrace{\alpha}, \underbrace{k}, \underbrace{g}, h^*, y^*, z^*)\ \} \tag{23}$$

First Branch-Split in the Proof x Immediately below the node storing the sentence (23) will occur the first branch-split in x's semantic tableaux proof. This split will be generated by the $\supset$Elimination Rule. The two children of (23)'s sentence are therefore given by equations (24) and (25) below:

$$\exists h^* \leq \underbrace{h}\ \exists y^* \leq \underbrace{y}\ \exists z^* \leq u_n\ \ \Upsilon(\underbrace{\alpha}, \underbrace{k}, \underbrace{g}, h^*, y^*, z^*)\ \} \tag{24}$$

$$\neg\, \Upsilon(\underbrace{\alpha}, \underbrace{k}, \underbrace{g}, \underbrace{h}, \underbrace{y}, u_n) \tag{25}$$

Construction of the Substring x_2 **and the Proof of its z-Tiny Size:** The substring x_2 will represent a closed subtree of z-tiny size that is rooted in Equation (24)'s sentence. The reason it is possible to build x_2 is that the hypothesis of Lemma 9 indicated that Paradox(y, z, α, k) was satisfied. From Lemma 7A, this implies x_2 exists. Moreover, its size must be z-tiny (because Lemma 7B implies x_2 has the same magnitude as y and Lemma 5 indicated y was actually much smaller than z-tiny).

The Remaining Fragments of the Proof-Tree x **and Their Small sizes:** To complete Lemma 9's proof, we must show how the subtree descending from (25) is also closed and adequately small. This sentence is the statement " $\neg\, \Upsilon(\underbrace{\alpha}, \underbrace{k}, \underbrace{g}, \underbrace{h}, \underbrace{y}, u_n)$ " where Υ was defined by (14). After applying the Semantic-Tableaux $\neg$ Elimination Rule to the preceding quoted sentence, our remaining part of the proof-tree is rooted in the sentence

$$\neg\, \text{Subst}(\underbrace{g}, \underbrace{h}) \vee \neg\, \text{SemPrf}_{\overbrace{\alpha}^{k}}(\underbrace{h}, \underbrace{y}, u_n) \tag{26}$$

Applying the $\vee-$Elimination Rule to (26), we get a branch-split generating the following two sibling nodes:

$$\neg\, \text{Subst}(\underbrace{g}, \underbrace{h}) \tag{27}$$

$$\neg\, \text{SemPrf}_{\overbrace{\alpha}^{k}}(\underbrace{h}, \underbrace{y}, u_n) \tag{28}$$

Moreover from Definition 1, it is apparent we can make another branch split below (28) that will generate the sibling nodes given in (29) and (30).

$$\neg\, \text{SemPrf}_{\underbrace{\alpha}}(\underbrace{h}, \underbrace{y}) \tag{29}$$

$$\neg \underbrace{y} < \text{Log}(u_n, \underbrace{k}) \tag{30}$$

Thus to show that the subtree descending from (25) is closed and also sufficiently small, we must analyze the three subtrees that descend from the corresponding sentences (27), (29) and (30). These three subtrees will be called x_3 , x_4 and x_0 , and their size analysis is given below:

1. The quantities g and h from (27)'s formula " Subst($\underbrace{g}$, $\underbrace{h}$) " must be integers less than y (simply because y is a proof of the theorem whose Gödel number $= h$) . Since, Subst is a Δ_0 formula, it then follows from Lemma 6 that the subtree x_3 descending from (27) has a size approximately equal to y's magnitude. The final point is that Lemma 5 indicated that y was actually much smaller than $z-$tiny. Hence, the subtree x_3 must certainly have a z-tiny magnitude (since its length is governed by y's size).
2. The proof that the subtree x_4 descending from (29) is z-tiny is almost identical to Item 1's analysis of x_3 essentially because SemPrf$\underbrace{\alpha}$($\underbrace{h}$, $\underbrace{y}$) is also a Δ_0 formula. Thus once again, Lemma 5 allows us to assume that y (and therefore again also h) are sharply smaller than z-tiny. Also, an obvious analog of Lemma 5 is similarly applicable to α . Hence since all three of SemPrf$\underbrace{\alpha}$($\underbrace{h}$, $\underbrace{y}$)'s terms are sufficiently small, we can again apply Lemma 6 to conclude that the subtree x_4 is z-tiny.
3. The proof that the subtree x_0 descending from (30) is z-adequately small is essentially an immediate consequence of Lemma 8. In particular, the tree t , constructed in Lemma 8's proof, contained a subtree, which was also called x_0, whose structure is identical to the object descending from (30)'s sentence. Thus by our preceding discussion, x_0 is z-adequately small.

The above observations have completed our proof of Lemma 9 because they have shown that each of the five subtrees $x_0 , x_1 , x_2 , x_3 , x_4$ are sufficiently small to satisfy Lemma 9 □

Lemma 10. *The axiom V_5 is valid in the Standard Model of the Natural Numbers.*

Proof Sketch. Almost all the details needed to establish Lemma 10 already appeared in Lemma 9's proof. This is because the formal statement of Lemma 9 is almost identical to the formal statement of axiom V_5 . Thus Lemma 9's formal statement indicated that any tuple (y , z , α , k) satisfying the formula (21) can be mapped onto an element x, representing a semantic tableaux proof of 0=1, whose bit-length is bounded by $O\{ [\text{LogLog}(z)]^C \}$ The point is that V_5's formal statement differs only by indicating that $x < z$.

The additional details to convert Lemma 9's $\text{Log}(x) < O\{ [\text{LogLog}(z)]^C \}$ bound into a strict $x < z$ inequality are extremely routine, albeit a bit tedious. For the sake of brevity, these final details are omitted. □

Finishing the Proof of Theorem 3: The combination of Lemmas 4 and 10 implies Theorem 3's validity because they show all $Q+V$'s axioms are valid in the Standard Model (hence establishing its consistency). □

Recapitulating This Proof: It is now possible to offer a pleasantly short 2-sentence intuitive summary of Theorem 3's proof. The heart of the proof rested on showing the existence of a proof-tree x that was sufficiently small to assure the validity of the axiom V_5's requirements for x. The intuitive reason such an x *must exist* is that its bit-length has the same order of magnitude as Lemma 8's proof-tree t, and Lemma 8 showed the latter was sufficiently small.

5 Final Remarks

Section 1 explained the basic reason for our interest in this subject. It was because our prior papers [23,24] established that significant exceptions to the semantic-tableaux version of the Second Incompleteness Theorem do occur when one transforms Multiplication from a total function into a 3-way relation. (Thus, our prior papers [23,24] showed that some axiom systems can verify **all** Peano Arithmetic's Π_1 theorems while *simultaneously* proving their *Semantic Tableaux* consistency when Multiplication is changed from a total function into a 3-way relation.) Since Theorem 4 isolates a single Π_1 sentence **preventing the same** effect when Multiplication serves as a total function, the combined results of our several papers establish a fairly tight characterization of exactly when the Second Incompleteness Theorem **is and is not** valid for Semantic Tableaux.

It should be emphasized the particular Π_1 sentence V defined by equations (12) through (15) is not the main point, since Section 4 noted many other Π_1 sentences can also satisfy Theorems 3 and 4. Rather what is noteworthy is that this paper has established the first documented example of some Π_1 sentence V with these Second Incompleteness properties.

A slightly longer version of the present article has also been prepared, and it can be emailed to any interested readers. It includes fuller proofs for Lemmas 1 and 10 and a stronger version of Theorem 4 (which indicates that the Semantic Tableaux Incompleteness Effect *is also valid* for axiom systems α *with infinite cardinality).* Moreover, with a little additional work, our variant of the Second Incompleteness Theorem can also be generalized to other cut-free methods of deduction, such as Herbrand deduction, the cut-free version of the Sequent Calculus, and Resolution.

Finally, we wish to end this article on a mildly amusing note. Many readers will smile with amusement when they learn the true reason that the Semantic Tableaux version of the Second Incompleteness Theorem breaks down when Multiplication is changed from a total function into a 3-way relation. It is essentially that Lemma 8 and its short proof then become no longer valid. (Lemma 8 was crucial because Lemma 9's proof used the x_0 and x_1 substrings from Lemma 8's proof as an interim step. Without it, our proof of the Semantic Tableaux version of the Second Incompleteness Theorem *collapses entirely !*)

References

1. Z. Adamowicz, "On Tableaux consistency in weak theories", circulating manuscript from the Mathematics Institute of the Polish Academy of Sciences (1999).

2. J. Benett, Ph. D. Dissertation, Princeton University, 1962.
3. A. Bezboruah & J. Shepherdson, "Gödel's Second Incompleteness Theorem for Q", Journal of Symbolic Logic 41 (1976), pp. 503–512.
4. S. Buss, Bounded Arithmetic, in Proof Theory Lecture Notes, Vol. 3, published by Bibliopolis, Naples, 1986.
5. M. Fitting, First Order Logic and Automated Theorem Proving, Springer-Verlag, 1990.
6. P. Hájek & P. Pudlák, Metamathematics of First Order Arithmetic, Springer-Verlag, 1991.
7. J. Krajícek, Bounded Propositional Logic and Complexity Theory, Cambridge University Press 1995.
8. G. Kreisel, "A Survey of Proof Theory, Part I" in Journal of Symbolic Logic 33 (1968), pp. 321–388, and " Part II" in Proceedings of Second Scandinavian Logic Symposium (1971), pp. 109-170, North Holland Press, Amsterdam.
9. G. Kreisel & G. Takeuti, "Formally Self-Referential Propositions for Cut-Free Classical Analysis and Related Systems", Dissertationes Mathematicae 118 (1974), pp. 1–55.
10. E. Mendelson, Introduction to Mathematical Logic, Wadsworth-Brooks-Cole Math Series, 1987. See page 157 for Q's definition.
11. E. Nelson, Predicative Arithmetic, Princeton University Press, 1986.
12. R. Parikh, "Existence and Feasibility in Arithmetic", Journal of Symbolic Logic 36 (1971), pp. 494–508.
13. J. Paris & A. Wilkie, "Δ_0 Sets and Induction", Proceedings of the Jadswin Logic Conference (Poland), Leeds University Press (1981), pp. 237–248.
14. P. Pudlák, "Cuts Consistency Statements and Interpretations", Journal of Symbolic Logic 50 (1985), pp. 423–442.
15. P. Pudlák, "On the Lengths of Proofs of Consistency", in Collegium Logicum: Annals of the Kurt Gödel Society Volume 2, pp. 65–86, published (1996) by Springer-Wien-NewYork in cooperation with Technische Universität Wien.
16. R. Robinson, "An Essentially Undecidable Axiom System", Proceedings of 1950 International Congress on Mathematics, pp. 729-730, and/or page 157 of ref. [10].
17. R. Smullyan, First Order Logic, Springer-Verlag, 1968.
18. R. Solovay, Private Communications (1994) about Pudlák's main theorem from [14]. See Appendix A of [24] for a 4-page summary of Solovay's idea.
19. R. Statman, "Herbrand's theorem and Gentzen's Notion of a Direct Proof", Handbook on Mathematical Logic, North Holland (1983), pp. 897–913.
20. G. Takeuti, "On a Generalized Logical Calculus", Japan Journal on Mathematics 23 (1953), pp. 39–96.
21. G. Takeuti, Proof Theory, Studies in Logic Volume 81, North Holland, 1987.
22. A. Wilkie & J. Paris, "On the Scheme of Induction for Bounded Arithmetic", Annals on Pure and Applied Logic 35 (1987), pp. 261–302.
23. D. Willard, "Self-Verifying Axiom Systems", Third Kurt Gödel Symp (1993), pp. 325–336, Springer-Verlag LNCS#713. See also [24] for more details.
24. D. Willard, "Self-Verifying Systems, the Incompleteness Theorem & Tangibility Reflection Principle", to appear in the Journal of Symbolic Logic.
25. D. Willard, "Self-Reflection Principles and NP-Hardness", Dimacs Series in Discrete Math&Theoretical Comp Science #39 (1997), pp. 297–320 (AMS Press).
26. D. Willard, "The Tangibility Reflection Principle", Fifth Kurt Gödel Colloquium (1997), Springer-Verlag LNCS#1289, pp. 319–334.
27. C. Wrathall, "Rudimentary Predicates and Relative Computation", Siam Journal on Computing 7 (1978), pp. 194–209.

Redundancy-Free Lemmatization in the Automated Model-Elimination Theorem Prover AI-SETHEO

Joachim Draeger

Institut für Informatik der Technischen Universität München
D–80290 München
Germany
++49 89 289 27918
draeger@informatik.tu-muenchen.de

Abstract. Lemmatization is a promising way for solving "hard" problems with automated theorem provers. However, valuable results can only be reached, if the employed lemmata are restricted to useful records. The automated model-elimination theorem prover AI-SETHEO was designed for using such a controlled lemmatization from the beginning. Mainly responsible for the success of the newest version of AI-SETHEO is the implementation of an algorithm for eliminating lemma redundancies. Here, a lemma f is called redundant to other lemmata $f_1, \ldots, f_n$, if f can be generated within very few inferences from $f_1, \ldots, f_n$. Conflicts between redundacies are resolved with an importance criterion. First experiments with AI-SETHEO give promising results.

1 Introduction

When dealing with difficult problems, automated theorem provers (ATPs) are still inferior to skilled human mathematicians. Obviously, the method of modularization used by humans is better suited for this area of applications than the brute-force search typically perfomed by ATPs [3]. It is suggestive to utilize a mechanized form of modularization like lemmatization [2] in ATPs as well. Technically, this approach can be realized as a procedure which generates unit-lemmata f_i and uses them for constructing the proof of the actual problem P (given as clause set) [1].

In order to avoid a disadvantageous *knowledge-intensive* search [1,9], only probably "useful" lemmata have to be retained in a small knowledge base F. This includes the restriction of the knowledge base elements $f \in F$ to *nontrivial* lemmata, i.e. lemmata requiring many inferences for their proof [2]. Only their use can guarantee an appreciable progress while processing a problem. However, the required nontriviality of all lemmata $f \in F$ with respect to the original problem P does not exclude a possible triviality of f with respect to $P \cup F \setminus \{f\}$. In such a case, f is called *redundant* to $F \setminus \{f\}$. Despite being useful for itself, f is not useful as element of F, because F and $F \setminus \{f\}$ represents the same state of

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 431–435, 2000.

knowledge. Experiments have given an overwhelming evidence that every useless and hence superfluous lemma contained in such a knowledge base leads to a considerable diminuation of the prover performance. Thus, it is suggestive to eliminate these *redundant* lemmata in the knowledge base F.

The paper is organized as follows. Section 2 is devoted to the basic ideas of redundancy elimination. Section 3 describes the implementation of this algorithm. Section 4 contains an assessment of experimental results. The paper closes with new perspectives in section 5.

2 Redundancy Elimination

As defined in the introduction, a lemma f is called redundant to $F \setminus \{f\}$, if f is trivial with respect to $P \cup F \setminus \{f\}$. Aiming at the removal of as many redundant lemmata as possible, a brute-force algorithm will give the best results. Unfortunately, the resulting super-exponential complexity is not tractable in practice. Hence, instead of this brute-force approach a heuristic approach based on a stepwise procedure is pursued. Indeed, a high computational complexity can be avoided in this way; however, some additional complications occur. The most severe one is the occurrence of conflicts.

Let us consider the following situation. If an equality $a = b$ is considered as useful and thus contained in the knowledge base F, it is very probable, that the lemma $b = a$ is an element of F, too. These two lemmata f_1, f_2 are trivial consequences of each other because of the equality axiom $X = Y \Rightarrow Y = X$. It means that f_1, f_2 are redundant to each other. In such cases, the intended redundancy elimination algorithm has to decide, which one of f_1, f_2 should be deleted and which one should remain in the knowledge base. This can be done with an importance evaluation of all lemmata $f \in F$. If a conflict is identified between two lemmata, the lemma with higher importance should retain in F. In this way the redundancies are stepwise eliminated as far as possible using the algorithm described below.

At first all proofs for all $f \in F$ requiring only very few inferences are systematically produced from $P \cup F$. A lemma f is identified as redundant to $F' \subseteq F \setminus \{f\}$, iff a proof exists using only the clauses contained in $P \cup F'$. A lemma f is considered as important, iff f is often usable and rarely producable. This discrimination allows a suitable handling of redundancy conflicts during the actual elimination of the identified redundancies; at its start all flags $u(f)$ labelling a lemma $f \in F$ as undeletable are set to false, i.e. $u(f) = 0$. Let $F_0 := F$ denote the original knowledge base.

The (or a) lemma $f \in \{g \in F \mid u(g) = 0\}$ with the lowest importance $E(f)$ is chosen as best candidate for an elimination. Every lemma $f' \in F_0 \setminus F$ already deleted must be redundant to a set $S_{f'} \subseteq F \setminus \{f\}$; otherwise f is labelled as undeletable, i.e. $u(f) := 1$, and the next best candidate for an elimination is chosen. Now, f is eliminated from F. Let $S_f^1, \ldots, S_f^n \subset F \setminus \{f\}$ denote the lemma sets, to which the lemma f is redundant; the lemmata $\{f_1, \ldots, f_n\} := S_f^i$ contained in the set S_f^i with the highest average importance $\sum_{f \in S_f^i} E(f)/|S_f^i|$ are

marked as undeletable, i.e. $u(f_1) := 1, \ldots, u(f_n) := 1$. The lemmata $f_1, \ldots, f_n$ must remain in the knowledge base, because otherwise f would eventually not be trivial to the resulting final knowledge base anymore. The described elimination is iterated as long as candidates for an elimination exist.

3 Implementation

The proposed redundancy elimination algorithm, which can be utilized in all theorem provers using lemmatization, was implemented in the ATP AI-SETHEO, which was designed for the usage of a controlled lemmatization from the beginning; hence the implementation could be done very easily. AI-SETHEO in its original implementation without redundancy elimination procedure is described in [2]. It is a completely automated system based on the generation, evaluation, and selection of unit-lemmata. Its core is the goal-oriented theorem prover SETHEO [6], which is based on the model elimination calculus [7,8] for first-order clause logic as refined special tableau method. In the model elimination calculus a given query is recursively decomposed into new sub-queries until each of them can be solved either by a unit-lemma of the axiomatization, or by one of the assumptions made during the decomposition. In this way possible decompositions are enumerated until a proof can be constructed [5]. The other two main modules of AI-SETHEO besides the theorem prover SETHEO itself are the lemma generator and the lemma selection procedure. In detail, the lemmatization mechanism including redundancy elimination of AI-SETHEO works as follows.

At first, AI-SETHEO tries to solve the original problem P with help of SETHEO [10] in a certain amount of time. If this attempt is unsuccessful, a set S of additional unit-lemmata is generated in a brute-force way with the DELTA-Iterator [11], which is also based on SETHEO. Then AI-SETHEO evaluates the usefulness of all $f \in S$. This is done by calculating a measure of nontriviality with respect to P, which simply counts the minimal number of inferences contained in the proof of f, and a relevancy measure [4,13]. A lemma f considered as useful in both respects is transferred to the knowledge base F. Now the redundancy elimination is applied to F. For efficiency reasons it is scheduled as two stage process. The first stage consists in the removal of subsumable lemmata. The prover system avoids such specialisations, because general lemmata can finish the actual proof task alike and represent much more powerful aids. This criterion can easily be applied. It makes the application of the general redundancy elimination to F as second filter stage much more easily. The general redundancy elimination is performed as described in section 2. The resulting considerably compressed knowledge base F is appended to the original formula P. Then the whole process is repeated with the modified formula $P \cup F$. The iteration process stops in the case of success or when a time limit is reached. In this way, many more proofs are found than with the application of SETHEO on P [2,4,13].

4 Experiments

The performance of AI-SETHEO without and with redundancy elimination was tested on all hard problems contained in the problem collection TPTP [12]. Here, a problem is called hard, if SETHEO can not solve it within 100 seconds. The very promising results are shown in the following table. The entries in the first three columns give the theory domain in the TPTP, the total number of problems in this domain and the number of hard problems in this domain. The entries in the last three columns give the performances of SETHEO and AI-SETHEO without and with redundancy elimination for an overall time limit of 1000s. All experiments were carried out on Sun Ultra 10.

Domain	Problems Total	Problems Hard	SETHEO	AI-SETHEO - RedElim	AI-SETHEO + RedElim
BOO	68	52	5	1	21
CAT	58	29	2	15	25
COL	163	68	11	10	31
FLD	281	188	8	13	27
GEO	165	103	6	19	48
GRP	375	249	13	24	48
HEN	64	35	1	9	34
LAT	35	34	0	4	7
LCL	281	181	10	41	78
LDA	23	22	0	6	14
NUM	309	285	2	10	14
PUZ	60	15	3	7	6
RNG	100	82	3	5	17
SET	1018	735	23	35	100
Others	1005	421	15	13	28
Total	4005	2499	102	212	498

As can be seen, significantly more problems can be solved with redundancy elimination than without. With respect to 'hard' problems an increase of performance by an average factor 2 was reached; in several domains, the performance increase peaked up to a factor of 3 and more. The amount of the size reduction of the lemma set due to the redundancy elimination algorithm is subject to strong variations; it strongly depends on the actual problem under consideration. Hence for this parameter no typical numbers can be given. Unfortunately, the redundancy elimination is not always advantageous. It prefers the use of the most general version of a lemma, which produces sometimes an unnecessarily large search space compared to a lemma specifically instantiated for the actual problem under consideration. Hence it is possible, that in a few cases the search space is enlarged rather than reduced after application of the redundancy elimination contrary to the intention of this algorithm. Indeed, some solutions found by AI-SETHEO without redundancy elimination were lost in this way.

5 Outlook

Despite the prototypical implementation of the presented algorithm, the results are very promising. Although the proposed methods have been tested using a model elimination theorem prover for first-order predicate calculus, the underlying concepts are more general and can be applied to other logics, calculi, and inference rules as well. The paper confirms the overall tendency, that a better discrimination between useful and useless lemmata leads to a smaller knowledge base, improving the performance of the theorem prover. Indeed, the number of solved problems is considerably higher with application of the redundancy elimination algorithm than without. The full potential of controlled lemmatization has yet to be discovered. Further improvements are possible, e.g. the selection of lemmata on demand controlled by failed proof trials. It can be expected, that the implementation of such techniques will yield further improvements.

6 Acknowledgments

This work is supported by the Deutsche Forschungsgemeinschaft within the Sonderforschungsbereich 342, subproject A5 (PARIS).

References

1. Astrachan, O.; Stickel, M.; 1992. Caching and Lemmaizing in Model Elimination Theorem Provers, in *CADE-92*, LNCS 607
2. Draeger, J.; 1998. Acquisition of Useful Lemma Knowledge in Automated Reasoning, in *AIMSA-98*, LNCS 1480
3. Draeger, J.; 1998. *Modularisierte Suche in Theorembeweisern*. Ph.D. Dissertation, Technical University Munich
4. Draeger, J.; Wolf, A.; 1999. Strategy Parallelism and Lemma Evaluation, in *FLAIRS-99*
5. Goller, C.; 1997. A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems, Ph.D. Dissertation, Technical University Munich
6. Goller, C; Letz, R; Mayr, K; Schumann, J; 1994. SETHEO V3.2: Recent Developments, in *CADE-94*, LNCS 814
7. Loveland, D.; 1968. Mechanical Theorem-Proving by Model Elimination, JACM 15(2)
8. Loveland, D.; 1978. Automated Theorem Proving: a Logical Basis, North-Holland
9. Mitchell, T.; 1983. Learning and Problem Solving, in *IJCAI-83*
10. Moser, M.; et al. 1997. SETHEO and E-SETHEO. The CADE-13 Systems. *JAR* 18(2):237
11. Schumann, J.; 1994. DELTA — A Bottom-up Preprocessor for Top-Down Theorem Provers, in *CADE-94*, LNAI 814
12. Sutcliffe, G.; Suttner, C.; Yemenis, T.; 1994. The TPTP Problem Library, in *CADE-94*, LNAI 814
13. Wolf, A.; Draeger, J.; 1999. Strategy Parallelism and Lemma Evaluation, in *TABLEAUX-99*, LNCS 1617

E-SETHEO: An Automated^3 Theorem Prover

Gernot Stenz[1] and Andreas Wolf[2]

[1] Institut für Informatik der Technischen Universität München
80290 München, Germany
stenzg@in.tum.de
[2] NorCom Information Technology AG
Stefan-George-Ring 6, 81929 München, Germany
awo@norcom.de

Abstract. We have developed a method for strategy evaluation and selection based on test data generated from the problem domain. We present the theorem prover e-SETHEO, which automatically handles training data management, strategy evaluation and selection, and actual proof tasks. We also give some experimental data produced with this system. We address the problem of test set extraction and give an assessment of our work.

1 Introduction

When using systems for *Automated Theorem Proving* (ATP), often tremendously large search spaces have to be examined. Such a search problem is usually solved by a uniform search procedure. In automated deduction, different search strategies may behave significantly different on a given problem. Unfortunately, in general, it cannot be decided in advance which strategy is the best for a given problem. This motivates the competitive use of different strategies, especially when the available resources are restricted. For us, a strategy is one particular way of traversing the search space. Many ways of combining and applying different strategies in parallel have already been studied. Some of these are applicable to automated theorem proving. A discussion of the related work is available in [7] as well as an introduction to the paradigm of strategy parallelism. The system abstract [8] of p-SETHEO, an earlier (and much simpler) version of the prover described in this abstract, may be interesting for the reader, too. Here, we only sketch the key facts. Strategy parallelism is, from a certain point of view, a competition approach. Different strategies are applied to the same problem and the first successful strategy stops all others. Since not all strategies are equally promising or require equal effort, it is advisable to divide the available resources in an adequate way. The selection of more than one search strategy in combination with techniques to partition the available resources such as time and processors is called *strategy parallelism* [7]. Different, competitive agents traverse the same search space via different paths in such a way that the repeated consideration of identical parts is largely avoided. Such a selection of strategies together with a resource allocation for the strategies is called a *schedule*.

2 Implementation

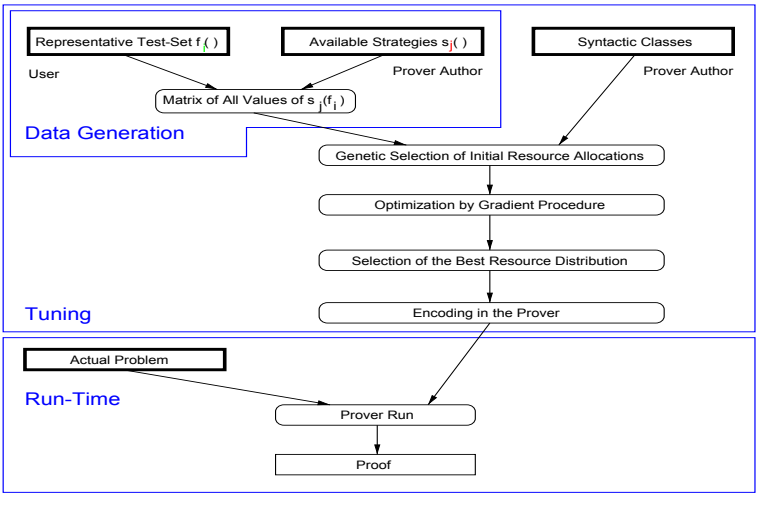
Using SETHEO as the basic underlying inference machine, we have developed p-SETHEO [8], a prototypical strategy parallel theorem prover. This system in

Roy Dyckhoff (Ed.): TABLEAUX 2000, LNAI 1847, pp. 436–440, 2000.

the meanwhile has been further developed into e-SETHEO[1], the most important improvements being the *full automation of the prover configuration process* and the augmentation by the new E prover, a superposition calculus equality prover [3]. We have used e-SETHEO to collect experimental data and successfully participated with that system in the ATP system competition at the CADE-16 conference. While in the early stages of development nearly all stages had been variants of SETHEO (that is, e-SETHEO would invocate different instantiations of SETHEO with different parameterizations), in the meanwhile we have begun to incorporate a much wider variety of different strategies covering special problem classes. E-SETHEO is able to incorporate all state of the art ATP systems, provided these ATP systems adhere to minimum standards regarding issues such as resource allocation or input-output behaviour.

3 Prover Configuration

E-SETHEO has been designed to automatically perform the prover configuration for the given problem set. A principal scheme of the functionality of the system is given in the Figure below.



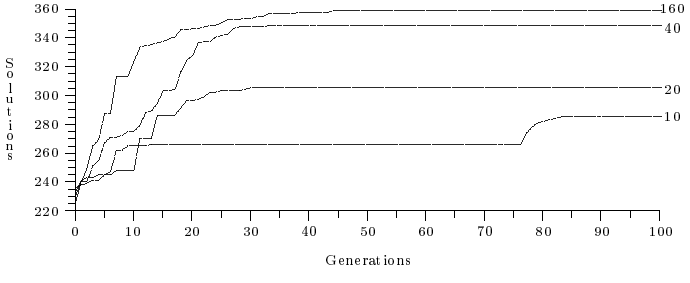
We capture the strategy allocation problem by using a set of training examples from the given domain and optimizing the admissible strategies for this training set using a genetic algorithm [1] followed by the application of a gradient procedure to the best individuals. Providing the necessary training data, however, is a very extensive task. We want to determine a combination of strategies which solves a maximal number of problems from the training set within the given resources. To compute this combination, we have to determine the solution times on all problems in the training set for all admissible strategies. We perform *automated data generation* on a loosely connected workstation cluster.

[1] e-SETHEO differs from its sister system AI-SETHEO in that it employs different single provers in parallel, whereas AI-SETHEO pursues a modularization of big proof tasks into smaller ones.

Using a system of independent distributed software components, we execute the necessary prover runs and generate the performance data matrices for the prover tuning phase. This tuning is done in the course of the *automated strategy evaluation and schedule determination*. The multitude of settings of the basic SETHEO inference machine and the considerable number of additional prover tools employed by e-SETHEO result in a vast number of different configurations in which the prover system can be used. It is obvious that it is not feasible to test all these possible configurations for their performance on a given problem domain. Using heuristics, intuition and experience, a number of about one hundred of these configurations has been identified as potentially useful and implemented as strategies. We obtain the strategy selection and resource partitioning using a number of pseudo-optimal solutions we acquire by evolving a set of schedules for each problem class by the genetic algorithm described in [4]. The best of these schedules in each class are selected for refinement by applying the gradient method explained in [9]. This process results in a set of schedules, one for each syntactic class, that are then used for configuring e-SETHEO. The run-time *automated proof search* follows the principles already used by p-SETHEO [8].

4 Experimental Results

Our experiments were conducted in two phases. First we intended to verify the feasibility of the approach described above with reduced problem and strategy sets. Therefore we used the 547 eligible TPTP problems [6] of the CADE-15 ATP systems competition as our training data set. Our prover p-SETHEO [8] employed 91 different strategies at that time, these formed our strategy set. We extracted all these strategies and ran each strategy on all problems using the standard sequential SETHEO [2]. The successful results of all those runs were collected. 398 problems can be solved by at least one of the strategies in at most 300 seconds. Then we ran the genetic algorithm described in [4] followed by the gradient procedure from [9] on the collected data.



The Figure above shows the average number of problems solved after 0 to 100 generations for 10, 20, 40, and 160 individuals (numbers at the curves) in 300 seconds on a single processor system. Our experimental results indicated that the combination of a genetic approach and a gradient one is extremely useful for automatically configuring a strategy parallel theorem prover, and therefore we tested each of the 110 strategies on all 4004 TPTP problems. These test runs were conducted on a cluster of Sun Ultra 10/60 workstations of 384 MB memory

each with a time limit of 300 seconds per problem and strategy and took about two months. Using the data from these runs we generated a set of schedules for the different syntactic problem classes. After having configured e-SETHEO with these schedules, we ran e-SETHEO on the TPTP v2.2.0 again, with 1 processor, and 300 seconds resources. The results of these tests are shown in the Table below. A detailed description of the experiments can be found in [4] and [5].

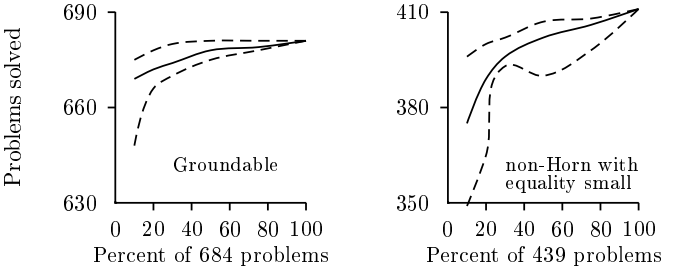
Problem Class	problems in class	# solved by some strategy	# solved by best strategy	# solved by schedule	strategies in schedule
Groundable	753	685	410	682	4
Unit Equality	446	365	330	341	5
Pure Equality	132	96	85	93	6
Horn w. Equality	226	183	175	183	3
Horn w/o Equality	373	293	275	291	3
non-Horn w/o Eq. (large)	268	98	74	96	6
non-Horn w/o Eq. (small)	266	227	191	227	6
non-Horn w. Eq. (large)	841	140	78	132	9
non-Horn w. Eq. (small)	699	445	317	416	12
TOTAL	4004	2532	2201	2461	-

5 Test Set Extraction

The applicability of the system described in the previous sections strongly depends on the amount of time required to configure the prover. Given an application environment, training the prover system on the entire domain does not make sense, or even more probably, is not possible. So, it is necessary to minimize the training period as much as possible while maintaining a sufficient overall performance of the final system. We decided to try a simple randomized approach to this problem: Using the given categorization of problems into problem classes, we randomly selected subsets of varying sizes of each class and used these subsets for configuring our prover system in the way specified in the previous sections. We tested the validity of this approach by doing the following for each problem class: We extracted random subsets consisting of 10, 20, 30, 50 and 75% of the problems in the problem class, generated and optimized a schedule for each such subset, and finally tested the performance of all schedules on the entire problem class. This entire process was repeated 10 times and the results were compared with our optimized schedules based on the entire TPTP library. The results for the class of groundable problems and for the class of non-Horn problems with equality given in a short formula are depicted in the Figure below.

The diagrams contain three curves. The upper/lower dashed curve indicates the performance of the best/worst of the 10 schedules, and the solid curve describes the median element in each schedule set. It becomes apparent from the data displayed in the Figure that the problem classes differ in the homogeneity of their problems with respect to search space behaviour. The Figure shows the performance of schedules depending on the size of the training sets for a homogeneous and a non-homogeneous formula class. While for the class of groundable

problems a test set of 10% is enough to obtain 98% of the performance of the schedule based on the full problem domain (with the best of the 10 schedules), for the other class (small non-Horn w. eq.) to achieve those 98% performance the test set has to comprise 30% of the problem domain.



6 Assessment

The search procedures of ATP systems are not generic enough to be applied to new problems without modification. Therefore we consider the adaption of the prover system to the problem domain a basic necessity. Due to the generic nature of our tuning mechanism we are optimistic that we can adapt e-SETHEO to perform well on even large arbitrary sets of application problems. A problem that remains to be solved is the categorization of problems into problem classes. Fixed problem classes have worked reasonably well in our given research context, but still a more generic approach to the categorization problem should be investigated. There are large formula classes without an apparent inner structure implied by syntactic characteristics. The categorization of problems according to their syntax remains a reasonable idea but has to be supplemented by other methods. One might use the existing knowledge about the problems in a given class to make reasonable subdivisions. Future work is also required on the refinement of the test set extraction. How can we determine what a representative problem from a certain domain should look like? Such knowledge might reduce the effort necessary to configure the prover system as well as help minimizing the required test set size.

References

1. J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1992.
2. M. Moser, et al. SETHEO and E-SETHEO. The CADE-13 Systems. *JAR*, 18(2):237–246, 1997.
3. S. Schulz. System Abstract: E 0.3. *CADE-16*, LNAI 1632, pp. 297–60. 1999.
4. G. Stenz and A. Wolf. Strategy Selection by Genetic Programming. *FLAIRS-12*, pp. 346–350. 1999.
5. G. Stenz and A. Wolf. E-SETHEO: Design, Configuration and Use of a Parallel Automated Theorem Prover. *Australian AI-99*, LNAI 1747, pp. 231–243. 1999.
6. G. Sutcliffe, et al. The TPTP Prob. Lib. *CADE-12*, LNAI 814, pp. 252–266. 1994.
7. A. Wolf and R. Letz. Strategy Parallelism in ATP. *IJPRAI*, 13(2):219–245, 1999.
8. A. Wolf. p-SETHEO: Strategy Parallelism in ATP. *TABLEAUX-98*, LNAI 1397, pp. 320–324. 1998.
9. A. Wolf. Strategy Selection for Automated Theorem Proving. *AIMSA-98*, LNAI 1480, pp. 452–465. 1998.

Author Index