Resolution-Based Decision Procedures for Subclasses of First-Order Logic

Dissertation

zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät
der Universität des Saarlandes
von

Ullrich Hustadt

Saarbrücken, 1999

Dekan:	Prof. Dr. Wolfgang Paul
Berichterstatter:	Prof. Dr. Harald Ganzinger
	Priv. Doz. Dr. Hans Jürgen Ohlbach
Tag des Kolloquiums:	08. November 1999

Abstract

This thesis studies decidable fragments of first-order logic which are relevant to the field of non-classical logic and knowledge representation. We show that refinements of resolution based on suitable liftable orderings provide decision procedures for the subclasses $\mathbf{E}^+$, $\overline{\mathrm{K}}$, and $\overline{\mathrm{DK}}$ of first-order logic. By the use of semantics-based translation methods we can embed the description logic $\mathcal{ALB}$ and extensions of the basic modal logic K into fragments of first-order logic. We describe various decision procedures based on ordering refinements and selection functions for these fragments and show that a polynomial simulation of tableaux-based decision procedures for these logics is possible. In the final part of the thesis we develop a benchmark suite and perform an empirical analysis of various modal theorem provers.

Zusammenfassung

Diese Arbeit untersucht entscheidbare Fragmente der Logik erster Stufe, die mit nicht-klassischen Logiken und Wissensrepräsentationsformalismen im Zusammenhang stehen. Wir zeigen, daß Entscheidungsverfahren für die Teilklassen $\mathbf{E}^+$, $\overline{\mathrm{K}}$, und $\overline{\mathrm{DK}}$ der Logik erster Stufe unter Verwendung von Resolution eingeschränkt durch geeigneter liftbarer Ordnungen realisiert werden können. Durch Anwendung von semantikbasierten Übersetzungsverfahren lassen sich die Beschreibungslogik $\mathcal{ALB}$ und Erweiterungen der Basismodallogik K in Teilklassen der Logik erster Stufe einbetten. Wir stellen eine Reihe von Entscheidungsverfahren auf der Basis von Resolution eingeschränkt durch liftbare Ordnungen und Selektionsfunktionen für diese Logiken vor und zeigen, daß eine polynomielle Simulation von tableaux-basierten Entscheidungsverfahren für diese Logiken möglich ist. Im abschließenden Teil der Arbeit führen wir eine empirische Untersuchung der Performanz verschiedener modallogischer Theorembeweiser durch.

Extended abstract

This thesis investigates decision procedures for description logics, propositional modal logics and fragments of first-order logic related to these non-classical logics. It is not the aim of this thesis to develop novel calculi for these logics, but to exploit the possibilities of resolution refined by orderings and selection functions to obtain decision procedures. To this end, we utilise the framework of Bachmair and Ganzinger [8].

To demonstrate the various techniques which will be used throughout the thesis, we first consider the class $\mathbf{E}^+$, a well-investigated solvable class of clauses. We show how to use renaming to transform formulae and clauses into a suitable more "well-behaved" form; how to use ordering refinements to restrict the application of resolution and factoring; how to prove termination of an ordering refinement; and how to establish relationships between various decision procedures by means of simulation.

The feature which makes the class $\mathbf{E}^+$ interesting is the presence of compound ground terms in the clauses which has the consequence that no liftable ordering decides $\mathbf{E}^+$, when applied a priori. We show that a satisfiability equivalence preserving transformation can be used to simplify the structure of clauses and that applying this transformation during proof search maintains the simpler structure. We prove that there exists a bound on the number of applications of the transformation during proof search. Thus, the termination of the procedure is not affected by dynamic applications of the transformation. Due to the simpler structure of the clauses all ordering refinements proposed in the literature [27, 28, 130] now determine the same resolution-based decision procedure for $\mathbf{E}^+$. In addition, we are able to show that despite the changes to the structure of the clauses induced by the transformation it is possible to polynomially simulate a decision procedure based on a non-liftable ordering by an ordering refinement based on a liftable ordering.

We also consider the class $\overline{\mathrm{K}}$ which is based on the class K introduced by Maslov [97]. The class $\overline{\mathrm{K}}$ not only covers many of the classical decidable fragments of first-order logic, but also many modal logics and description logics. Until now only for a subclass of $\overline{\mathrm{K}}$ a resolution-based decision procedure was available, which is based on a non-liftable ordering refinement [39]. Like for the class $\mathbf{E}^+$ we are able to devise a structural transformation for the clauses in $\overline{\mathrm{KC}}$ the class of clause sets corresponding to $\overline{\mathrm{K}}$, to obtain so-called strongly CDV-free clauses. We show that with an appropriate liftable ordering refinement of resolution, the derived clauses remain strongly CDV-free without further intervention. We prove that given a finite signature there exist only finitely many strongly CDV-free clauses. It follows that the procedure terminates for any finite set of clauses in $\overline{\mathrm{KC}}$.

From the field of knowledge representation we consider description logics focussing in particular on the satisfiability problem for knowledge bases of the logic $\mathcal{ALB}$ ("Attribute language over Boolean algebras on concepts and roles"). $\mathcal{ALB}$ is an extension of the well-known description logic $\mathcal{ALC}$ [123] by the operations complement, intersection, and union on binary relations. Using a semantics-based translation we are able to map $\mathcal{ALB}$ knowledge bases into first-order formulae. The clausal form of the first-order formulae we obtain from the class of so-called DL-clauses, which is a subclass of $\overline{\mathrm{KC}}$. Due to the more stringent structure of the clauses every ordering refinement compatible with the multiset extension of the strict subterm ordering will result in a resolution-based decision procedure for the satisfiability problem of a set of DL-clauses.

While for the logic $\mathcal{ALB}$ no alternative decision procedure is known, there exist a variety of tableaux-based decision procedures for the description logic $\mathcal{ALC}$ and a range of its extensions. It is therefore interesting to study the relative length of proofs of a resolution-based decision procedure compared to those of a tableaux-based decision procedure. We consider a resolution-based procedure which is based on a particular selection function of negative literals. We show for every refutation of a standard tableaux-based procedure there exists a refutation of this resolution-based procedure which is at most twice as long. Thus, the resolution-based procedure is able to polynomially simulate the tableaux-based procedure.

Closely related to description logics are propositional modal logics. In particular, the multi-modal logic $\mathsf{K}_{(m)}$ is a notational variant of the description logic $\mathcal{ALC}$. The major difference between description logics and modal logics is the presence of additional axiom schemata in extensions of K. We investigate extensions of K with the axiom schemata $\mathsf{4}$, $\mathsf{5}$, B, D, T, and their combinations. We consider two semantics-based translation methods for modal logic into first-order logic, the relational translation and the semi-functional translation [101]. As far as the relational translation of modal formulae in the modal logic K and its extension by an arbitrary combination of the axiom schemata B, D, and T is concerned, the clausal form of the first-order formulae belong to the class of DL-clauses. The same holds for the semi-functional translation. So, the decision procedure we devised for the class of DL-clauses can also be applied to modal logics. Additionally, the resolution-based procedure based on a selection function we described in the same context allows for a polynomial simulation of prefix tableaux calculi for these modal logics. For extensions of the modal logic $\mathsf{K4}$ a decision procedure based on the ordered chaining calculus is described by Ganzinger, Hustadt, Meyer und Schmidt [46].

For extensions of $\mathsf{K5}$ and $\mathsf{K4}$ the semi-functional translation yields clause sets which do not belong to one of the classes we have considered before. While for extensions of $\mathsf{K5}$ by the axiom schemata $\mathsf{4}$, D, and T there nevertheless exists a decision procedure based on an ordering refinement, for extensions of $\mathsf{K4}$ by the axiom schemata D and T only a combination of an ordering refinement and a selection function provides the basis for a terminating resolution procedure. This procedure is of particular interest since there is no obvious relation between inference by this procedure and inference found in tableaux-based calculi for $\mathsf{K4}$.

The final part of the thesis is concerned with the empirical performance analysis of modal theorem provers. We compare the theorem provers KSAT, $\mathcal{KRIS}$, the Logics Workbench, and a decision procedure based on the optimised functional translation [107] and the first-order theorem prover SPASS [134]. The aim of our comparison is not to show the superiority of a particular theorem prover, but to determine which collections of benchmark formulae are suitable for an empirical performance analysis and which techniques have positive or negative effects on the performance of modal theorem provers. We present guidelines for the generation of suitable collections of randomly generated benchmark formulae and show how the syntactical characteristics of these formulae influence the behaviour of modal theorem provers. The analysis shows that techniques for redundancy elimination, simplification, and search control have an important impact on the performance of modal theorem provers.

Ausführliche Zusammenfassung

Die vorliegende Arbeit untersucht Entscheidungsverfahren für Beschreibungslogiken, Modallogiken und Fragmente der Logik erster Stufe, die mit diesen Logiken in Zusammenhang stehen. Im Vordergrund steht dabei nicht die Entwicklung völlig neuartiger Verfahren, sondern die Verwendung von Ordnungs- und Selektionseinschränkungen in resolutionsbasierten Beweisverfahren. Wir bedienen uns dazu des Ansatzes von Bachmair und Ganzinger [8].

Wir betrachten zunächst die Klasse $\mathbf{E}^+$, um die in dieser Arbeit verwendeten Techniken an Hand einer der am besten untersuchten Klasse von Klauseln zu demonstrieren. Insbesondere zeigen wir, wie strukturelle Transformation verwendet werden kann, um die Struktur von Klauseln zu vereinfachen; wie Ordnungseinschränkungen verwendet werden können, um die Anwendung von Resolution einzuschränken; wie man die Terminierung solcher Ordnungseinschränkungen beweist; und wie man die relative Beweislänge unterschiedlicher Verfahren untersucht.

Die Klasse $\mathbf{E}^+$ hat dadurch Bedeutung erlangt, daß durch die Präsenz von komplexen Grundtermen in den Klauseln die a priori Anwendung liftbarer Ordnung nicht zu einem Entscheidungsverfahren führt. In der Arbeit wird gezeigt, daß es mittels struktureller Transformation möglich ist, die Struktur der Klauseln in den betrachteten Klauselmengen zu vereinfachen und diese einfachere Struktur durch Anwendung der Transformation während der Beweissuche zu erhalten. Die Terminierung des Verfahrens wird durch die dynamische Anwendung der Transformation nicht beeinträchtigt. Durch die einfachere Form der Klauseln bedingt, führen nun alle bisher in der Literatur zur Behandlung von $\mathbf{E}^+$ vorgeschlagenen Ordnungseinschränkungen [27, 28, 130] zu dem gleichen Entscheidungsverfahren für die Klasse $\mathbf{E}^+$. Wir sind zudem in der Lage zu zeigen, daß ein Entscheidungsverfahren für die Klasse $\mathbf{E}^+$ basierend auf einer liftbaren Ordnung in der Lage ist Entscheidungsverfahren, die auf einer nicht-liftbare Ordnung basieren, polynomiell zu simulieren.

Wir wenden uns dann der Klasse $\overline{\mathrm{K}}$ zu. Diese Klasse beruht auf der von Maslov [97] definierten Klasse K. Die Klasse $\overline{\mathrm{K}}$ umfaßt nicht nur viele der klassischen entscheidbaren Fragmente der Logik erste Stufe, sondern auch viele der im Bereich der Wissensrepräsentation wichtigen Beschreibungslogiken und Modallogiken. Bisher existierte nur für eine Teilklasse von $\overline{\mathrm{K}}$ ein resolutionsbasiertes Entscheidungsverfahren unter Verwendung einer nicht-liftbaren Ordnung [39]. Wieder lassen sich die Klauseln in der zu $\overline{\mathrm{K}}$ korrespondierenden Klauselklasse $\overline{\mathrm{KC}}$ durch eine strukturelle Transformation in eine geeignete Form bringen, die bei Verwendung einer geeigneten liftbaren Ordnung unter Resolution erhalten bleibt. Wir zeigen, daß es über einer endlichen Signatur nur endlich viele Klauseln dieser Form geben kann, woraus die Terminierung des Verfahrens folgt.

Im Bereich der Beschreibungslogiken untersuchen wir das Erfüllbarkeitsproblem für Wissensbasen über der Logik $\mathcal{ALB}$ („Attribute language over Boolean algebras on concepts and roles“), einer Erweiterung der bekannten Beschreibungslogik $\mathcal{ALC}$ [123] um die Operationen Komplement, Durchschnitt, und Vereinigung auf binären Relationen. Unter Verwendung einer semantikbasierten Übersetzung können wir Wissensbasen über $\mathcal{ALB}$ in Formeln der Logik erster Stufe abbilden. Die Klauselform dieser Formeln bildet die Klasse der DL-Klauseln, einer Teilklasse von $\overline{\mathrm{KC}}$. Durch die einfachere Struktur von DL-Klauseln bedingt, liefert jede Ordnungseinschränkung, die kompatibel zur Multimengenerweiterung der strikten Subtermordnung ist, ein resolutionsbasiertes

Entscheidungsverfahren für das Erfüllbarkeitsproblem von Mengen von DL-Klauseln.

Während für die Logik $\mathcal{ALB}$ bisher kein anderes Entscheidungsverfahren bekannt ist, gibt es für die Beschreibungslogik $\mathcal{ALC}$ und einige ihrer Erweiterungen eine Reihe von tableauxbasierten Entscheidungsverfahren. Es ist deshalb interessant die relative Beweislänge von tableauxbasierten und resolutionsbasierten Entscheidungsverfahren zu untersuchen. Für ein Resolutionsverfahren welches auf einer reinen Selektionseinschränkung basiert können wir zeigen, daß es zu jeder Widerlegung in einem tableauxbasierten Verfahren eine Widerlegung in diesem Resolutionsverfahren gibt, die höchstens doppelt so lang ist. Daraus folgt, daß dieses Resolutionsverfahren ebenso effizient ist wie ein tableauxbasiertes Verfahren.

Nahe verwandt zu Beschreibungslogiken sind aussagenlogische Modallogiken. Insbesondere ist die Multimodallogik $\mathsf{K}_{(m)}$ eine notationelle Variante der Beschreibungslogik $\mathcal{ALC}$. Der wesentliche Unterschied zwischen diesen beiden Klassen von Logiken sind zusätzliche Axiomenschemata in Erweiterungen von K. Die hier untersuchten Schemata sind $\mathsf{4}$, $\mathsf{5}$, B, D, T, und deren Kombinationen. Wir betrachten zwei verschiedene semantikbasierte Übersetzungsverfahren von Modallogiken in die Logik erster Stufe, die relationale Übersetzung und die semi-funktionale Übersetzung [101]. Für die relationale Übersetzung gehören die Klauseln, die wir für die Modallogik K und deren Erweiterung mit einer beliebigen Kombination der Schemata B, D, und T erhalten zur Klasse der DL-Klauseln. Dasselbe gilt bei der semi-funktionalen Übersetzung. Die entsprechenden Entscheidungsverfahren lassen sich deshalb auch auf diese Modallogiken anwenden. Insbesondere liefert auch hier das auf einer Selektionseinschränkung basierende Entscheidungsverfahren eine polynomielle Simulation von Präfixtableauxverfahren für die entsprechenden Modallogiken. Für Erweiterungen der Modallogik $\mathsf{K4}$ wurde ein Entscheidungsverfahren in der Arbeit von Ganzinger, Hustadt, Meyer und Schmidt [46] vorgestellt.

Die Erweiterungen von $\mathsf{K5}$ und $\mathsf{K4}$ liefern unter der semi-funktionalen Übersetzung Klauselmengen, die nicht mehr zu einer der vorher behandelten Klassen gehören. Während für die Erweiterungen von $\mathsf{K5}$ mit den Axiomenschemata $\mathsf{4}$, D, und T noch eine reine Ordnungseinschränkung für eine Entscheidungsverfahren ausreichend ist, basiert das Entscheidungsverfahren für Erweiterungen von $\mathsf{K4}$ um die Axiomenschemata D und T auf einer Kombination von Ordnungseinschränkung und Selektionseinschränkung. Das für Erweiterungen von $\mathsf{K4}$ vorgestellte Verfahren ist insbesondere deshalb interessant, da es keinen unmittelbaren Zusammenhang zwischen den Ableitungen dieses Verfahrens und Ableitungen von tableauxbasierten Verfahren gibt.

Im letzten Teil der Arbeit führen wir eine empirische Untersuchung von modallogischen Theorembeweisern durch. Wir vergleichen dabei die Theorembeweiser KSAT, $\mathcal{KRIS}$, die Logics Workbench, und ein Entscheidungsverfahren basierend auf der optimiert funktionalen Übersetzung [107] und dem Theorembeweiser SPASS [134] für die Logik erster Stufe. Das Ziel dieser Untersuchung ist nicht, die Überlegenheit eines dieser Theorembeweiser zu zeigen, sondern festzustellen, welche Sammlungen von Beispielformeln als Grundlage empirischer Untersuchungen geeignet sind und welche Techniken positive oder negative Effekte auf die Performanz der Theorembeweiser haben. Wir erarbeiten Richtlinien zur Erzeugung geeigneter Sammlungen von zufällig erzeugten modallogischen Formeln und zeigen wie die Eigenschaften solcher Formeln das Verhalten von Theorembeweisern beeinflußen können. Es zeigt sich, daß Techniken der Redundanzeliminiation, Simplifikation, und der Steuerung der Beweissuche wesentlichen Anteil an der beobachtbaren Performanz haben.

Acknowledgements

I am indebted to Hans Jürgen Ohlbach for introducing me to the field of modal logic and, in particular, translation-based approaches for reasoning in modal logics. I am grateful to Harald Ganzinger for introducing me to automated theorem proving and for his constant support during the past ten years. In particular, I wish to thank Renate Schmidt for her encouragement, inspiration, and criticism. The work with all three has been a tremendous challenge and zest at the same time.

It has been a joy to work at the Max-Planck-Institut für Informatik, not only due the excellent working conditions, but foremost due to all the people helping to create a friendly and inspiring environment. I thank Hubert Baumeister, Alexander Bockmayr, Christoph Meyer, Andreas Nonnengart, Jürgen Stuber, Miroslava Tzakova, Uwe Waldmann, and Christoph Weidenbach for many fruitful discussions and valuable advice.

Saarbrücken, March 1999 Ullrich Hustadt

Contents

Introduction **1**

1 Basic notions **7**

1.1 Preliminary definitions 7
1.2 The resolution calculus 11
1.3 Structural transformation 13
1.4 Simulation 14
1.5 Solvable classes 15

2 The class E+ **19**

2.1 The class $\mathbf{E}^+$ and variable uniform clauses 20
2.2 Resolution and factoring on variable uniform clauses 23
2.3 On the relationship between the decision procedures for $\mathbf{E}^+$ 34
2.4 Related work 36
2.5 Conclusion 37

3 The classes K and K-bar **39**

3.1 The class $\overline{\mathrm{K}}$ 40
3.2 The class $\overline{\mathrm{KC}}$ and quasi-regular clauses 42
3.3 Resolution and factoring on quasi-regular clauses 44
3.4 A decision procedure for $\overline{\mathrm{KC}}$ 52
3.5 The class $\overline{\mathrm{DK}}$ 59
3.6 Related Work 62
3.7 Conclusion 65

4 Description logics **67**

4.1 Syntax and semantics of description logics 67
4.2 $\mathcal{ALB}$ and DL-clauses 71
4.3 Resolution and factoring on DL-clauses 76
4.4 Variations of $\mathcal{ALB}$ 79
4.5 A decision procedure for $\mathcal{ALB}_{\mathrm{D}}$ based on selection 84

4.6 Conclusion . . . 92

5 Modal logics **95**

5.1 Syntax and semantics of modal logics . . . 96
5.2 The relational and optimised functional translation . . . 101
5.3 The semi-functional translation . . . 104
5.4 Conclusion . . . 116

6 Performance evaluation **117**

6.1 Analytical versus empirical performance studies . . . 117
6.2 Scientific testing and scientific benchmarking . . . 119
6.3 Theorem provers for the modal logic $\mathsf{K}_{(m)}$ . . . 120
6.4 A benchmark suite for scientific benchmarking . . . 130
6.5 Evaluation of theorem provers and a benchmark suite . . . 133
6.6 Broadening the evaluation . . . 141
6.7 Where are the hard problems? . . . 143
6.8 Conclusion . . . 148

Conclusion **153**

A Properties of regular terms and literals **155**

B Glossary of solvable classes **163**

B.1 The classes $\mathbf{E}^+$ and $\mathbf{E}_1$ . . . 163
B.2 The classes $\overline{\mathrm{K}}$, $\overline{\mathrm{DK}}$, $\overline{\mathrm{KC}}$, and $\overline{\mathrm{DKC}}$ . . . 163
B.3 DL-clauses and fluted DL-clauses . . . 164
B.4 Small SF-clauses and SF-clauses . . . 165

Bibliography **167**

List of figures **177**

List of tables **178**

Index of subjects **179**

Index of symbols **187**

Introduction

Since the advent of automated theorem proving, research on decision procedures has been in the focus of this field. In 1960 Wang implemented three decision procedures, that is, procedures that were sound, complete, and terminating: a procedure for deciding validity in propositional logic, a procedure for selecting (deriving) theorems in propositional logic, and a procedure based on the sequent calculus for deciding validity in a fragment of the full first-order predicate calculus. The decidable fragment Wang focused on, **AE** predicate logic, contains the first-order formulae in prenex form with quantifier prefix $\forall x_1 \ldots \forall x_m \, \exists y_1 \ldots \exists y_n$.

At the beginning of the sixties, researchers started to build theorem provers for full first-order logic. One of the most important steps in this effort was the work in 1965 of Robinson on *resolution* [117]. Robinson showed that an automated theorem prover with resolution as the only rule of inference was complete for first-order logic. This avoided the need for choices between different inference rules, thus providing a basis for a straightforward implementation of a first-order theorem prover.

Soon researchers started to investigate whether the resolution calculus can be restricted in a way that preserves the completeness of the calculus for a given subclass of first-order logic while also guaranteeing termination. Kallick [87] was first to show, in 1968, that this is in fact possible. His decision procedure is for testing the satisfiability of formulae with quantifier prefix $\forall x_1 \, \forall x_2 \, \exists y_1$. However, the resolution procedure of Kallick is incomplete for full first-order logic.

In 1964, Maslov [96] independently invented the inverse method for automated theorem proving in first-order logic and applied this method to provide a decision procedure for a fragment of first-order logic, today known as *Maslov's Class* K. Based on his result it is possible to obtain decision procedures for a variety of classical fragments of first-order logic including the initially Ackermann class, the monadic class, the initially extended Skolem class, and the initially extended Gödel class.

A major breakthrough was the work of Joyner [86] in 1976. Unlike Kallick who used a specialised theorem prover in his approach, Joyner aimed at the development of resolution strategies which are complete in the general case and provide a decision procedure for various decidable fragments of first-order logic. In this way, he also hoped to gain insight into the reasons for the decidability of these fragments. He presents three decision procedures using a refinement of resolution based on A-orderings for the Ackermann class, the Monadic class, and the initially extended Skolem class. Notably, these classes are also covered by Maslov's inverse method.

In the following decades the use of ordering refinements for the development of resolution-based decision procedures for fragments of first-order logic has become standard. An overview of the results obtained by the beginning of the nineties is given in a monograph by Fermüller, Leitsch, Tammet and Zamov [39]. It contains resolution-based decision procedures for the classes

$\mathcal{PVD}$, $\mathcal{OCCI}$, $\mathbf{E}^+$, $\mathbf{E}_1$, $\mathbf{M}^+$, One-Free, the Bernays-Schönfinkel class, and a subclass of the dual of Maslov's class K.

Decidability issues play a prominent role in most fields of computer science. We will study solvable classes relevant for the field of *knowledge representation*, where decidability is commonly regarded to be a minimum prerequisite. The initial motivation for the proposal of a new formalism for knowledge representation, as opposed to using existing formalisms like first-order logic, has been the *restricted language thesis* by Levesque and Brachman [92]: General-purpose knowledge representation systems should restrict their languages by omitting constructs which require non-polynomial worst-case response times for their inferential services. However, it became obvious that there is no formalism which is at the same time tractable and reasonably expressive [32]. Consequently, research in the area of knowledge representation has shifted to more expressive logics, which are intractable but still decidable. The *description logic* $\mathcal{ALC}$ and various extensions have been of particular interest. $\mathcal{ALC}$ can be seen as the smallest extension of propositional logic with additional quantificational operators which make the logic suitable for describing "structured objects."

The fact that $\mathcal{ALC}$ can be viewed as a notational variation of the *basic multi-modal logic* $\mathsf{K}_{(m)}$, links description logics to popular logics in the field of non-classical logics [118]. What distinguishes description logics from modal logics, is the presence of knowledge bases in the first and the presence of modal axiom schemata in the second. In the field of modal logic, a theme of current research is the investigation and understanding of modal logics in the setting of first-order logic [4]. It is well-known that the basic modal logic K can be translated into the two-variable fragment of first-order logic (FO^2). Although the satisfiability problem for FO^2 is decidable and every satisfiable formula has a model with finite domain, FO^2 lacks many of the model theoretic and proof theoretic properties of K. An alternative fragment of first-order logic, the guarded fragment, which seems to be better suited, has been proposed by Andréka, van Benthem, and Németi [4]. The definition of the guarded fragment is based on the observation that quantifiers in the translations of modal formulae occur only in guarded form. The guarded fragment is decidable and shares the finite model property and the tree model property with modal logic [4, 60].

However, if our main concern is to find fragments of first-order logic which generalise modal logics and description logics, while preserving decidability, then there exists at least one alternative to the two-variable fragment of first-order logic and the guarded fragment. Maslov's Class K and its dual $\overline{\mathrm{K}}$, both distinct from the guarded fragment, although not intended to be a characterisation or generalisation of the fragment of first-order logic corresponding to modal logics, also cover the relational translation of a range of propositional modal logics. One of our aims is to advance our knowledge in this direction.

The purpose of this thesis is the development of efficient inference procedures for important subclasses of first-order logic. The context in which we conduct our investigation is that of resolution, in particular, the resolution framework of Bachmair and Ganzinger [8]. Two factors which lead to non-termination are: (i) the growth of the depth of clauses and (ii) the growth of the number of literals in clauses during a theorem proving derivation. A sound and complete refinement of resolution that prevents this expansion provides a decision procedure [86]. A class of refinements of resolution which have successfully been applied to a variety of solvable classes are *ordering refinements*.

Given a particular fragment of first-order logic one faces several problems in developing a decision procedure based on a refinement of resolution. First, one has to devise a transformation from the fragment of first-order logic into a class of clauses. Second, one has to find instances of the parameters of the resolution framework which are able to control the growth of the depth and size of clauses during a theorem proving derivation. In our case this amounts to finding an appropriate combination of ordering, selection function, and redundancy elimination criteria. Third, we have to prove the termination of the ordering refinement. This amounts to a tedious check that all cases that may occur during a theorem proving derivation have been taken into account and are handled appropriately.

The difficulty lies in the interdependency between the three steps. With the use of structural transformations the structure of clauses obtained from formulae in a fragment of first-order logic is virtually arbitrary. If the clauses are not well-structured enough it might be difficult or impossible to find suitable settings of the parameters of the resolution framework. Having found the right, well-structured form the solution seems almost trivial. Whether or not the right transformation, ordering, selection function, and redundancy elimination criteria have been chosen will only be revealed during the proof of termination, by tedious case analysis. If the case analysis fails, we have to revise our choice and recheck all the different cases. Once all three problems have been solved the solution seems to be seamless and straightforward, leaving the development cycle undocumented.

There is a fourth problem which deserves attention: For a given fragment of first-order logic there may exist more than one refinement of resolution which can serve as a decision procedure for this fragment. The task then is to determine suitable assessment criteria and to perform a comparison of the various refinements. The primary criteria we are interested in are the generality of the approach and performance. This led us to choose the resolution framework of Bachmair and Ganzinger [8] in which special emphasis is put on simplification and redundancy elimination. Through its range of parameters, including in particular liftable orderings and selection, it is accepted to provide the basis for practical and efficient decision procedures. On the basis of the results of this thesis we believe that it has the potential to cover a wide range of solvable first-order classes.

This thesis addresses these four problems for a variety of decidable fragments of first-order logic which are relevant to the field of non-classical logic and knowledge representation.

Chapter 1 provides the basic definitions for the following chapters. In particular, we describe the resolution framework, structural transformations (renamings), and various solvable classes mentioned throughout the thesis.

Chapter 2 investigates the class $\mathbf{E}^+$ introduced by Tammet [128]. $\mathbf{E}^+$ has become one of the most well-studied classes with respect to resolution decision procedures. The aim in this chapter is to give an overview of the basic techniques which we will use in the following chapters by a working example. In particular, we will see (i) how to use renaming to transform formulae and clauses into a suitable more "well-behaved" form, (ii) how to use ordering refinements to restrict the application of resolution and factoring, (iii) how to prove termination of an ordering refinement, and (iv) how to establish relationships between various decision procedures and logics by means of simulation.

Chapter 3 studies the dual of Maslov's class K, called $\overline{\mathrm{K}}$, and the class $\overline{\mathrm{DK}}$ containing all finite conjunctions of formulae in $\overline{\mathrm{K}}$. Although Maslov [97] described a decision procedure for the validity problem in K based on the inverse method in the late sixties, only in the early nineties a resolution-based decision procedure was described by Zamov [39, chap. 6] for a subclass of $\overline{\mathrm{K}}$. Zamov's techniques are based on non-liftable orderings which have limitations regarding the application of some standard simplification rules which are important to obtain efficient decision procedures. We show that $\overline{\mathrm{K}}$ and $\overline{\mathrm{DK}}$ are solvable using a resolution refinement based on a *liftable* ordering, thus improving the result of Zamov with respect to the fragment of first-order logic we cover and with respect to the applicability of the procedure.

In Chapter 4 we turn our attention to description logics. First, we consider the satisfiability problem for knowledge bases over an extension $\mathcal{ALB}$ of the description logic $\mathcal{ALC}$. We characterise two classes of clauses, namely the class of *DL-clauses* and the class of *fluted DL-clauses*. Both classes are subclasses of $\overline{\mathrm{K}}$ and contain all clauses we obtain from the translating $\mathcal{ALB}$ knowledge bases into first-order logic. The satisfiability problem for DL-clauses and fluted DL-clauses can be solved by a very general ordering refinement of resolution. For a logic $\mathcal{ALB}_{\mathsf{D}}$ in-between $\mathcal{ALC}$ and $\mathcal{ALB}$, we are able to provide a decision procedure using a refinement of resolution based solely on a selection function. We show that this decision procedure is able to polynomially simulate standard tableaux-based decision procedure for the satisfiability problem in $\mathcal{ALC}$.

In Chapter 5 we address decidability issues of classes of first-order formulae which most closely resemble translated modal formulae under the relational and the semi-functional translation. The relational translation of formulae in the basic modal logic K and its extensions by the axiom schemata D, B, and T results in clauses which belong to the class of DL-clauses. Therefore, the ordering refinement developed for the class of DL-clauses provides a decision procedure for these modal logics. It also follows from the considerations in Chapter 4 that the refinement of resolution based on a selection function provides also a decision procedure for the satisfiability problem in K and it simulates standard prefix tableaux calculi [42, 59, 98]. These results together with the decision procedure described by Ganzinger, Hustadt, Meyer and Schmidt [46] for extensions of $\mathsf{K4}$ now cover the classical normal modal logics K, $\mathsf{K4}$, KB, KD, $\mathsf{KD4}$, KT, $\mathsf{KT4}$, KTB, and their multi-modal versions, as well as $\mathsf{S5}$.

We also look at the semi-functional translation of modal logics developed by Nonnengart [101]. The semi-functional translation of formulae in the basic modal logic K and extensions by the axiom schemata D, B, and T results in clauses which belong to the class $\overline{\mathrm{K}}$. Thus, the ordering refinement of Chapter 3 provides a decision procedure for the satisfiability problem of these logics. By contrast, the clauses we obtain from the semi-functional translation of formulae in the modal logics $\mathsf{K(D)4}$, $\mathsf{K(D)5}$, and $\mathsf{K(D)45}$ do not belong to the class $\overline{\mathrm{K}}$. We define two classes of clauses, the class of *SF-clauses* and the class of *small SF-clauses*, to cover these logics. We describe a resolution decision procedure based on an ordering refinement for the class of small SF-clauses and a decision procedure based on an ordering refinement and a selection function for the class of SF-clauses. The second decision procedure is similar to the one presented by Ganzinger, Hustadt, Meyer, and Schmidt [46]. Interestingly, using the semi-functional translation none of the extensions of $\mathsf{K4}$ requires the additional inference rules of ordered chaining.

Besides the resolution-based decision procedures described in Chapter 5 there are various other procedures for establishing the theoremhood and satisfiability of modal formulae, namely procedures based on tableaux calculi, sequent calculi, and extensions of SAT procedures. The simulation results of Chapter 4 and 5 shed some light on the relative performance we can expect of resolution-based algorithms compared to tableaux-based algorithms provided that the resolution-based algorithm follows a particular strategy matching the one used by the tableaux-based algorithm. However, if this is not the case, and the algorithms follow unrelated strategies, the analytical results do not predict their relative performance. Chapter 6 considers issues related to empirical evaluations of theorem provers for modal logic. It considers some of the problems related to establishing an appropriate benchmark suite for modal logics and problems related to the evaluation of theorem provers based on different calculi. We outline an approach called *scientific benchmarking*. The aim of scientific benchmarking is not to find or declare the best-performing system. Instead the focus is on different techniques, strategies, and heuristics, which are used in the different theorem provers for improved performance on particular problem sets. Extensive benchmarks are performed for the theorem provers $\mathcal{KRIS}$, KSAT, the Logics Workbench, and the translation approach using the optimised functional translation and the theorem prover SPASS.

Chapter 1

Basic notions

This chapter defines the basic notions needed in this thesis. Of particular importance are Section 1.2, 1.3, and 1.5. We start with preliminary definitions in Section 1.1. Section 1.2 describes the resolution framework we will use as a basis for the decision procedures developed in this thesis. Section 1.3 describes the technique of structural transformation which is a key element in our decision procedures. In Section 1.4 we give a brief introduction into the simulation of proof systems. Section 1.5 gives an overview of solvable classes which we will mention at various points in the thesis. The presentation of the definitions in this chapter is restricted to the single-sorted case. A generalisation to the many-sorted case (without subsort and operator overloading) is straightforward [94].

1.1 Preliminary definitions

Terms, literals, and clauses

Let F, P, and V be three disjoint (countable) sets. The elements of F are called *function symbols*, the elements of P *predicate symbols*, and the elements of V *variables*. With each function symbol and predicate symbol we associate a non-negative number, called its *arity*. A function symbol of arity 0 is a *constant symbol* or *constant*, a predicate symbol of arity 0 is a *propositional variable*. A tuple $(\mathsf{F}, \mathsf{P}, \mathsf{V})$ defines a *signature*.

A *term* is either a variable or an expression $f(t_1, \ldots, t_n)$ where f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms. A term which is neither a variable nor a constant is called *compound*. The set of all terms built from function symbols in F and variables in V is denoted by $\mathsf{T}(\mathsf{F}, \mathsf{V})$. Terms containing no variables are called *ground terms*. The set of all ground terms is denoted by $\mathsf{GT}(\mathsf{F})$. A term s is said to be a *subterm* of a term t if either $s = t$, or else $t = f(t_1, \ldots, t_n)$ and s is a subterm of one of the terms t_i, $1 \leq i \leq n$. By a *strict subterm* of t we mean a subterm distinct from t. The *depth* $\mathrm{dp}(t)$ *of a term* t is inductively defined as follows: (i) if t is a variable or a constant then $\mathrm{dp}(t) = 1$, and (ii) if $t = f(t_1, \ldots, t_n)$, then $\mathrm{dp}(t) = 1 + \max(\{\mathrm{dp}(t_i) \mid 1 \leq i \leq n\})$. The *arity of a term* t, denoted by $\mathrm{arity}(t)$, is defined as follows: (i) If t is a constant, then $\mathrm{arity}(t) = 0$, (ii) if t is compound term $f(t_1, \ldots, t_n)$, then $\mathrm{arity}(t) = n$, and (iii) if t is a variable, then $\mathrm{arity}(t)$ is undefined.

An *atomic formula* (or an *atom*) is an expression $p(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms in $\mathsf{T}(\mathsf{F}, \mathsf{V})$ and p is a n-ary predicate symbol in P. A *literal* is an expression A (a *positive* literal) or $\neg A$ (a *negative* literal) where A is an atomic formula. '$(\neg)A$' denotes either A or $\neg A$. For a literal $L = (\neg)p(t_1, \ldots, t_n)$ the terms $t_1, \ldots, t_n$ are the *arguments* or *argument terms* of L. By $\arg_{set}(L)$ we denote the *set of arguments* of L and by $\arg_{mul}(L)$ we denote the *multiset of arguments* of L. We let $\overline{L}$ denote the *complement* of a literal L, that is, $\overline{L}$ denotes $\neg L$, if L is a positive literal, and A, if L is a negative literal $\neg A$. Furthermore, we let $|\,L\,|$ denote the *norm* of a literal L , that is, $|\,L\,|$ denotes L, if L is a positive literal, and A, if L is a negative literal of the form $\neg A$. The *depth* of a literal L is defined by $\mathrm{dp}(L) = 1 + \max(\{\mathrm{dp}(t) \mid t \in \arg_{set} L\})$.

The set of *first-order formulae* over a signature (F, P) and a set of variables V is inductively defined as follows: (i) every atom is a first-order formula, (ii) if φ and ψ are formulae and x is a variable, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, $\varphi \leftrightarrow \psi$, $\forall x\colon \varphi$, and $\exists x\colon \varphi$ are first-order formulae. The notions of scope of a universal quantifier $\forall$ and an existential $\exists$, and free and bound variables are defined in the usual way. A *subformula* of a first-order formula φ is a subexpression of φ. A formula φ is in *prenex form* if $\varphi = Q_1 x_1 \ldots Q_n x_n\, \psi$ where $Q_1, \ldots, Q_n$ are quantifiers and the formula ψ is quantifier free. The *matrix* of a formula φ is the formula ψ obtained from φ by deleting all occurrences of quantifiers. A formula $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ is a *disjunction* or *disjunctive formula* and each φ_i is a *disjunct* of φ. A formula $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ is a *conjunction* or *conjunctive formula* and each φ_i is a *conjunct* of φ. A formula is in *conjunctive normal form* iff its matrix is a conjunction of disjunctions of atoms and their negation. An occurrence of a subformula of an equivalence has *zero polarity*. For subformulae not below an equivalence, an occurrence has *positive polarity* if it is inside the scope of an even number of (explicit or implicit) negations, and an occurrence has *negative polarity* if it is one inside the scope of an odd number of negations. A formula φ without function symbols is *rectified* iff no variable occurs both bound and free in φ and no variable is bound by two different quantifier occurrences. A *schema* is a formula without function symbols that is rectified and closed.

A formula φ is in *negation normal form* if for every subformula $\neg\psi$ of φ, ψ is an atomic formula. For every formula φ there exists an equivalent formula $\mathrm{nnf}(\varphi)$ in negation normal form. We assume that $\mathrm{nnf}(\varphi)$ is computed by two consequtive transformations. First, every occurrence of $\psi \leftrightarrow \phi$ with positive polarity is replaced by $(\psi \rightarrow \phi) \wedge (\phi \rightarrow \psi)$ and every occurrence of $\psi \leftrightarrow \phi$ with negative polarity is replaced by $(\psi \wedge \phi) \vee (\neg\psi \wedge \neg\phi)$. This form of *linearisation* avoids a possible exponential explosion during the conversion to clausal form. Second, the following transformation rules are applied exhaustively:

$$
\begin{array}{ccc}
\neg(\phi \vee \psi) \Rightarrow \neg\phi \wedge \neg\psi & \neg\neg\phi \Rightarrow \phi & \neg(\phi \wedge \psi) \Rightarrow \neg\phi \vee \neg\psi \\
\neg\forall x\colon \phi \Rightarrow \exists x\colon \neg\phi & \phi \rightarrow \psi \Rightarrow \neg\phi \vee \psi & \neg\exists x\colon \phi \Rightarrow \forall x\colon \neg\phi \\
\neg\bot \Rightarrow \top & & \neg\top \Rightarrow \bot.
\end{array}
$$

A *multiset* over a set S is a mapping C from S to the natural numbers $\mathbb{N}$. A *clause* is a multiset of literals. We use $\bot$ to denote the empty clause. We write $L \in C$ if $C(L) \geq 0$ for a literal L. A *subclause* D of a clause C, is a submultiset D of C. A clause D is a *strict subclause* of C, denoted by $D \subset C$, if $D \subseteq C$ and $D \neq C$. We use $C \setminus D$ to denote the multiset-difference of a clause C and a subclause D of C. A *Horn clause* is a clause containing at most one positive literal. The *depth* of a clause C is defined by $\mathrm{dp}(C) = \max(\{\mathrm{dp}(L) \mid L \in C\})$.

In the following, an *expression* will be a term, an atom, a literal, or a clause. The set of all variables occurring in an expression E, or in a set of expressions S, is denoted by $\mathcal{V}(E)$, or $\mathcal{V}(S)$. Two expressions E and E' are *variable-disjoint* if $\mathcal{V}(E) \cap \mathcal{V}(E') = \emptyset$. Analogously, for sets of expressions. $|N|$ denotes the *cardinality* of a set.

A *substitution* is a mapping from variables to terms which is the identity mapping almost everywhere. A substitution σ can be represented as a finite set of pairs $\sigma = \{x_1/t_1, \dots , x_n/t_n\}$, where $x_i \neq t_i$ for all i, $1 \leq i \leq n$. The *identity substitution* is denoted by ι. The value of a substitution σ for a variable x is denoted by $x\sigma$. A substitution can be homomorphically extended to a mapping from terms to terms. Analogously, for atoms, literals, and clauses. The result of the *application* of a substitution σ to an expression E is denoted by $E\sigma$. The set $\mathrm{D}\sigma = \{x \in \mathsf{V} \mid x\sigma \neq x\}$ is the *domain* of a substitution σ and the set $\mathrm{C}\sigma = \{x\sigma \mid x \in \mathrm{D}\sigma\}$ is the *codomain* of σ. If $V \subseteq \mathsf{V}$ is a set of variables, then the *restriction* $\sigma_{|V}$ *of* σ *to* V is defined by a substitution θ such that $\mathrm{D}\theta = \mathrm{D}\sigma \cap V$ and $x\sigma = x\theta$ for every x, $x \in \mathrm{D}\theta$. The *composition* $\theta\sigma$ of the substitutions σ and θ is defined as $x\theta\sigma = (x\theta)\sigma$ for all variables x in V. A substitution σ is *idempotent* if $\sigma\sigma = \sigma$. A *variable renaming* is an injective substitution σ such that $\mathrm{C}\sigma \subseteq \mathsf{V}$.

An expression E' is an *instance* of an expression E if there exists a substitution σ such that $E' = E\sigma$. An expression E' is a *variant* of an expression E if there exists a variable renaming σ such that $E' = E\sigma$. We say E and E' are *identical modulo variable renaming*. We usually consider clauses C and D to be identical if they are identical modulo variable renaming. A substitution σ is a (syntactical) *unifier* of expressions $E_1, \dots , E_n$ if $E_i\sigma = E_j\sigma$ for all i, j, $1 \leq i, j \leq n$, and $E_1, \dots , E_n$ are said to be *unifiable*. A unifier σ is a *most general unifier* of $E_1, \dots , E_n$ if for every unifier θ of $E_1, \dots , E_n$ there exists a substitution ρ such that $\theta_{|\mathcal{V}(\{E_1,\dots,E_n\})} = (\sigma\rho)_{|\mathcal{V}(\{E_1,\dots,E_n\})}$. If $E_1, \dots , E_n$ are unifiable, then there exists a most general unifier of $E_1, \dots , E_n$. For appropriate algorithms for the computation of most general unifiers see [88, 95].

As usual, positions of an expression are denoted by sequences of natural numbers. The length of a sequence λ is denoted by $|\lambda|$. The set of all positions of an expression E is denoted by $\mathrm{Pos}(E)$. If λ is a position in E, then $E|_\lambda$ denotes the subexpression at position λ, and $E[\lambda \leftarrow E']$ is the result of replacing the subexpression of E at position λ by E'. We write $E[E_1]$ to indicate that the expression E contains E_1 as a subexpression. Then $E[E_2]$ denotes the result of replacing the same occurrence of E_1 in E by the expression E_2. If E_1 and E_2 are expressions such that E_1 occurs at a position λ in E_2, that is $E_2|_\lambda = E_1$, then E_1 *occurs in* E_2 *at depth* $|\,\lambda\,|$. If E_1 and E_2 are expressions such that Λ is the set of all positions in E_2 such that $E_2|_\lambda = E_1$, then $\max(\{|\,\lambda\,| \mid \lambda \in \Lambda\})$ is the *maximal depth of an occurrence of* E_1 *in* E_2.

Orderings

Let U be a set. A binary relation R on U is a subset of $U \times U$. The *inverse* R^{-1} of R is the set $\{(y, x) \mid (x, y) \in R\}$. A binary relation R is (i) *reflexive* if for every $x \in U$, $(x, x) \in R$, (ii) *symmetric* if for every $x, y \in U$, $(x, y) \in R$ implies $(y, x) \in R$, (iii) *antisymmetric* if for every $x, y \in U$, $(x, y) \in R$ and $(y, x) \in R$ implies $x = y$, (iv) *transitive* if for every $x, y, z \in U$, $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$, and (v) *asymmetric* if for every $x, y \in U$, $(x, y) \in R$ implies $(y, x) \notin R$. A binary relation R is an *equivalence relation* if R is reflexive, symmetric, and transitive.

For a binary relation $\rightarrow$ we use $\leftrightarrow$, $\rightarrow^+$, and $\rightarrow^*$ to denote its symmetric, transitive, and reflexive-transitive closure, respectively. A binary relation $\rightarrow$ over a set U is *well-founded* if there is no infinite chain $d_1 \rightarrow d_2 \rightarrow \cdots$ of elements in U. An element $d \in U$ is in *normal form* with respect to $\rightarrow$ if there is no $d' \in U$ such that $d \rightarrow d'$. If $d \rightarrow^* d'$ and d' is in normal form with respect to $\rightarrow$, then d' is a *normal form of* d.

We say a binary relation $\rightarrow$ on expressions is *stable under contexts* if $E_1 \rightarrow E_2$ implies

$E[\lambda \leftarrow E_1] \to E[\lambda \leftarrow E_2]$ for all expressions E, E_1, and E_2 such that $E[\lambda \leftarrow E_1]$ and $E[\lambda \leftarrow E_2]$ are well-formed. The relation $\to$ is *stable under substitutions* if $E_1 \to E_2$ implies $E_1\sigma \to E_2\sigma$ for all expressions E_1, E_2 and all substitutions σ. If $\to$ is stable under substitutions, we also say $\to$ is *liftable*. A *rewrite relation* is a binary relation which is stable under context and stable under substitutions.

A *partial ordering* $\succeq$ on a set U is a binary relation on U which is reflexive, antisymmetric, and transitive. A *strict partial ordering* $\succ$ on a set U is a binary relation on U which is asymmetric and transitive. A strict partial ordering $\succ$ on a set U is *total* if for every $x, y \in U$ either $x \succ y$, $y \succ x$, or $x = y$ holds.

Given a strict partial ordering on U, a subset U' of U, and an element x of U, x is $\succ$*-maximal* with respect to U' if there is no element y of U' such that $y \succ x$ holds. The element x is *strictly* $\succ$*-maximal* with respect to U' if there is no element y of U' such that $y \succ x$ or $y = x$ holds.

A *quasi-ordering* $\succsim$ is any reflexive and transitive binary relation. The associated equivalence relation $\sim$ is the intersection of $\succsim$ with its inverse. To any quasi-ordering $\succsim$ we can associate a strict partial ordering $\succ$ which is the difference between $\succsim$ and its inverse. The resulting strict partial ordering is also called the *strict part* of $\succsim$. A quasi-ordering is well-founded if its strict part is well-founded.

A partial ordering $\succeq$ on a set U can be extended to a partial ordering $\succeq^{set}$ on (finite) sets over U as follows: $M \succeq^{set} N$ if for every element x of N there exists an element y of M such that $y \succeq x$. A strict partial ordering $\succ$ on U can be extended to an ordering $\succ^{mul}$ on (finite) multisets over U as follows: $M \succ^{mul} N$ if (i) $M \neq N$ and (ii) whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some $y \succ x$.

An ordering $\succ_1$ is a *refinement* of an ordering $\succ_2$ if $\succ_2 \subset \succ_1$ holds. An ordering $\succ_1$ on a set U is a *refinement with respect to a subset S of U* of $\succ_2$ if $(\succ_2 \cap S \times S) \subset (\succ_1 \cap S \times S)$. An ordering $\succ$ is *compatible* with a binary relation R if R is a subset of $\succ$.

Let f be an n-ary function symbol with which we associate a mapping (called the *status* of f) that assigns to each strict partial ordering $\succ$ on terms an ordering $\succ^f$ on n-tuples of terms. In particular, the function symbol f is said to have *multiset status*, if $\succ^f$ is defined by $(s_1, \dots, s_n) \succ^f (t_1, \dots, t_n)$ if $\{s_1, \dots, s_n\} \succ^{mul} \{t_1, \dots, t_n\}$. It is said to have *lexicographic status*, if there exists a permutation π of $\{1, \dots, n\}$, such that $(s_1, \dots, s_n) \succ^f (t_1, \dots, t_n)$ if (i) $(s_{\pi(1)}, \dots, s_{\pi(n)}) \succ^{lex} (t_{\pi(1)}, \dots, t_{\pi(n)})$ and (ii) $f(s_1, \dots, s_n) \succ t_i$ for all i, $1 \leq i \leq n$. (Here $\succ^{lex}$ denotes the n-fold lexicographic combination of the ordering $\succ$.)

Let $\succ$ be an ordering, called a *precedence*, on a given set of function symbols (and predicate symbols), and suppose that each function (and predicate) symbol has either multiset or lexicographic status. Then the corresponding *recursive path ordering* $\succ_{rpo}$ is recursively defined by: $s = f(s_1, \dots, s_m) \succ_{rpo} g(t_1, \dots, t_n) = t$ if (i) $s_i \succeq_{rpo} t$, for some i with $1 \leq i \leq m$, or (ii) $f \succ g$ and $s \succ_{rpo} t_j$, for all j with $1 \leq j \leq n$, or (iii) $f = g$ and $(s_1, \dots, s_m) \succ^f_{rpo} (t_1, \dots, t_m)$. Any recursive path ordering is a well-founded strict partial ordering on terms which is stable with respect to substitutions and total on ground terms [126].

An *A-ordering* $\succ$ is an ordering on atoms which is stable under substitutions. An *atom ordering* is a well-founded, total ordering on ground atoms. A *literal ordering* is a well-founded, total ordering on ground literals.

1.2 The resolution calculus

A *condensation* $\mathrm{Cond}(C)$ of a clause C is a minimal subclause of C which is also an instance of it. A clause C is *condensed* if there exists no condensation of C which is a strict subclause of C. The components in the variable partition of a clause are called *split components*. This implies split components do not share variables. If $C_1, \ldots, C_n$ are the split components of C, then we say C can be *decomposed* into $C_1, \ldots, C_n$. A clause which is its own split component is *indecomposable*.

Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint clauses such that the atoms A_1 and A_2 are unifiable with most general unifier σ. Then $(D_1 \cup D_2)\sigma$ is a *(binary) resolvent* of C_1 and C_2. C_1 is called the *positive premise* and C_2 is called the *negative premise*. Let $C_1 = \{L_1, L_2\} \cup D_1$ be a clause such that the literals L_1 and L_2 are unifiable with most general unifier σ. Then $(\{L_1\} \cup D_1)\sigma$ is a *factor* of C_1. For a discussion of various refinements of resolution confer [10, 91].

The refinements of resolution on which the decision procedures developed in the following chapters are based, will be accommodated in the resolution calculus of Bachmair and Ganzinger [8, 10]. It makes use of a certain class of admissible orderings $\succ$ and a selection function S of negative literals.

An ordering $\succ$ on literals is *admissible*, if (i) it is well-founded and total on ground literals, and stable under substitutions, (ii) $\neg A \succ A$ for all ground atoms A, and (iii) if $B \succ A$, then $B \succ \neg A$ for all ground atoms A and B. Any atom ordering $\succ$ can be extended to a literal ordering $\succ'$ by taking the multiset extension of $\succ$ and by identifying any positive literal A with the multiset $\{A\}$ and any negative literal $\neg A$ with the multiset $\{A, A\}$. The ordering $\succ'$ satisfies Conditions (ii) and (iii). Any literal ordering $\succ$ satisfying Conditions (ii) and (iii) can be extended to an admissible ordering on literals by lifting it to non-ground literals L_1 and L_2 as follows: $L_1 \succ L_2$ if and only if $L_1\sigma \succ L_2\sigma$, for all ground substitutions σ. We say a literal L is *$\succ$-maximal with respect to a clause C* if for any L' in C, $L' \not\succ L$, and L is *strictly $\succ$-maximal with respect to C* if for any L' in C, $L \not\succeq L$. Any ordering $\succ$ on literals can be extended to clauses by taking the multiset extension of $\succ$.

A *selection function S* assigns to each clause a possibly empty set of occurrences of negative literals. If C is a clause, then the literal occurrences in $S(C)$ are *selected*. No restrictions are imposed on the selection function.

The calculus consists of general *expansion rules* of the form

$$\frac{N}{N_1 \mid \cdots \mid N_n}$$

each representing a finite derivation of the leaves $N_1, \ldots, N_k$ from the root N. The following rules describe how derivations can be expanded at leaves.

Deduce:
$$\frac{N}{N \cup \{\mathrm{Cond}(C)\}}$$

if C is either a resolvent or a factor of clauses in N.

Delete:
$$\frac{N \cup \{C\}}{N}$$

if C is a tautology or N contains a clause which is a variant of C.

Split:
$$\frac{N \cup \{C \cup D\}}{N \cup \{C\} \mid N \cup \{D\}}$$
if C and D are variable-disjoint.

Resolvents and factors are derived by the following rules.

Ordered Resolution:
$$\frac{C \cup \{A_1\} \quad D \cup \{\neg A_2\}}{(C \cup D)\sigma}$$
where (i) σ is the most general unifier of A_1 and A_2, (ii) no literal is selected in C and $A_1\sigma$ is strictly $\succ$-maximal with respect to $C\sigma$, and (iii) $\neg A_2$ is either selected, or $\neg A_2\sigma$ is $\succ$-maximal in $D\sigma$ and no literal is selected in D.[1]

Ordered Factoring:
$$\frac{C \cup \{A_1, A_2\}}{(C \cup \{A_1\})\sigma}$$
where (i) σ is the most general unifier of A_1 and A_2, and (ii) no literal is selected in C and $A_1\sigma$ is $\succ$-maximal with respect to $C\sigma$.

If C_2 is derived from C_1 by ordered factoring based on the ordering $\succ$, then C_2 is a *$\succ$-factor of C_1*. Likewise, if C_3 is derived from C_1 and C_2 by ordered resolution, then C_3 is a *$\succ$-resolvent of C_1 and C_2*.

The ordering $\succ$ is used *a posteriori* in the inference rules, that is, $\succ$-maximality of a literal is determined after instantiation of the premises by the most general unifier σ. If we determine the $\succ$-maximality of literals with respect to the uninstantiated premises, then we say the ordering is applied *a priori*. Note that due to the stability of any admissible ordering $\succ$ under substitutions, the maximality of a literal $L\sigma$ with respect to $C\sigma$ implies the maximality of L with respect to C.

The expansion rule "Delete" represents the weakest form of *redundancy elimination* which is sufficient for the purpose of this thesis. However, the performance of resolution-based theorem provers crucially depends on the use of more powerful techniques. Bachmair and Ganzinger [8] define the following notion of redundancy. Let N be a set of clauses. A ground clause C is *redundant with respect to* N if there are ground instances $C_1\sigma, \dots, C_n\sigma$ of clauses in N such that $C_1\sigma, \dots, C_n\sigma \models C$ and for each i, $C \succ C_i\sigma$. A non-ground clauses C is *redundant with respect to* N if every ground instance of C is redundant with respect to N. The notion of redundancy is lifted to the non-ground case in the expected way. An inference is redundant if one of the premises is redundant, or its conclusion is redundant or an element of N. A clause set N is *saturated up to redundancy* if all inferences from non-redundant premises are redundant with respect to N.

If a clause C is not condensed then C is redundant with respect to $\mathrm{Cond}(C)$. It is therefore admissible to replace any clause C by its condensation Cond(C). Note that whenever a clause C is neither condensed nor indecomposable, an application of condensing should have precedence over an application of splitting to avoid unnecessary paths in the theorem proving derivation. Therefore, we have incorporated condensation into the "Deduce" expansion rule and assume from now on that clauses are condensed.

A *(theorem proving) derivation* from a set N of clauses is a finitely branching tree T with root N constructed by applications of the expansion rules. A derivation T is a *refutation* if for every path $N = N_0, N_1, \dots$, the clause set $\bigcup_j N_j$ contains the empty clause.

[1] As usual we implicitly assume that the premises have no common variables.

A derivation T from N is called *fair* if for any path $N = N_0, N_1, \ldots$ in the tree T, with *limit* $N_\infty = \bigcup_j \bigcap_{k \geq j} N_k$, it is the case that each clause C that can be deduced from non-redundant premises in N_∞ is contained in some set N_j.

Theorem 1.1 (Bachmair, Ganzinger, Waldmann [8, 15]).
Let T be a fair theorem proving derivation from N. If $N, N_1, \ldots$ is a path in T with limit N_∞, then N_∞ is saturated up to redundancy. Furthermore, N is satisfiable if and only if there exists a path in T with limit N_∞ such that N_∞ is satisfiable.

Theorem 1.2 (Bachmair, Ganzinger, Waldmann [8, 15]).
Let T be a fair theorem proving derivation from N. N is unsatisfiable if and only if for every path $N = N_0, N_1, \ldots$, the clause set $\bigcup_j N_j$ contains the empty clause.

We will restrict our attention to derivations which are generated by strategies in which “Delete”, “Split”, and “Deduce” are applied in this order. In addition, no application of the “Deduce” expansion rule with premises and consequence which are identical (modulo variable renaming) may occur twice on the same path in the derivation. Furthermore, we require that derivations are *fair*. In all the cases we consider this is not an additional restriction. It will be straightforward to see that any theorem proving derivation generated according to the strategy outlined above is fair.

1.3 Structural transformation

Structural transformation, also known as renaming, is a standard technique of many areas, besides automated deduction, for transforming formulae into a more suitable normal form [7, 113]. In this thesis we will use this technique for the conversion to clausal form, the embedding into suitable classes of clauses, and for swapping signs of literals so that particular literals can be selected.

Let $\mathrm{Pos}(\varphi)$ be the set of positions of a first-order formula φ. If λ is a position in φ, then $\varphi|_\lambda$ denotes the subformula of φ at position λ and $\varphi[\lambda \leftarrow \psi]$ is the result of replacing φ at position λ by ψ.

We associate with each element λ of $\Lambda \subseteq \mathrm{Pos}(\varphi)$ a new predicate symbol Q_λ and a new literal $Q_\lambda(x_1, \ldots, x_n)$, where $x_1, \ldots, x_n$ are the free variables of $\varphi|_\lambda$. Let

$$\begin{aligned} \mathrm{Def}^+_\lambda(\varphi) &= \forall x_1 \ldots x_n\ (Q_\lambda(x_1, \ldots, x_n) \rightarrow \varphi|_\lambda) \quad \text{and} \\ \mathrm{Def}^-_\lambda(\varphi) &= \forall x_1 \ldots x_n\ (\varphi|_\lambda \rightarrow Q_\lambda(x_1, \ldots, x_n)). \end{aligned}$$

The *definition* of Q_λ is the formula

$$\mathrm{Def}_\lambda(\varphi) = \begin{cases} \mathrm{Def}^+_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has positive polarity,} \\ \mathrm{Def}^-_\lambda(\varphi) & \text{if } \varphi|_\lambda \text{ has negative polarity,} \\ \mathrm{Def}^+_\lambda(\varphi) \wedge \mathrm{Def}^-_\lambda(\varphi) & \text{otherwise.} \end{cases}$$

Now, define $\mathrm{Def}_\Lambda(\varphi)$ inductively by:

$$\mathrm{Def}_\emptyset(\varphi) = \varphi \quad \text{and}$$

$$\mathrm{Def}_{\Lambda\cup\{\lambda\}}(\varphi) = \mathrm{Def}_{\Lambda}(\mathrm{Def}_{\lambda}(\varphi) \wedge \varphi[\lambda \leftarrow Q_{\lambda}(x_1, \ldots, x_n)]),$$

where λ is maximal in $\Lambda\cup\{\lambda\}$ with respect to the prefix ordering on positions. The corresponding clauses will be called *definitional clauses*. A *definitional form* of φ is $\mathrm{Def}_{\Lambda}(\varphi)$, where Λ is a subset of positions of subformulae (usually, non-atomic or non-literal subformulae).

Theorem 1.3.
Let φ be a first-order formula. For any $\Lambda \subseteq \mathrm{Pos}(\varphi)$, $\mathrm{Def}_{\Lambda}(\varphi)$ can be computed in polynomial time and φ is satisfiable iff $\mathrm{Def}_{\Lambda}(\varphi)$ is satisfiable.

Structural transformation can be defined more generally so that for variant subformulae only one new predicate symbol is introduced.

The use of structural transformation for the conversion to clausal from has two major advantages.

1. If we translate a first-order formula φ directly to its clausal form $\mathrm{Cls}(\varphi)$, the size of $\mathrm{Cls}(\varphi)$ can be exponential in the size of φ. If Λ is the set of all positions of φ, then the size of $\mathrm{Cls}(\mathrm{Def}_{\Lambda}(\varphi))$ is polynomial/linear in the size of φ.

2. For the fragments of first-logic we will consider, the application of structural transformation considerably simplifies the form of clauses we obtain from φ. This eases our task to devise suitable decision procedures for the fragments under consideration. In fact, only with the help of an appropriate form of structural transformation are we able to solve this task for some of the fragments of first-order logic we will consider, namely $\mathbf{E}^+$ and $\overline{\mathrm{KC}}$.

We assume in the following that the clausal form $\mathrm{Cls}(\varphi)$ of a first-order formula is computed using, for example, the algorithm presented by Chang and Lee [21]. It is important that outer Skolemisation is used and the scope of quantifiers is not reduced. Inner Skolemisation or strong Skolemisation, considered advantageous in [103], can destroy the covering or regularity property of terms. For further discussion see Section 3.2.

1.4 Simulation

The following definitions are adapted from Eder [34]. An excellent overview of simulation results for proof systems of propositional logic is given in [131].

For a finite alphabet Σ, let Σ^* denote the set of all finite strings over Σ. A *language* over an alphabet Σ is a subset of Σ^*.

A *proof system for L* is a mapping $S : \Sigma_1^* \to L$ for some alphabet Σ_1, such that S is surjective and S can be computed in polynomial time by a Turing machine. Every string ρ is said to be a *proof* of $S(\rho)$ in S. A crucial property of a proof system is that, given a string ρ, there is a feasible method for checking whether or not ρ is a proof, and if so, of what it is a proof [131].

Let $\mathcal{C}$ be a calculus given by a set of axioms and a set of inference rules. We assume that the set of axioms and inference rules is decidable in polynomial time. A *derivation* of φ_n in $\mathcal{C}$ is a sequence of formulae $\varphi_1, \ldots, \varphi_n$ such that each φ_i, $1 \leq i \leq n$ is either an axiom or derived from formulae occurring in $\varphi_1, \ldots, \varphi_{i-1}$ by an inference rule. Alternatively, we may define that a derivation in $\mathcal{C}$ is a tree labelled with formulae such that every formula labelling a leave node is an axiom and every formula labelling an interior node is derived by a rule of inference from the formulae labelling its parent nodes. In both cases we can encode derivations as strings over an appropriate alphabet. Then the calculus $\mathcal{C}$ induces a proof system S in the following way: if a string ρ encodes a derivation of φ in $\mathcal{C}$, then let $S(\rho) = \varphi$, otherwise let $S(\rho) = \tau$ where τ is a standard theorem of $\mathcal{C}$, for example, one of its axioms.

The resolution calculus of Section 1.2 is an example for a refutational calculus, that is, given a finite, unsatisfiable set of clauses it derives a contradiction.

Let $S_1 : \Sigma_1^* \rightarrow L$ and $S_2 : \Sigma_2^* \rightarrow L$ be proof systems for L. Then S_1 *polynomially simulates* S_2, if (*) there is a polynomial p such that for every natural number n and for every element τ of L the following holds: If there is a proof of τ in S_2 whose length is n, then there is a proof of τ in S_1 whose length is less than $p(n)$. More generally, $S_1 : \Sigma_1^* \rightarrow L_1$ polynomially simulates $S_2 : \Sigma_2^* \rightarrow L_2$ if L_2 can be polynomially reduced to L_1 and (*) is true.

If S_1 is a proof system induced by calculus $\mathcal{C}_1$, then the length of a proof ρ in S_1 is polynomially bounded by the number of applications of inference rules in the derivation encoded by ρ. Let S_1 and S_2 be proof systems induced by calculi $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. To show that S_1 p-simulates S_2 it is therefore sufficient to prove that for every formula φ and every derivation D_2 of φ in S_2, there exists a derivation D_1 of φ in S_1 such that the number of applications of inference rules in D_1 is polynomially bounded by the number of applications of inference rules in D_2.

All the simulation results in this thesis achieve this by showing that there exists a number n such that each application of an inference rule in D_1 corresponds to at most n applications of inference rules in D_2. It follows that the length of D_2 is polynomially bounded by the length of D_1. We call this a *step-wise simulation* of S_2 by S_1. Note that a step-wise simulation is independent of whether the considered derivations are proofs.

The notion of p-simulation was introduced by Cook and Reckhow [22] for the purpose of studying the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem. However, as far as analytical investigations of the relative efficiency of calculi and theorem provers are concerned, it is only one measure which needs consideration. Notice that for any derivation by the resolution calculus of Section 1.2 there exists a derivation of the same length by the unrefined resolution calculus, while the opposite does not hold. Nevertheless, for all practical applications refinements of resolution have considerable performance advantages. This is due to the reduction in the size of the *search space* achieved by restricting the applicability of the inference rules by ordering restrictions and selections functions and the use of powerful redundancy criteria for removing superfluous clauses.

Informally, the *search space* of a given set N of formulae is the set S of all formulae that can be derived from N in a calculus $\mathcal{C}$. More formal definitions are given in [114, 18]. It is then possible to consider the relative size of the search space of calculi and particular theorem proving strategies. Again the consideration of a step-wise simulation can provide insights. If a proof system S_1 is able to step-wise simulate a proof system S_2 and vice versa, then the relative size of the search space of S_1 and S_2 can only differ by a constant factor.

1.5 Solvable classes

A class of (first-order) formulae is *solvable* if and only if there is a decision procedure for satisfiability, that is, an effective procedure that determines for every formula in the class whether it has a model.

For an historical overview of the work on solvability problems see [33]. We will only consider classes without equality. Classical examples of solvable classes which we will mention in this thesis are given below. Recall that schemata are first-order formulae without function symbols that are rectified and closed. Whenever the quantified variable is not relevant for the characterisation of the form of a formula, we write $\forall$ and $\exists$ instead of $\forall x$ and $\exists x$, respectively.

The initially extended Ackermann class
The initially extended Ackermann class is the class of all schemata in prenex normal form with a prefix of the form $\exists \ldots \exists \forall \exists \ldots \exists$.

The Bernays-Schönfinkel class
The Bernays-Schönfinkel class is the class of all schemata in prenex normal form with a prefix of the form $\exists \ldots \exists \forall \ldots \forall$.

The initially extended Gödel class
The initially extended Gödel class is the class of all schemata in prenex normal form with a prefix of the form $\exists \ldots \exists \forall \forall \exists \ldots \exists$.

The monadic class (with equality)
The monadic class (with equality) is the class of all schemata (with equality) such that all predicate symbols are monadic.

The solvability results for the Bernays-Schönfinkel and the initially extended Gödel class together with the unsolvability results for the class of prenex formulae with prefix $\forall \forall \forall \exists$ and the class of prenex formulae with prefix $\forall \exists \forall$ settle the solvability problem for all fragments of first-order logic defined in terms of the prefix of formulae in prenex form.

Examples of decidable fragments of first-order logic which are not solely defined in terms of the prefix of formulae in prenex form are the following.

The initially extended Skolem class
The initially extended Skolem class is the class of all schemata in prenex normal form with a prefix of the form $\exists z_1 \ldots \exists z_k \forall y_1 \ldots \forall y_m \exists x_1 \ldots \exists x_n$ such that each atom of the matrix has among its arguments either (i) at least one of the x_i, or (ii) at most one of the y_i, or (iii) all of $y_1, \ldots, y_m$.

The two-variable fragment of first-order logic (FO^2)
The two-variable fragment of first-order logic is the class of formulae without function symbols over a set of variables V with only two elements.

The guarded fragment GF [**30**, **29**]
The guarded fragment GF is a subclass of first-order logic without non-constant function symbols which is inductively defined as follows: (i) $\top$ and $\perp$ are in GF, (ii) if A is an atom, then A is in GF, (iii) if φ and ψ are in GF, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, and $\varphi \leftrightarrow \psi$ are in GF, (iv) if φ is in GF, A is an atom, and $\bar{x}$ is a sequence of variables, such that every free variable of φ is an argument of A, then $\forall \bar{x}\colon A \rightarrow \varphi$ and $\exists \bar{x}\colon A \wedge \varphi$ are in GF. The atom A in case (iv) is called a *guard*. Note that free variables in a formula are implicitly existentially quantified.

The loosely guarded fragment LGF [**29**]
The loosely guarded fragment LGF is a subclass of first-order logic without function symbols which is inductively defined as follows: (i) $\top$ and $\perp$ are in LGF, (ii) if A is an atom, then A is in LGF, (iii) if φ and ψ are in LGF, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, and $\varphi \leftrightarrow \psi$ are in LGF, (iv) If φ is in LGF, $A_1, \ldots, A_n$ are atoms, $\bar{x}$ is a sequence of variables, such that every free variable occurs together with every x_i of $\bar{x}$ in one of the A_j, then $\forall \bar{x}\colon (A_1 \wedge \ldots \wedge A_n) \rightarrow \varphi$ and

$\exists\bar{x}\colon (A_1 \wedge \ldots \wedge A_n) \wedge \varphi$ are in LGF. Again, free variables in a formula are implicitly existentially quantified.

The class of fluted formulae (fluted logic) [115]
Let X_m be an ordered set $\{x_1, \ldots, x_m\}$ of variables. The class of fluted formulae over X_m is inductively defined as follows: (i) if R is a n-ary predicate symbol with $n \leq m$, then the atom $R(x_{m-n+1}, \ldots, x_m)$ is a fluted formula over X_m, (ii) if φ and ψ are a fluted formulae over X_m, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$, and $\varphi \leftrightarrow \psi$ are fluted formulae over X_m, (iii) if φ is a fluted formula over X_{m+1}, then $\exists x_{m+1}\colon \varphi$ and $\forall x_{m+1}\colon \varphi$ are fluted formulae over X_m.

The class One-Free [129]
A closed formula φ without non-constant function symbols belongs to the class One-Free if and only if any subformula ψ of φ starting with a quantifier contains no more than one free variable.

Instead of characterising solvable classes as sets of first-order formulae, we can use a characterisation as classes of clauses, or clauses of clause sets. The presence of Skolem functions potentially allows for arbitrarily complex terms in clauses. In this case, the clauses no longer correspond to the clausal form of some first-order formulae. The following classes are defined in [39].

The class $\mathcal{OCCI}$
Let C_+ and C_- denote the subclause of all positive literals and the subclause of all negative literals of a clause C, respectively. A set of clauses N belongs to $\mathcal{OCCI}$ iff for all clauses C in N the following holds: (i) for every variable x in $\mathcal{V}(C_+)$ there is exactly one occurrence of x in C_+, and (ii) for every $x \in \mathcal{V}(C_+) \cap \mathcal{V}(C_-)$, the maximal depth of occurrences of x in C_+ is less than or equal to the maximal depth of occurrences of x in C_-.

The class $\mathbf{M}^+$
Two terms t_1 and t_2 are *congruent* iff both terms are compound and $\arg_{mul}(t_1) = \arg_{mul}(t_2)$. A literal L is *uniform* if either $\arg_{set}(L) \subseteq \mathsf{V} \cup \mathsf{F}_0$ or L has a compound argument term t such that each argument of L is either congruent to t, an argument of t or a constant. A clause set N belongs to $\mathbf{M}^+$ iff for all clauses C in N the following holds: (i) every literal in C is uniform, (ii) C contains no occurrence of nested, non-constant function symbols, and (iii) C contains at most two literals.

The class $\mathcal{PVD}$
Let C_+ and C_- denote the subclause of all positive literals and the subclause of all negative literals of a clause C, respectively. A set of clauses N belongs to $\mathcal{PVD}_+$ iff for all clauses C in N the following holds: (i) $\mathcal{V}(C_+) \subseteq \mathcal{V}(C_-)$, and (ii) for every $x \in \mathcal{V}(C_+)$, the maximal depth of an occurrence of x in C_+ is less than or equal to the maximal depth of an occurrence of x in C_-.

A *sign renaming* is a mapping on clause sets consistently replacing occurrences of literals L by $\overline{L}$ depending on the predicate symbol of L.

A set of clauses N belongs to $\mathcal{PVD}$ (positive variable dominated) if there exists a sign mapping γ such that $\gamma(N)$ is in $\mathcal{PVD}_+$.

The class $\mathcal{S}^+$
A set of clauses N belongs to $\mathcal{S}^+$ iff for all clauses $C \in N$ and all literals L in C the following holds: (i) if t is a compound term in C then $\mathcal{V}(t) = \mathcal{V}(C)$, and (ii) either $\mathcal{V}(L)$ is a singleton set or $\mathcal{V}(L) = \mathcal{V}(C)$.

Chapter 2

The class $\mathbf{E}^+$

The class $\mathbf{E}^+$ was introduced by Tammet [128] and has become one of the most well-studied classes with respect to resolution decision procedures. Unlike classical solvable classes which are fragments of first-order logic, the class $\mathbf{E}^+$ is a class of clauses. $\mathbf{E}^+$ has the pleasant property that given a clause set N in $\mathbf{E}^+$ any clause C derivable by unrestricted resolution and unrestricted factoring again belongs to (some clause set in) $\mathbf{E}^+$. Furthermore, C will not contain more variables than the maximal number of variables occurring in a clause in N [28]. Consequently, the only obstacle to decidability by unrestricted resolution is the fact that there is no upper bound on the maximal depth of terms.

Tammet presents a refined calculus based on ordered resolution using a non-liftable ordering $\succ_v$ and shows termination. On $\mathbf{E}^+$ ordered resolution with respect to $\succ_v$ has the important property that the maximal depth of variables in a resolvent never exceeds the maximal depth of variables in its parent clauses. This immediately establishes decidability of this refinement. However, it was open whether the calculus is complete. De Nivelle [26] was able to establish the completeness of the calculus using a new general technique for proving completeness of non-liftable orderings.

More recently, de Nivelle [28] showed that $\mathbf{E}^+$ can be decided by ordered resolution using a liftable ordering $\succ_d$. In contrast to the ordering $\succ_v$ which is applied a priori, the ordering $\succ_d$ is applied a posteriori. In this resolution refinement the maximal depth of variables in a resolvent can exceed the maximal depth of variables in the parent clauses.

This chapter has mainly introductory character. We will use the class $\mathbf{E}^+$ to demonstrate the basic techniques which we will use in the following chapters for more complicated and more interesting first-order fragments. In particular, we will see (i) how to use renaming to transform formulae and clauses into a suitable more "well-behaved" form, (ii) how to use ordering refinements to restrict the application of resolution and factoring, (iii) how to prove termination of an ordering refinement, and (iv) how to establish relationships between various decision procedures by means of simulation. Along the way, a satisfiability equivalence preserving transformation of clause sets in $\mathbf{E}^+$ will be presented which allows for the definition of a complete decision procedure based on a variety of ordering refinements. We show how some general problems due to the use of renaming techniques can be solved. This allows us to establish a relationship between resolution decision procedures using ordering refinements based on liftable orderings to decision procedures using ordering refinements based on non-liftable orderings.

2.1 The class E^+ and variable uniform clauses

This section defines the class $\mathbf{E}^+$ and gives some technical lemmata. As in Fermüller et al. [39, p. 99] we define:

Definition 2.1 (Covering and weakly covering terms and literals).
A compound term t is *covering* if for every compound subterm s of t the sets of variables of s and t are identical, that is, $\mathcal{V}(s) = \mathcal{V}(t)$. A compound term t is *weakly covering* if for every non-ground, compound subterm s of t $\mathcal{V}(s) = \mathcal{V}(t)$ holds.

An atom or literal L is *covering* if each argument of L is either a constant, a variable, or a covering term t with $\mathcal{V}(t) = \mathcal{V}(L)$. An atom or literal L is *weakly covering*[1] if each argument of L is either a ground term, a variable, or a weakly covering term t with $\mathcal{V}(t) = \mathcal{V}(L)$.

Definition 2.2 (Variable uniform clauses).
A clause C is *variable uniform* if

1. every literal in C is weakly covering, and
2. for each literal L_1 and L_2 in C either $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ or $\mathcal{V}(L_1) \cap \mathcal{V}(L_2) = \emptyset$ holds.

Definition 2.3 (The class $\mathbf{E}^+$).
A clause set N belongs to the class $\mathbf{E}^+$ iff all clauses C in N are variable uniform.

Definition 2.4.
For any literal, L let $\mathrm{GT}(L)$ be the set of all ground subterms of L. Define $\mathrm{dp}^x_{\max}(L)$ to be the maximal depth of occurrences of the variable x in L. If x does not occur in L, let $\mathrm{dp}^x_{\max}(L) = 0$. Let $\mathrm{dp}^{\mathsf{V}}_{\max}(L)$ be the maximal depth of variable occurrences in L. If L is ground, let $\mathrm{dp}^{\mathsf{V}}_{\max}(L) = 0$. Let $\mathrm{dp}^{\mathrm{GT}}_{\max}(L)$ be the maximal depth of elements of $\mathrm{GT}(L)$.

Let N be a finite set of variable uniform clauses. Let Γ_N be the set of all non-ground compound terms in N. With every literal L we can associate a finite set $L*$ of literals which can be obtained from L by applying a substitution σ replacing variables in L by elements of Γ_N with the restriction that the maximal depth of a variable in $L\sigma$ does not the exceed $\mathrm{dp}^{\mathsf{V}}_{\max}(N)$. By $\mathrm{card}^{\mathsf{V}}_{\max}(L)$ we denote maximal number of different variables in elements of $L*$. Note that for any ground literal L $\mathrm{card}^{\mathsf{V}}_{\max}(L)$ is equal to zero.

For example, $\mathrm{dp}^x_{\max}(p(x, f(y), y)) = 1$, $\mathrm{dp}^y_{\max}(p(x, f(y), y)) = 2$, and $\mathrm{dp}^{\mathsf{V}}_{\max}(p(x, f(y), y) = 2$. If a literal L does not contain any compound ground terms, then $\mathrm{dp}(L) = \mathrm{dp}^{\mathsf{V}}_{\max}(L) + 1$.

Lemma 2.5 ([39, p. 102]). *Let σ be a most general unifier of two weakly covering literals L_1 and L_2. Then the following properties hold for $L_1\sigma$:*

1. *$L_1\sigma$ is weakly covering.*
2. $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2))$.
3. $\mathrm{dp}(L_1\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)) + \max(\mathrm{dp}^{\mathrm{GT}}_{\max}(L_1), \mathrm{dp}^{\mathrm{GT}}_{\max}(L_2))$.
4. $\mathrm{GT}(L_1\sigma) \subseteq \mathrm{GT}(L_1) \cup \mathrm{GT}(L_2)$ *or $L_1\sigma$ is ground.*

[1] Note that Fermüller et al. [39] present two definitions of the notion of weakly covering literals. The definition on page 81 of Fermüller et al. [39] is more restrictive. It does not allow variable arguments in weakly variable uniform literals.

5. $|\mathcal{V}(L_1\sigma)| \leq \max(|\mathcal{V}(L_1)|, |\mathcal{V}(L_2)|)$.

Actually, Fermüller proves a stronger result than Lemma 2.5(2) for the case that $L_1\sigma$ is non-ground.

Lemma 2.6 ([39, p. 102–104]). *Let σ be a most general unifier of two weakly covering literals L_1 and L_2 with $L_1\sigma$ non-ground. Then the following properties hold:*

1. $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2))$.
2. $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)$ *or* $\mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$.

It is possible to give a more precise characterisation of the form of $L_1\sigma$. Let G be a set of ground terms. Then $L_1 \geq_G L_2$ if either $L_1 = L_2$ or L_2 can be obtained from L_1 by substituting some variables of L_1 by variables or elements of G.

Lemma 2.7 ([39, p. 102–104]). *Let σ be a most general unifier of two weakly covering literals L_1 and L_2. Let Γ denote the set $\mathrm{GT}(L_1) \cup \mathrm{GT}(L_2)$. Then*

1. *Either $L_1 \geq_\Gamma L_1\sigma$ or $L_2 \geq_\Gamma L_2\sigma$.*
2. *Either the codomain of $\sigma\restriction_{\mathcal{V}(L_1)}$ or $\sigma\restriction_{\mathcal{V}(L_2)}$, or both, contain no non-ground, compound terms.*

Lemma 2.8 ([39, p. 104]). *Let L_1 and L_2 be weakly covering literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$. For every substitution σ if $L_1\sigma$ is weakly covering then $L_2\sigma$ is weakly covering.*

In weakly covering atoms the maximal depth of occurrences is the same for every variable. Since this property is important for our decision procedure, we present a formal proof.

Lemma 2.9. *Let L be a weakly covering literal. Then for each variable x and y in $\mathcal{V}(L)$ the following holds*

$$\mathrm{dp}^{x}_{\max}(L) = \mathrm{dp}^{y}_{\max}(L) = \mathrm{dp}^{\mathsf{V}}_{\max}(L). \tag{2.1}$$

Proof. For ground literals L (2.1) trivially holds.

Suppose L is a non-ground literal such that all compound arguments of L are ground. So all occurrences of variables in L are arguments of L. They all occur at the same depth which coincides with the maximal depth of occurrences of variables in L. Again equation (2.1) holds.

Suppose L is a non-ground literal containing at least one non-ground compound argument. Let $f(t_1, \dots, t_n)$ be a subterm of L at maximal depth d such that one of the argument terms $t_1, \dots, t_n$ is a variable. We show that all variables of L are among the argument terms $t_1, \dots, t_n$. Suppose not. Since $f(t_1, \dots, t_n)$ is a non-ground, compound, and weakly covering term, it contains all variables of L. If there is a variable x of L which is not identical to one of $t_1, \dots, t_n$, then some term t_i has a strict subterm of the form $g(s_1, \dots, s_m)$ such that $x = s_j$ for some term s_j. This contradicts our assumption that $f(t_1, \dots, t_n)$ is a subterm at maximal depth having a variable occurrence among its arguments. So, every variable L is identical to one of the argument terms $t_1, \dots, t_n$. For any variable x in $\mathcal{V}(L)$ the maximal depth of occurrences of x in L is equal to d+1. Thus (2.1) holds. ∎

Lemma 2.10. *Let L_1 and L_2 be weakly covering literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$.*

1. *For any substitution σ such that $L_1\sigma$ is non-ground, if $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2) + k$, for some k, $k \geq 0$, then $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) + k$.*

2. *If $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$, then for any substitution σ, $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma)$.*

Proof. Suppose $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) \geq \mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma)$. Then σ instantiates the variables of L_1 either with variables or ground terms. Since L_1 and L_2 share the same variables, it is not possible that $L_2\sigma$ is ground while $L_1\sigma$ is non-ground or vice versa. If $L_1\sigma$ is ground, then $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = 0 = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma)$. If $L_1\sigma$ is non-ground, then $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma)$ and $\mathrm{dp}^{\mathsf{V}}_{\max}(L_2) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma)$. Thus, $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) + k$.

Suppose $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) < \mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma)$. Then $L_1\sigma$ is non-ground, and some variable of L_1 has been instantiated by a compound term. Let x be a variable in $\mathcal{V}(L_1) \cap \mathrm{D}(\sigma)$ such that $x\sigma$ is a non-ground, compound term of maximal depth in $\mathrm{C}(\mathcal{V}(L_1) \cap \mathrm{D}(\sigma))$. Then $x\sigma$ contains a variable y such that $\mathrm{dp}^{y}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma)$. By Lemma 2.9, $\mathrm{dp}^{x}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)$. Therefore, $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{y}_{\max}(x\sigma) + \mathrm{dp}^{x}_{\max}(L_1) - 1$.

Since $\mathcal{V}(L_1) = \mathcal{V}(L_2)$, the variable x also occurs in L_2. Furthermore, $\mathcal{V}(L_1) \cap \mathrm{D}(\sigma) = \mathcal{V}(L_2) \cap \mathrm{D}(\sigma)$ and $x\sigma$ is also a non-ground, compound term of maximal depth in $\mathrm{C}(\mathcal{V}(L_2) \cap \mathrm{D}(\sigma))$. Again by Lemma 2.9, $\mathrm{dp}^{x}_{\max}(L_2) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$. So,

$$\begin{aligned}
\mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) + k &= \mathrm{dp}^{y}_{\max}(x\sigma) + \mathrm{dp}^{x}_{\max}(L_2) + k \\
&= \mathrm{dp}^{y}_{\max}(x\sigma) + \mathrm{dp}^{\mathsf{V}}_{\max}(L_2) + k \\
&= \mathrm{dp}^{y}_{\max}(x\sigma) + \mathrm{dp}^{\mathsf{V}}_{\max}(L_1) \\
&= \mathrm{dp}^{y}_{\max}(x\sigma) + \mathrm{dp}^{x}_{\max}(L_1) \\
&= \mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma).
\end{aligned}$$

□

Lemma 2.11. *Let L_1 and L_2 be literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ and σ be an arbitrary substitution. If $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$ and for every $x \in \mathcal{V}(L_1)$, $\mathrm{dp}^{x}_{\max}(L_1) > \mathrm{dp}^{x}_{\max}(L_2)$, then $\mathrm{dp}(L_1\sigma) > \mathrm{dp}(L_2\sigma)$.*

Proof. If $\mathrm{dp}(L_2\sigma) \leq \mathrm{dp}(L_1)$, then the lemma holds trivially. Suppose $\mathrm{dp}(L_2\sigma) > \mathrm{dp}(L_1)$. Since $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, some of the variables of L_1 have been instantiated by compound terms. In particular, there is a variable $x \in \mathcal{V}(L_1)$ such that $\mathrm{dp}^{x}_{\max}(L_1) + \mathrm{dp}(x\sigma) = \mathrm{dp}(L_1\sigma)$. However, $\mathrm{dp}^{x}_{\max}(L_2) > \mathrm{dp}^{x}_{\max}(L_1)$ and therefore

$$\begin{aligned}
\mathrm{dp}(L_2\sigma) &\geq \mathrm{dp}^{x}_{\max}(L_2) + \mathrm{dp}(x\sigma) \\
&> \mathrm{dp}^{x}_{\max}(L_1) + \mathrm{dp}(x\sigma) \\
&= \mathrm{dp}(L_1\sigma).
\end{aligned}$$

□

Lemma 2.12. *Let L_1 and L_2 be literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ and for every $x \in \mathcal{V}(L_1)$, $\mathrm{dp}^{x}_{\max}(L_1) \geq \mathrm{dp}^{x}_{\max}(L_2)$. Let σ be a substitution such that $\mathrm{dp}(L_2\sigma) > \mathrm{dp}(L_2)$. Then $\mathrm{dp}(L_1\sigma) \geq \mathrm{dp}(L_2\sigma)$.*

Proof. Without loss of generality we assume that the domain of σ is a subset of $\mathcal{V}(L_2)$. Since $\mathrm{dp}(L_2\sigma) > \mathrm{dp}(L_2)$, the substitution σ instantiates at least one of the variables of L_2 by a compound term. In particular, there exists a a variable x in L_2 such that $\mathrm{dp}^x_{\max}(L_2) + \mathrm{dp}(x\sigma) = \mathrm{dp}(L_2\sigma)$. Since $\mathrm{dp}^x_{\max}(L_1) \geq \mathrm{dp}^x_{\max}(L_2)$, we obtain

$$\begin{aligned}\mathrm{dp}(L_1\sigma) &\geq \mathrm{dp}^x_{\max}(L_1) + \mathrm{dp}(x\sigma)\\ &\geq \mathrm{dp}^x_{\max}(L_2) + \mathrm{dp}(x\sigma)\\ &= \mathrm{dp}(L_2\sigma).\end{aligned}$$

□

Lemma 2.13. *Let $\{L_1, L_2\} \cup C$ be an indecomposable, variable uniform clause such that L_1 and L_2 are unifiable with most general unifier σ. If $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) \neq \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$, then $L_1\sigma$ is ground.*

Proof. We assume that $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ holds and $L_1\sigma$ is non-ground. According to Lemma 2.6(1),

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1).$$

Based on Lemma 2.9 we infer that no variable occurring in L_1 is instantiated by a non-ground compound term. On the other hand, we have

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) = \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2).$$

This is only possible, if one of the variables of L_2 has been instantiated by a non-ground compound term. However, L_1 and L_2 share the same variables which entails that one of the variables of L_1 has been instantiated by a non-ground compound term. We have derived a contradiction. □

2.2 Resolution and factoring on variable uniform clauses

In the literature on the class $\mathbf{E}^+$ the following four orderings play a vital role.

Definition 2.14.
Let L_1 and L_2 be literals. We define the orderings $\succ'_v$, $\succ_v$, $\succ_n$, and $\succ_d$ as follows.

$L_1 \succ'_v L_2$ iff $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$.

$L_1 \succ_v L_2$ iff (i) $\mathcal{V}(L_1) = \mathcal{V}(L_2) \neq \emptyset$,[2] and
(ii) for every x in $\mathcal{V}(L_1)$ we have $\mathrm{dp}^x_{\max}(L_1) > \mathrm{dp}^x_{\max}(L_2)$.

$L_1 \succ_n L_2$ iff (i) $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, and
(ii) either $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ or both L_1 and L_2 are ground.

$L_1 \succ_d L_2$ iff (i) $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, and
(ii) for every x in $\mathcal{V}(L_2)$ we have $\mathrm{dp}^x_{\max}(L_1) > \mathrm{dp}^x_{\max}(L_2)$.

In contrast to the ordering $\succ_d$, the orderings $\succ_v$, $\succ'_v$, and $\succ_n$ are not stable under substitutions. Note that $\succ'_v$ is an extension of $\succ_v$, but coincides with $\succ_v$ on indecomposable, variable uniform clauses. Also, $\succ_n$ coincides with $\succ_d$ on variable uniform clauses.

[2] The definition of $\succ_v$ differs from the original definition in [130] in the additional requirement that L_1 and L_2 are non-ground. This is necessary to ensure that $\succ_v$ is irreflexive on ground literals.

Tammet [39, p. 82] shows that resolution based on an a priori $\succ_v$ ordering refinement always terminates. As mentioned before, completeness was an open problem, since $\succ_v$ is not stable under substitutions. De Nivelle [27] shows that ordered inferences based on the ordering $\succ_v'$ is complete and terminating.

The motivation for using the orderings $\succ_v$ and $\succ_v'$ can be illustrated by the following example. Resolving

$$C_1 = \{\neg p(f(g(h(a)), x)), \neg p(f(g(x), x))\}$$

and

$$C_2 = \{p(f(g(h(a)), h(y)))\}$$

on the first literal of C_1 results in the clause

$$C_3 = \{\neg p(f(g(h(x)), h(x)))\}.$$

The maximal depth of the variable x in C_3 is greater than the maximal depth of x in C_1 and C_2. It is not immediate how an upper bound on this growth of the maximal depth of variables could be established. Resolving C_1 with C_2 on the second literal of C_1 (which is both $\succ_v$- and $\succ_v'$-maximal) results in the ground clause

$$C_4 = \{\neg p(f(g(h(a)), h(a)))\}.$$

Clearly, the value of $\mathrm{dp}^{\vee}_{\mathrm{max}}(C_4)$ is smaller than the maximum of $\mathrm{dp}^{\vee}_{\mathrm{max}}(C_1)$ and $\mathrm{dp}^{\vee}_{\mathrm{max}}(C_2)$.

Tammet shows that restricting resolution to the $\succ_v$-maximal literals of a clause ensures that the maximal variable depth of resolvents is less than or equal to the maximum of the maximal variable depths of the parent clauses. This property is one of the prerequisites for termination. Because the literals of C_1 have a common (ground) instance $\neg p(f(g(h(a)), h(a)))$, there is no ordering $\succ$ which is stable under substitutions such that the second literal of C_1 is strictly $\succ$-maximal in C_1.

There are three important points to note concerning de Nivelle's approach. First, it is vital that the ordering $\succ_n$ is used a posteriori. De Nivelle illustrates this by the following example. If the ordering $\succ_n$ is applied a priori to the clause

$$C_5 = \{\neg p(x, s(s(s(0)))), p(s(x), s(s(s(0))))\}$$

then both literals in C_5 are $\succ_n$-maximal, since their depths are equal. Given the clause

$$C_6 = \{p(0, s(s(s(0))))\}$$

we are then able to derive the clauses $\{p(s^n(0), s(s(s(0))))\}$ for arbitrarily large n, $n \geq 1$. Thus, an a priori $\succ_n$ ordering refinement does not ensure termination. Note that the same example can be used to demonstrate that an a posteriori $\succ_v$ ordering refinement does not provide a decision procedure, since in ground instances of C_5 both literals are $\succ_v$-maximal which allows for the derivation of an infinite set of clauses.

Second, the maximal depth of variables in resolvents can be greater than the maximum of the maximal depths of variables in the parent clauses. This can be illustrated by the resolvent of C_5 with itself, which is

$$C_7 = \{\neg p(x, s(s(s(0)))), p(s(s(x)), s(s(s(0))))\}.$$

However, there is an upper bound on the maximal depth of variables. If d_c is the maximal depth of clauses in a clause set N in $\mathbf{E}^+$, then the maximal depth of variables in any clause derivable from N with respect to the $\succ_n$-refinement will not exceed d_c. By contrast, if d_v is the maximal depth of variables in N, then the maximal depth of variables in any clause derivable from N with respect to the $\succ_v$-refinement will not exceed d_v. Consequently, if d_c is greater than d_v, then potentially more clauses are derivable using the $\succ_n$-refinement as opposed to the $\succ_v$-refinement. For example, based on the $\succ_v$-refinement the clause C_7 is not derivable from C_5. This could be considered to be a disadvantage.

Using an ordering which is stable under substitutions has the advantage that more powerful redundancy elimination techniques can be used compared to calculi based on non-liftable orderings. This can compensate the disadvantage of the $\succ_d$-refinement over the $\succ_v$-refinement noted before.

The basic idea of our decision procedure for the class $\mathbf{E}^+$ is as follows. All the orderings coincide with each other on non-ground variable uniform clauses if

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L) = \mathrm{dp}(L) \tag{2.2}$$

holds for every literal L in a clause set. We will adopt the strategy of replacing occurrences of non-ground literals L violating condition (2.2) in a set N of clauses by literals L' for which condition (2.2) holds in a satisfiability equivalence preserving way. Then, any of the orderings of Definition 2.14 may be used to restrict the inference. It will also be irrelevant whether maximality is determined a priori or a posteriori.

The notion of depth distorted clauses introduced next is an (easily computable) approximation identifying potentially critical cases.

Definition 2.15.
A clause C is *depth distorted* if it contains literals L_1 and L_2 such that $\mathrm{dp}^{\mathsf{V}}_{\max}(C) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ and $\mathrm{dp}(L_1) \leq \mathrm{dp}(L_2)$. The literal L_2 is called *$\succ_d$-distorting literal*. A clause which is not depth distorted is called *depth undistorted*. A non-ground literal L_1 such that $\mathrm{dp}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)$ is *depth dominated*.

Note that a $\succ_d$-distorting literal L_2 is depth dominated. The motivation for the notion of depth distorted clauses is, that they contain a literal L_2 such that L_2 is not $\succ_v$-maximal, but potentially $\succ_d$-maximal. It is possible, that with respect to any substitution σ used in an inference step with a depth distorted clause, $L_2\sigma$ is not $\succ_d$-maximal. The clauses C_1, C_5, and C_7 are examples of depth distorted clauses.

Let N be a set of clauses. For every n, $1 \leq n$, let v_n be a new n-ary function symbol, and h be a new unary function symbol[3]. Let L be a literal with variables $x_1, \ldots, x_n$, $0 \leq n$. We define

[3]That is, neither v_n nor h occur in N

a function $\mathrm{d}_\downarrow$ on literals as follows.

$$\mathrm{d}_\downarrow(L) = \begin{cases} p & \text{if } L \text{ is ground,} \\ p(x_1, \ldots, x_n) & \text{if } \mathrm{dp}^{\mathsf{V}}_{\max}(L) = 1, \\ p(h^{k-2}(v_n(x_1, \ldots, x_n))) & \text{if } \mathrm{dp}^{\mathsf{V}}_{\max}(L) = k \geq 2, \end{cases}$$

where p is a new predicate symbol of appropriate arity uniquely associated with L (up to renaming of variables). It is straightforward to see that for every literal L, $\mathrm{dp}^{\mathsf{V}}_{\max}(L) = \mathrm{dp}^{\mathsf{V}}_{\max}(\mathrm{d}_\downarrow(L))$.

We define the following renaming transformation:

$N \Rightarrow_{\mathcal{D}} N' \cup \mathrm{Eqv}^A_L$ iff (i) L is an occurrence of a $\succ_d$-distorting literal
(ii) $A = \mathrm{d}_\downarrow(L)$ and Eqv^A_L is the set of clauses $\{\{\neg A, L\}, \{\neg L, A\}\}$
(iii) N' is obtained from N by replacing any occurrence of L by A.

In Eqv^A_L we denote $\{\neg A, L\}$ by Def^A_L and $\{\neg L, A\}$ by Def^L_A. These clauses are called *definitions*. As far as replacing occurrences of L by A is concerned, we assume that for any occurrence of L we have $\mathcal{V}(L) = \{x_1, \ldots, x_n\}$. This can be achieved by renaming the variables of clauses in N appropriately.

Note that L is not $\succ_d$-distorting in Def^A_L and Def^A_L (and neither is A). So, every transformation step eliminates at least one occurrence of an $\succ_d$-distorting literal. Therefore, any sequence of applications of $\Rightarrow_{\mathcal{D}}$ to N terminates. We denote the resulting clause set by $N{\downarrow_{\mathcal{D}}}$.

By $\mathrm{Eqv}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ we denote the union of all sets Eqv^A_L introduced in the process of transforming N to $N{\downarrow_{\mathcal{D}}}$. Similarly, $\mathrm{Def}^{\rightarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ and $\mathrm{Def}^{\leftarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ denote the union of all sets Def^A_L and Def^L_A, respectively. We can show the following.

Theorem 2.16.
Let N be a finite set of clauses. Then $N{\downarrow_{\mathcal{D}}}$ can be computed in polynomial time and is satisfiable if and only if N is satisfiable.

Proof. Since renaming is satisfiability equivalence preserving. □

In fact, $\mathrm{Def}^{\rightarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ instead of $\mathrm{Eqv}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ would have sufficed, that is, N is satisfiable iff $N{\downarrow_{\mathcal{D}}} \setminus \mathrm{Def}^{\leftarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ is satisfiable. The clauses in $\mathrm{Def}^{\leftarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ will be used later (in Section 2.3) to relate our procedure to existing decision procedures.

The clauses in $N{\downarrow_{\mathcal{D}}}$ still contain depth dominated literals and literals with compound ground terms. Consequently, inference may produce depth distorted clauses. For example, we can derive

$$\{q(g(x), a), q(g(a), x)\}$$

from

$$\{q(g(x), a), \neg p(g(g(x)))\}$$

and

$$\{q(g(a), x), p(g(g(x)))\}.$$

This may seem odd. By Lemma 2.5(4) inference steps by resolution and factoring from clauses not containing compound ground terms will never result in non-ground clauses containing such terms. This raises the question why we do not rename all occurrences of literals containing compound ground terms. The problem is that even with this form of renaming there will be compound literals in $\mathrm{Def}^{\rightarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$, so inference steps with one of the clauses in $\mathrm{Def}^{\rightarrow}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ will reintroduce compound ground terms and depth dominated literals into derived clauses.

Consider the clause set consisting of the three clauses

$$\begin{array}{ll} (3) & \{p(g(g(x)))\} \\ (4) & \{q(g(z), z), \neg p(g(g(g(z)))), \neg p(g(f(g(a), z)))\} \\ (5) & \{q(x, y), p(g(f(x, y)))\}. \end{array}$$

The third literal in (4) is both $\succ_d$-distorting and contains the only compound ground term. The transformation $\Rightarrow_{\mathcal{D}}$ will replace clause (4) by

$$(6) \quad \{q(g(z), z), \neg p(g(g(g(z)))), p_1(h(v_1(z)))\}$$

and add the clauses

$$\begin{array}{ll} (7) & \{\neg p_1(h(v_1(z))), \neg p(g(f(g(a), z)))\} \\ (8) & \{p_1(h(v_1(z))), p(g(f(g(a), z)))\}. \end{array}$$

The derivation continues as follows.

$$\begin{array}{lll} [(6)2,\mathrm{R},(3)1] & (9) & \{q(g(z), z), p_1(h(v_1(z)))\} \\ [(9)2,\mathrm{R},(7)1] & (10) & \{q(g(z), z), \neg p(g(f(g(a), z)))\} \\ [(10)2,\mathrm{R},(5)2] & (11) & \{q(g(z), z), q(g(a), z)\}. \end{array}$$

Here [(6)2,R,(3)1] denotes that the second literal of clause (6) is resolved with the first literal of clause (3). Analogously, [(6)1,F,(6)2] denotes an inference by ordered factoring on the first and second literal of clause (6).

Clause (11) is depth distorted. To avoid the generation of clause (11) we could try to prevent resolution on the first literal of (7) by choosing an appropriate refinement of $\succ_d$. Now, assume that we are only allowed to resolve on the second literal of (7). Then we obtain the following alternative derivation.

$$\begin{array}{lll} [(6)2,\mathrm{R},(3)1] & (9') & \{q(g(z), z), p_1(h(v_1(z)))\} \\ [(7)2,\mathrm{R},(5)2] & (10') & \{q(g(a), y), \neg p_1(h(v_1(y)))\} \\ [(9')2,\mathrm{R},(10')2] & (11') & \{q(g(z), z), q(g(a), z)\}. \end{array}$$

Clause (11') is depth distorted. This shows, even if all literals containing compound terms are renamed, inferences by ordered resolution (with respect to $\succ_d$ or a refinement of it) may derive new depth distorted clauses.

Instead of applying $\Rightarrow^*_{\mathcal{D}}$ once before the theorem proving derivation, we augment the set of expansion rules by dynamic renaming:

$\downarrow_{\mathcal{D}}$-Renaming: $\dfrac{N}{N{\downarrow_{\mathcal{D}}}}$

if N contains a depth distorted clause.

We require that "$\downarrow_{\mathcal{D}}$-Renaming" is applied eagerly during a theorem proving derivation. The addition of "$\downarrow_{\mathcal{D}}$-Renaming" does not affect the completeness of the calculus. Note that adding the clauses in $\mathrm{Eqv}_{\mathcal{D}}(N{\downarrow_{\mathcal{D}}})$ to N without modifying N itself, preserves satisfiability equivalence. We could assume without loss of generality that all possible definitions are added before commencing the derivation. Now, given an appropriate refinement $\succ'_d$ of $\succ_d$, "$\downarrow_{\mathcal{D}}$-Renaming" becomes a simplification operation in the sense of Bachmair and Ganzinger [8].

We define a precedence $\succ_{\mathsf{P}}$ on predicate symbols such that the predicate symbol of $\mathrm{d}_{\downarrow}(L)$ is smaller than the predicate symbol of L. Let p_L denote the predicate symbol of a literal L. On literals $\succ_{\mathsf{P}}$ is defined by $L_1 \succ_{\mathsf{P}} L_2$ if (i) $\mathrm{dp}(L_1) = \mathrm{dp}(L_2)$ (ii) for every x in $\mathcal{V}(L_2)$ we have $\mathrm{dp}^x_{\max}(L_1) = \mathrm{dp}^x_{\max}(L_2)$, and (iii) $p_{L_1} \succ_{\mathsf{P}} p_{L_2}$. The ordering $\succ_{\mathsf{P}}$ is stable under substitutions and disjoint from $\succ_d$. The union $\succ'_d$ of $\succ_d$ and $\succ_{\mathsf{P}}$ is a refinement of $\succ_d$ and again stable under substitutions.

Theorem 2.17.
The expansion rule "$\downarrow_{\mathcal{D}}$-Renaming" is a simplification.

Proof. Let $C \cup \{L\}$ be a clause with L $\succ_d$-distorting. For any ground instance $(C \cup \{L\})\sigma$, we can use the definition $\{\neg L, A\}$ to derive $(C \cup \{A\})\sigma$, that is, instead of applying $\Rightarrow_{\mathcal{D}}$ we simply perform an inference step.

Using the definition $\{\neg A, L\}$ we could derive $(C\cup\{L\})\sigma$ from $(C\cup\{A\})\sigma$, that is, $(C\cup\{A\})\sigma$ and $\{\neg A, L\}\sigma$ logically imply $(C \cup \{L\})\sigma$. It remains to show that the premises are smaller than the conclusion with respect to $\succ'_d$. Note that $\mathrm{dp}(L\sigma) \geq \mathrm{dp}(A\sigma)$ holds. If $\mathrm{dp}(L\sigma) > \mathrm{dp}(A\sigma)$, then $L\sigma \succ_d A\sigma$, since $L\sigma$ and $A\sigma$ are ground. If $\mathrm{dp}(L\sigma) = \mathrm{dp}(A\sigma)$, then $p_{L\sigma} \succ_{\mathsf{P}} p_{A\sigma}$ and therefore $L\sigma \succ_{\mathsf{P}} A\sigma$. In both cases $L\sigma \succ'_d A\sigma$. Therefore, $(C \cup \{A\})\sigma$ is smaller than $(C \cup \{L\})\sigma$ with respect to the multiset extension of $\succ'_d$. Now consider the clauses $\{\neg A, L\}\sigma$ and $(C\cup\{L\})\sigma$. The literal L is $\succ_d$-distorting in $C \cup \{L\}$, that is, there is a literal L_2 in C with

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_2) = \mathrm{dp}^{\mathsf{V}}_{\max}(C) > \mathrm{dp}^{\mathsf{V}}_{\max}(L).$$

With respect to A we have

$$\mathrm{dp}(L_2) \geq \mathrm{dp}^{\mathsf{V}}_{\max}(L_2) > \mathrm{dp}^{\mathsf{V}}_{\max}(L) = \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A) = \mathrm{dp}(\neg A).$$

Consequently, $\mathrm{dp}(L_2\sigma) > \mathrm{dp}(\neg A\sigma)$ and $L_2\sigma \succ_d \neg A\sigma$ hold. Thus, $\{\neg A, L\}\sigma$ is smaller than $(C \cup \{L\})\sigma$ with respect to the multiset extension of $\succ'_d$, since $C\sigma$ contains a literal $L_2\sigma$ with $L_2\sigma \succ'_d \neg A\sigma$. ☐

We will now show some basic properties of depth undistorted clauses.

Lemma 2.18. *Let $\{L_1\} \cup C$ be an indecomposable, depth undistorted, variable uniform clause. If L_1 is $\succ_n$-maximal with respect to C, then $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(C)$.*

Proof. The lemma is obviously true if C is the empty clause. Suppose C is not the empty clause. Since $\{L_1\} \cup C$ is indecomposable and variable uniform, we have $\mathcal{V}(C) = \mathcal{V}(L_1) = \mathcal{V}(L_2) \neq \emptyset$ for all literals L_2 in C.

Assume that there is a literal $L_2 \neq L_1$ in C such that $\mathrm{dp}^{\mathsf{V}}_{\max}(L_2) = \mathrm{dp}^{\mathsf{V}}_{\max}(C) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)$. Since $L_2 \not\succ_n L_1$, we must have $\mathrm{dp}(L_2) \leq \mathrm{dp}(L_1)$. This means, L_1 is a $\succ_d$-distorting literal and $\{L_1\} \cup C$ is not a depth undistorted clause. ☐

To lay the grounds for an investigation of the relationship between $\succ'_v$ and $\succ_n$ on depth undistorted clauses, we now prove that the orderings $\succ_v$ and $\succ'_v$ as well as $\succ_n$ and $\succ_d$ coincide on weakly covering literals sharing the same set of variables and the stability of all these orderings under non-ground substitutions.

Lemma 2.19. *Let L_1 and L_2 be weakly covering literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$.*

1. *If σ is a substitution such that $L_1\sigma$ is non-ground, and $\succ$ is one of the orderings $\succ'_v$, $\succ_v$, $\succ_n$, and $\succ_d$, then $L_1 \succ L_2$ implies $L_1\sigma \succ L_2\sigma$.*

2. *$L_1 \succ_v L_2$ if and only if $L_1 \succ'_v L_2$.*

3. *$L_1 \succ_d L_2$ if and only if $L_1 \succ_n L_2$.*

Proof. Suppose $L_1\sigma$ and $L_2\sigma$ are non-ground and $L_1 \succ'_v L_2$ holds. Then there is a natural number k, $k \geq 1$, such that

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2) + k > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2).$$

By Lemma 2.10(1), it follows that

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma) + k > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma).$$

So, $L_1\sigma \succ'_v L_2\sigma$.

Let L_1 and L_2 be weakly covering literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ and $L_1 \succ_v L_2$. By Lemma 2.9, $\mathrm{dp}^{x}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)$ and $\mathrm{dp}^{x}_{\max}(L_2) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$, for every variable x in $\mathcal{V}(L_1) = \mathcal{V}(L_2)$. Let x be an arbitrary variable in $\mathcal{V}(L_1)$. Since $L_1 \succ_v L_2$, we have

$$\mathrm{dp}^{x}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2) = \mathrm{dp}^{x}_{\max}(L_2).$$

Thus, $L_1 \succ'_v L_2$. In the same way we can show that $L_1 \succ_v L_2$ entails $L_1 \succ'_v L_2$. By the previous case,

$$L_1 \succ_v L_2 \text{ implies } A \succ'_v L_2 \text{ implies } A\sigma \succ'_v L_2\sigma \text{ implies } A\sigma \succ_v L_2\sigma.$$

Suppose $L_1\sigma$ and $L_2\sigma$ are non-ground and $L_1 \succ_n L_2$ holds. We have that $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ by (ii) of the Definition of $\succ_n$, and by Lemma 2.10(1) also $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma)$. By (i) of the Definition of $\succ_n$ and Lemma 2.11 it follows that $\mathrm{dp}(L_1\sigma) > \mathrm{dp}(L_2\sigma)$. Thus, $L_1\sigma \succ_n L_2\sigma$.

Let L_1 and L_2 be weakly covering literals such that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$. If L_1 and L_2 are ground, then the condition that for every $x \in \mathcal{V}(L_1)$ we have $\mathrm{dp}^{x}_{\max}(L_1) > \mathrm{dp}^{x}_{\max}(L_2)$ is true, while the condition $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ is false. On the assumption that we only compare literals sharing the same set of variables, we can rewrite the definitions of $\succ_n$ and $\succ_d$ as follows.

$L_1 \succ_n L_2$ iff (i) $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, and
(ii) either $L_1 \succ'_v L_2$ or both L_1 and L_2 are ground.

$L_1 \succ_d L_2$ iff (i) $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, and
(ii) either $L_1 \succ_v L_2$ or both L_1 and L_2 are ground.

Hence, the difference between $\succ_n$ and $\succ_d$ is that between $\succ'_v$ and $\succ_v$. However, we have already shown that $\succ'_v$ and $\succ_v$ coincide on literals sharing the same set of variables. Thus, $\succ_n$ and $\succ_d$ also coincide on these literals.

Finally, we have to show that $L_1 \succ_d L_2$ implies $L_1\sigma \succ_d L_2\sigma$. This follows immediately from the fact that $\succ_n$ and $\succ_d$ coincide on literals sharing the same set of variables and the fact that $\succ_n$ is stable under non-ground substitutions. $\square$

The following lemma is concerned with the relationship between $\succ'_v$ and $\succ_n$ on depth undistorted clauses.

Lemma 2.20. *Let $\{L\} \cup C$ be an indecomposable, depth undistorted, variable uniform clause. The literal L is $\succ_n$-maximal with respect to C if and only if L is $\succ'_v$-maximal with respect to C.*

Proof. The lemma holds if C is the empty clause. If C is not the empty clause, then all literals in C as well as L are non-ground. In this case, we have $L_1 \succ_n L_2$ if and only if $L_1 \succ'_v L_2$ and $\mathrm{dp}(L_1) > \mathrm{dp}(L_2)$, that is, $\succ'_v$ is a refinement of $\succ_n$. Consequently, if L is $\succ'_v$-maximal with respect C, then L is also $\succ_n$-maximal with respect to C.

Suppose L is $\succ_n$-maximal with respect to C. By Lemma 2.18 we have $\mathrm{dp}^{\mathsf{V}}_{\max}(L) = \mathrm{dp}^{\mathsf{V}}_{\max}(C)$, that is, there can be no literal L_2 in C such that the maximal variable depth of L_2 is greater than the maximal variable depth of L. Thus, L is $\succ'_v$-maximal with respect to C. □

Lemma 2.21. *Let $\{L\} \cup C$ be an indecomposable, depth undistorted, variable uniform clause and σ be a substitution such that $L\sigma$ is weakly covering. If $L\sigma$ is $\succ_n$-maximal with respect to $C\sigma$, then L is $\succ'_v$-maximal with respect to C.*

Proof. The orderings $\succ_n$ and $\succ_d$ coincide on $\{L\} \cup C$ and $(\{L\} \cup C)\sigma$, so $L\sigma$ is also $\succ_d$-maximal. Since $\succ_d$ is stable under substitutions, L is $\succ_d$-maximal and therefore $\succ_n$-maximal with respect to C. By Lemma 2.20, L is also $\succ'_v$-maximal with respect to C. □

The Lemmata 2.20 and 2.21 show that it is not essential in our framework to apply $\succ_n$ a posteriori. Reconsider the depth distorted clause C_5

$$C_5 = \{\neg p(x, s(s(s(0)))), p(s(x), s(s(s(0))))\}.$$

The transformation $\Rightarrow_{\mathcal{D}}$ will replace clause C_5 by C'_5

$$C'_5 = \{p_1(x), p(s(x), s(s(s(0))))\}.$$

While in C_5 both literals are $\succ_n$-maximal, only $p(s(x), s(s(s(0))))$ is $\succ_n$-maximal in C'_5. There exists no infinite derivation from C'_5 using the a priori $\succ_n$-refinement.

Theorem 2.22.
Let $\{L_1, L_2\} \cup C$ be an indecomposable, variable uniform clause such that L_1 and L_2 are unifiable with most general unifier σ. Then

$$\mathrm{dp}^{\mathsf{V}}_{\max}((\{L_1\} \cup C)\sigma) \leq \mathrm{dp}^{\mathsf{V}}_{\max}(\{L_1, L_2\} \cup C). \tag{2.12}$$

Proof. Suppose $L_1\sigma$ is a ground literal. Since $\mathcal{V}(C) \subseteq \mathcal{V}(L_1)$, the factor $(\{L_1\} \cup C)\sigma$ is ground and (2.12) holds.

Suppose $L_1\sigma$ is a non-ground literal. By Lemma 2.13 we have $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$ and by Lemma 2.10(2) this implies $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2\sigma)$. None of the variables in L_1 and C is instantiated by a non-ground compound term. Consequently, $\mathrm{dp}^{\mathsf{V}}_{\max}((\{L_1\} \cup C)\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(\{L_1, L_2\} \cup C)$. □

Theorem 2.23.
Let $\{A_1\} \cup D_1$ and $\{\neg A_2\} \cup D_2$ be indecomposable, variable uniform clauses such that A_1 is $\succ_n$-maximal with respect to D_1, $\neg A_2$ is $\succ_n$-maximal with respect to D_2, and A_1 and A_2 are unifiable with most general unifier σ. Then

$$\mathrm{dp}^{\mathsf{V}}_{\max}((D_1 \cup D_2)\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(\{A_1\} \cup D_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\{\neg A_2\} \cup D_2)). \tag{2.13}$$

Proof. The inequality (2.13) holds if $(D_1 \cup D_2)\sigma$ is ground. In the remainder of the proof we assume that $(D_1 \cup D_2)\sigma$, and therefore its parent clauses, is non-ground. By Lemma 2.5(2) and Lemma 2.18,

$$\begin{aligned}\mathrm{dp}^{\mathsf{V}}_{\max}(A_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A_2\sigma) &\leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(\{A_1\} \cup D_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\{\neg A_2\} \cup D_2))) \\ &= \max(\mathrm{dp}^{\mathsf{V}}_{\max}(A_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A_2)).\end{aligned} \tag{2.14}$$

Let L_3 and L_4 be arbitrary literals in D_1 and D_2, respectively. Since $\{A_1\} \cup D_1$ and $\{\neg A_2\} \cup D_1$ are indecomposable, variable uniform clause, we have $\mathcal{V}(L_3) = \mathcal{V}(A_1)$ and $\mathcal{V}(L_4) = \mathcal{V}(\neg A_2)$. Again by Lemma 2.18, $\mathrm{dp}^{\mathsf{V}}_{\max}(L_3) \leq \mathrm{dp}^{\mathsf{V}}_{\max}(A_1)$ and $\mathrm{dp}^{\mathsf{V}}_{\max}(L_4) \leq \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A_2)$. Since L_3 and L_4 are non-ground we obtain by Lemma 2.10,

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_3\sigma) \leq \mathrm{dp}^{\mathsf{V}}_{\max}(A_1\sigma) = \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A_2\sigma) \geq \mathrm{dp}^{\mathsf{V}}_{\max}(L_4\sigma). \tag{2.15}$$

Taking (2.14) and (2.15) together we obtain

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_3\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(\{A_1\} \cup D_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\{\neg A_2\} \cup D_2)))$$

and

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_4\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(\{A_1\} \cup D_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\{\neg A_2\} \cup D_2))).$$

This proves (2.13). □

By Lemma 2.21 and Theorem 2.23 it follows:

Corollary 2.24. *Let $\{A_1\} \cup D_1$ and $\{\neg A_2\} \cup D_2$ be indecomposable, variable uniform clauses such that $A_1\sigma$ is (strictly) $\succ_n$-maximal with respect to D_1, $\neg A_2\sigma$ is $\succ_n$-maximal with respect to D_2, and A_1 and A_2 are unifiable with most general unifier σ. Then*

$$\mathrm{dp}^{\mathsf{V}}_{\max}((D_1 \cup D_2)\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(\{A_1\} \cup D_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\{\neg A_2\} \cup D_2)). \tag{2.16}$$

It remains to show that there is not only a bound on the maximal depth of variables in derived clauses, but also on the depth of the clauses.

Theorem 2.25.
Let N be a set of variable uniform clauses. Let $\max^{\mathrm{dp}}_N$ the maximal depth of clauses in N and $\max^{\mathsf{V}}_N$ be the maximal depth of variables in clauses in N. Then, for any clause C derivable from N

$$\mathrm{dp}^{\mathsf{V}}_{\max}(C) \leq \max^{\mathsf{V}}_N \tag{2.17}$$

and

$$\mathrm{dp}(C) \leq \max_N^{\mathsf{V}} + \max_N^{\mathrm{dp}}. \tag{2.18}$$

If C is non-ground, then

$$\mathrm{dp}(t) \leq \max_N^{\mathrm{dp}}, \tag{2.19}$$

for every ground term t in C. Otherwise,

$$\mathrm{dp}(t) \leq \max_N^{\mathsf{V}} + \max_N^{\mathrm{dp}}. \tag{2.20}$$

Proof. The proof proceeds by induction on the length of the derivation of C. The inequalities (2.17) to (2.20) hold for any clause C which is an element of N.

Suppose C is the result of applying the "Splitting" rule to a clause D. Since C is a subclause of D and (2.17) to (2.20) hold for D by the induction hypothesis, they hold for the clause C as well.

Suppose C has been added by an application of the "$\downarrow_{\mathcal{D}}$-Renaming" rule. Then C is not a ground clause. It is straightforward to check that literals in C have at most the depth and maximal depth of variables of literals already occurring in the clause set and that this rule does not introduce any new ground terms. Therefore, (2.17), (2.18) and (2.19) hold for C.

Let C be a factor of an indecomposable, variable uniform clause C_1. By Theorem 2.22, (2.17) is true for C. If C_1 is ground, then C is a subclause of C_1 and (2.18) and (2.20) hold. Let $C_1 = \{L_1, L_2\} \cup D_1$ and $C = (\{L_1\} \cup D_1)\sigma$ where σ is the most general unifier of L_1 and L_2. Suppose C is non-ground. By Lemma 2.13, $\mathrm{dp}^{\mathsf{V}}_{\max}(L_1) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_2)$. By Lemma 2.5(2),

$$\begin{aligned}\mathrm{dp}^{\mathsf{V}}_{\max}(L_1\sigma) &\leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(L_1), \mathrm{dp}^{\mathsf{V}}_{\max}(L_2))\\ &= \mathrm{dp}^{\mathsf{V}}_{\max}(L_1)\\ &\leq \max_N^{\mathsf{V}}.\end{aligned} \tag{2.21}$$

Furthermore, for any literal L_3 in C_1 we have $\mathrm{dp}^{\mathsf{V}}_{\max}(L_3) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_3\sigma)$ and therefore, $\mathrm{dp}^{\mathsf{V}}_{\max}(C) = \mathrm{dp}^{\mathsf{V}}_{\max}(C_1)$. By Lemma 2.5(4), any ground term in $L_1\sigma$, and therefore in the codomain of σ, already occurs in L_1 or L_2. Since C_1 is variable uniform, none of the literals in C contains a ground term which does not occur in L_1 or L_2. Therefore, inequality (2.19) holds for C_1 and we obtain

$$\begin{aligned}\mathrm{dp}(C) &\leq \mathrm{dp}^{\mathsf{V}}_{\max}(C_1) + \max(\mathrm{dp}^{\mathrm{GT}}_{\max}(L_1), \mathrm{dp}^{\mathrm{GT}}_{\max}(L_2)) &&\\ &\leq \mathrm{dp}^{\mathsf{V}}_{\max}(C_1) + \max_N^{\mathrm{dp}} && \text{by inequality (2.19)}\\ &\leq \max_N^{\mathrm{dp}} + \max_N^{\mathsf{V}} && \text{by inequality (2.21).}\end{aligned} \tag{2.22}$$

Suppose C is ground. By Lemma 2.7 and the fact that $\mathcal{V}(L_1) = \mathcal{V}(L_2)$, we know that the codomain of $\sigma\restriction_{\mathcal{V}(L_1)}$ contains only ground terms in $\mathrm{GT}(L_1) \cup \mathrm{GT}(L_2)$. Therefore, inequality (2.22) also holds in this case which shows that C satisfies the inequalities (2.18) and (2.20).

Let C be a resolvent of two indecomposable, variable uniform clauses C_1 and C_2. If both C_1 and C_2 are ground, then $\mathrm{dp}(C) \leq \mathrm{dp}(C_1) = \mathrm{dp}(C_2)$ and the inequalities (2.18) and (2.20) hold.

Otherwise, let C_1, C_2, and C be of the form $\{A_1\} \cup D_1$, $\{\neg A_2\} \cup D_2$, and $(D_1 \cup D_2)\sigma$, respectively, where A_1 is $\succ'_v$-maximal with respect to D_1, $\neg A_2$ is $\succ'_v$-maximal with respect to D_2, and σ is the most general unifier of A_1 and A_2.

By Theorem 2.23, inequality (2.17) is satisfied by C. By Lemma 2.5(2), the depth of $A_1\sigma$ and $\neg A_2\sigma$ is bounded by

$$\text{(2.23)} \quad \mathrm{dp}(\neg A_2\sigma) = \mathrm{dp}(A_1\sigma) \leq \max(\mathrm{dp}^{\mathsf{V}}_{\max}(A_1), \mathrm{dp}^{\mathsf{V}}_{\max}(\neg A_2)) + \max(\mathrm{dp}^{\mathrm{GT}}_{\max}(A_1), \mathrm{dp}^{\mathrm{GT}}_{\max}(\neg A_2)).$$

However, this fact alone does not provide a bound on the depth of literals in C. Let $L\sigma$ be a literal of maximal depth in $D_1\sigma$ and $D_2\sigma$. If $\mathrm{dp}(L\sigma) \leq \mathrm{dp}(L)$, then $\mathrm{dp}(C) \leq \max(\mathrm{dp}(C_1), \mathrm{dp}(C_2))$ and inequality (2.18) holds for C. Suppose $\mathrm{dp}(L\sigma) > \mathrm{dp}(L)$. Since A_1 and $\neg A_2$ are $\succ'_v$-maximal with respect to D_1 and D_2, respectively, and C_1 and C_2 are variable uniform, either $\mathrm{dp}^x_{\max}(A_1) \geq \mathrm{dp}^x_{\max}(L)$ for every $x \in \mathcal{V}(A_1)$, or $\mathrm{dp}^x_{\max}(\neg A_2) \geq \mathrm{dp}^x_{\max}(L)$ for every $x \in \mathcal{V}(\neg A_2)$. By Lemma 2.12 it follows that either $\mathrm{dp}(A_1\sigma) \geq \mathrm{dp}(L\sigma)$ or $\mathrm{dp}(\neg A_2\sigma) \geq \mathrm{dp}(L\sigma)$. In both cases,

$$\mathrm{dp}(L\sigma) \leq \max(\mathrm{dp}(A_1\sigma), \mathrm{dp}(\neg A_2\sigma)) = \mathrm{dp}(A_1\sigma).$$

By the induction hypothesis, the inequalities (2.17) and (2.19) hold for C_1 and C_2. Together with (2.23) this is sufficient to show that inequality (2.18) holds for C.

It remains to consider (2.19) and (2.20). By (2.23) and the induction hypothesis,

$$\mathrm{dp}(\neg A_2\sigma) = \mathrm{dp}(A_1\sigma) \leq \max^{\mathrm{dp}}_N + \max^{\mathsf{V}}_N.$$

We have already proved that the depth of any literal $L\sigma$ in $(D_1 \cup D_2)\sigma$ is less or equal to the depth of L or the depth of $A_1\sigma$. Since the depth of L also does not exceed $\max^{\mathrm{dp}}_N + \max^{\mathsf{V}}_N$, we conclude that (2.20) holds for any term in C. ☐

Theorem 2.26.
Let N be a finite set of variable uniform clauses. Then any derivation from N by ordered resolution and ordered factoring based on either $\succ'_v$, $\succ_v$, $\succ_n$, or $\succ_d$ augmented with an eager application of the "$\downarrow_{\mathcal{D}}$-Renaming" expansion rule terminates.

Proof. By Theorem 2.25 there is a bound on the depth of any clause derivable from N by a fair theorem proving derivation based on one of the orderings in Definition 2.14.

Given a finite signature, there can only be finitely many such clauses and the derivation eventually terminates. However, applications of the "$\downarrow_{\mathcal{D}}$-Renaming" rule can extend the signature by the introduction of new function and predicate symbols. The rule "$\downarrow_{\mathcal{D}}$-Renaming" makes use of one distinguished unary function symbol h and n^{ar}_N distinguished function symbols v_k, $1 \leq k \leq n$, where n^{ar}_N is the maximal arity of function symbols in N. Thus, the number of new function symbols introduced is bounded. It remains to show that the number of new predicate symbols is bounded as well. This amounts to verifying that "$\downarrow_{\mathcal{D}}$-Renaming" can only be applied a bounded number of times.

Suppose for any depth distorted clause C occurring in the theorem proving derivation, the predicate symbol of $\succ_d$-distorting literals in C already occurs in N. Due to the depth bound on any literal occurring in the theorem derivation, the number of different such $\succ_d$-distorting literals is bounded. Thus, also the number of applications of the "$\downarrow_{\mathcal{D}}$-Renaming" rule extending the signature is bounded.

If there were an unbounded number of applications of the "$\downarrow_{\mathcal{D}}$-Renaming" rule, then the $\succ_d$-distorting literals subject to the transformation $\Rightarrow_{\mathcal{D}}$ have themselves been introduced by "$\downarrow_{\mathcal{D}}$-Renaming". Thus, there is an infinite sequence $C_1, C_2, \ldots$ of depth distorted clauses occurring in the theorem proving derivation and a corresponding infinite sequence of literals $L_1, L_2, \ldots$, such that for every $i \geq 1$, L_i is a $\succ_d$-distorting literal in C_i and L_{i+1} is an instance of $\mathrm{d}_{\downarrow}(L_i)$. Note that all the L_i are non-ground and that $\mathrm{dp}(\mathrm{d}_{\downarrow}(L_i)) = \mathrm{dp}^{\mathsf{V}}_{\max}(\mathrm{d}_{\downarrow}(L_i))$. Let σ_i be a substitution such that $L_{i+1} = \mathrm{d}_{\downarrow}(L_i)\sigma_i$. We assume that the domain of σ_i is a subset of $\mathcal{V}(\mathrm{d}_{\downarrow}(L_i))$.

Since L_{i+1} is $\succ_d$-distorting, $\mathrm{dp}(L_{i+1}) > \mathrm{dp}^{\mathsf{V}}_{\max}(L_{i+1})$, which implies that σ_i instantiates at least one of the variables of $\mathrm{d}_{\downarrow}(L_i)$ by a compound term. Let t_i be a compound term with maximal variable depth in the codomain of σ_i. If t_i is non-ground, then

$$\mathrm{dp}^{\mathsf{V}}_{\max}(L_{i+1}) > \mathrm{dp}^{\mathsf{V}}_{\max}(\mathrm{d}_{\downarrow}(L_i)) = \mathrm{dp}^{\mathsf{V}}_{\max}(L_i). \tag{2.24}$$

If t_i is ground, then

$$|\mathcal{V}(L_{i+1})| < |\mathcal{V}(\mathrm{d}_{\downarrow}(L_i))| = |\mathcal{V}(L_i)|. \tag{2.25}$$

With every literal L we can associate a complexity measure $c_L = \langle \mathrm{dp}^{\mathsf{V}}_{\max}(L), |\mathcal{V}(L)| \rangle$. We define an ordering $\succ_c$ on complexity measures by the lexicographic combination of the orderings $>$ and $<$ on the natural numbers. By (2.24) and (2.25), we have $c_{L_{i+1}} \succ_c c_{L_i}$ for all $i \geq 1$. However, by Theorem 2.25, no literal occurring in a theorem proving derivation has a maximal variable depth exceeding $\max^{\mathsf{V}}_N$. (Trivially, there is also a lower bound on the number of variables in the literals L_i.) Thus, there can be no infinite ascending chain $c_{L_1} \prec_c c_{L_2} \prec_c \cdots$ and no infinite chain of literals $L_1, L_2, \ldots$ of the kind defined above.

Therefore, the number of applications of the "$\downarrow_{\mathcal{D}}$-Renaming" rule extending the signature is bounded. We eventually obtain a finite signature Σ which is stable for the remainder of the theorem proving derivation. This derivation is terminating, since by Theorem 2.25, there exist only finitely many distinct clauses (modulo variable renaming) over Σ. □

2.3 On the relationship between the decision procedures for E^+

In this section we will briefly discuss the differences between the refinement proposed in Section 2.2 and those used by de Nivelle [28] and Tammet [128]. In particular, we will consider the sizes of the saturated clause sets.

First, let us consider the use of the a posteriori $\succ_d$ refinement without the additional "$\downarrow_{\mathcal{D}}$-Renaming" rule of our decision procedure. We have already observed, that in this case the maximal depth of variables in resolvents can be strictly greater than the maximum of the maximal depths of variables in the parent clauses. Theorem 2.22 and 2.23 show that this is not the case for our decision procedure. Consequently, the saturated clause set is potentially smaller. Whether this is actually the case depends on the particular clause set under consideration. For example, on the set of clauses

(26) $\{p(f(x)), q(x, f(f(a)))\}$
(27) $\{\neg p(f(a))\}$

no inference step by the posteriori $\succ_d$-refinement is possible. However, on the transformed clause set

(28) $\{p(f(x)), q_1(x)\}$
(29) $\{\neg p(f(a))\}$
(30) $\{\neg q_1(x), q(x, f(f(a)))\}$
(31) $\{q_1(x), \neg q(x, f(f(a)))\}$

we will need two resolution inference steps to obtain a saturated clause set independently of the particular ordering chosen, since the ordering no longer prevents an inference step on the first literal of (28).

[(28)1,R,(29)1] (32) $\{q_1(a)\}$
[(30)1,R,(32)1] (33) $\{q(a, f(f(a)))\}$.

So, there is no way to tell beforehand, which decision procedure will perform best on a particular problem.

As intended by the construction of our decision procedure, there is a close relationship to the decision procedure based on the a priori $\succ_v$-refinement of resolution. However, the renaming of literals by the transformation $\Rightarrow_{\mathcal{D}}$ might prevent particular inference steps by factoring and resolution. We will now discuss this problem in more detail. Reconsider the clause C_1

(34) $\{\neg p(f(g(h(a)), x)), \neg p(f(g(x), x))\}$

which has a ground factor

[(34)1,F,(34)2] (35) $\{\neg p(f(g(h(a)), h(a)))\}$.

The literal $\neg p(f(g(h(a)), x))$ is $\succ_d$-distorting in C_1. Renaming will replace C_1 by the set:

(36) $\{p_1(v_1(x)), \neg p(f(g(x), x))^*\}$
(37) $\{\neg p_1(v_1(x))_+, \neg p(f(g(h(a)), x))^*\}$
(38) $\{p_1(v_1(x)), p(f(g(h(a)), x))^*\}$.

We have marked the $\succ'_d$-maximal literals with $\_^*$. In addition we will make use of a selection function S_v which selects the literal $\neg A$ in the clause $\mathrm{Def}^A_L = \{\neg A, L\}$. The selected literal is marked with $\_{}_+$ in clause (37). No factoring inference step is possible on clause (36). If clause (35) is part of a refutation of the clause set containing clause (34), then a refutation without (35), which is still possible, might be twice as long. In general, a refutation can be exponentially longer in the number of eliminated factoring inference steps. Since the clause set introduced by the transformation $\Rightarrow_{\mathcal{D}}$ not only contains Def^A_L, which is sufficient to preserve satisfiability equivalence, but also Def^L_A, we can simulate the factoring inference step by two additional resolution inference steps:

[(36)2,R,(38)2] (39) $\{p_1(v_1(h(a))), p_1(v_1(h(a)))\}$
[(39)1,F,(39)2] (40) $\{p_1(v_1(h(a)))\}$
[(37)1,R,(40)1] (41) $\{\neg p(f(g(h(a)), h(a)))\}$.

Similarly, resolution steps possible before the renaming transformation but which are no longer possible after the transformation can be simulated. In our first example we had the clause C_2

(42) $\{p(f(g(h(a)), h(y)))\}$

in addition to C_1 and were able to construct a refutation using two resolution inference steps.

[(34)2,R,(42)1] (43) $\{\neg p(f(g(h(a)), h(a)))\}$
[(43)1,R,(42)1] (44) $\perp$.

In the transformed clause set, after obtaining

[(36)2,R,(42)1] (45) $\{p_1(v_1(h(a)))\}$

by resolving (36) and (42), a resolution inference between (45) and (42) is not possible. However, the literal introduced for the $\succ_d$-distorting literal in (34) is now maximal in (45). We can now use clause (37) to obtain (43) by one additional resolution inference step and then complete the refutation as before.

Theorem 2.27.
The resolution decision procedure based on the ordering $\succ_d$ and the selection function S_v p-simulates the resolution decision procedure of Tammet [128] based on the non-liftable ordering $\succ_v$.

2.4 Related work

Previous work on the class $\mathbf{E}^+$ is by Tammet [128], de Nivelle [28], and Fermüller [39].

Tammet was first to claim that the a priori $\succ_v$-refinement of resolution provides a decision procedure for the class $\mathbf{E}^+$. His argument contains a gap, though. He shows that if C is the resolvent of two indecomposable, variable uniform clauses C_1 and C_2 with respect to the a priori $\succ_v$-refinement such that C contains a term deeper than the deepest term in C_1 and C_2, then for all literal $L\sigma$ in C

$$\begin{array}{ll} \mathrm{card}^{\mathsf{V}}_{\max}(L\sigma) = 0 & \text{if } \mathrm{card}^{\mathsf{V}}_{\max}(L) = 0, \text{ and} \\ \mathrm{card}^{\mathsf{V}}_{\max}(L\sigma) < \mathrm{card}^{\mathsf{V}}_{\max}(L) & \text{if } \mathrm{card}^{\mathsf{V}}_{\max}(L) > 0. \end{array}$$

He then argues that this implies the existence of a bound on the depth of terms such that all literals of this depth have a $\mathrm{card}^{\mathsf{V}}_{\max}$ equal to zero, they do not contain variables, and their depth cannot grow.

As we have already noted, there are infinitely many literals L with $\mathrm{card}^{\mathsf{V}}_{\max}(L)$ equal to zero. There is no bound on the depth of these literals and so there are infinitely many ground clauses we can construct using these literals. To give an example, reconsider the clauses

$$C_5 = \{\neg p(x, s(s(s(0)))), p(s(x), s(s(s(0))))\}$$

and

$$C_6 = \{p(0, s(s(s(0))))\}.$$

Ignoring the ordering restriction by $\succ_v$, we are able to derive an infinite sequence of clauses of the form $\{p(s^n(0), s(s(s(0))))\}$. While $\mathrm{card}^{\mathsf{V}}_{\max}(p(s^n(0), s(s(s(0))))) = 0$, for the corresponding literal in C_5 we have $\mathrm{card}^{\mathsf{V}}_{\max}(p(s(x), s(s(s(0))))) = 1$. Thus, the clauses satisfy the restriction on $\mathrm{card}^{\mathsf{V}}_{\max}$ described above without ensuring termination.

Since the a priori $\succ_v$-refinement of resolution is a decision procedure for $\mathbf{E}^+$, it is not possible to construct a counterexample to the termination proof of Tammet which obeys the ordering restriction.

De Nivelle's exposition [28] of the termination proof for the a posteriori $\succ_n$-refinement of resolution on $\mathbf{E}^+$ establishes an upper bound on the maximal depth of variables in derived clauses. A proof of the existence of an upper bound on the depth of derived clauses would proceed along the lines of Theorem 2.25.

Finally, the decision procedure of Fermüller [39] is also based on the a priori $\succ_v$-refinement. Like our procedure he uses an additional inference rule “Fill” to deal with depth distorted clauses. Instead of renaming $\succ_d$-distorting literals, he adds particular ground instances of depth distorted clauses to the set of clauses to ensure the completeness of the procedure.[4]

2.5 Conclusion

It is interesting to compare the class $\mathbf{E}^+$ with its subclass $\mathbf{E}_1$ defined as follows. A clause C belongs to $\mathbf{E}_1$ if (i) every literal in C is covering, and (ii) for each literal L_1 and L_2 in C either $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ or $\mathcal{V}(L_1) \cap \mathcal{V}(L_2) = \emptyset$ holds. In contrast to $\mathbf{E}^+$, the class $\mathbf{E}_1$ corresponds to a fragment of first-order logic. Clauses in $\mathbf{E}_1$ contain no compound ground terms and by Lemma 2.5(4) inference steps by resolution and factoring will never result in non-ground clauses containing such terms. Therefore, we have $\mathrm{dp}^{\mathsf{V}}_{\max}(L) = \mathrm{dp}(L)$ for any literal occurring in a non-ground, derived clause. From Lemmata 2.5 and 2.7 we easily infer that resolution and factoring with respect to the $\succ_d$-refinement of resolution will only derive clauses with at most the maximal depth and maximal variable depth of their parent clauses. Termination of the $\succ_d$-refinement of resolution on $\mathbf{E}_1$ follows immediately.

The class $\mathbf{E}^+$ differs from $\mathbf{E}_1$ only in that it allows for compound ground terms where $\mathbf{E}_1$ allows only constants. The adjustments to the resolution decision procedure and its termination proof necessary to accommodate this slight extension are surprisingly complicated. This observation sheds some light on the difficulty we may encounter when we turn our attention from traditional fragments of first-order logic without free function symbols to classes of formulae with free function symbols. Such classes would naturally arise from (program) verification problems and could also constitute an application area where assuring the termination of any derivation could level some scepticism with respect to the use of theorem proving techniques.

Among the techniques used in this chapter, the following two are of particular interest. First, the use of renaming as an expansion rule applied during a theorem proving derivation instead of a preprocessing step [7, 19, 104, 113]. This technique may have applications to other decidable fragments for which no resolution decision procedure exists as yet. I conjecture that dynamic renaming is necessary to decide *fluted logic* [115] by resolution. Second, the simulation of a resolution decision procedure based on a non-liftable ordering by a resolution decision procedure based on a liftable ordering. This provides some additional insight into the relationship between these two classes of resolution refinements. For further discussions see [27]. In general, simulation results provide useful insights into the relative proof and search complexity of calculi and theorem proving strategies [34, 114]. For the classes under consideration in the following chapters they will allow us to relate resolution decision procedures to tableaux decision procedures found in the literature.

[4]Since the completeness of the $\succ_v$-refinement was unknown at this time.

Chapter 3

The classes K and $\overline{\mathrm{K}}$

Since the early work of Kallick [87] decision procedures based on resolution have been in the focus of research in automated theorem proving. Two parameters of resolution must be controlled to assure a resolution-type decision procedure: the nesting of compound terms in resolvents and the size of clauses. For most of the solvable classes known in the literature, unrestricted resolution is only a semi-decision procedure, since one or both of these parameters grow unboundedly.

One possible solution to the problem of keeping the two parameters within a finite bound is the use of ordering refinements of resolution (in Section 4.5 we will give an example of the use of a selection refinement). Most of the results on decision procedures based on ordered resolution consider classes where all literals in the clausal form of the formulae under consideration share the same variables. This property is preserved by unrestricted resolution (and factoring). Then an ordering is utilised to ensure that for all clauses in a derivation the maximal depth of variable occurrences does not increase. Consequently, there exists a bound on the nesting of compound terms in the derivation and decidability follows.

The class $\overline{\mathrm{K}}$ is an example of a class where such an approach is not sufficient. The class $\overline{\mathrm{K}}$ is based on Maslov's class K [97]. More precisely, it is intended to be the dual of K, that is, for every formula φ in K the formula $\mathrm{nnf}(\neg\varphi)$ is in $\overline{\mathrm{K}}$. While Maslov is interested in validity of formulae in the class K, we will consider the dual problem, namely, the satisfiability of formulae in the class $\overline{\mathrm{K}}$. The class $\overline{\mathrm{K}}$ contains a variety of the classical decidable fragments of first-order logic such as the monadic class, the initially extended Skolem class and the Gödel class.

According to Maslov [97] the inverse method provides a means to decide the validity of disjunctions of formulae in the class K. He provides only a proof sketch. Although Kuehner noted in 1971 that there is a one-to-one correspondence between derivations in the inverse method and resolution, only in 1993 a decision procedure for a subclass of the class $\overline{\mathrm{K}}$ based on lock resolution is described by Zamov. Section 3.6 discusses his results.

In this chapter we describe a resolution-based decision procedure for $\overline{\mathrm{K}}$ as well as for the class $\overline{\mathrm{DK}}$ consisting of conjunctions of formulae in $\overline{\mathrm{K}}$, thereby extending the result of Zamov. An additional renaming transformation of certain problematic clauses allows for the embedding of the classes under consideration into a class for which standard liftable orderings ensure closure under resolution and factoring as well as termination. Sections 3.1 and 3.2 define the class $\overline{\mathrm{K}}$ and a corresponding class of clause sets, called $\overline{\mathrm{KC}}$. The basic lemmata in Section 3.2 are similar to those of Zamov [39, chap. 6]. For this reason we present the proofs in Appendix A. It should be noted, however, that our definitions of similarity and k-regularity in Section 3.2 are different to Zamov's, in particular, our notions allow for the presence of constants. Section 3.4 describes the renaming transformation and presents the termination proof. Section 3.5 extends the results to the class $\overline{\mathrm{DK}}$. Section 3.6 as well as the conclusion discuss related work and related decidable fragments of first-order logic. A short version of this chapter is [83].

3.1 The class $\overline{\mathrm{K}}$

Definition 3.1 (φ-prefix).
Let φ be a schema in negation normal form and ψ a subformula of φ. The *φ-prefix* of the formula ψ is a sequence of quantifiers of the schema φ which bind the free variables of ψ.

If a φ-prefix is of the form

$$\exists y_1 \ldots \exists y_m \forall x_1 Q_1 z_1 \ldots Q_n z_n,$$

where $m \geq 0$, $n \geq 0$, $Q_i \in \{\exists, \forall\}$ for all i, $1 \leq i \leq n$, then

$$\forall x_1 Q_1 z_1 \ldots Q_n z_n$$

is the *terminal φ-prefix.* For a φ-prefix

$$\exists y_1 \ldots \exists y_m$$

the terminal φ-prefix is the empty sequence of quantifiers.

Definition 3.2 (The class $\overline{\mathrm{K}}$).
The schema φ in negation normal form belongs to the class $\overline{\mathrm{K}}$ if there are k quantifiers $\forall x_1$, $\ldots$, $\forall x_k$, $k \geq 0$, in φ not interspersed with existential quantifiers, such that for every atomic subformula ψ of φ the terminal φ-prefix of ψ

1. either is of length less than or equal to 1, or
2. ends with an existential quantifier, or
3. is of the form $\forall x_1 \forall x_2 \ldots \forall x_k$

We say the variables $x_1, \ldots, x_k$, $k \geq 0$, are the *fixed universally quantified variables of* φ and φ is of *grade* k, indicating the number of fixed universally quantified variables.

Example 3.3:
The formula φ_1

$$\exists a_1 \exists a_2 \forall x_1 \forall x_2 \exists y_1 \forall z_1 \exists y_2 \colon p(a_1, a_2) \wedge p(a_2, y_1) \wedge (q(x_1, a_1, x_2) \vee r(x_1, y_2, z_1))$$

is an element of class $\overline{\mathrm{K}}$ of grade 2: The variables x_1 and x_2 are the fixed universally quantified variables of φ_1. Every atomic subformula ψ satisfies the restrictions on the quantifier prefix of φ_1 binding the variables in ψ. The terminal φ_1-prefix of the literal $p(a_1, a_2)$ is empty, so property (1) of Definition 3.2 is satisfied. The φ_1-prefix of the literal $p(a_2, y_1)$ is $\exists a_2 \forall y_1$, its terminal φ_1-prefix is $\forall y_1$. It is of length 1 and satisfies property (1) of Definition 3.2. The terminal φ_1-prefix of $q(x_1, a_1, x_2)$ is $\forall x_1 \forall x_2$. Due to our choice of the fixed universally quantified variables, the literal satisfies property (3). Finally, the terminal φ_1-prefix of $r(x_1, y_2, z_1)$ is $\forall x_1 \forall z_1 \exists y_2$. It ends in an existentially quantified variable. So, property (2) holds.

The formula φ_2

$$\begin{aligned}\forall x_1 \forall x_2 \exists x_3 \forall x_4 \colon q(x_4, x_4) \wedge \\ (r(x_1, x_3) \vee p(x_1, x_2)) \wedge \\ (p(x_1, x_2) \vee r(x_1, x_3)) \wedge q(x_4)\end{aligned}$$

belongs to the class $\overline{\mathrm{K}}$, because there exist universally quantified variables x_1 and x_2 such that the φ_2-prefix of $q(x_4, x_4)$, which has the form $\forall x_4$, is of length 1, the φ_2-prefix of $r(x_1, x_3)$, which has the form $\forall x_1 \exists x_3$, ends in an existential quantifier, and the φ_2-prefix of $p(x_1, x_2)$ is of the form $\forall x_1 \forall x_2$.

The formula φ_3

$$\exists y_1 \forall x_1\, p(y_1, x_1)$$

belongs to the class $\overline{\mathrm{K}}$, since the terminal φ_3-prefix of the unique atomic subformula $p(y_1, x_1)$ of φ_3 is of length 1.

The following two formulae do not belong to the class $\overline{\mathrm{K}}$. Consider the formula φ_4

$$\forall x_1 \forall x_2 \forall x_3 \colon p(x_1, x_2, x_3) \wedge q(x_1, x_2).$$

φ_4 has two atomic subformulae $p(x_1, x_2, x_3)$ and $q(x_1, x_2)$ with corresponding (terminal) φ_4-prefixes $\forall x_1 \forall x_2 \forall x_3$ and $\forall x_1 \forall x_2$. Neither of the φ_4-prefixes is of length 1 nor ends in an existential quantifier. In addition, we are not able to choose variables $y_1, \ldots, y_k$ from $\{x_1, x_2, x_3\}$ such that both φ_4-prefixes are equal to $\forall y_1 \ldots \forall y_k$, because the φ_4-prefixes are not equal to each other.

The formula φ_5

$$\forall x_1 \exists x_2 \forall x_3 \colon \neg p(x_1, x_2, x_3) \vee p(x_1, x_2, x_3)$$

does not belong to $\overline{\mathrm{K}}$, since the φ_5-prefix $\forall x_1 \exists x_2 \forall x_3$ of the two occurrences of $p(x_1, x_2, x_3)$ is not of length 1, does not end in an existential quantifier, nor does it consist of universal quantifiers only. Note that φ_5 is obviously satisfiable.

It is also important to see some important classes of formulae which do not belong to $\overline{\mathrm{K}}$. For example, formulae like

(Transitivity) $\quad \forall x_1 \forall x_2 \forall x_3 \colon \neg r(x_1, x_2) \vee \neg r(x_2, x_3) \vee r(x_1, x_3)$

are not in $\overline{\mathrm{K}}$, since no subset of $\{x_1, x_2, x_3\}$ will suffice as fixed universally quantified variables. For the same reason, connectivity formulae

(Euclideanness) $\quad \forall x_1 \forall x_2 \forall x_3 \colon \neg r(x_1, x_2) \vee \neg r(x_1, x_3) \vee r(x_2, x_3)$

(Confluence) $\quad \forall x_1 \forall x_2 \forall x_3 \exists x_4 \colon \neg r(x_1, x_2) \vee \neg r(x_1, x_3) \vee (r(x_2, x_4) \wedge r(x_3, x_4))$

are not in $\overline{\mathrm{K}}$. Important properties (of binary relations) which do belong to $\overline{\mathrm{K}}$ are the following.

(Reflexivity) $\quad \forall x_1 \colon r(x_1, x_1)$

(Irreflexivity) $\quad \forall x_1 \colon \neg r(x_1, x_1)$

(Symmetry) $\quad \forall x_1 \forall x_2 \colon \neg r(x_1, x_2) \vee r(x_2, x_1)$

(Seriality) $\quad \forall x_1 \exists x_2 \colon r(x_1, x_2)$

(Density) $\quad \forall x_1 \forall x_2 \colon \neg r(x_1, x_2) \vee \exists y_1 \colon (r(x_1, y_1) \wedge r(y_1, x_2))$

3.2 The class $\overline{\mathrm{KC}}$ and quasi-regular clauses

Since our intention is a resolution-based decision procedure for the class $\overline{\mathrm{K}}$ we are interested in the clause sets corresponding to formulae in $\overline{\mathrm{K}}$.

Definition 3.4 (The class $\overline{\mathrm{KC}}$).
Without loss of generality we can restrict ourselves to formulae in prenex form whose matrix is in conjunctive normal form, that is, schemas in $\overline{\mathrm{K}}$ have the form

$$\exists y_1 \ldots \exists y_m \forall x_1 \ldots \forall x_k Q_1 z_1 \ldots Q z_l \bigwedge_{i=1,\ldots,n} \bigvee_{j=1,\ldots,m_i} L_{i,j} \tag{3.1}$$

where $m \geq 0$, $k \geq 0$, $l \geq 0$, $n > 0$, $m_i > 0$, and $L_{i,j}$ are literals. We assume that outer Skolemisation is used in the process of transforming (3.1) to clausal form, that is, if $\forall z_1 \ldots \forall z_p$ is the subsequence of all universal quantifiers of the φ-prefix of subformula $\exists z{:}\,\varphi$ of φ, then $\varphi[z/f(z_1, \ldots, z_p)]$ is the outer Skolemisation of $\exists z{:}\,\varphi$. The class of clause sets obtained from formulae of the form (3.1) in the class $\overline{\mathrm{K}}$ is denoted by $\overline{\mathrm{KC}}$.

The remainder of this Section is devoted to the definition of a syntactic characterisation of the clauses in $\overline{\mathrm{KC}}$.

Definition 3.5 (Dominating term).
The term t *dominates* the term s, denoted by $t \gtrsim_Z s$, if at least one of the following conditions is satisfied:

1. $t = s$
2. $t = f(t_1, \ldots, t_n)$, s is a variable and $s = t_i$ for some i, $1 \leq i \leq n$.
3. $t = f(t_1, \ldots, t_n)$, $s = g(t_1, \ldots, t_m)$, $n \geq m \geq 0$.

Lemma 3.6. *The relation $\gtrsim_Z$ is a quasi-ordering on terms.*

Proof. See Lemma A.1 and Corollary A.2 in Appendix A. □

Lemma 3.7. *Let s and t be compound terms. Let σ be a substitution. If $s \gtrsim_Z t$, then $s\sigma \gtrsim_Z t\sigma$.*

Proof. See Lemma A.3 in Appendix A. □

We can extend the relation $\gtrsim_Z$ to sets of terms and literals in the following way. The set T_1 of terms *dominates* the set T_2 of terms if for every term t_2 in T_2 there exists a term t_1 in T_1 such that t_1 dominates t_2. Two terms s and t are *similar* if s dominates t and t dominates s.

Definition 3.8 (Dominating literal).
The literal L_1 *dominates* the literal L_2, denoted by $L_1 \gtrsim_Z L_2$, if the set of non-constant arguments of L_1 dominates the set of non-constant arguments of L_2.

Note that $\gtrsim_Z$ is a quasi-ordering on literals. We define $\sim_Z$ as $\gtrsim_Z \cap \gtrsim_Z^{-1}$ and $\succ_Z$ as $\gtrsim_Z \setminus \sim_Z$.

Example 3.9:
The literal $p(x, y)$ dominates $q(a, x, y)$, but not $q(f(a), x, y)$.

Definition 3.10 (Similar literals).
Two literals L_1 and L_2 are *similar* if the set of non-constant arguments of L_1 dominates the set of non-constant arguments of L_2, and vice versa.

Example 3.11:
The literals $p(x, y)$, $q(a, x, y)$, and $q(y, x)$ are similar. So are $p(f(a), x)$ and $q(g(a), x)$. Note that $p(f(a), x)$ is not similar to $q(g(b), x)$, nor does one dominate the other one.

Definition 3.12 (Regular literal).
Let F be a set of function symbols and let V be a set of variables. Based on the quasi-ordering $\succsim_Z$ we are able to characterise a subset of the set $\mathsf{T}(\mathsf{F}, \mathsf{V})$ of all terms in the following way: A term is called *regular* if it dominates all its arguments. We denote the set of all regular terms built from F and V by $\mathsf{T}_{\mathsf{reg}}(\mathsf{F}, \mathsf{V})$. We extend this notion to sets of terms and literals as follows. A set of terms is called *regular* if it contains no compound term or it contains some regular compound term which dominates all terms of this set. A literal is called *regular* if the set of its arguments is regular.

The extension of regularity to sets of literals is less straightforward. We need two more definitions: A literal L is *singular* if it contains no compound term and $\mathcal{V}(L)$ is a singleton, otherwise it is *non-singular*. A regular literal containing a compound term is *deep*, otherwise it is *shallow*.

Definition 3.13 (Regular clause).
A clause C of literals is *k-regular* if the following conditions hold:

1. C contains regular literals only.
2. k is a non-negative integer not greater than the minimal arity of the non-constant function symbols occurring in C. If C does not contain compound terms, then k is arbitrary.
3. C contains some literal which dominates every literal in the set C.
4. If L_1 and L_2 are non-singular, shallow literals in C, then L_1 and L_2 are similar.
5. If L_1 is a non-singular, shallow literal in C, then for all compound terms t occurring in any literal in C,
$$\arg_{set}(L_1) \setminus \mathsf{F}_0 \sim_Z \arg_{set}^{1\ldots k}(t) \setminus \mathsf{F}_0$$
holds.

A clause is *regular* if it is k-regular for some $k \geq 0$. A clause is again called *quasi-regular* if all of its indecomposable components are regular.

Example 3.14:
The clause C_1
$$C_1 = \{p(a, y, z), q(f(a, y, z))\}$$
is 3-regular. The set of non-constant arguments of the non-singular literal $p(a, y, z)$, that is, $\{y, z\}$, is similar to the subset of non-constant terms of the set of the first three arguments of $f(a, y, z)$, that is, $\{y, z\}$.

Note that $\{p(x_1, x_2, x_3)\}$ can be considered a 2-regular clause, although the corresponding first-order formula $\forall x_1 \forall x_2 \forall x_3 \colon p(x_1, x_2, x_3)$ is of grade 3.

3.3 Resolution and factoring on quasi-regular clauses

Next we investigate the closure of quasi-regular clauses under resolution and factoring. We show in Theorem 3.15 that every split component D of a clause corresponding to a formula φ of the form (3.1) is k-regular. Theorem 3.19 shows that the resolvent of two indecomposable k-regular clauses is again k-regular if we restrict ourselves to resolution on maximal literals with respect to $\gtrsim_Z$. Theorem 3.20 shows that the factor of an indecomposable k-regular clause is k-regular. The proofs of these theorems are based on the Lemmata 3.16 to 3.18. Then in Theorem 3.25 we show that the number of k-regular clauses over a finite signature is bounded modulo variable renaming.

Lemma 3.15. *Every split component D of a clause C in the clausal form of a formula φ of the form (3.1) is k-regular.*

Proof. We first show that C consists of regular literals only. Let L_1 be a literal in C and let L_2 be the corresponding literal in φ. We consider the following cases:

1. The φ-prefix of L_2 is empty. The set of arguments of L_1 is empty and L_1 is trivially regular.
2. The φ-prefix of L_2 is non-empty and its terminal φ-prefix is either empty or consists only of universal quantifiers. Each argument of L_1 is either a constant or a variable. Thus, L_1 is regular.
3. The terminal φ-prefix of L_2 ends with $\exists y$. Then L_1 contains a term $t_1 = f_1(x_1, \dots, x_n)$ (replacing the variable y after Skolemisation) where $x_1, \dots, x_n$ are the universally quantified variables in the terminal φ-prefix of L_2. Let t_2 be an argument of L_1. If t_2 is a constant, then t_1 trivially dominates t_2. If t_2 is a variable, then it is among the universally quantified variables of L_2. So, t_2 is a variable argument of t_1 and therefore dominated by t_1. Suppose t_2 is a compound term $f_2(y_1, \dots, y_m)$. We have $\mathcal{V}(t_2) \subseteq \mathcal{V}(t_1)$ and therefore $m \leq n$. Furthermore, we can assume that the order of variables in the Skolem terms t_1 and t_2 is determined by the order of variables in the φ-prefix of L_2. So, $x_i = y_i$ for every i, $1 \leq i \leq m$, and t_1 dominates t_2. Thus, t_1 dominates all arguments of L_1.

Now we show that D is k-regular. Some simple cases are:

1. D is a singleton set $\{L\}$. Conditions (1) and (3) of Definition 3.13 are satisfied, since D contains only one regular literal L. If L contains a compound term, then Conditions (4) and (5) are fulfilled, since D contains no shallow, non-singular literals. Otherwise, Condition (4) is fulfilled, since any literal is similar to itself; and Condition (5) is vacuous.
2. D contains only ground literals. Then D is a singleton set, because ground literals are variable disjoint to any other literal. Thus, D is regular by the previous case.

It is important to note that in all further cases, D does not contain ground literals. In all remaining cases we assume that D contains at least two literals.

1. Let L_1 be a deep literal in D such that its dominating term t_1 has maximal arity of all terms occurring in D. Let L_2 be an arbitrary literal in D.

   Suppose L_2 is deep. Since L_2 is regular it contains a dominating compound term t_2. The arity n_1 of t_1 is greater than or equal to the arity n_2 of t_2. By the assumptions we have made about Skolemisation, $t_1 = f_1(x_1, \ldots, x_{n_1})$ and $t_2 = f_2(x_1, \ldots, x_{n_2})$. So, t_1 dominates t_2. Since L_2 is regular, t_2 dominates every argument of L_2. Due to the transitivity of $\succsim_Z$, t_1 dominates every argument of L_2. It follows that L_1 dominates L_2.

If the literal L_2 does not contain compound terms and L_2 is not singular, then the terminal φ-prefix of L_2 is $\forall x_1 \ldots \forall x_k$. The set of its non-constant arguments is equal to the set of the first k arguments of the term t_1. All constant arguments of L_2 are trivially dominated by t_1. Therefore L_1 dominates L_2.

Furthermore, if L_3 is a non-singular, shallow literal in D, then its terminal prefix is $\forall x_1 \ldots \forall x_k$. Thus the sets of non-constant arguments of L_2 and L_3 are equal.

Finally, if L_2 is a singular literal and does not contain function symbols then its only variable argument is an argument of t_1, since D is indecomposable. Therefore, L_1 dominates L_2.

2. Suppose D does not contain literals with compound terms. Then every literal occurring in D is obtained from some literal with terminal φ-prefix $\forall x_1 \ldots \forall x_k$ or $\forall z$. Let L_1 and L_2 be literals in D where L_1 is obtained from a literal with terminal φ-prefix $\forall x_1 \ldots \forall x_k$.

   If L_2 is obtained from a literal with the same φ-prefix, then the sets of non-constant arguments of L_1 and L_2 are equal. Thus L_1 and L_2 are similar.

   If L_2 is obtained from a literal with terminal φ-prefix $\forall z$, then the only variable of L_2 is among the arguments of L_1, since D is indecomposable.

   So, in both cases L_1 dominates L_2.

3. All literals in D are obtained from literals with terminal φ-prefix of length 1. Since D is indecomposable, all literals in D contain the same variable. It follows that all literals in D are similar.

So we have shown that in any case D contains some literal which dominates all literals from D, that is, D is a regular component. Furthermore, the minimal arity of each compound term (if there is any) is k. We have also shown that Conditions (3)–(5) of Definition 3.13 are satisfied in all the cases we have to consider. Thus D is k-regular. $\square$

Lemma 3.15 no longer holds if we make use of strong Skolemisation or techniques reducing the scope of quantifiers. Consider the following examples from [103]. Outer Skolemisation of the formula $\forall x, y\, \exists z\colon p(x, z) \vee p(x, y)$ results in $\forall x, y\colon p(x, f(x, y)) \vee p(x, y)$ and the clausal form is 2-regular. If we reduce the scope of the $\forall y$ quantifier to $\forall x\, (\exists z\colon p(x, z)) \vee (\forall y\colon p(x, y))$ before Skolemisation, then the clausal form of $\forall x, y\colon p(x, f(x)) \vee p(x, y)$ is not regular. Similarly, outer Skolemisation of $\forall x, y\colon p(x, y) \vee (\exists z\colon q(y, z) \wedge r(x, z))$ yields the clauses $\{p(x, y), q(y, f(x, y))\}$ and $\{p(x, y), r(x, f(x, y))\}$ which are both 2-regular. In contrast, strong Skolemisation yields the clauses $\{p(x, y), q(y, f(z, y))\}$ and $\{p(x, y), r(x, f(x, y))\}$ or the clauses $\{p(x, y), q(y, f(x, y))\}$ and $\{p(x, y), r(x, f(x, z))\}$. In both cases one of the clauses is not regular.

Lemma 3.16 (Properties of regular terms).

1. *Let t be a compound regular term $f(t_1, \ldots, t_n)$. Then all variables occurring in t are arguments of t. Furthermore, if t_i is a compound term, then all variables occurring in t_i occur in $\{t_1 \ldots, t_{i-1}\}$.*
2. *If t is a regular term and t dominates a term s, then s is regular too.*

3. *If a regular term t dominates the term s and σ is a substitution such that $t\sigma$ is regular, then $t\sigma$ dominates $s\sigma$.*
4. *If t is a regular term and σ is a substitution such that the codomain of σ contains only constants and variables, then $t\sigma$ is a regular term.*

Proof. See [39, pages 136–137] or Appendix A. □

Lemma 3.17 (Properties of regular literals).

1. *Let $L_1 = (\neg)p(s_1, \dots, s_n)$ and $L_2 = (\neg)p(t_1, \dots, t_n)$ be unifiable deep literals. If s_i is a dominating term of L_1, then also t_i must be a dominating term of L_2.*
2. *Assume that L_1 and L_2 are regular literals and σ is a most general unifier of L_1 and L_2. Then $L_1\sigma$ is regular.*
3. *Let C be a regular clause, L a dominating literal of C and t a dominating term of L. Then t dominates each argument of each literal in C.*

Proof. See [39, pages 140–144] or Appendix A. □

Lemma 3.18. *Let $\{L_1\} \cup D$ be an indecomposable, k-regular clause such that L_1 dominates each literal in D and σ is a substitution such that $L_1\sigma$ is regular. Suppose that k is not greater than the minimal arity of function symbols occurring in $L_1\sigma$. Then $L_1\sigma$ dominates each literal in $D\sigma$ and the clause $(\{L_1\} \cup D)\sigma$ is k-regular.*

Proof. Let L_2 be some arbitrary literal in D. First, we will show that $L_1\sigma$ dominates $L_2\sigma$ and $L_2\sigma$ is regular. To this end we consider the following cases:

1. L_1 is a deep literal. Since L_1 dominates L_2, the dominating term t_1 of the literal L_1 dominates each argument of the literal L_2 by Lemma 3.17(3). Since the literal $L_1\sigma$ is regular, the term $t_1\sigma$ is regular and dominates each argument of the literal $L_2\sigma$ by Lemma 3.16(3). It follows that $L_1\sigma$ dominates $L_2\sigma$.

   It remains to prove that $L_2\sigma$ is a regular literal.

   (a) L_2 is a deep literal. The term t_1 dominates the dominating term t_2 of literal L_2 by Lemma 3.17(3), therefore $t_1\sigma$ dominates $t_2\sigma$ by Lemma 3.16(3). By Lemma 3.16(2), $t_2\sigma$ is regular since $t_1\sigma$ is regular. Furthermore, the term $t_2\sigma$ dominates each argument of literal $L_2\sigma$ by Lemma 3.16(3). It follows that $L_2\sigma$ is regular.

   (b) L_2 is a shallow literal and $L_2\sigma$ is deep. Then there exists a function symbol g of arity m, terms $s_1, \dots, s_m$, and a variable x in L_2 such that $x\sigma = g(s_1, \dots, s_m)$ and g has maximal arity among all the function symbols in the codomain of $\sigma_{\restriction \mathrm{FV}(L_2)}$. Note that g has to be the function symbol of some term in the codomain of $\sigma_{\restriction \mathrm{FV}(L_2)}$, because the arity of a subterm of a regular term is smaller than the arity of the term itself. We show that $x\sigma$ dominates every argument of $L_2\sigma$. Let t be some argument of $L_2\sigma$. Then t is either a term of the form $h(u_1, \dots, u_l)$ for $l > 0$, a constant c, or a variable z. In the first case, t is in the codomain of $\sigma_{\restriction \mathrm{FV}(L_2)}$. Thus, $l \leq m$ holds. Since $t_1\sigma$ dominates $g(s_1, \dots, s_m)$ and $h(u_1, \dots, u_l)$, we have $s_1 = u_1, \dots, s_l = u_l$. So, $g(s_1, \dots, s_m)$ dominates t. In the second case, $x\sigma$ trivially dominates the constant argument c. In the third case, we have to show that the variable z is an argument of $g(s_1, \dots, s_m)$. Since, $g(s_1, \dots, s_m)$ and z are arguments of $L_2\sigma$, L_2 has at least two arguments. Let y be the variable argument of L_2 such that $y\sigma = z$ (y may be identical to z). We know that

$$\arg_{set}(L_2) \setminus \mathsf{F}_0 \sim_Z \arg_{set}^{1\dots k}(t_1) \setminus \mathsf{F}_0$$

holds. Thus, the variable y, which belongs to $\arg_{set}(L_2) \setminus \mathsf{F}_0$, is one of the first k arguments of t_1. We conclude that z has to be one of the first k arguments of $t_1\sigma$. Because of the assumption that the minimal arity of a function symbol in $L_1\sigma$ is not smaller than k, we know that m is greater than or equal to k. Since $t_1\sigma$ dominates $g(s_1, \dots, s_m)$, the first k arguments of these terms are identical. Thus, z occurs among the first k arguments of $g(s_1, \dots, s_m)$.

(c) $L_2\sigma$ is shallow. Since the set of argument terms of $L_2\sigma$ does not contain a compound term, $L_2\sigma$ is trivially regular.

So we have shown that $L_1\sigma$ dominates $L_2\sigma$ and both literals are regular.

2. Both L_1 and L_2 are non-singular, shallow literals. Since $\{L_1\} \cup D$ is regular, L_1 and L_2 are similar and their sets of arguments differ in constants only. Since $L_1\sigma$ is regular, $L_2\sigma$ is regular too. The literals $L_1\sigma$ and $L_2\sigma$ are similar.

3. L_1 is a non-singular, shallow literal, and L_2 is a singular literal. In this case the variable x, occurring in L_2, occurs in L_1 too, since $\{L_1\} \cup D$ is indecomposable. The literal $L_1\sigma$ is regular, hence $x\sigma$ is regular too. It follows that $L_2\sigma$ is regular. It is also obvious that $L_1\sigma$ dominates $L_2\sigma$.

4. Both L_1 and L_2 are singular literals. Similar to the previous case.

Next, we have to show that any two non-singular, shallow literals $L_2\sigma$ and $L_3\sigma$ in $(\{L_1\} \cup D)\sigma$ are similar. Since $L_2\sigma$ and $L_3\sigma$ are non-singular, shallow literals, L_2 and L_3 are non-singular, shallow literals as well. Since $\{L_1\} \cup D$ is k-regular, L_2 and L_3 are similar, that is, L_2 and L_3 have the same set of non-constant arguments. After applying the substitution σ, the set of non-constant arguments will still be the same for $L_2\sigma$ and $L_3\sigma$. Thus, they are similar.

Finally, we show that the set of non-constant arguments of a non-singular, shallow literal $L_2\sigma$ is similar to the subset of non-constant terms of the set of the first k arguments of any compound term occurring in any literal in $(\{L_1\} \cup D)\sigma$. If $(\{L_1\} \cup D)\sigma$ contains no compound term, then the property trivially holds. Otherwise, $L_1\sigma$ contains a compound term $t_1\sigma$ such that $t_1\sigma$ dominates every compound term $t_2\sigma$ in $(\{L_1\} \cup D)\sigma$. Since $L_2\sigma$ is a non-singular, shallow literal, L_2 is a non-singular, shallow literal as well and $\sigma_{|\mathrm{FV}(L_2)} = \{x_1/w_1, \dots, x_n/w_n\}$ where w_i is either a constant or a variable for all i, $1 \le i \le n$.

Suppose t_1 is compound term. Since $\{L_1\} \cup D$ is regular, all the variables $x_1, \dots, x_n$ occur in the set of the first k arguments of t_1. This implies that $w_1, \dots, w_n$ also occur in the first k arguments of $t_1\sigma$. Since $t_1\sigma$ dominates $t_2\sigma$ and the arity of $t_2\sigma$ is greater or equal to k, $w_1, \dots, w_n$ occur in the set of the first k arguments of $t_2\sigma$ as well.

Suppose t_1 is not a compound term. Then L_1 contains no compound term at all. Assume the opposite, that is, there exists a compound term t_2 in L_1. Since L_1 is regular, t_1 is an argument of t_2. Therefore, $t_1\sigma$ is an argument of $t_2\sigma$. Thus $t_1\sigma$ does not dominate $t_2\sigma$. This contradicts our assumption that t_1 is a dominating term for $(\{L_1\} \cup D)\sigma$. Nevertheless, L_1 dominates L_2. It follows that L_1 is a non-singular, shallow literal. Since $L_1\sigma$ dominates $(\{L_1\} \cup D)\sigma$, there are no compound terms in $(\{L_1\} \cup D)\sigma$. □

Lemma 3.19. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, k-regular clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let A_1 and $\neg A_2$ be dominating literals in C_1 and C_2, respectively. Then every split component of $(D_1 \cup D_2)\sigma$ is a k-regular clause.*

Proof. According to Lemma 3.17(2) the literals $A_1\sigma$ and $\neg A_2\sigma$ are regular. Since the clause $(C_1 \cup C_2)\sigma$ does not contain any function symbol which does not occur in $C_1 \cup C_2$, k is not greater than the minimal arity of function symbols occurring in $(C_1 \cup C_2)\sigma$. By Lemma 3.18, $C_1\sigma$ and $C_2\sigma$ are quasi-regular and every split component of these clauses is k-regular.

Let D be a split component of $(D_1 \cup D_2)\sigma$. To prove that D is k-regular, we distinguish the following cases:

1. D contains a deep literal. Let L_1 be a deep literal in D such that L_1 contains a compound term $t_1 = f(u_1, \ldots, u_m)$ with maximal arity among all compound terms in D. Let L_2 be an arbitrary literal in D. We show that L_1 dominates L_2. Let t_2 be an argument of L_2. Suppose t_2 is a compound term $g(v_1, \ldots, v_n)$. $A_1\sigma$ contains a compound term $t = h(w_1, \ldots, w_l)$ such that t dominates t_1 and t_2, that is, $u_1 = w_1 = v_1, \ldots, u_n = w_n = v_n$ and $n \leq m \leq l$ hold. So, t_1 dominates t_2. If t_2 is a constant, then t_1 trivially dominates t_2. If t_2 is a variable, then t_2 is an argument of t. If t_2 is one of the first m arguments of t, then t_2 is also an argument of t_1 and t_1 dominates t_2. It remains to show that t_2 is one of the first m arguments of t. Suppose not. If L_2 is a deep literal, then it contains some compound term t_3 dominating t_2, that is, t_2 is an argument of t_3. But t_3 is also dominated by t_1 as shown above. Therefore, t_2 has to be an argument of t_1 which contradicts the assumption. Let L_2 be a shallow, non-singular literal. L_2 either occurs in $D_1\sigma$ or $D_2\sigma$. Since $C_1\sigma$ and $C_2\sigma$ are k-regular clauses, the set of non-constant arguments of L_2 is similar to the subset of non-constant arguments of the set of the first $k \leq m$ arguments of any compound term in these clause sets. Thus, t_2 occurs in the set of the first k arguments of t. Finally, let L_2 be a singular literal. Then t_2 must occur in some literal L_3 which is not singular, since otherwise D would be decomposable. But we have just shown that in this case, L_3 will be dominated by L_1. Thus, L_1 will dominate L_2 as well.

2. D contains no deep literal, but a non-singular literal. Let L_1 be an arbitrary non-singular literal. Without loss of generality we can assume that L_1 belongs to D_1. Let L_2 be some non-singular literal in D. We need to show that L_1 and L_2 are similar. If L_2 belongs to $D_1\sigma$, then L_1 and L_2 are similar, because $C_1\sigma$ is k-regular. Suppose L_2 belongs to $D_2\sigma$. Since $A_1\sigma$ and $A_2\sigma$ are equal, $A_1\sigma$ dominates L_2. If A_1 is a deep literal, then the subset of non-constant arguments of the set of the first k arguments of a dominating compound term in A_1 is similar to the set of non-constant arguments of L_2. The set of non-constant arguments of L_2 is similar to the set of non-constant arguments of L_1. If A_1 is itself a non-singular, shallow literal, then its set of non-constant arguments is similar to the set of non-constant arguments of L_1 and L_2. Again, the set of non-constant arguments of L_1 and L_2 have to be similar.

   Let L_2 be a singular literal. The non-constant argument of L_2 is an argument of L_1, because D is indecomposable. Thus, L_1 dominates L_2.

3. D contains only singular literals. Therefore D is trivially k-regular. □

Lemma 3.20. *Let $C = \{L_1, L_2\} \cup D$ be a k-regular clause such that L_1 and L_2 are unifiable with most general unifier σ. Then every split component of $(\{L_1\} \cup D)\sigma$ is a k-regular clause.*

Proof. Suppose L_1 (or L_2) is a dominating literal in C. By Lemma 3.17(2), $L_1\sigma$ is regular. Furthermore, k is not greater than the minimal arity of function symbols in $L_1\sigma$. By Lemma 3.18 $C\sigma$ is k-regular. So, $(\{L_1\} \cup D)\sigma$ is k-regular.

Suppose neither L_1 nor L_2 is a dominating literal in C. Since σ does not introduce new function symbols, the minimal arity of function symbols remains unchanged. Let L_3 be a dominating literal in C. We distinguish the following cases:

1. L_1 and L_2 are deep literals. By Lemma 3.17(1), if $t_1 = f(u_1, \ldots, u_n)$ is the dominating term of L_1, then $t_2 = f(v_1, \ldots, v_n)$ is the dominating term of L_2. Let $t_3 = g(s_1, \ldots, s_m)$ be the dominating term of L_3. Since t_3 dominates t_1 and t_2, we have $m \geq n$ and for all i, $1 \leq i \leq n$, $u_i = s_i = v_i$. Thus, t_1 and t_2 are identical, σ is the identity substitution and $(\{L_1\} \cup D)\sigma$ is obviously k-regular.

2. L_1 is a deep literal and L_2 is a shallow literal. Then L_1 contains a compound term t_1 such that every variable argument of L_2 is a strict subterm of t_1. Thus, L_1 and L_2 are not unifiable.

3. L_1 and L_2 are shallow literals. The codomain of σ contains only variables and constants. Let t_3 be the dominating term of L_3. By Lemma 3.16(4) $t_3\sigma$ is regular. Let s_3 be an arbitrary argument of L_3. By Lemma 3.16(3) and Lemma 3.16(2), $s_3\sigma$ is regular and $t_3\sigma$ dominates $s_3\sigma$. Thus, $L_3\sigma$ is regular. By Lemma 3.18 $L_3\sigma$ dominates each literal in $(\{L_1\} \cup D)\sigma$ and $(\{L_1\} \cup D)\sigma$ is k-regular. □

Thus, the split components of the conclusion of an arbitrary factoring inference step on a k-regular clause are k-regular clauses.

Lemma 3.19 and Lemma 3.20 are already sufficient to obtain an upper bound on the number of variables in clauses derivable from a set N of indecomposable, k-regular clauses. Let ar_{pred} and ar_{fun} be the maximal arity of predicate symbols and function symbols in N, respectively. Then no clause in N contains more than $\max(ar_{pred}, ar_{fun})$ variables, neither does any clause derivable by ordered resolution on dominating literals and factoring. However, it is possible to prove the following stronger result.

Lemma 3.21. *Let C be a k-regular clause and let D be a factor of C. Then $|\mathcal{V}(D)| \leq |\mathcal{V}(C)|$.*

Proof. Through factoring the number of different variables in the resulting clauses cannot increase, and hence the number of variables in D does not exceed the number of variables in C. □

Lemma 3.22. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, k-regular clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let A_1 and $\neg A_2$ be dominating literals in $\{A_1\} \cup C_1$ and $\{\neg A_2\} \cup C_2$, respectively. Then $|\mathcal{V}(D_1 \cup D_2)\sigma| \leq \max(|\mathcal{V}(C_1)|, |\mathcal{V}(C_2)|)$.*

Proof. By Lemma 3.17(2) the literals $A_1\sigma$ and $\neg A_2\sigma$ are regular. Since the clause $(C_1 \cup C_2)\sigma$ does not contain any function symbol which does not occur in $C_1 \cup C_2$, k is not greater than the minimal arity of function symbols occurring in $(C_1 \cup C_2)\sigma$. By Lemma 3.18 $C_1\sigma$ and $C_2\sigma$ are k-regular, and $A_1\sigma$ and $\neg A_2\sigma$ are dominating literals in $C_1\sigma$ and $C_2\sigma$, respectively. Since $A_1\sigma = A_2\sigma$, $A_1\sigma$ is a dominating literal in both $C_1\sigma$ and $C_2\sigma$.

We distinguish the following cases:

1. $A_1\sigma$ is a singular literal with variable x_1. Then all literals in $D_1\sigma$ and $D_2\sigma$ are singular, with variable x_1. Since neither A_1 nor $\neg A_2$ can be ground, also $\{A_1\} \cup D_1$ and $\{\neg A_2\} \cup D_2$ contain at least one variable. Thus, the lemma holds.

2. $A_1\sigma$ is a shallow, non-singular literal. Then A_1 and A_2 are also shallow, non-singular literals. The substitution σ instantiates variables by variables or constants. So, $|\mathcal{V}(A_1\sigma)| \leq |\mathcal{V}(A_1)|$ and also $|\mathcal{V}(A_1\sigma)| = |\mathcal{V}(\neg A_2)\sigma| \leq |\mathcal{V}(A_1)|$. We obtain
$$\begin{aligned}|\mathcal{V}(A_1\sigma)| &\leq \min(|\mathcal{V}(A_1)|, |\mathcal{V}(\neg A_2)|)\\ &\leq \max(|\mathcal{V}(A_1)|, |\mathcal{V}(\neg A_2)|).\end{aligned}$$
Since A_1 and $\neg A_2$ contain all the variables of C_1 and C_2, respectively, and $A_1\sigma$ contains all the variables of $C_1\sigma$ and $C_2\sigma$ the desired result follows.

3. $A_1\sigma$ is a deep literal. Then either A_1 or $\neg A_2$ contains a compound term. Without loss of generality we assume A_1 contains a dominating compound term $t = f(t_1, \ldots, t_n)$. Let $\mathcal{V}_1 = \{t_1, \ldots, t_n\} \cap \mathsf{V}$. By Lemma 3.16(1) if t_i is a compound term, then all variables occurring in t_i occur in $\{t_1, \ldots, t_{i-1}\}$. So, $\mathcal{V}(t) \subseteq \mathcal{V}_1$. Since t is a dominating term in A_1 and therefore a dominating term in C_1, all variables of C_1 occur in $\mathcal{V}_1$. Recall that $A_1\sigma$ is a dominating literal in $C_1\sigma$ and $C_2\sigma$. It follows from Lemma 3.17(3) that $t\sigma$ is a dominating term in $C_1\sigma$ and $C_2\sigma$. Let $\mathcal{V}_2 = \{t_1\sigma, \ldots, t_n\sigma\} \cap \mathsf{V}$. Again, by Lemma 3.16(1) if $t_i\sigma$ is a compound term, then all variables occurring in $t_i\sigma$ occur in $\mathcal{V}_2$, that is, $\mathcal{V}(t\sigma) \subseteq \mathcal{V}_2$. Furthermore, all variables in $(C_1 \cup C_2)\sigma$ occur in $t\sigma$. Since instantiation of t by σ will not turn a compound term argument into a variable, we have $|\mathcal{V}_2| \leq |\mathcal{V}_1|$. It follows that
$$\begin{aligned}|\mathcal{V}(D_1\sigma \cup D_2\sigma)| &\leq |\mathcal{V}(A_1\sigma)|\\ &\leq |\mathcal{V}(A_1)|\\ &\leq \max(|\mathcal{V}(C_1), \mathcal{V}(C_2)|).\end{aligned}$$
□

Corollary 3.23. *Let N be a set of indecomposable, k-regular clauses. Let n_{var} be the maximum number of distinct variables in any clause in N. Then for any clause C derivable by resolution on $\succsim_Z$-maximal literals or factoring, the number of variables in C is less than or equal to n_{var}.*

The next lemma forms the basis for approximating the maximal number of k-regular clauses over a given finite signature.

Lemma 3.24. *Let F be a finite set of function symbols and V be a set of variables. Let T be the set of words over $\mathsf{F} \cup \mathsf{V}$ defined by:*
$$T := \{w \in (\mathsf{F} \cup \mathsf{V})^* \mid |w| \leq (ar_{fun} + 1)\}$$
where ar_{fun} is the maximal arity of function symbols in F. Then there is a subset T_r of T such that there is an isomorphism i between T_r and the set of regular terms $\mathsf{T}_{\mathsf{reg}}(\mathsf{F}, \mathsf{V})$.

Proof. Let C be the set of constants in F. A straightforward morphism will not do due to the length restriction on words in T. For example, if f and g are function symbols of arity ar_{fun} and $ar_{fun}-1$, then mapping the term $f(x_1, \ldots, x_{ar_{fun}-1}, g(x_1, \ldots, x_{ar_{fun}-1}))$ to $fx_1 \ldots x_{ar_{fun}-1} gx_1 \ldots x_{ar_{fun}-1}$ results in a word of length $2ar_{fun}$.
Define i as follows.

$$i(t) = \begin{cases} t, & \text{if } t \in \mathsf{V} \cup \mathsf{C}, \\ f \cdot it(t_1) \cdot \cdots \cdot it(t_n), & \text{if } t = f(t_1, \ldots, t_n),\ n \geq 1, \end{cases}$$

where $\_ \cdot \_$ denotes the concatenation of two strings and

$$it(t) = \begin{cases} t, & \text{if } t \in \mathsf{V} \cup \mathsf{C}, \\ f, & \text{if } t = f(t_1, \ldots, t_n),\ n \geq 1. \end{cases}$$

Note that $it(f(t_1, \ldots, t_n))$ "forgets" the argument terms $t_1, \ldots, t_n$. So, i is certainly not a isomorphism for arbitrary terms. For example, it maps the terms $f(g(x, a))$ and $f(g(a, x))$ to the same word fg. We have to prove that i is injective on regular terms. Suppose it is not. Then there are terms s and t such that $i(s) = i(t)$ although $s \neq t$. By case analysis, we can show that s and t have the following form:

$$\begin{aligned} t &= f(t_1, \ldots, t_l, g(u_1, \ldots, u_k), t_{l+2}, \ldots, t_n) \\ s &= f(t_1, \ldots, t_l, g(v_1, \ldots, v_k), t'_{l+2}, \ldots, t'_n) \end{aligned}$$

where for some i, $1 \leq i \leq k$, $u_i \neq v_i$. But t and s are regular, that is, t dominates $g(u_1, \ldots, u_k)$ which implies that $u_1 = t_1, \ldots, u_k = t_k$, and $k \leq l$. In the same way, s dominates $g(v_1, \ldots, v_k)$ and thus $v_1 = t_1, \ldots, v_k = t_k$. Therefore, $u_j = v_j$ for every j, $1 \leq j \leq k$ which contradicts our assumption.

Furthermore, i is surjective on

$$T_r := i(\mathsf{T}_{\mathsf{reg}}(\mathsf{F}, \mathsf{V})). \tag{3.2}$$

□

Theorem 3.25.
Let N be a set of k-regular clauses. Let ar_{fun} be the maximal arity of function symbols in N, let n_{var} be the maximal number of variables in clauses in N, and let n_{fun} be the number of function symbols in N, respectively. The number of different terms in N cannot exceed

$$n_{terms} = (n_{fun} + n_{var})^{ar_{fun}+1}.$$

Furthermore, the number of clauses in N modulo variable renaming cannot exceed

$$n_{clauses} = 2^{(2 \times n_{pred} \times {n_{terms}}^{ar_{pred}})}$$

where n_{pred} be the number of predicate symbols in N and ar_{pred} the maximal arity of a predicate symbol.

Proof. It is easy to see that the number of words in the set T_r defined by (3.2) using n_{fun} function symbols of maximal arity ar_{fun} and at most n_{var} variables is less than or equal to n_{terms}.

There are at most $n_{atoms} = n_{pred} \times {n_{terms}}^{ar_{pred}}$ different regular atoms and $2 \times n_{atoms}$ different literals. Finally, we have to estimate an upper bound for the number of regular clauses. A clause is just a subset of the set of all regular literals. Thus the maximal number of clauses is $2^{(2 \times n_{atoms})}$. The number of non-tautological clauses is $3^{n_{atoms}}$, since every atom either does not occur in a clause, or it occurs positively, or negatively. □

Note that these upper bounds on the number of terms and clauses are not tight. For example, if we have only a unary function symbol f and one variable x, then the number of terms will not exceed $(1+1)^2 = 4$ according to the previous theorem. Actually, there are only two terms, namely x and $f(x)$.

3.4 A decision procedure for $\overline{\mathrm{KC}}$

The acyclical relation constructed from the 'dominates'-relation is not stable under substitutions, that is, we cannot construct an admissible ordering on literals based on $\succ_Z$. The first problem with $\succ_Z$ occurs on the term-level. Consider the terms $f(x,y)$ and y. Obviously, $f(x,y) \succ_Z y$ holds. However, for the substitution σ replacing y by $g(z)$, $f(x,y)\sigma \not\succ_Z y\sigma$ holds. Note that $f(x,y)\sigma$ is no longer regular. According to Lemma 3.16(3) $t \succ_Z s$ implies $t\sigma \succ_Z s\sigma$ for any substitution σ such that t and $t\sigma$ are regular. This problem is mainly caused by the dual use of $\succ_Z$: On the one hand it is used to define the structure of regular terms, literals and clauses and on the other hand it is used to determine the literals of a clause to resolve upon. The problem can be solved by using an ordering which is compatible with $\succ_Z$ on regular terms and is still stable under substitutions. The second problem with $\succ_Z$ occurs on the atom-level. As a simple example consider the atoms $p(x,y,x)$ and $p(x,x,a)$. We have $p(x,y,x) \succ_Z p(x,x,a)$. But since the atoms have a common instance $p(a,a,a)$, there exists no ordering $\succ$ stable under substitutions such that $p(x,y,x) \succ p(x,x,a)$ holds.

It is important to remember that we have to restrict resolution inference steps in a clause $\{p(x,y,x), p(x,x,a)\}$ to the first literal. For otherwise, we can no longer guarantee that resolvents of k-regular clauses are still k-regular. For example, resolution with $\{p(z,x,z), \neg p(x,x,a)\}$ (on the second literal in each clause) results in $\{p(z,x,z), p(x,y,x)\}$ which is not k-regular.

As far as the selection of suitable literals to resolve upon is concerned, clauses meeting the following two conditions cause problems.

1. The clause C contains a singular literal which has constant arguments or duplicate variable arguments.

   Suppose the opposite. Then all singular literals are monadic. All predicate symbols of shallow, non-singular literals have arity greater than one. Thus, there are no common instances of singular literals and non-singular literals. It is straightforward to define an ordering $\succ$ stable under substitutions such that the singular literals are not $\succ$-maximal in the clause C.

2. The clause C contains a shallow, non-singular literal or there is a compound term t in C such that

   $$|\mathcal{V}(\arg_{set}^{1\ldots k}(t))| \geq 2.$$

   If C contains no shallow, non-singular literals and for all compound terms t in C we have $|\mathcal{V}(\arg_{set}^{1\ldots k}(t))| \leq 1$, then all shallow literals in C contain exactly one variable.

At first glance this condition seems to be too general. In a 2-regular clause like

$$\{q(f(x,y,z)), p(x,a)\}$$

the literals $q(f(x,y,z))$ and $p(x,a)$ have no common instances and it is straightforward to define a liftable ordering $\succ$ such that $q(f(x,y,z))$ is strictly $\succ$-maximal. However, a resolution inference step with

$$\{\neg q(f(x,y,z)), p(x,y)\}$$

results in

$$\{p(x,a), p(x,y)\}$$

which is the prototypical example of a problematic clause.

This analysis motivates the following definition.

Definition 3.26.
A literal L is *CDV* (containing constants or duplicate variables) if L is singular, and there is an argument which is a constant or there are duplicate (variable) arguments. Otherwise a literal is *CDV-free*.

A clause C is *CDV* if it contains a CDV literal and a shallow, non-singular literal, but no deep literal. Otherwise C is *CDV-free*. The intuition of CDV-free clauses is that $\succ$-maximal literals are also $\succ_Z$-maximal for suitable admissible orderings $\succ$ on literals. Note that there are CDV-free clauses which contain CDV literals. Any CDV clause contains at least two literals.

An indecomposable, k-regular clause C is *strongly CDV-free* if it satisfies at least one of the following conditions.

1. C contains no CDV literal, or
2. C contains no shallow, non-singular literal and for all compound terms t occurring in any literal in C,
$$|\mathcal{V}(\arg_{set}^{1\ldots k}(t))| = 1.$$

Note that the second condition is satisfied if C contains no non-singular literals.

A set of clauses N is *CDV-free* if every clause in N is CDV-free.

Example 3.27:
The clause

$$\{p(f(x,y)), q(x,y), r(x,a)\}$$

is CDV-free, but not strongly CDV-free. It contains the literal $r(x,a)$ which is CDV and a shallow, non-singular literal $q(x,y)$. Also the 2-regular clause

$$\{p(f(x,y)), r(x,a)\}$$

is not strongly CDV-free: Apart from the CDV literal $r(x,a)$ it contains a deep, non-singular literal $p(f(x,y))$ such that the term $f(x,y)$ contains more than one variable. There is a subtle point to note. If we consider $\{p(f(x,y), r(x,a))\}$ as a 1-regular clause, then it is strongly CDV-free.

The clause

$$\{p(x,x,a), q(x,b), p(c,x,c)\}$$

is strongly CDV-free, since it contains no non-singular literal. The clause

$$\{p(x,y,c), q(x,y), p(x,a,y)\}$$

is strongly CDV-free, since it contains no singular literal.

Note that CDV-freeness of a clause is not preserved under resolution. A simple example is a resolution inference step between $C_1 = \{p(f(x,y)), r(x,y), q(x,a)\}$ and $C_2 = \{\neg p(z)\}$ with conclusion $\{r(x,y), q(x,a)\}$. This is due to the fact that the CDV literal in C_1 is *shielded* by the term $f(x,y)$. Since this term is no longer present in the conclusion, the CDV literal becomes unshielded.

However, C_1 is not strongly CDV-free. We will now show that ordered factoring and ordered resolution preserve strong CDV-freeness.

Lemma 3.28. *Let $C = \{L_1, L_2\} \cup D$ be an indecomposable, strongly CDV-free, k-regular clause such that L_1 and L_2 are unifiable with most general unifier σ and L_1 is a dominating literal in C. Then $(\{L_1\} \cup D)\sigma$ is strongly CDV-free.*

Proof. For every literal $L\sigma$ in $(\{L_1\} \cup D)\sigma$ the set $\mathcal{V}(L\sigma)$ is a subset of $\mathcal{V}(L)$ and the depth of $L\sigma$ is equal to the depth of L. Thus, no singular literal L in C will become a non-singular literal $L\sigma$ in $(\{L_1\} \cup D)\sigma$ nor will any deep literal L satisfying the third condition of strong CDV-freeness turn into a deep literal $L\sigma$ violating this condition.

If C is strongly CDV-free, since it contains no non-singular literal, then the factoring inference step will not introduce such a literal in C and $(\{L_1\} \cup D)\sigma$ is still strongly CDV-free.

If C is strongly CDV-free, since it contains no CDV literal, then we can argue as follows. Suppose $(\{L_1\} \cup D)\sigma$ contains a literal $L\sigma$ which is CDV. Then L is not singular, since any singular CDV-free literal has the form $(\neg)p(x)$ for some predicate symbol p and variable x. Instantiation with σ cannot introduce additional constants or duplicate variables into such a literal. Since deep literals also remain deep after instantiation, L can only be a shallow, non-singular literal. However, all shallow, non-singular literals in a k-regular clause contain the same set of variables and for any shallow, non-singular literal L and for all compound terms t occurring in the clause, the set of non-constant arguments of L are similar to the subset of non-constant arguments of the first k arguments of t. If instantiation with σ turns L into a singular literal with variable x, then it does so with every shallow, non-singular literal in the clause. Furthermore, the subset of non-constant terms of the first k arguments of any term t will contain only one variable, namely x. Thus, $(\{L_1\} \cup D)\sigma$ is strongly CDV-free due to the definition of strong CDV-freeness. ◻

Lemma 3.29. *Let L be a singular, CDV-free literal and σ be a substitution such that $\mathrm{C}(\sigma)$ contains only variables and constants. Then $L\sigma$ is either ground or CDV-free.*

Proof. Let $\mathcal{V}(L)$ be the singleton set $\{x\}$. If x is not an element of $\mathrm{D}(\sigma)$, then $L\sigma = L$ is still CDV-free. If $x\sigma$ is a constant, then $L\sigma$ is ground and therefore not singular. If $x\sigma$ is a variable, then L and $L\sigma$ are identical up to the renaming of variables. So, $L\sigma$ is CDV-free. ◻

Lemma 3.30. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, strongly CDV-free, k-regular clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let A_1 and $\neg A_2$ be dominating literals in C_1 and C_2, respectively. Then every split component of $(D_1 \cup D_2)\sigma$ is strongly CDV-free.*

Proof. The general observation that no singular or deep literal L in one of the premises will become a shallow, non-singular literal $L\sigma$ in the conclusion $(D_1 \cup D_2)\sigma$ remains true.

We distinguish the following cases:

1. Both A_1 and $\neg A_2$ are singular literals. Then neither C_1 and C_2, nor $(D_1 \cup D_2)\sigma$ contain a non-singular or deep literal. So, the conclusion of the resolution inference step is strongly CDV-free.

2. A_1 is a singular literal and $\neg A_2$ is a shallow, non-singular literal. D_1 and $D_1\sigma$ contain no non-singular or deep literal and D_2 contains no deep literal. The literal $\neg A_2\sigma$ is singular. Since all shallow, non-singular literals in D_2 are similar to $\neg A_2$, $D_2\sigma$ contains no non-singular or deep literal.

3. A_1 and $\neg A_2$ are shallow, non-singular literals. Neither D_1 nor D_2 contains a deep literal. If $A_1\sigma = A_2\sigma$ is singular, then $(D_1 \cup D_2)\sigma$ contains no non-singular literals. Suppose $A_1\sigma = A_2\sigma$ is again a shallow, non-singular literal. Let L be a CDV-free, singular literal in either D_1 or D_2. Since $\mathrm{C}(\sigma)$ contains only variables and constants, $L\sigma$ is either ground or CDV-free by Lemma 3.29.

4. A_1 is a deep literal and $\neg A_2$ is singular. Note that C_2 contains only singular literals and that $\mathcal{V}(C_2)$ is a singleton set. So, $\neg A_2$ is not necessarily CDV-free. Without loss of generality we can assume that σ maps the only variable occurring in $\neg A_2$ to some compound term t. That means, $\neg A_2\sigma$, and likewise any literal in $D_2\sigma$, is a deep literal. The elements of $\mathrm{C}(\sigma|_{\mathcal{V}(A_1)})$ are either variables or constants. So, if L is a CDV-free literal in D_1, then $L\sigma$ is still CDV-free or ground.

   Suppose D_1 contains a CDV literal. Then D_1 satisfies the second condition of strong CDV-freeness. Instantiation with σ will not introduce additional variables into compound terms and the term t also satisfies the requirements of the second condition. So, $(D_1 \cup D_2)\sigma$ is strongly CDV-free.

   Suppose D_1 contains a shallow, non-singular literal L. Note that D_1 does not contain a CDV literal. If $L\sigma$ is still CDV-free, then $D_1\sigma$ contains no CDV literal and $(D_1 \cup D_2)\sigma$ is strongly CDV-free. If $L\sigma$ is a CDV literal, then we argue as in the proof of Lemma 3.28 that $(D_1 \cup D_2)\sigma$ satisfies the second condition of strong CDV-freeness.

5. A_1 is a deep literal and $\neg A_2$ is a shallow, non-singular literal. Note that C_2 contains no CDV literal. The unifier σ maps some of the variables of $\neg A_2$ to compound terms. Thus, $D_2\sigma$ contains only deep literals and, possibly, CDV-free literals. We can follow the lines of the previous case to show that $(D_1 \cup D_2)\sigma$ is strongly CDV-free.

6. Both A_1 and $\neg A_2$ are deep literals. The important point to note in this case is the following. Let s and t be compound terms in $C_1\sigma$ or $C_2\sigma$. Let L be a shallow, non-singular literal in C_1 or C_2. Then
$$\arg_{set}^{1\ldots k}(s) = \arg_{set}^{1\ldots k}(t)$$

and

$$\arg_{set}(L) \setminus \mathsf{F}_0 \sim_Z \arg_{set}^{1\ldots k}(s) \setminus \mathsf{F}_0.$$

Consequently, if L_1 is a shallow, non-singular literal in D_1 and L_2 is a shallow, non-singular literal in D_2, either both $L_1\sigma$ and $L_2\sigma$ are singular literals and $\mathcal{V}(\arg_{set}^{1\ldots k}(t))$ is a singleton set for any compound term t in $(D_1 \cup D_2)\sigma$ or neither $L_1\sigma$ and $L_2\sigma$ is a shallow, non-singular literal.

It follows that $(D_1 \cup D_2)\sigma$ is strongly CDV-free. □

We have shown that the property of strong CDV-freeness is preserved under inferences by ordered factoring and ordered resolution. Since clause sets in $\overline{\mathrm{KC}}$ are not necessarily strongly CDV-free, we define a satisfiability equivalence preserving transformation which transform any clause set N in $\overline{\mathrm{KC}}$ into a strongly CDV-free clause set N'.

$N \Rightarrow_{\mathcal{M}} N' \cup \mathrm{Def}_L^A$ iff
(i) L is an occurrence of a CDV literal in a clause $C \in N$ which is not strongly CDV-free,
(ii) A is an atom of the form $p(x)$ where p is a new predicate symbol with respect to N and x is the variable occurring in L,
(iii) Def_L^A is a clause of the form $\{\neg A, L\}$, and
(iv) N' is obtained from N by replacing any occurrence of L by A.

Again, the clauses Def_L^A added by the transformation $\Rightarrow_{\mathcal{M}}$ are called *definitions*. Note that A is CDV-free and that the clause $\{\neg A, L\}$ is strongly CDV-free. As each transformation step removes at least one CDV literal in one of the clauses which are not strongly CDV-free, we eventually obtain a strongly CDV-free set of clauses by a sequence of transformation steps. We denote the resulting clause set by $N{\downarrow_{\mathcal{M}}}$ and the set of all definitions by $\mathrm{Def}_{\mathcal{D}}^{\rightarrow}(N{\downarrow_{\mathcal{M}}})$.

Lemma 3.31. *Let N be a set of clauses. Then $N{\downarrow_{\mathcal{M}}}$ can be computed in polynomial time and is satisfiable if and only if N is satisfiable.*

Proof. Since renaming is satisfiability equivalence preserving. □

An admissible ordering $\succ$ on literals suitable for our purpose has to satisfy one condition:

(3.3) If a literal L is $\succ$-maximal in a clause $C \in \mathsf{C}$, then there is no literal L' in C with $L' \succ_Z L$.

Note that it is not relevant whether $\succ$ is applied a priori like $\succ_Z$ or a posteriori: Since $\succ$ is stable under substitutions, if $L\sigma$ is $\succ$-maximal in $C\sigma$ then L is $\succ$-maximal in C.

We have seen that no ordering stable under substitutions satisfying this condition can exist if C is the class of all (indecomposable) k-regular clauses. We will now show that if C is the class of all (indecomposable) strongly CDV-free, k-regular clauses, we are able to define such an ordering.

Let $\succ_\Sigma$ be a total precedence on the predicate symbols and functions symbols such that

- $f \succ_\Sigma g$ if f is a n-ary function symbol, g is a m-ary function symbol, and $n > m$ holds;
- $f \succ_\Sigma p$ if f is a n-ary function symbol, $n \geq 1$, and p is a predicate symbol;
- $p \succ_\Sigma q$ if p is a n-ary predicate symbol, $n \geq 2$, and q is a unary predicate symbol;

- $p \succ_\Sigma c$ if p is a predicate symbol and c a constant symbol.

Every predicate symbol and function symbol has multiset status. Let $\succ_S$ be the recursive path ordering based on the precedence $\succ_\Sigma$. The ordering $\succ_S$ is extended to literals in the usual way. The resulting ordering on literals is again denoted by $\succ_S$. It is an admissible ordering according to the definition in Section 1.2.

Lemma 3.32. *Let C be an indecomposable, strongly CDV-free, k-regular clause. If $L_1 \succ_Z L_2$ for literals L_1 and L_2 in C, then $L_1 \succ_S L_2$.*

Proof. We distinguish the following cases according to the type of L_1:

1. L_1 is non-singular and shallow. Then L_2 is singular. Therefore, L_2 contains exactly one variable x and x is an argument of L_2 and L_1. If x is the only argument of L_2, then the multiset of arguments of L_1 is obviously greater than the multiset of arguments of L_2 and the predicate symbol of L_1 has precedence over the predicate symbol of L_2 by $\succ_\Sigma$. So, $L_1 \succ_S L_2$ holds. If L_2 contains more than one argument, then L_2 is not CDV-free, contradicting our assumption that C is strongly CDV-free.

2. L_1 is deep. L_1 contains a compound term t_1 dominating all the arguments of L_1. Since $L_1 \succ_Z L_2$ and $\succ_Z$ is transitive, for every argument of t_2 of L_2, $t_1 \succ_Z t_2$ holds. The term t_2 is either a variable or a compound term. In the first case, according to the definition of $\succ_Z$, t_2 is an argument of t_1, that is, t_2 is a subterm of t_1. In the second case, t_1 has the form $f(u_1, \ldots, u_m)$ and t_2 has the form $g(u_1, \ldots, u_n)$ such that $m > n$ holds. Due to the definition of the precedence on function symbols, $f \succ_\Sigma g$ holds. Therefore, we need to verify that $f(u_1, \ldots, u_m) \succ_S u_j$ holds, for all j with $1 \leq j \leq n$, but this is clear. □

We are now ready to present a decision procedure for the class $\overline{\text{KC}}$. Our calculus consists of the expansion rules "Delete", "Split", and "Deduce" described in Section 1.2. Recall that we restrict our attention to theorem proving derivations which are generated by strategies in which "Delete", "Split", and "Deduce" are applied in this order of (descending) priorities. In addition, no application of the "Deduce" expansion rule with identical premises and identical consequence may occur twice on the same path in the derivation. For any finite set N of k-regular clauses, any theorem proving derivation from N is fair.

Theorem 3.33.
Let $\succ$ be an admissible ordering on literals satisfying Condition (3.3). Let N be a finite set of k-regular clauses in $\overline{\text{KC}}$. Then any derivation from $N{\downarrow_{\mathcal{M}}}$ by ordered resolution and ordered factoring based on $\succ$ terminates.

Proof. By Lemma 3.31 $N{\downarrow_{\mathcal{M}}}$ is satisfiable if and only if N is satisfiable and $N{\downarrow_{\mathcal{M}}}$ contains only strongly CDV-free, k-regular clauses.

We can construct a fair theorem proving derivation from $N{\downarrow_{\mathcal{M}}}$ based on an ordering $\succ$ satisfying Condition (3.3) on indecomposable, strongly CDV-free, k-regular clauses. The ordering $\succ_S$ is an example of such an ordering.

Lemma 3.19 states that if we apply resolution to $\succ_Z$-maximal literals and split the resulting clauses, then the split components of any resolvent of two indecomposable, k-regular clauses are k-regular again. Lemma 3.20 states the same for factoring. Lemma 3.30 and Lemma 3.28 state that resolution and factoring on $\succ_Z$-maximal literals preserves strong CDV-freeness for the split components of the resulting clauses. Since $\succ$-maximal literals in indecomposable, strongly CDV-free, k-regular clauses are also $\succ_Z$-maximal literals according to Condition (3.3), these results remain valid for ordered resolution and ordered factoring based on $\succ$. Therefore, any split component of a clause derivable from N will be k-regular.

Since the "Split" expansion rule has priority over the "Deduce" expansion rule, we are sure that whenever we derive a decomposable resolvent or factor, it will be decomposed into its split components before further applications of "Deduce".

By Lemma 3.21 and Lemma 3.22 the number of variables in derived clauses will not exceed the maximal number of variables in clauses in $N{\downarrow_\mathcal{M}}$. By Theorem 3.25 there exists an upper bound for the number of clauses in a set of k-regular clauses. That means any theorem proving derivation T from $N{\downarrow_\mathcal{M}}$ will be finite.

If $N{\downarrow_\mathcal{M}}$ is unsatisfiable, then every leave N'' of T will contain the empty clause and if N is satisfiable, then some leave will not contain the empty clause due to Theorem 1.2. $\square$

Corollary 3.34. *The class* $\overline{\mathrm{KC}}$ *is decidable.*

Consider the following set N of clauses

$$\begin{array}{rl}
(4) & \{p(a,a,x), r(a,x,y)\} \\
(5) & \{\neg r(a,a,x), p(a,x,y)\} \\
(6) & \{r(a,a,x), \neg p(a,x,y)\} \\
(7) & \{\neg p(a,a,x), \neg r(a,x,y)\}
\end{array}$$

We show how our decision procedure constructs an expansion from N. First, we note that none of the clause in N is strongly CDV-free. The result of transforming N into a satisfiability equivalent set N_1 of strongly CDV-free clauses is

$$\begin{array}{rl}
(8) & \{p_1^+(x), r(a,x,y)\} \\
(9) & \{r_1^-(x), p(a,x,y)\} \\
(10) & \{r_1^+(x), \neg p(a,x,y)\} \\
(11) & \{p_1^-(x), \neg r(a,x,y)\} \\
(12) & \{\neg p_1^+(x), p(a,a,x)\} \\
(13) & \{\neg r_1^-(x), \neg r(a,a,x)\} \\
(14) & \{\neg r_1^+(x), r(a,a,x)\} \\
(15) & \{\neg p_1^-(x), \neg r(a,a,x)\}
\end{array}$$

We will present only one branch of the theorem proving derivation from N_1.

$$\begin{array}{lrl}
[\,(8)2,\ \mathrm{R},\ (11)2] & (16) & \{p_1^+(x), p_1^-(x)\} \\
[(16)1,\ \mathrm{R},\ (12)1] & (17) & \{p(a,a,x), p_1^-(x)\} \\
[(17)1,\ \mathrm{R},\ (10)2] & (18) & \{r_1^+(a), p_1^-(x)\} \\
[(18)1, \mathrm{Spt}\qquad] & (19) & \{r_1^+(a)\} \\
[(19)1,\ \mathrm{R},\ (14)1] & (20) & \{r(a,a,a)\} \\
[(20)1,\ \mathrm{R},\ (11)2] & (21) & \{p_1^-(a)\} \\
[(21)1,\ \mathrm{R},\ (15)2] & (22) & \{\neg p(a,a,a)\} \\
[(22)1,\ \mathrm{R},\ \ (9)2] & (23) & \{r_1^-(a)\} \\
[(23)1,\ \mathrm{R},\ (13)2] & (24) & \{\neg r(a,a,a)\} \\
[(24)1,\ \mathrm{R},\ (20)1] & (25) & \bot
\end{array}$$

Here [(18)1, Spt] denotes an application of the "Split" expansion rule on the first literal in clause (18). It is straightforward to check that we are able to derive the empty clause in all remaining branches by similar sequences of inference steps. Hence, the initial set N is unsatisfiable.

A particularly interesting variant of our decision procedure can by obtained by the utilisation of a *selection function*. The selection function $S_{\mathcal{KC}}$ selects the monadic literal $\neg A$ in a clause Def_L^A introduced by the transformation $\Rightarrow_{\mathcal{M}}$. Note that positive occurrences of A in $N{\downarrow_{\mathcal{M}}}$ are not $\succ$-maximal in their clauses. Thus, inferences with clauses in Def_L^A are prohibited. Only when a clause C with a $\succ$-maximal literal A and with selected counterpart $\neg A$ in $\mathrm{Def}_{\mathcal{D}}^{\rightarrow}(N{\downarrow_{\mathcal{M}}})$ is produced by ordered resolution will an inference step with Def_L^A be performed. Effectively, such an inference step reintroduces (an instance of) the original literal occurrence L into C. Now, leaving these inference steps aside, a theorem proving derivation performed by ordered resolution with selection corresponds one-to-one to a theorem proving derivation based on the non-liftable ordering $\succ_Z$. To be able to simulate inference steps by ordered factoring we have use the approach described in Section 2.3.

The picture changes if we take the different notions of redundancy underlying these calculi into account. Note that the inference step leading to the clause $\{p_1^+(x), p_1^-(x)\}$ corresponds to the inference step

$$[\ (4)2,\quad \mathrm{R},\quad (7)2] \qquad (16') \qquad \{p(a,a,x), \neg p(a,a,x)\}$$

on the original clause set N which results in a tautological clause. In a decision procedure based on the non-liftable ordering $\succ_Z$ such tautological clauses are not redundant and, in general, cannot be eliminated without loosing completeness of the procedure. This is already evident in our example, since the only alternative inference step possible on N is the derivation of the tautological clause $\{r(a,a,x), \neg r(a,a,x)\}$ from clauses (2) and (3). Thus, the only clauses derivable from the unsatisfiable clause set N based on the $\succ_Z$-refinement of resolution are tautological.

In contrast, we can make use of the notion of redundancy introduced by Bachmair and Ganzinger [10]. For example, given the clause set N_2 containing the clauses

$$
\begin{array}{ll}
(26) & \{p(a,x,y), r(b,x,y)\} \\
(27) & \{\neg p(a,a,z), \neg r(b,a,z)\}
\end{array}
$$

which are strongly CDV-free, the tautological conclusion

$$[(26)2,\quad \mathrm{R},\ (27)2] \qquad (28) \qquad \{p(a,a,x), \neg p(a,a,x)\}$$

is redundant and can be eliminated.

3.5 The class $\overline{\mathrm{DK}}$

In this section we consider the class $\overline{\mathrm{DK}}$ containing all possible (finite) conjunctions of formulae of the class $\overline{\mathrm{K}}$. Note that the grade of the formulae in such a conjunction can vary. The class of clause sets obtained from formulae of the class $\overline{\mathrm{DK}}$ is denoted by $\overline{\mathrm{DKC}}$.

We will prove that the satisfiability problem for formulae in $\overline{\mathrm{DK}}$ is decidable using the procedure of the previous section.

Definition 3.35 (k-originated Skolem function).
Let φ be a formula of the class $\overline{\mathrm{K}}$ and let φ be of grade k. Let N be the corresponding set of clauses. Then we call a non-constant Skolem function f occurring in some clause in N *k-originated*.

Definition 3.36 (Strongly k-regular clause).
Let C be a k-regular clause such that all non-constant Skolem functions occurring in C are k-originated. Then C is *strongly k-regular.*

There is a rather subtle point to note about the definition of k-regular and strongly k-regular clauses. The value of k can be chosen almost arbitrarily if the clause does not contain compound terms. For example, if we talk about 3-regular clauses in the following, this includes clauses like

$$\{p_1(x_1, b, x_2, x_3), p_1(x_1, x_2, a, x_3), p_3(x_3)\}$$

and

$$\{p_4(x_1, x_2), p_4(x_2, x_1)\}$$

as well as

$$\{p_5(x_3, c, x_2, x_1, x_4)\}.$$

To distinguish clauses like these from clauses actually containing k-originated function symbols, we introduce the notion of *inhabited clauses.*

Definition 3.37.
A clause containing at least one non-constant Skolem function symbol is called *inhabited clause.*

Lemma 3.38. *Let φ be a formula of class $\overline{\mathrm{K}}$ and let φ be of grade k. Let N be the corresponding set of clauses. Then every clause in N is strongly k-regular.*

Proof. Every clause in N is k-regular according to Lemma 3.15. Following Definition 3.35, all Skolem functions in N are k-originated. Thus, N contains strongly k-regular clauses only. □

Corollary 3.39. *Let φ be a formula of class $\overline{\mathrm{DK}}$. Let N be the corresponding set of clauses. Then every clause in N is strongly k-regular for some $k \in \mathbb{N}$.*

Lemma 3.40. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, strongly k-regular clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let A_1 and $\neg A_2$ be dominating literals in C_1 and C_2, respectively. Then the split components of the resolvent $(D_1 \cup D_2)\sigma$ are strongly k-regular.*

Proof. According to Lemma 3.20 the split components of $(D_1 \cup D_2)\sigma$ are k-regular. Since all the Skolem functions of $(D_1 \cup D_2)\sigma$ already occur in one of the parent clauses C_1 and C_2, all the non-constant Skolem functions in $(D_1 \cup D_2)\sigma$ are k-originated. Thus, the split components of $(D_1 \cup D_2)\sigma$ are strongly k-regular. □

Lemma 3.41. *Let C be a strongly k-regular clause. Let D be a factor of C. Then the split components of D are strongly k-regular.*

Proof. By Lemma 3.20 the split components of clause D are k-regular. Since all the Skolem functions of D already occur in C, all the non-constant Skolem functions in D are k-originated. Thus, the split components of D are strongly k-regular. □

Lemma 3.42. *Let $C_1 = \{A_1\} \cup D_1$ be a k_1-regular clause and $C_2 = \{\neg A_2\} \cup D_2$ be a k_2-regular clause such that A_1 and A_2 are unifiable with most general unifier σ and A_1 and $\neg A_2$ are dominating literals in C_1 and C_2, respectively. Let all the non-constant Skolem functions in C_1 and C_2 be k_1'-originated and k_2'-originated, respectively. Then k_1' is equal to k_2' and all the non-constant Skolem functions in $(D_1 \cup D_2)\sigma$ are k_1'-originated.*

Proof. If neither C_1 nor C_2 contain function symbols, the lemma is trivially true. Without loss of generality we assume that at least one function symbol occurs in C_1. So, the clause C_1 contains a deep literal. As a consequence, the literal A_1 is deep as well. Therefore, there exists a compound term t_1 in A_1 such that t_1 dominates every argument of every literal in C_1.

According to the form of $\neg A_2$, we distinguish the following cases.

1. $\neg A_2$ is a singular literal. Let x be the variable of $\neg A_2$. Then $x\sigma$ is a compound term. The resolvent $(D_1 \cup D_2)\sigma$ will contain only non-constant Skolem functions of C_1. Since all non-constant Skolem functions in C_1 are k_1'-originated, this will be the case for $(D_1 \cup D_2)\sigma$ as well.

2. $\neg A_2$ is a non-singular and shallow literal. Similar to the previous case, we can assume without loss of generality that the domain of σ contains all the variables of $\neg A_2$ and therefore all the variables in C_2. Again $(D_1 \cup D_2)\sigma$ contains only non-constant function symbols occurring in C_1.

3. $\neg A_2$ is a deep literal. Then there is a term t_2 in $\neg A_2$ dominating every argument of every literal in C_2. According to Lemma 3.17(1) the terms t_1 and t_2 occur at the same argument position in A_1 and $\neg A_2$, respectively. Since A_1 and A_2 are unifiable, t_1 and t_2 are unifiable too. To this end, the top function symbol of t_1 and t_2 is equal. Let us denote this function symbol by g. The function symbol g is k_1'-originated and k_2'-originated according to our assumptions. Thus k_1' and k_2' have to be equal. The conclusion that all Skolem functions in $(D_1 \cup D_2)\sigma$ are k_1'-originated is now a straightforward consequence of the equality of k_1' and k_2'. □

Corollary 3.43. *Let C_1 be an inhabited, strongly k_1-regular clause and C_2 be an inhabited, strongly k_2-regular clause such that $k_1 \neq k_2$. Then C_1 and C_2 have no ordered resolvent with respect to $\succ_Z$.*

By Lemma 3.42 and Corollary 3.43:

Lemma 3.44. *Let C_1 be an indecomposable, strongly k_1-regular clause and C_2 an indecomposable, strongly k_2-regular clause such that C_1 and C_2 are variable-disjoint. Let C be a resolvent of C_1 and C_2, such that the literals resolved up on are maximal with respect to $\succ_Z$. Every split component of C is strongly k-regular for some k.*

Theorem 3.45.
Let $\succ$ be an admissible ordering on literals satisfying Condition (3.3). Let N be a finite set of k-regular clauses in $\overline{\mathrm{DKC}}$. Then any derivation from $N{\downarrow_{\mathcal{M}}}$ by ordered resolution and ordered factoring based on $\succ$ terminates.

Proof. The procedure described in Section 3.4 provides also a decision procedure for $\overline{\mathrm{DKC}}$. The proof follows the lines of the proof of Theorem 3.33. □

Corollary 3.46. *The class $\overline{\mathrm{DKC}}$ is decidable.*

3.6 Related Work

Maslov [97, p. 69] defines the class K in the following way.

> **Definition (The Class K).** Let us denote by K the following class of formulae of the predicate calculus (without function symbols and equality): the formula φ in negation normal form (which may be non-closed) belongs to K if and only if there exist variables $x_1, x_2, \ldots, x_k$ which do not lie within the scope of any universal quantifier such that all the non-empty φ-prefixes of atomic subformulae of φ are of length 1, or end in a universal quantifier, or are of the form $\exists x_1 \ldots \exists x_k$.

Formulae can be non-closed and free variables are implicitly universally quantified. He then states the following theorem [97, p. 70].

> **Theorem 2.** The class of arbitrary disjunctions of formulae from K is decidable with respect to deducibility in the calculus Π.

The calculus Π is the *inverse method* [97]. The inverse method can be regarded to be dual to resolution which has been introduced independently by Robinson [117]. Given a set N of clauses, resolution attempts to show the unsatisfiability of N by deriving the empty clause. In contrast, the inverse method attempts to compute the logical consequences of N. Bachmair and Ganzinger [10] establish a one-to-one correspondence between the inverse method and positive hyper-resolution.

Maslov's argument of termination is limited to the following [97, p. 75]:

> Applying Lemma 5, we obtain a branched process for constructing single-member favourable collections which yields an empty collection for every deducible sequence (7)[1]. Let the number of all single-member F-collections be finite, and let this process terminate (before attainment of an empty collection) at some branch even if sequent (7) is non-deducible. Theorem 2 has been proved.

Our proof presented in Sections 3.2 to 3.4 reworks and improves the results of Zamov [39, chapter 6], who uses a refinement of resolution to provide a decision procedure for the class K. Thus, Zamov's approach fits better to our framework. However, there are several problems with the presentation in [39]. First, the definition of $\overline{\mathrm{K}}$, simply called K, does not coincide with the class of complementary K-formulae. We will denote Zamov's class by $\overline{\mathrm{K}}_Z$. It is defined as follows.

> **Definition (Class $\overline{\mathrm{K}}_Z$).** The formula F belongs to the class $\overline{\mathrm{K}}_Z$ if there exist variables $x_1, \ldots, x_k$, $k \geq 0$, which are not in the scope of any existential quantifier, such that each non-empty F-prefix of an atomic subformula of F
>
> - either is of length 1,
> - ends with an existential quantifier,
> - is of the form $\forall x_1 \forall x_2 \ldots \forall x_k$.

[1] That is, a disjunction of formulae from K.

Zamov also uses a slightly different definition of F-prefix, which does not require that the formula F is in negation normal form nor that is is closed. It is not mentioned whether it is a prerequisite that the formula F satisfies either of these properties. Nevertheless it is easy to see that negation normal form is a prerequisite for the formulae under consideration. For example, the formula φ

$$\neg\exists x_1\, \exists x_2\, \exists x_3 \colon \neg(r(x_1, x_2) \wedge r(x_2, x_3) \rightarrow r(x_1, x_3)),$$

which is logically equivalent to the transitivity formula for the binary relation r, literally satisfies the conditions in the definition of the class $\overline{\mathrm{K}}_Z$: The φ_6-prefixes of $r(x_1, x_2)$, $r(x_2, x_3)$, $r(x_1, x_3)$ are $\exists x_1\, \exists x_2$, $\exists x_2\, \exists x_3$, and $\exists x_1\, \exists x_3$, respectively, which all end in an existential quantifier. However, transitivity formulae are out of the scope of Maslov's, Zamov's, and our method. Alternatively, in addition to the assumption that all formulae are in negation normal form, we could redefine the meaning of 'universal' and 'existential quantifier' to take into account the polarity of their occurrence.

Likewise, free non-constant function symbols can be used to circumvent the restrictions on admissible formulae intended by the definition of $\overline{\mathrm{K}}_Z$. Therefore, it is not in Zamov's intention to allow such function symbols. The question whether free constant symbols are allowed is closely related to the question whether free variables are admissible. Note that the definition of Maslov explicitly allows non-closed formulae. One possible interpretation of the definition of $\overline{\mathrm{K}}_Z$ is that formulae in $\overline{\mathrm{K}}_Z$ can be non-closed as well, but the free variables are implicitly existentially quantified. Consequently, a formula like φ_7

$$\forall x_1\, \forall x_2 \colon p(y, x_1, x_2)$$

is an element of class $\overline{\mathrm{K}}_Z$ of grade 2: The universally quantified variables of φ_7 are x_1 and x_2, the φ_7-prefix of the atomic subformula $p(y, x_1, x_2)$ is $\forall x_1\, \forall x_2$ and satisfies the third condition of the definition above. But contrary to the standard interpretation of free variables, y is existentially quantified. The formula φ_8

$$\exists y\, \forall x_1\, \forall x_2 \colon p(y, x_1, x_2)$$

is certainly not in the class $\overline{\mathrm{K}}_Z$: The universally quantified variables of φ_7 are x_1 and x_2 again, the φ_7-prefix of the atomic subformula $p(y, x_1, x_2)$ is $\exists y\, \forall x_1\, \forall x_2$. Obviously, it is neither of length 1, nor does it not end in an existential quantifier, and is not of the form $\forall x_1\, \forall x_2$.

Restricting ourselves to schemata in negation normal form instead of arbitrary formulae and using terminal φ-prefixes instead of φ-prefixes we can avoid this problem and obtain Definition 3.2 as presented in Section 3.1.

Second, there are problems with the notion of dominating literal and regular set of literals as defined by Zamov when constant symbols are present. His definitions are as follows. The literal L_1 *dominates* the literal L_2, if the set of arguments of L_1 dominates the set of arguments of L_2. A set of terms T_1 is *similar* to a set of terms T_2 if T_1 dominates T_2 and T_2 dominates T_1. A set M of literals is called *k-regular* if the following conditions hold: (i) M contains regular literals only, (ii) the non-negative integer k is not greater than the minimal arity of function symbols occurring in literals of the set M, (iii) M contains some literal which dominates every literal in the set M, and (iv) all shallow, non-singular literals of M are similar and the set of non-constant arguments of any such literal is similar to the set of the first k arguments of any compound term occurring in any literal from M. A set M of literals is called *regular* if it is k-regular for some k, $k \geq 0$.

According to these definitions, the literals $q(x,y)$ and $p(a,x,y)$ are not similar. So, the clause $\{q(x,y), p(a,x,y)\}$ is not regular. Neither is $\{q(a,x,y), p(f(a,x,y))\}$. (But, according to our definitions, $q(x,y)$ and $p(a,x,y)$ are similar literals and the clauses $\{q(x,y), p(a,x,y)\}$ and $\{q(a,x,y), p(f(a,x,y))\}$ are regular.)

Even if we start with regular clauses, resolution eventually generates clauses which are not regular. Consider the 3-regular clauses

$$\{p(x,y,z), q(f(x,y,z)), r(f(x,y,z))\} \qquad \text{and} \qquad \{\neg r(f(a,y',z'))\}.$$

There is exactly one resolvent which is

$$\{p(a,y,z), q(f(a,y,z))\}.$$

This clause is not regular, since the set of non-constant arguments of the non-singular literal $p(a,y,z)$, that is, $\{y,z\}$, is not similar to the set of the first three arguments of $f(a,y,z)$, that is, $\{a,y,z\}$ (nor to one of the sets $\{a,y\}$ or $\{a\}$).

Third, one of the fundamental steps in Zamov's proof is the following lemma (see [39, page 147]) which does not hold in general.

> **Lemma** Assume that $C_1 \cup \{A\}$ and $C_2 \cup \{B\}$ are indecomposable regular clauses and the clause $(C_1 \cup C_2)\sigma$ is a resolvent of these clauses by resolution upon A and B. Then if the literals A and B are dominating for their clauses, then $(C_1 \cup C_2)\sigma$ is quasi-regular.

Here is a counter example. The clauses

$$\{\neg p(f(x,y,z),z), q(x,y), r(f(x,y,z))\} \tag{3.29}$$

and

$$\{p(f(x',y',z'), g(x'))\} \tag{3.30}$$

are both indecomposable and regular clauses. The clause (3.29) is 2-regular and clause (3.30) is 1-regular. But the resolvent

$$\{q(x,y), r(f(x,y,g(x)))\} \tag{3.31}$$

is not quasi-regular. The minimal arity of a function symbol in clause (3.31) is one. There is one non-singular literal $q(x,y)$ in the clause with non-constant arguments x and y. These arguments have to be similar to the first argument of any compound term in clause (3.31). This is of course impossible.

Lemma 3.19 requires that both clauses are k-regular and we show in Section 3.5 that clauses like (3.31) cannot occur in derivations from a clause set in $\overline{\text{KC}}$ or $\overline{\text{DKC}}$.

Finally, the completeness proof is not without problems. Zamov considers the set N' of all ground instances of clauses in a clause set N. He defines an ordering $>$ on ground terms and occurrences of ground literals in N' which is then lifted to the non-ground case. He claims that $>$ is a π-ordering and that due to the completeness of π-orderings a refutation of N' with respect to the ordering refinement $>$ exists if N' is unsatisfiable.

However, $>$ is not a π-ordering according to the definition in [39, chapter 4], which is a reformulation of Maslov's original definition imposing stricter conditions on an ordering, but only according to Maslov's original definition. Orderings satisfying Maslov's original definition do not guarantee completeness in general as Weidenbach [132, pp. 8-10] shows. Since it is essential for Zamov's argument that a ground instance of a non-singular, shallow literal is strictly greater than a ground instance of a singular, shallow literal, it is not possible to define a liftable ordering $>'$ which coincides with $>$.

As discussed in Section 3.4 it is possible to define a decision procedure based on the non-liftable ordering $\succ_Z$. A completeness proof for this procedure could be obtained by the method of de Nivelle [27].

3.7 Conclusion

The classes $\overline{\text{KC}}$ and $\overline{\text{DKC}}$ discussed in this chapter are among the most interesting decidable classes of clause sets. It covers a variety of the classical decidable fragments of first-order logic such as the initially extended Ackermann class, the Monadic class, the initially extended Skolem class, the initially extended Gödel class, and the two-variable fragment of first-order logic (using an appropriate structural transformation). We will also see that many of the logics discusses in Chapter 4 and 5 are contained in $\overline{\text{KC}}$.

Although neither $\overline{\text{KC}}$ nor $\overline{\text{DKC}}$ include some of the more recently introduced classes of clause sets, like $\mathcal{S}^+$ and $\mathbf{E}^+$, they have a property which is unique among these classes. Unlike the initially extended Skolem class, $\mathcal{S}^+$, $\mathbf{E}^+$, the class One-Free, or the loosely guarded fragment, not all compound terms in k-regular clauses contain all the variables of the clause. This leads to the possibility that the clause depth of a resolvent is greater than the maximal clause depth of its parent clauses even if we use a non-liftable ordering to restrict permissible inference steps. Unlike the loosely guarded fragment we also have no restriction on the polarity of particular literals, that is, there is no notion of *guards* in k-regular clauses.

This leads to the interesting question whether extensions of $\overline{\text{KC}}$ and $\overline{\text{DKC}}$ can be obtained by the introduction of this notion. Note that the requirement in the definition of k-regular clauses that all shallow, non-singular literals have to be similar is rather restrictive compared to the requirement in the definition of guarded clauses. It excludes for example clauses like $\{p(x,y,z), \neg r(x,y)\}$, $\{\neg p(x,y,z), \neg r(y,z), q(x,y,z)\}$, and $\{\neg q(x,y,z), r(x,z)\}$. (The first clause is not guarded because the only literal with all three variables is positive.) Each clause contains a literal containing all variables of the clauses. If we restrict resolution inferences to these literals, then no resolvent will contain more variables than its parents. However, with two inference steps we derive the transitivity clause $\{\neg r(x,y), \neg r(y,z), r(x,z)\}$ and inferences with this clause will no longer obey a bound on the number of variables. This observation motivates the restriction embodied in the definition of k-regular clauses. It also explains why the restriction on the polarity of guards cannot be easily dispensed with. A way leading to an extension of $\overline{\text{DKC}}$ is to consider whether the requirement that (shallow) clauses have a *negative* literal containing all the variables of the clause is sufficient to retain decidability.

One further direction of future research concerns the relationship of $\overline{\text{DKC}}$ and the classes $\mathbf{E}^+$ and One-Free. These classes are more liberal concerning the structure of terms occurring in one of the classes. For example, non-regular terms like $f(f(x))$ or $f(g(a), h(x))$ are admissible in $\mathbf{E}^+$. This leads to the question whether it is possible to (slightly) generalise the notion of regular terms to extend the class $\overline{\mathrm{DKC}}$.

Chapter 4

Description logics

Two research areas where decidability issues play a particularly prominent role are: extended modal logics and description logics. Although it is not difficult to see that most of the logics under consideration can be translated to first-order logic, it is not obvious what the characteristics of the corresponding classes of first-order formulae are which makes these logics decidable. Furthermore, the fact that the class of first-order formulae resulting from the translation of modal formulae or expressions in a description logic is decidable, does not indicate how a resolution-based decision procedure for this class can be obtained. Recent important results describe resolution decision procedures for the guarded fragment [29, 30] and the class One-Free [30, 129]. Both use non-liftable ordering restrictions. The most popular description logic is $\mathcal{ALC}$. It can be embedded by the optimised functional translation into a subclass of the Bernays-Schönfinkel class. The subclass, the basic path logic, can be decided by resolution and condensing using any compatible ordering or selection strategy [121].

In this chapter we consider both the satisfiability problem of concepts in a description logic and the satisfiability of knowledge bases. We describe a characterisation of clause sets obtained from the relational translation of terminological knowledge bases. We show that there are two fundamentally different approaches for obtaining resolution-based decision procedures: one is based on ordered resolution and one on selection. While the first approach covers a wider range of terminological logics, the latter approach is closely linked to tableaux-based decision procedures for description logic. We formally confirm this link by showing that the resolution-based decision procedure based on selection is able to polynomially simulate tableaux-based decision procedures for extensions of $\mathcal{ALC}$.

4.1 Syntax and semantics of description logics

We describe the language of the *universal terminological logic* $\mathcal{U}$ [111], largely adopting the modern notation introduced by Schmidt-Schauß and Smolka [123].

The *signature* is given by a tuple $\Sigma = (\mathsf{O}, \mathsf{C}, \mathsf{R})$ of three disjoint alphabets, the set C of *concept symbols*, the set R of *role symbols*, and the set O of *object symbols*. Concept symbols and role symbols are also called *atomic concepts* and *atomic roles*.

The set of *concept terms* (or just *concepts*) and *role terms* (or just *roles*) is inductively defined as follows. Every concept symbol is a concept term and every role symbol is a role term. Now assume that C and D are concepts, R and S are roles, and U and V are bindings. Then

- $\top$ (*top concept*), $\bot$ (*bottom concept*), $C \sqcap D$ (*concept intersection*), $C \sqcup D$ (*concept union*), $\neg C$ (*concept complement*), $\forall R.C$ (*universal restriction*), $\exists R.C$ (*existential restriction*), $\exists_{\geq n} R$, $\exists_{\leq n} R$ (*number restrictions*), $\exists_{\geq n} R.C$, $\exists_{\leq n} R.C$ (*qualified number restrictions*), $(R{=}S)$ (*role value maps*), and $\exists_{\mathsf{B}} U{:}C$ are *concept terms*,
- ∇ (*top role*), $\triangle$ (*bottom role*), id (*identity role*), $R \sqcap S$ (*role intersection*), $R \sqcup S$ (*role union*), $R \circ S$ (*role composition*), $\neg R$ (*role complement*), R^{-1} (*role converse*), R^{+} (*role closure*), $R|C$ (*domain restriction*), and $R\downharpoonright C$ (*range restriction*) are role terms, and
- $(\subseteq R\, S)$, $(\supseteq R\, S)$, $U \sqcap V$ are *bindings*.

Let CE be the set of all concepts, RE the set of all roles, and BE the set of all bindings. The tuple $\langle \mathsf{CE}, \mathsf{RE}, \mathsf{BE} \rangle$ forms the *term language* of the universal terminological language. An element of either CE, RE, and BE is a *(terminological) term* or *(terminological) expression*.

The definition differs from the original presentation in the addition of the qualified number restrictions $\exists_{\geq n} R.C$ and $\exists_{\leq n} R.C$. In the presence of the role restriction operator, $\exists_{\geq n} R.C$ and $\exists_{\leq n} R.C$ can be expressed by $\exists_{\geq n} R\downharpoonright C$ and $\exists_{\leq n} R\downharpoonright C$, respectively. Bindings are special role terms, namely $(\subseteq R\, S)$, $(\supseteq R\, S)$ and their conjunctions.

The set of sentences S over the term language $\langle \mathsf{C}, \mathsf{B}, \mathsf{R} \rangle$ is divided into *terminological sentences* and *assertional sentences*. If C and D are concepts, and R and S are roles, then $C \stackrel{.}{\sqsubseteq} D$, $C \doteq D$, $R \stackrel{.}{\sqsubseteq} S$, and $R \doteq S$ are terminological sentences. If C is a concept, R is a role, and a, b are individual objects then $a \in C$ and $(a, b) \in R$ are assertional sentences. A *knowledge base* is a finite set of terminological and assertional sentences. The set of assertional sentences of a knowledge base is usually called the *ABox*. The set of terminological sentences of a knowledge base is called the *TBox*. We say, a symbol S_0 *uses* a symbol S_1 in a TBox T *directly* if and only if T contains a sentence of the form $S_0 \doteq E$ or $S_0 \stackrel{.}{\sqsubseteq} E$ such that S_1 occurs in E. A symbol S_0 *uses* S_n if and only if there is a chain of symbols $S_0, \ldots, S_n$ such that S_i uses S_{i+1} directly, for every i, $1 \leq i \leq n{-}1$. A knowledge base Γ contains a *terminological cycle* if and only if some symbol uses itself in the TBox of Γ. Commonly, the following restrictions are imposed on the set of admissible terminological sentences in knowledge bases [123]:

- The concepts on the left-hand sides of terminological sentences have to be concept symbols,
- a concept symbol may occur at most once on the left-hand side of a terminological sentence, and
- there are no terminological cycles.

A knowledge base obeying these restrictions will be called a *descriptive knowledge base*. In a descriptive knowledge base terminological sentences $A \stackrel{.}{\sqsubseteq} C$ and $P \stackrel{.}{\sqsubseteq} R$ are called *concept* and *role specialisations*, respectively. $A \doteq C$ is a *concept definition* and A is a *defined concept*. Similarly, $P \doteq R$ is a *role definition* and P is a *defined role*.

The semantics of the terminological logic $\mathcal{U}$ is defined by a *terminological interpretation* which is a pair $(\mathcal{D}, v)$ consisting of a domain $\mathcal{D}$ and an interpretation function v. It maps the object symbols to elements of $\mathcal{D}$, concept symbols to subsets of $\mathcal{D}$ and the role symbols to subsets of $\mathcal{D} \times \mathcal{D}$. It is a standard requirement that v obeys the *unique name assumption*, that is, $v(a) \neq v(b)$ holds for every pair of object symbols $a \neq b \in \mathsf{O}$.

The interpretation function v extends in a natural way to complex concepts and roles:

$$
\begin{aligned}
v(\top) &= \mathcal{D} \\
v(\bot) &= \emptyset \\
v(C \sqcap D) &= v(C) \cap v(D) \\
v(C \sqcup D) &= v(C) \cup v(D) \\
v(\neg C) &= \mathcal{D} \setminus v(C) \\
v(\forall\, R.C) &= \{d \in \mathcal{D} \mid e \in v(C) \text{ for all } e \text{ with } (d,e) \in v(R)\} \\
v(\exists\, R.C) &= \{d \in \mathcal{D} \mid e \in v(C) \text{ for some } e \text{ with } (d,e) \in v(R)\} \\
v(\exists_{\geq n} R) &= \{d \in \mathcal{D} \mid |\{e \mid (d,e) \in v(R)\}| \geq n\} \\
v(\exists_{\leq n} R) &= \{d \in \mathcal{D} \mid |\{e \mid (d,e) \in v(R)\}| \leq n\} \\
v(\exists_{\geq n} R.C) &= \{d \in \mathcal{D} \mid |\{e \mid (d,e) \in v(R) \wedge e \in v(C)\}| \geq n\} \\
v(\exists_{\leq n} R.C) &= \{d \in \mathcal{D} \mid |\{e \mid (d,e) \in v(R) \wedge e \in v(C)\}| \leq n\} \\
v(R{=}S) &= \{d \in \mathcal{D} \mid \forall e\colon (d,e) \in v(R) \leftrightarrow (d,e) \in v(S)\} \\
v(\exists_{\mathsf{B}} U{:}C) &= \{d \in \mathcal{D} \mid e \in v(C) \text{ for some } e \text{ with } (d,e) \in v(U)\} \\
v(\bigtriangledown) &= \mathcal{D} \times \mathcal{D} \\
v(\bigtriangleup) &= \emptyset \\
v(\mathsf{id}) &= \{(d,d) \in \mathcal{D} \times \mathcal{D} \mid d \in \mathcal{D}\} \\
v(R \sqcap S) &= v(R) \cap v(S) \\
v(R \sqcup S) &= v(R) \cup v(S) \\
v(R \circ S) &= \{(d,e) \in \mathcal{D} \times \mathcal{D} \mid \exists c\colon (d,c) \in v(R) \wedge (c,e) \in v(S)\} \\
v(\neg R) &= (\mathcal{D} \times \mathcal{D}) \setminus R \\
v(R^{-1}) &= \{(d,e) \in \mathcal{D} \times \mathcal{D} \mid (e,d) \in v(R)\} \\
v(R^{+}) &= v(R)^{+} \\
v(R{\restriction}C) &= \{(d,e) \in v(R) \mid d \in v(C)\} \\
v(R{\downharpoonleft}C) &= \{(d,e) \in v(R) \mid e \in v(C)\} \\
v(\subseteq R\, S) &= \{(d,e) \in \mathcal{D} \times \mathcal{D} \mid \forall c\colon (d,c) \in v(R) \to (e,c) \in v(S)\} \\
v(\supseteq R\, S) &= \{(d,e) \in \mathcal{D} \times \mathcal{D} \mid \forall c\colon (e,c) \in v(S) \to (d,c) \in v(S)\} \\
v(U \sqcap V) &= v(U) \cap v(V)
\end{aligned}
$$

Let $(\mathcal{D}, v)$ be a terminological interpretation. The satisfiability relation $\models$ is defined by:

$$
\begin{array}{lcc}
(\mathcal{D},v) \models a \in C & \text{iff} & v(a) \in v(C) \\
(\mathcal{D},v) \models (a,b) \in R & \text{iff} & (v(a),v(b)) \in v(R) \\
(\mathcal{D},v) \models C \mathrel{\dot{\sqsubseteq}} D & \text{iff} & v(C) \subseteq v(D) \\
(\mathcal{D},v) \models C \doteq D & \text{iff} & v(C) = v(D) \\
(\mathcal{D},v) \models R \mathrel{\dot{\sqsubseteq}} S & \text{iff} & v(R) \subseteq v(S) \\
(\mathcal{D},v) \models R \doteq S & \text{iff} & v(R) = v(S)
\end{array}
$$

Let Γ be a knowledge base. We say $(\mathcal{D}, v)$ *satisfies* Γ, written $(\mathcal{D}, v) \models \Gamma$, if $(\mathcal{D}, v)$ satisfies every sentence in Γ. In this case, $(\mathcal{D}, v)$ is a *(terminological) model* of Γ. We say a knowledge base Γ *entails* a sentence α, written $\Gamma \models \alpha$, if every model of Γ satisfies α.

An occurrence of a subexpression is a *positive occurrence* if it is one inside the scope of an even number of (explicit or implicit) negations (complements), and an occurrence is a *negative occurrence* if it is one inside the scope of an odd number of negations. For example, both occurrences of the subformula $\neg C \sqcap D$ in $(\exists R^{-1}.(\neg C \sqcap D)) \sqcap (\forall R \sqcup S.(\neg C \sqcap D))$ have positive polarity, R^{-1} has positive polarity, and $R \sqcup S$ has negative polarity.

A concept C is *coherent* or *satisfiable* if there exists a terminological interpretation $(\mathcal{D}, v)$ such that $v(C)$ is non-empty. Otherwise, C is *incoherent* or *unsatisfiable*. A concept C is *coherent with respect to* Γ if there exists a terminological model $(\mathcal{D}, v)$ of Γ such that $v(C)$ is non-empty.

The basic inference services provided by terminological systems can be classified as follows: (i) *subsumption of concepts*: decide whether $\emptyset \models C \dot{\sqsubseteq} D$ holds for concepts C and D in which case D *subsumes* C; (ii) *subsumption of concepts with respect to a TBox* T: decide whether $T \models C \dot{\sqsubseteq} D$ holds; (iii) *equivalence of concepts (with respect to a TBox* T*)*: decide whether C subsumes D and D subsumes C at the same time for two concepts C and D (with respect to a TBox T); (iv) *classification of a TBox* T: decide for all concept symbols A and B occurring in T whether A subsumes B or B subsumes A with respect to T; (v) *satisfiability of a concept (with respect to a TBox* T*)*: decide for a concept C whether it is satisfiable (with respect to T); (vi) *consistency of an ABox* A *with respect to a TBox* T: decide whether the knowledge base $A \cup T$ is satisfiable; (vii) *instance checking*: decide whether a given a knowledge base Γ entails a given assertional sentence of the form $a \in C$; (viii) *realization*: compute for an object symbol a in a knowledge base Γ the set of most specific (with respect to the subsumption relation) concept symbols C such that $\Gamma \models a \in C$. (ix) *retrieval*: compute for a given concept C in a knowledge base Γ those object symbols a such that Γ entails $a \in C$. All these inferential services can be realized by satisfiability tests for a knowledge base. For example, to determine whether $T \models C \sqsubseteq D$ holds, we test the satisfiability of $T \cup \{a \in C, a \in \neg D\}$ where a is some arbitrary object symbol. Note that all these inferential services are restricted to the consideration of concepts and objects. Although it is straightforward to define inferential services for roles in analogy to (i)–(ix), there are computational problems with their realization in terminological systems.

Schild [119] has shown that the subsumption problem for a sublanguage of $\mathcal{U}$ containing only role intersection, role complement, role composition, and the identity role, is undecidable. Schmidt-Schauß [122] has shown that the subsumption problem for a sublanguage of $\mathcal{U}$ containing only concept intersection, universal and existential restrictions, role composition, and role value maps is undecidable. From the literature on extended modal logics and algebraic logic it is known that role composition and role complement together with role intersection or union lead to undecidability [2]. Since decidability of the inferential services is one of the major design goals of terminological systems, the negative results by Schild and Schmidt-Schauß led to most of the role-forming operators, terminological sentences concerning roles, and all inferential services concerning roles to be abandoned.

The description logic $\mathcal{ALC}$ is the sublanguage of $\mathcal{U}$ containing the top and bottom concept, concept complement, concept intersection, concept union), universal restriction, and existential restriction. In the subsequent sections we focus on a language which we call $\mathcal{ALB}$ (short for ‘attribute language with Boolean algebras on concepts and roles’). It extends $\mathcal{ALC}$ with the top role, role complement, role intersection, role union, role converse, role value maps, domain restriction, and range restriction. For ease of presentation we consider only the consistency test operation for knowledge bases. As mentioned above this does no restrict the generality of the results.

4.2 $\mathcal{ALB}$ and DL-clauses

Before we define the translation of knowledge bases over $\mathcal{ALB}$ to first-order logic, we will first show that all concepts can be transformed into *negation normal form*. We let the *negation normal form* $\mathrm{nnf}(E)$ of an $\mathcal{ALB}$ expression E be obtained by the following rewrite rules:

$$\begin{array}{ccc}
\neg\top \Rightarrow \bot & \neg\neg E \Rightarrow E & \neg\forall R.C \Rightarrow \exists R.\neg C \\
\neg\triangledown \Rightarrow \vartriangle & \neg(E \sqcap F) \Rightarrow \neg E \sqcup \neg F & \neg\exists R.C \Rightarrow \forall R.\neg C \\
\neg\bot \Rightarrow \top & \neg(E \sqcup F) \Rightarrow \neg E \sqcap \neg F & \neg(R\upharpoonleft C) \Rightarrow \neg R \sqcup (\triangledown\upharpoonleft\neg C) \\
\neg\vartriangle \Rightarrow \triangledown & \neg(R^{-1}) \Rightarrow (\neg R)^{-1} & \neg(R\downharpoonleft C) \Rightarrow \neg R \sqcup (\triangledown\downharpoonleft\neg C).
\end{array}$$

By a *basic concept* we mean a concept symbol, $\top$, $\bot$, or its negation and by a *basic role* we mean a role symbol, $\triangledown$, $\vartriangle$, its negation or its converse. The following rewrite rules can be used to ensure that the role converse operator is only applied to basic roles.

$$\begin{array}{ccc}
(R \sqcap S)^{-1} \Rightarrow R^{-1} \sqcap S^{-1} & (R \sqcap S)\upharpoonleft C \Rightarrow R\upharpoonleft C \sqcap S\upharpoonleft C & (R \sqcap S)\downharpoonleft C \Rightarrow R\downharpoonleft C \sqcap S\downharpoonleft C \\
(R \sqcup S)^{-1} \Rightarrow R^{-1} \sqcup S^{-1} & (R \sqcup S)\upharpoonleft C \Rightarrow R\upharpoonleft C \sqcup S\upharpoonleft C & (R \sqcup S)\downharpoonleft C \Rightarrow R\downharpoonleft C \sqcup S\downharpoonleft C \\
(\neg R)^{-1} \Rightarrow \neg(R^{-1}) & (R\upharpoonleft C)^{-1} \Rightarrow R^{-1}\downharpoonleft C & (R\downharpoonleft C)^{-1} \Rightarrow R^{-1}\upharpoonleft C.
\end{array}$$

$\mathcal{ALB}$ expressions can be further simplified according to the following rewrite rules.

$$\begin{array}{cccc}
C \sqcap \top \Rightarrow C & R \sqcap \triangledown \Rightarrow R & \forall R.\top \Rightarrow \top & R\upharpoonleft\top \Rightarrow R \\
C \sqcup \top \Rightarrow \top & R \sqcup \triangledown \Rightarrow \triangledown & \exists R.\bot \Rightarrow \bot & R\upharpoonleft\bot \Rightarrow \vartriangle \\
C \sqcap \bot \Rightarrow \bot & R \sqcap \vartriangle \Rightarrow \vartriangle & \forall \vartriangle.C \Rightarrow \top & R\downharpoonleft\top \Rightarrow R \\
C \sqcup \bot \Rightarrow C & R \sqcup \vartriangle \Rightarrow R & \exists \vartriangle.C \Rightarrow \bot & R\downharpoonleft\bot \Rightarrow \vartriangle \\
\vartriangle\upharpoonleft C \Rightarrow \vartriangle & \vartriangle\downharpoonleft C \Rightarrow \vartriangle & \triangledown^{-1} \Rightarrow \triangledown & \vartriangle^{-1} \Rightarrow \vartriangle \\
R^{-1^{-1}} \Rightarrow R. & & &
\end{array}$$

The simplification of an expression E is denoted by $\mathrm{smp}(E)$. Although the decision procedure we present does not require that $\mathcal{ALB}$ expressions are in simplified form, the application of simplification does have an impact on the performance of the procedure as is shown in Chapter 6.

Lemma 4.1. *Let $(\mathcal{D}, v)$ be a terminological interpretation and E be an $\mathcal{ALB}$ expression. Then $v(E) = v(\mathrm{nnf}(E))$ and $v(E) = v(\mathrm{smp}(E))$.*

The definition of the translation mapping π of concepts and roles of $\mathcal{ALB}$ to first-order formulae follows the definition of the semantics. With every concept symbol $A \in \mathsf{C}$ and every role symbol $P \in \mathsf{R}$ we uniquely associate a unary predicate symbol p_A and a binary predicate symbol p_P, respectively. Then π is defined as follows:

$$\begin{array}{rlrl}
\pi(A, X) &= p_A(X) & \pi(P, X, Y) &= p_P(X, Y) \\
\pi(\top, X) &= \top & \pi(\triangledown, X, Y) &= \top \\
\pi(\bot, X) &= \bot & \pi(\vartriangle, X, Y) &= \bot \\
\pi(\neg C, X) &= \neg\pi(C, X) & \pi(\neg R, X, Y) &= \neg\pi(R, X, Y) \\
\pi(C \sqcap D, X) &= \pi(C, X) \wedge \pi(D, X) & \pi(R \sqcap S, X, Y) &= \pi(R, X, Y) \wedge \pi(S, X, Y)
\end{array}$$

$$
\begin{aligned}
\pi(C \sqcup D, X) &= \pi(C, X) \vee \pi(D, X) & \pi(R \sqcup S, X, Y) &= \pi(R, X, Y) \vee \pi(S, X, Y)\\
\pi(\forall R.C, X) &= \forall y\colon \pi(R, X, y) \rightarrow \pi(C, y) & \pi(R{\restriction}C, X, Y) &= \pi(R, X, Y) \wedge \pi(C, X)\\
\pi(\exists R.C, X) &= \exists y\colon \pi(R, X, y) \wedge \pi(C, y) & \pi(R{\downharpoonleft}C, X, Y) &= \pi(R, X, Y) \wedge \pi(C, Y)\\
\pi((R = S), X) &= \forall y\colon \pi(R, X, y) \leftrightarrow \pi(R, X, y) & \pi(R^{-1}, X, Y) &= \pi(R, Y, X).
\end{aligned}
$$

X and Y are meta-variables for variables and constants. The translation morphism Π maps $\mathcal{ALB}$ sentences to first-order logic.

$$
\begin{aligned}
\Pi(C \mathrel{\dot\sqsubseteq} D) &= \forall x\colon \pi(C, x) \rightarrow \pi(D, x) & \Pi(R \mathrel{\dot\sqsubseteq} S) &= \forall x, y\colon \pi(R, x, y) \rightarrow \pi(S, x, y)\\
\Pi(C \doteq D) &= \forall x\colon \pi(C, x) \leftrightarrow \pi(D, x) & \Pi(R \doteq S) &= \forall x, y\colon \pi(R, x, y) \leftrightarrow \pi(S, x, y)\\
\Pi(a \in C) &= \pi(C, \underline{a}) & \Pi((a, b) \in R) &= \pi(R, \underline{a}, \underline{b}).
\end{aligned}
$$

For all sentences α, $\Pi(\alpha)$ is a closed first-order formula. Finally, the extension of Π to (finite) sets of sentences maps knowledge bases to a conjunction of first-order formulae. Note that in the absence of id and number restrictions, the unique name assumption does not affect the satisfiability of a knowledge base. Therefore, it is not necessary to incorporate formulae resulting from the translation of the unique name assumption into Π.

Lemma 4.2. *Let Γ be a knowledge base and α a sentence. Then Γ entails α if and only if $\Pi(\Gamma)$ entails $\Pi(\alpha)$ (in first-order logic).*

Proof. Straightforward. □

To transform the first-order formulae resulting from the translation of $\mathcal{ALB}$ knowledge bases to clausal form, we make use of a *structural transformation*. Note that the translation Π preserves the structure of sentences, concepts, and roles. Thus, every occurrence of a concept or role in a knowledge base Γ is associated with a position in $\Pi(\Gamma)$. Let $\mathrm{Pos}_r(\varphi)$ be the set of positions of subformulae of φ corresponding to positions of non-atomic concepts and non-atomic roles in the knowledge base Γ. By Ξ we denote the transformation taking $\Pi(\Gamma)$ to the *definitional form* $\mathrm{Def}_{\mathrm{Pos}_r(\Pi(\Gamma))}(\Pi(\Gamma))$ *of* $\Pi(\Gamma)$. We assume that the variable ordering in a literal $Q_\lambda(x, y)$ introduced by Ξ follows the convention we have used in the definition of π, that is, for a subformula like $R(x, y) \star S(x, y)$ associated with $R \star S$ and a subformula like $R(y, x)$ associated with R^{-1} we introduce $Q_\lambda(x, y)$ (not $Q_\lambda(y, x)$).

Note that it is not necessary for the following considerations that $\mathrm{Def}_\lambda(\varphi)$ depends on the polarity of φ. It is admissable to use $\mathrm{Def}_\lambda(\varphi) = \mathrm{Def}^+_\lambda(\varphi) \wedge \mathrm{Def}^-_\lambda(\varphi)$ in all cases. Recall that for identical subformulae only one new predicate symbol needs to be introduced. The decidability result we present would still apply.

We now characterise the class of clauses which are the result of translating $\mathcal{ALB}$ knowledge bases to clausal form. These clauses are *DL-clauses*. We will show that ordered resolution and ordered factoring with respect to any ordering $\succ_{\mathcal{COV}}$ compatible with a particular complexity measure will result in clauses which are again DL-clauses. Furthermore, for a finite set of predicate and function symbols, the set of (non-variant) clauses is finitely bounded. So, saturation (up to redundancy) of a set of DL-clauses is guaranteed to terminate, producing either the empty clause, or a finite saturated set not containing the empty clause.

Let C be a clause and t be compound term in C. t is called *(variable) embracing* if for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(t) \neq \emptyset$ implies $\mathcal{V}(L) \subseteq \mathcal{V}(t)$. A literal L in C is called *(variable) embracing* if

(i) for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(L) \neq \emptyset$ implies $\mathcal{V}(L') \subseteq \mathcal{V}(L)$ (that is, embracing literals contain all variables occurring in their split component of the clause), and (ii) if L contains a compound term t, then t is embracing.

Recall, that a literal L is *singular* if it contains no compound term and $\mathcal{V}(L)$ is a singleton, otherwise it is *non-singular*. A literal is *clean* if it is either a ground literal or there are no occurrences of constant symbols in it. A literal is *flat* if it is non-ground and contains no compound term.

Let Γ be a knowledge base. The first column of Table 4.1 lists all possible forms of subformulae in $\Xi\Pi(\Gamma)$ and the second column list the corresponding clausal form. Recall the definition of a regular literal from Definition 3.12. In the context of this chapter a regular literal has either no compound term arguments, or if it does, then there is a compound term argument which contains all the variables of the literal and does not itself have a compound term argument. We will show that all clauses in Table 4.1 are DL-clauses.

Definition 4.3 (DL-literals).
A literal L is a *DL-literal* iff

1. L is regular,
2. L is either monadic or dyadic and contains at most 2 variables,
3. L is ground whenever L contains a constant symbol, and
4. the maximal arity of function symbols in L is 1.

Definition 4.4 (DL-clause).
A clause C is a *DL-clause* iff

1. if C contains a compound term t, then t is embracing,
2. C is ground whenever C contains a constant symbol,
3. all literals in C are DL-literals, and
4. the argument multisets of all flat, dyadic literals coincide.

Property (1) is actually a restriction of property (3): the literals $r(x, y)$ and $r(x, f(x))$ are DL-literals, but in the clause

$$\{r(x, y), r(x, f(x))\}$$

the term $f(x)$ is not embracing.

Property (2) further refines property (3) in the presence of ground literals: the literals $p(x)$ and $q(a)$ are both DL-literals, but the clause

$$\{p(x), q(a)\}$$

is not a DL-clause, since the first literal is not ground although the clause contains a constant symbol.

Property (4) excludes clauses like $\{p(x, x), q(x, y)\}$ which do not occur in Table 4.1. The problem is that both literals are maximal with respect to any ordering which is stable under substitutions. Nevertheless, in order to avoid possibly unbounded chains of variables across literals we need to restrict resolution inferences to the literal $q(x, y)$. By contrast, clauses like

$$\{p(x, x), q(x, x)\} \qquad \text{and} \qquad \{p(x, y), q(x, y)\},$$

TBox concept definitions/restrictions and additional definitions	
$\forall x\colon p_0(x) \leftrightarrow \neg p_1(x)$	$\{\{\neg p_0(x)^*, \neg p_1(x)^*\}, \{p_0(x)^*, p_1(x)^*\}\}$
$\forall x\colon p_0(x) \leftrightarrow (p_1(x) \wedge p_2(x))$	$\{\{\neg p_0(x)^*, p_1(x)^*\}, \{\neg p_0(x)^*, p_2(x)^*\},$ $\{\neg p_1(x)^*, \neg p_2(x)^*, p_0(x)^*\}\}$
$\forall x\colon p_0(x) \leftrightarrow (p_1(x) \vee p_2(x))$	$\{\{\neg p_0(x)^*, p_1(x)^*, p_2(x)^*\}, \{\neg p_1(x)^*, p_0(x)^*\},$ $\{\neg p_2(x)^*, p_0(x)^*\}\}$
$\forall x\colon p_0(x) \leftrightarrow (\forall y\colon p_1(x,y))$	$\{\{\neg p_0(x), p_1(x,y)^*\}, \{\neg p_1(x,f(x))^*, p_0(x)\}\}$
$\forall x\colon p_0(x) \leftrightarrow (\exists y\colon p_1(x,y))$	$\{\{\neg p_0(x), p_1(x,f(x))^*\}, \{\neg p_1(x,y)^*, p_0(x)\}\}$
$\forall x\colon p_0(x) \leftrightarrow (\forall y\colon p_1(x,y) \leftrightarrow p_2(x,y))$	$\{\{\neg p_0(x), \neg p_1(x,y)^*, p_2(x,y)^*\},$ $\{\neg p_0(x), \neg p_2(x,y)^*, p_1(x,y)^*\},$ $\{\neg p_1(x,f(x))^*, \neg p_2(x,f(x))^*, p_0(x)\},$ $\{p_1(x,f(x))^*, p_2(x,f(x))^*, p_0(x)\}\}$
$\forall x\colon p_0(x) \rightarrow p_1(x)$	$\{\{\neg p_0(x)^*, p_1(x)^*\}\}$
$\forall x\colon (\neg)p_0(x)$	$\{\{(\neg)p_0(x)^*\}\}$
TBox role definitions/restrictions and additional definitions	
$\forall x,y\colon p_0(x,y) \leftrightarrow \neg p_1(x,y)$	$\{\{\neg p_0(x,y)^*, \neg p_1(x,y)^*\}, \{p_0(x,y)^*, p_1(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow (p_1(x,y) \wedge p_2(x,y))$	$\{\{\neg p_0(x,y)^*, p_1(x,y)^*\}, \{\neg p_0(x,y)^*, p_2(x,y)^*\},$ $\{\neg p_1(x,y)^*, \neg p_2(x,y)^*, p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow (p_1(x,y) \vee p_2(x,y))$	$\{\{\neg p_0(x,y)^*, p_1(x,y)^*, p_2(x,y)^*\},$ $\{\neg p_1(x,y)^*, p_0(x,y)^*\}\{\neg p_2(x,y)^*, p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow p_1(y,x)$	$\{\{\neg p_0(x,y)^*, p_1(y,x)^*\}, \{\neg p_1(y,x)^*, p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow (p_1(x,y) \wedge p_2(x))$	$\{\{\neg p_0(x,y)^*, p_1(x,y)^*\}, \{\neg p_0(x,y)^*, p_2(x)^*\},$ $\{\neg p_1(x,y)^*, \neg p_2(x)^*, p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow (p_1(x,y) \wedge p_2(y))$	$\{\{\neg p_0(x,y)^*, p_1(x,y)^*\}, \{\neg p_0(x,y)^*, p_2(y)^*\},$ $\{\neg p_1(x,y)^*, \neg p_2(y)^*, p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow p_2(x)$	$\{\{\neg p_0(x,y)^*, p_2(x)\}, \{\neg p_2(x), p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow p_2(y)$	$\{\{\neg p_0(x,y)^*, p_2(y)\}, \{\neg p_2(y), p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \leftrightarrow (p_1(x,y) \rightarrow p_2(y))$	$\{\{\neg p_0(x,y)^*, \neg p_1(x,y)^*, p_2(y)\}, \{p_0(x,y)^*, p_1(x,y)^*\},$ $\{\neg p_2(y), p_0(x,y)^*\}\}$
$\forall x,y\colon p_0(x,y) \rightarrow p_1(x,y)$	$\{\{\neg p_0(x,y)^*, p_1(x,y)^*\}\}$
$\forall x\colon (\neg)p_0(x,y)$	$\{\{(\neg)p_0(x,y)^*\}\}$
ABox	
$(\neg)A(a)$	$\{\{(\neg)A(a)^*\}\}$
$(\neg)P(a,b)$	$\{\{(\neg)P(a,b)^*\}\}$

Table 4.1: Clausal form of formulas in definitional form

are DL-clauses. Note that the clause $\{p(x,x), q(x,y)\}$ belongs to the class One-Free (which is used by Tammet to describe a decidable subclass of first-order logic into which $\mathcal{ALC}$ can be translated). It is the absence of such clauses that allows us to rely on an ordering which is stable with respect to substitution to obtain a decision procedure via ordered resolution. The major advantage is that the full power of redundancy techniques can be used in the framework we adopt.

It follows that every DL-clause is a strongly 1-regular clause. All indecomposable, non-ground DL-clauses are strongly CDV-free. Thus, we will restrict ourselves to indecomposable DL-clauses by applying the "Split" expansion rule eagerly during any derivation.

Table 4.1 contains a list of all possible subformulae of $\Xi\Pi(\Gamma)$ and the corresponding sets of clauses.

Lemma 4.5. *Let Γ be a knowledge base. Every clause in the clausal form of $\Xi\Pi(\Gamma)$ is a DL-clause.*

Proof. Straightforward. □

Lemma 4.6. *Let $t = f(t_1, \ldots, t_n)$ be a regular term. For any function symbol g occurring in $\{t_1, \ldots, t_n\}$, the arity of g is smaller than the arity of f.*

Proof. The cases 't is a constant' and 't is not a constant but g is', are easy. Suppose g is not a constant symbol, that is, there exists a strict subterm s of t of the form $s = g(s_1, \ldots, s_m)$ with $m > 0$. Since t is regular, t dominates s. That means $s_1 = t_1, \ldots, s_m = t_m$ and $m \leq n$ holds. Suppose $m = n$ holds. Since s is a subterm of some t_i, $1 \leq i \leq n$, and t_i is a strict subterm of s, we have to conclude that t_i is a strict subterm of itself. Of course, this is impossible. Hence $m < n$. □

Lemma 4.7. *Let L be a regular literal and σ a substitution such that*

- *$L\sigma$ is regular and*
- *the maximal arity of all function symbols in $L\sigma$ is 1.*

Then the depth of $L\sigma$ is less than or equal to 3.

Proof. All argument terms of regular literals are regular terms. According to Lemma 4.6, if a term $g(s_1, \ldots, s_m)$ is a strict subterm of some regular term $t = f(t_1, \ldots, t_n)$, then $\mathrm{arity}(g) < \mathrm{arity}(f)$. But as any function symbol has either arity 0 or 1, the arguments of any compound term t are either constants or variables. This implies $\mathrm{dp}(L\sigma) \leq 3$. □

This lemma includes the case that σ is the identity substitution.

Lemma 4.8. *The depth of an indecomposable DL-clause C is less than or equal to 3 and the number of variables in C is less than or equal to 2.*

Proof. By Lemma 4.7 the depth of a DL-clause is less than or equal to 3. It remains to exhibit the bound on the number of variables in C. If C is ground, then the lemma holds trivially. If C is non-ground, then it does not contain any constant symbol. Therefore, every literal in C is non-ground. If C contains a compound term t, then by $t = f(x)$ and x is the only variable in C, since t is embracing. Suppose C does not contain a compound term. If C contains only monadic literals, then these literals are flat and contain exactly one variable. So, C contains one variable. Finally, if C contains dyadic literals, then these literals are flat and the multiset of arguments coincide. That is, there are variables x and y and for all dyadic literals L in C we have $\mathcal{V}(L) = \{x, y\}$. Any monadic literal L' in C either has the argument x or the argument y. Thus, C contains two variables. $\square$

Lemma 4.8 also holds if we consider condensed clauses instead of indecomposable clauses.

Corollary 4.9. *Over a finite signature there are only a finitely bounded number of indecomposable DL-clauses (modulo variable renaming).*

4.3 Resolution and factoring on DL-clauses

Next we investigate the closure of DL-clauses under resolution and factoring. It is obvious that using unrefined resolution the resolvent of two DL-clauses is not a DL-clause in general. The size and depths of derived clauses is not bounded. Since the class of DL-clauses is a subclass of $\overline{\mathrm{KC}}$, we could make use of the results obtained in Chapter 3. However, the additional properties of DL-clauses enable us to show that a more general class of ordering refinements provide a decision procedure for the class of DL-clauses.

For every literal L, let the complexity measure c_L be the multiset of arguments of L. We define the *strict subterm ordering* $\succ^s$ on terms by $s \succ^s t$ if and only if t is a strict subterm of s. The relation $\succ^s$ is a partial ordering on terms. In addition, $\succ^s$ is stable with respect to substitutions. We compare complexity measures by the multiset extension of the strict subterm ordering $\succ^s_{mul}$. The ordering $\succ_{\mathcal{COV}}$ is any admissible ordering compatible with the ordering $\succ^s_{mul}$ on the complexity measure c_L. In Table 4.1 all literals marked by $\_^*$ are potentially maximal.

Lemma 4.10. *Let $C = \{L_1\} \cup D$ be an indecomposable DL-clause. If L_1 is $\succ_{\mathcal{COV}}$-maximal with respect to D and contains no compound term, then no literal in D contains a compound term.*

Proof. Since L_1 does not contain a compound term, the arguments of L_1 are either constants or variables. If L_1 is ground, then the set D is empty and the lemma is trivially true. Otherwise all the arguments of L_1 are variables. Now suppose there is some literal $L_2 \in D$ containing a compound term $t_2 = f(x_2)$. Since C is a DL-clause, $f(x_2)$ is embracing, that is, all variables of L_1 are subterms of t_2. Consequently, $L_2 \succ_{\mathcal{COV}} L_1$, contradicting the maximality of L_1. $\square$

Lemma 4.11. *Let $C = \{L_1\} \cup D$ be an indecomposable DL-clause. If L_1 is a flat, monadic literal which is $\succ_{\mathcal{COV}}$-maximal with respect to D, then there is no (flat) dyadic literal in D.*

Proof. The literal L_1 has the form $p(x)$ where x is a variable. By the previous lemma, D contains no compound term. Suppose there is a flat, dyadic literal in D, that is, there is a literal L_2 of the form $q(x_1, x_2)$ where at least one of the x_i is a variable. Since C is an indecomposable DL-clause, at least one of x_1 and x_2 is identical to x, that is, $\arg_{mul}(L_1) \subset \arg_{mul}(L_2)$ holds. Consequently, we have $L_2 \succ_{\mathcal{COV}} L_1$, contradicting the assumption that L_1 is maximal. $\square$

Lemma 4.12. *Let $C = \{L_1\} \cup D$ be an indecomposable DL-clause. If L_1 is $\succ_{\mathcal{COV}}$-maximal with respect D, then L_1 is $\succ_Z$-maximal with respect to D.*

Proof. In essence by the definition of $\succ_{\mathcal{COV}}$. If L_1 is a ground literal, then C is a singleton set and the lemma is obviously true. In the following we assume that C contains no constants. Suppose there is a literal L_2 in D. It is straightforward to check that the only possibilities for $L_2 \succ_Z L_1$ to hold are:

1. $\arg_{set}(L_2) = \{x, y\}$ and $\arg_{set}(L_1) = \{x\}$ for variables x and y. We have to distinguish two cases: Either L_1 has the form $p(x)$ or the form $p(x, x)$ for some predicate symbol p. In the first case $L_2 \succ_{\mathcal{COV}} L_1$ holds, contradicting our assumption that L_1 is maximal with respect to $\succ_{\mathcal{COV}}$. In the second case L_1 and L_2 are both flat, dyadic literals, but $\arg_{mul}(L_2) \neq \arg_{mul}(L_1)$ contradicting that C is a DL-clause.

2. $\arg_{set}(L_2) \supset \{f(x)\}$ and $\arg_{set}(L_1) = \{x\}$ for some function symbol f and variable x. So, L_2 contains an argument which is greater than any argument of L_1 with respect to the strict subterm ordering. This contradicts that L_1 is maximal with respect to $\succ_{\mathcal{COV}}$. □

Note that $\succ_{\mathcal{COV}}$ is a strict refinement of $\succ_Z$. For example, in the clause

$$\{p(x, f(x)), q(f(x))\}$$

both literals are maximal with respect to $\succ_Z$, but only the first literal is maximal with respect to $\succ_{\mathcal{COV}}$.

Lemma 4.12 shows that $\succ_{\mathcal{COV}}$ satisfies Condition 3.3. By Theorem 3.33 this is already sufficient to show that ordered resolution and ordered factoring based on $\succ_{\mathcal{COV}}$ provides a decision procedure for the class of DL-clause. However, Theorem 3.33 does not ensure that the conclusion of an inference step by ordered resolution of two DL-clauses is again a DL-clause. This will be our concern in the remainder of this section.

We know that every indecomposable, 1-regular clause C contains a literal L which dominates (with respect to $\succ_Z$) all the literals in C. In particular, L contains all the variables of C. It follows by Lemma 4.12, that all $\succ_{\mathcal{COV}}$-maximal literals in an indecomposable DL-clause C contain all the variables occurring in C.

Corollary 4.13. *Let $C = \{L\} \cup D$ be an indecomposable DL-clause such that L is maximal in C with respect to $\succ_{\mathcal{COV}}$. Then $\mathcal{V}(L) = \mathcal{V}(C)$.*

Lemma 4.14. *Let $C = \{L_1, L_2\} \cup D$ be an indecomposable DL-clause such that L_1 and L_2 are unifiable with most general unifier σ. The split components of $(\{L_1\} \cup D)\sigma$ are DL-clauses.*

Proof. Since C is not a unit clause, it contains at least two literals and it contains no constant symbols. Let E be a split component of $(\{L_1\} \cup D)\sigma$. We show that E satisfies the properties (1)–(4) of Definition 4.4:

1. By Lemma 3.20 a factor of a 1-regular clause is again 1-regular. By Lemma 3.17(3) and the fact that the arity of function symbols is at most 1, we obtain that any compound term is embracing in E.

2. Since C contains no constant symbols, σ will not introduce any constant symbols. Thus, E does not contain any constant symbols.

3. We have to show that all literals in E are DL-literals. Because E is a split component of a factor of C, we have $|\mathcal{V}(E)| \leq |\mathcal{V}(C)| \leq 2$, and E contains no predicate symbols or function symbols which do not already occur in C. Since E is 1-regular, all literals in E are regular.

4. Finally, we must show that all flat, dyadic literals in E contain the same multiset of arguments. Suppose $L_3\sigma$ and $L_4\sigma$ are flat, dyadic literals in E. We know that the multiset of arguments of L_3 and L_4 are equal. Thus, the multisets of arguments of $L_3\sigma$ and $L_4\sigma$ are equal as well. □

Lemma 4.15. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be two variable-disjoint, indecomposable DL-clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let $A_1\sigma$ and $\neg A_2\sigma$ be $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$ and $D_2\sigma$, respectively. The split components of $(D_1 \cup D_2)\sigma$ are DL-clauses.*

Proof. Let E be a split component of $(D_1 \cup D_2)\sigma$. We show that E satisfies the properties (1)–(4) of Definition 4.4:

1. By Lemma 4.12, A_1 and $\neg A_2$ are $\succ_Z$-maximal in C_1 and C_2, respectively. By Lemma 3.17(3) and the fact that the arity of function symbols is at most 1, we obtain that any compound term is embracing in E.

2. By Lemma 4.13, A_1 and $\neg A_2$ contain all variables occurring in C_1 and C_2, respectively. Suppose one of A_1 or $\neg A_2$ is a ground literal. Then $A_1\sigma$ and $\neg A_2\sigma$ as well as $D_1\sigma$ and $D_2\sigma$ are ground. The split component E contains a single ground literal. Otherwise, neither C_1 and C_2 contains a constant. Consequently, E will not contain a constant.

3. The proof that E satisfies property (3) is analogous to case 3 of Lemma 4.14.

4. Finally, we have to show that all flat, dyadic literals in E contain the same multiset of arguments. If either A_1 or $\neg A_2$ is a ground literal, then E is a ground clause and the condition is trivially satisfied. Depending on whether A_1 or $\neg A_2$ contain a compound term we distinguish the following cases:

   (a) Suppose neither A_1 nor $\neg A_2$ contains a compound term, that is, all the arguments of A_1 and $\neg A_2$ are variables. Without loss of generality we can assume that $|\mathcal{V}(A_1)| \geq |\mathcal{V}(\neg A_2)|$ holds. It is straightforward to check that we can also assume for σ that $\mathrm{D}(\sigma) = \mathcal{V}(A_1)$ and $\mathrm{C}(\sigma) = \mathcal{V}(\neg A_2)$ holds. Thus, $A_1\sigma$ is identical to A_2.

   Suppose $A_1\sigma = A_2$ is a flat, dyadic literal. Then A_1 is a flat, dyadic literal as well. Let L_3 be a flat, dyadic literal in D_1. L_3 contains the same arguments as A_1, so $L_3\sigma$ contains the same arguments as $A_1\sigma$, $\neg A_2\sigma$, and any flat dyadic literal in $D_2\sigma = D_2$.

   Suppose $\neg A_2$ and A_1 are both monadic literals. According to Lemma 4.11, there can be no flat, dyadic literals in D_1 and D_2. So, there are no flat, dyadic literals in E.

   (b) Suppose only one of A_1 or $\neg A_2$ contains a compound term. Without loss of generality we assume that A_1 contains a compound term, that is, A_1 contains a functional term of the form $f(x)$ where x is a variable. x is the only variable occurring in C_1. Again, it is not difficult to verify that $\mathcal{V}(\mathrm{C}(\sigma)) = \{x\}$ holds. Hence, x is the only variable occurring in E and the multiset of arguments of any flat, dyadic literal in E is $\{x, x\}$.

   (c) Suppose A_1 and $\neg A_2$ each contain a compound term, that is, $A_1 = p(s_1, \dots, s_n)$, $n \leq 2$, and there is some i, $1 \leq i \leq n$, such that $s_i = f(x_i)$, $A_2 = p(t_1, \dots, t_n)$, $n \leq 2$, and there is some j, $1 \leq j \leq n$, such that $t_j = g(y_j)$. Since A_1 and A_2 are unifiable, regular literals, we know that $i = j$ and $f = g$ holds. Assume that σ has the form $\{y_i/x_i\}$. Since x_i is also the only variable occurring in $(D_1 \cup D_2)\sigma$, the multiset of arguments of any flat, dyadic literal in E is $\{x_i, x_i\}$. □

Theorem 4.16.
Let Γ be an $\mathcal{ALB}$ knowledge base and N be the clausal form of $\Xi\Pi(\Gamma)$. Then any derivation from N by ordered resolution and (ordered) factoring based on $\succ_{\mathcal{COV}}$ terminates.

Proof. Lemma 4.15 states that if we apply resolution to $\succ_{\mathcal{COV}}$-maximal literals only and split the resulting clauses, then the components of any resolvent of two indecomposable, DL-clauses are DL-clauses also. Lemma 4.14 states the same for factorisation. Therefore, any split component of a clause derivable from N will be a DL-clause. Since the "Split" expansion rule is applied eagerly before any further application of "Deduce", any decomposable conclusion of an inference step will be decomposed into its split components before further application of inference rules. Therefore, before an application of "Deduce" any set N' of clauses in the theorem proving derivation consists only of DL-clauses.

By Corollary 4.9 there is only a finitely bounded number of indecomposable DL-clauses modulo variable renaming. Since "Delete" is applied to the clause set to eliminate variant clauses, this means any derivation is finitely bounded. □

4.4 Variations of $\mathcal{ALB}$

The notion of DL-literals is quite liberal compared to which kind of literals actually occur in Table 4.1. For example, $p(x, x)$ and $q(f(x), g(x))$ are DL-literals. But in none of the literals in Table 4.1 does a variable occur twice in a flat literal nor do two compound terms occur in one literal. If such literals occur in a $\succ_{\mathcal{COV}}$-derivation, they are generated by inference steps. For example, a $\succ_{\mathcal{COV}}$-factor of the clause

$$\{p(x, y), p(y, x), q(x, y)\}$$

is

$$\{p(x, x), q(x, x)\}.$$

To determine the cases in which such clauses are generated, we take a closer look at part (4) of the proof of Lemma 4.14.

Lemma 4.17. *In the absence of the role converse operator in a knowledge base, the factoring substitution will always be the identity substitution and flat, singular, dyadic literals do not occur. Factoring can be reduced to detecting and (eagerly) deleting duplicate occurrences of literals in clauses.*

Proof. We analyse the relationship between the most general unifier σ and the literals L_1 and L_2 in a factoring inference step. The cases are:

1. Suppose neither L_1 nor L_2 contains a compound term, that is, all the arguments of L_1 and L_2 are variables. We split this case into three further cases:

   (a) Suppose L_1 is a non-singular, flat, dyadic literal, that is, L_1 has the form $p(x, y)$ for some predicate symbol p and variables x, y. Since all flat, dyadic literals contain the same multiset of arguments, L_2 has either the form $p(x, y)$ or $p(y, x)$. In the first case, σ is the identity substitution and E is identical to $\{L_1\} \cup D$. In the second case, we can assume that σ has the form $\{y/x\}$. All the flat, dyadic literals in E have the form $q(x, x)$ where q is some predicate symbol.

(b) Suppose L_1 is a singular, flat, dyadic literal, that is, both L_1 and L_2 have the form $p(x, x)$ for some predicate symbol p and variable x. Then σ is the identity substitution.

(c) Finally, suppose L_1 is a monadic literal. Again L_2 has to be a monadic literal. Since L_1 and L_2 are maximal in C, there is no dyadic literal in C (due to Lemma 4.11). So L_1 and L_2 share the same variable and σ is the identity substitution.

2. Suppose only one of L_1 or L_2 contains a compound term. We show that this assumption leads to a contradiction. Without loss of generality assume that L_1 contains a compound term. The literal L_1 contains a functional term of the form $f(x)$ where x is a variable. The variable x is the only variable occurring in C. Since $x\sigma$ can neither be a compound term (due to Lemma 4.7), a constant (due to the absence of constants in C), nor a variable different to x, σ is the identity substitution. As a consequence $L_2\sigma = L_2$ does not contain a compound term and is not identical to $L_1\sigma = L_1$.

3. Suppose L_1 and L_2 each contain a compound term. Again it is straightforward to check that σ has to be the identity substitution.

So, in all but one case the substitution σ is the identity substitution. Suppose a clause C does not contain literals of the form $q(x, x)$. A $\succ_{\mathcal{COV}}$-factor of C will contain such a literal if and only if C contains literals of the form $p(x, y)$ and $p(y, x)$ for some predicate symbol p and variables x, y. For all but two clauses in Table 4.1, the flat, dyadic literals of a clause do not only contain the same multiset of argument, but identical terms occur at the same argument position in all literals. It straightforward to check that the only two clauses violating this principle are the result of the translation of a role term of the form R^{-1}. $\square$

This motivates the introduction of the notion of fluted DL-literals and clauses. The next definitions differ from Definitions 4.3 and 4.4 in (5), and (3) and (4), respectively.

Definition 4.18 (Fluted DL-literal).
A literal L is a *fluted DL-literal* iff

1. L is regular,
2. L is either monadic or dyadic and contains at most 2 variables,
3. L is ground whenever L contains a constant symbol,
4. the maximal arity of functions symbols in L is 1, and
5. there is at most one compound term t in L and t can only occur in the last argument position of L.

Definition 4.19 (Fluted DL-clause).
A clause C is a *fluted DL-clause* iff

1. C is a 1-regular clause of grade k where $k \leq 2$ holds,
2. C is ground whenever C contains a constant symbol,
3. all literals in C are fluted DL-literals, and
4. there exist distinct variables x and y such that all flat, dyadic literals in C are of the form $p(x, y)$ for some predicate symbol p.

Except for the clauses which originate from the translation of converse role expressions the clauses in Table 4.1 are fluted DL-clauses.

Preservation results similar to Lemma 4.14 and Lemma 4.15 can be shown for fluted DL-clauses.

Lemma 4.20. *Let $C = \{L_1, L_2\} \cup D$ be an indecomposable, fluted DL-clause such that L_1 and L_2 are unifiable with most general unifier σ, and let $L_1\sigma$ be $\succ_{\mathcal{COV}}$-maximal with respect to $D\sigma$. Then $(\{L_1\} \cup D)\sigma$ is a strict subclause of C and again a fluted DL-clause.*

Proof. The first three items of the proof of Lemma 4.14 are still valid. Two additional items need to be shown: (i) there are distinct variables x and y such that all flat, dyadic literals in D have the form $r(x, y)$ for some predicate symbol r, and (ii) compound terms occur only in the last argument position of a literal. Depending on whether L_1 or L_2 contain a compound term we distinguish the following cases:

1. Suppose neither L_1 nor L_2 contains a compound term, that is, all the arguments of L_1 and L_2 are variables.

   Suppose L_1 is a non-singular, flat, dyadic literal, that is, L_1 has the form $p(x, y)$ for some predicate symbol p and variables x, y. So, L_2 is identical to L_1, σ is the identity substitution, E is identical to $\{L_1\} \cup D$.

   Suppose L_1 and L_2 are monadic literals. Since L_1 and L_2 are maximal in C, there is no dyadic literal in C (due to Lemma 4.11). L_1 and L_2 share the same variable and σ is the identity substitution.

2. Suppose only one of L_1 or L_2 contains a compound term. As before, this leads to a contradiction.

3. Suppose L_1 and L_2 each contain a compound term. Again it is straightforward to check that σ has to be the identity substitution. □

Note that it is essential that we use ordered factoring in Lemma 4.20. For example, the clause $\{p(x), r(x, x)\}$ derived from $\{p(x), r(x, y), p(y)\}$ by a factoring inference step on the atoms $p(x)$ and $p(y)$ whose common instance is not $\succ_{\mathcal{COV}}$-maximal is not fluted.

Corollary 4.21. *Let C_2 be an ordered factor of an indecomposable, fluted DL-clause C_1. Then there exists a condensation of C_1 that subsumes C_2.*

Lemma 4.22. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, fluted DL-clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let $A_1\sigma$ and $\neg A_2\sigma$ be $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$ and $D_2\sigma$, respectively. Then the split components of $(D_1 \cup D_2)\sigma$ are fluted DL-clauses.*

Proof. The first three items of the proof of Lemma 4.15 remain valid. As for the previous lemma, we still have to prove: (i) there are distinct variables x and y such that all flat, dyadic literals in D have the form $r(x, y)$ for some predicate symbol r, and (ii) compound terms occur only at the final argument position of a literal. Depending on whether A_1 or A_2 contain a compound term we distinguish the following cases:

1. Suppose all the arguments of A_1 and A_2 are variables. Without loss of generality assume that $|\mathcal{V}(A_1)| \geq |\mathcal{V}(\neg A_2)|$ and $A_1\sigma = A_2\sigma = A_2$.

   Suppose $A_2\sigma = A_2$ is a flat, dyadic literal. So is A_1. Let L_3 be a flat, dyadic literal in D_1. L_3 differs from A_1 only in the predicate symbol, so the same holds for $L_3\sigma$ and $A_1\sigma$.

   Suppose $A_2\sigma = A_2$ is not a dyadic literal. So, both A_1 and $\neg A_2$ are monadic literals. By Lemma 4.11, D_1 and D_2 do not contain flat, dyadic literals in D_1 and D_2. Hence, neither does D.

2. Suppose only one of A_1 or $\neg A_2$ contains a compound term. Assume that A_1 contains a compound term of the form $f(x)$. The variable x is the only variable occurring in C_1. Again it is straightforward to verify that x is the only variable occurring in D.

   Suppose A_2 is a monadic literal. By Lemma 4.11, there are no flat, dyadic literals in D_2. So there are no flat, dyadic literals in $D_2\sigma$. Hence the flat, dyadic literals in $D_1\sigma$ remain unchanged.

   Suppose A_2 is a dyadic literal and A_2 has the form $p(y, z)$ for distinct variables y and z. Then A_1 has the form $p(x, f(x))$. Thus, $\sigma = \{y/x, z/f(x)\}$ and all instances $L_3\sigma$ of flat, dyadic literals L_3 are identical to A_1. Thus, there are no flat, dyadic literals in $D_2\sigma$, while the flat, dyadic literals in $D_1\sigma$ remain unchanged.

3. Suppose A_1 and A_2 each contain a compound term, that is, A_1 has the form $p(x, f(x))$ and A_2 has the form $p(y, f(y))$. x is the only variable occurring in C_1 and y is the only variable occurring in C_2. This implies there are no flat, dyadic literals in C_1 and C_2. As a consequence there are no flat, dyadic literals in D. □

Theorem 4.23.
Let Γ be a $\mathcal{ALB}$ knowledge base such that no role converse operator occurs in Γ and let N be the clausal form of $\Xi\Pi(\Gamma)$. Then any derivation from N by ordered resolution based on $\succ_{\mathcal{COV}}$ (without factoring) terminates.

Proof. If Γ contains no role converse operator, then all clauses in the clausal form N of Γ are fluted DL-clauses. Lemma 4.20 and Lemma 4.22 show that all $\succ_{\mathcal{COV}}$-factors and $\succ_{\mathcal{COV}}$-resolvents derived from N are again fluted DL-clauses. Corollary 4.21 shows we can replace the inference rule of ordered factoring by condensation. However, condensation is already an integral part of the "Deduce" expansion rule. The rest of the proof is as for Theorem 4.16. □

We now take a closer look at those concept and role forming operators of $\mathcal{U}$ that have been excluded from $\mathcal{ALB}$. All concepts, roles, and bindings formed using operators of $\mathcal{U}$ not included in $\mathcal{ALB}$ except for role closure can be translated to first-order logic as follows.

$$
\begin{aligned}
\pi(\exists_{\geq n} R, X) &= \exists y_1, \dots y_n : y_1 \not\approx y_2 \wedge \dots \wedge y_{n-1} \not\approx y_n \wedge R(X, y_1) \wedge \dots \wedge R(X, y_n) \\
\pi(\exists_{\leq n} R, X) &= \forall y_1, \dots, y_{n+1} : R(X, y_1) \wedge \dots \wedge R(X, y_{n+1}) \rightarrow y_1 \approx y_2 \vee \dots \vee y_n \approx y_{n+1} \\
\pi(\exists_{\geq n} R.C, X) &= \pi(\exists_{\geq n}(R{\restriction}C), X) \\
\pi(\exists_{\leq n} R.C, X) &= \pi(\exists_{\leq n}(R{\restriction}C), X) \\
\pi(\exists_{\mathsf{B}} U{:}C, X) &= \exists y : \pi(U, X, y) \wedge \pi(C, y) \\
\pi(\mathsf{id}, X, Y) &= (X \approx Y)
\end{aligned}
$$

Positive occurrences of $R \circ S$	$\{\{\neg p_0(x,y), p_R(x, f(x,y))\}, \{\neg p_0(x,y), p_S(f(x,y),y)\}\}$
Negative occurrences of $R \circ S$	$\{\{\neg p_R(x,z), \neg p_S(z,y), p_0(x,y)\}\}$
Positive occurrences of $(\subseteq U\, V)$	$\{\{\neg p_0(x,y), \neg p_U(x,z), p_V(y,z)\}\}$
Negative occurrences of $(\subseteq U\, V)$	$\{\{p_U(x, f(x,y)), p_0(x,y)\}, \{\neg p_V(y, f(x,y)), p_0(x,y)\}\}$
Positive occurrences of $(\supseteq U\, V)$	$\{\{\neg p_0(x,y), \neg p_U(y,z), p_V(x,z)\}\}$
Negative occurrences of $(\supseteq U\, V)$	$\{\{p_V(y, f(x,y)), p_0(x,y)\}, \{\neg p_U(x, f(x,y)), p_0(x,y)\}\}$

Table 4.2: Clausal form role composition and binding forming operators

$$\begin{aligned}
\pi(R \circ S, X, Y) &= \exists z : \pi(R, X, z) \wedge \pi(S, z, Y) \\
\pi(\subseteq R\, S, X, Y) &= \forall z : \pi(R, X, z) \rightarrow \pi(S, Y, z) \\
\pi(\supseteq R\, S, X, Y) &= \forall z : \pi(S, Y, z) \rightarrow \pi(R, X, z)
\end{aligned}$$

In the presence of id or number restrictions, we have to modify the translation Π to reflect the unique name assumption. For a knowledge base Γ, the translation Π maps Γ to

$$\bigwedge_{a \not\approx b \in \mathsf{O}} a \not\approx b \wedge \bigwedge_{\alpha \in \Gamma} \Pi(\alpha).$$

The introduction of equality reasoning into our calculus goes beyond the techniques developed in the previous chapters and sections.

However, it is possible to deal with knowledge bases containing negative occurrences of id only. Bachmair, Ganzinger, and Voronkov [14] present a modification method for the elimination of equality by the transformation of clauses. It follows from their results that, in our particular case, the reflexivity of '$\approx$' is the only property of equality we need to obtain a complete calculus, since (i) terms in the clausal form of $\Xi\Pi(\Gamma)$ are flat, (ii) there are only negative occurrences of equality atoms, (iii) the right-hand side of an equality atom is always a variable. Therefore, transformations of the clauses to accommodate for monotonicity, symmetry, and transitivity of equality are not required. Thus, it is sufficient to modify the translation Π to incorporate $\forall x : x \approx x$ without adding further inference rules like superposition to the calculus. The clauses in the clausal form of $\Xi\Pi(\Gamma)$ are again DL-clauses. For example,

$$\{\neg p_i(x), x \not\approx y, \neg r(x,y), q(y)\}$$

reduces to

$$\{\neg p_i(x), \neg r(x,x), q(y)\}$$

by an ordinary resolution inference step.

As far as role composition is concerned, we see in Table 4.2 that negative occurrences of $R \circ S$ lead to the introduction of clauses that do not contain an embracing literal. Similarly, for positive occurrences of $(\subseteq U\,V)$ and $(\supseteq U\,V)$. Such clauses do not belong to any of the decidable classes we have considered thus far. We will reconsider these cases in Section 4.5.

Positive occurrences of $R \circ S$, and negative occurrences of $(\subseteq U\,V)$, and $(\supseteq U\,V)$ introduce clauses which are 2-regular with at most two variables. It is straightforward to modify Definition 4.3 to allow for function symbols of arity 2 and to modify property (1) of Definition 4.4 to allow for 2-regular clauses with at most two variables. All the lemmata of Section 4.3 still hold for this extended class of DL-clauses. For 2-regular clauses with more than one variable, a resolution inference step between clauses of depth 3, like

$$\{\neg p(f(x_1, x_2), x_2), q(f(x_1, x_2), x_2)\} \qquad \text{and} \qquad \{p(f(y_1, y_2), g(y_1))\}$$

may produce a clause

$$\{q(f(x_1, g(x_1)), g(x_1))\}$$

of depth greater than 3. Such a case cannot occur for the extended class of DL-clauses. Since unary and binary function symbols never occur together in a clause of the clausal form of $\Xi\Pi(\Gamma)$, we can consider all clauses containing a unary function symbol as 1-regular and all clauses containing a binary function symbol as 2-regular. Based on Corollary 3.43 we conclude that no resolution inference steps between 1-regular and 2-regular clauses are possible. Consequently, we are not able to derive clauses which contain a unary as well as a binary function symbol which would be necessary for the generation of a 2-regular clause of depth greater than 3.

Let $\mathcal{U}^-$ be the reduct of $\mathcal{U}$ without number restrictions, qualified number restrictions, and role closure.

Theorem 4.24.
Let Γ be a knowledge base over $\mathcal{U}^-$ such that id *and expressions of the form $(\supseteq U\,V)$ and $(\subseteq U\,V)$ occur only negatively, and expressions of the form $(R \circ S)$ occur only positively. Let N be the clausal form of $\Xi\Pi(\Gamma)$. Then any derivation from N by ordered resolution and ordered factoring based on $\succ_{cov}$ terminates.*

4.5 A decision procedure for $\mathcal{ALB}_{\mathsf{D}}$ based on selection

In this section we describe an alternative decision procedure for a reduct of $\mathcal{ALB}$. The procedure is based on a particular selection function while there is no restriction on the ordering we use. The derivations are in essence exactly as for tableaux-based approaches. However, compared to tableaux-based approaches the procedure has the advantage that (i) it provides more flexibility concerning the theorem proving strategy, and (ii) it allows the application of general redundancy criteria.

We focus on descriptive knowledge bases Γ over the reduct of $\mathcal{ALB}$ without role complement, role value maps, and top role. We call this reduct $\mathcal{ALB}_{\mathsf{D}}$. An extension of this approach to general knowledge bases is described in [84].

We assume, all expressions occurring in a knowledge base Γ are in negation normal form.

As only negative literals can be selected, it is necessary to transform the given knowledge base. Formally, let $\mathrm{D}_{\doteq}(\Gamma)$ denote the set of symbols $S_0 \in \mathsf{C} \cup \mathsf{R}$ such that Γ contains a terminological sentence $S_0 \doteq E$. We obtain the knowledge base $\overline{\Gamma}$ over $(\mathsf{O}, \overline{\mathsf{C}}, \mathsf{R})$ in the following way. Extend C to $\overline{\mathsf{C}}$ by adding a concept symbol $\overline{A}$ for every concept symbol A in $\mathrm{D}_{\doteq}(\Gamma)$ and role symbols P^u

and P^d for every role symbol P in $\mathrm{D}_{\doteq}(\Gamma)$. We obtain $\overline{\Gamma}$ from Γ by the following transformation steps:

1. Replace concept definitions $A \doteq C$, by $A \dot{\sqsubseteq} C$ and $\neg A \dot{\sqsubseteq} \neg C$, and replace role definitions $P \doteq R$, by $P^d \dot{\sqsubseteq} R$ and $R \dot{\sqsubseteq} P^u$.

2. Replace every occurrence of a concept $\neg A$, for A in $\mathrm{D}_{\doteq}(\Gamma)$, by $\overline{A}$.

3. Replace every positive occurrence of a role symbol $P \in \mathrm{D}_{\doteq}(\Gamma)$ by P^d, and every negative occurrence of P by P^u.

4. For every concept symbol A in $\mathrm{D}_{\doteq}(\Gamma)$, add the terminological sentence $A \dot{\sqsubseteq} \neg\overline{A}$ and add for every role symbol P in $\mathrm{D}_{\doteq}(\Gamma)$, the terminological sentence $P^d \dot{\sqsubseteq} P^u$.

5. Turn all concepts and roles in the resulting knowledge base into negation normal form.

Lemma 4.25. *The knowledge base Γ is satisfiable if and only if $\overline{\Gamma}$ is satisfiable.*

Proof. The proof is in two steps, twice exploiting the preservation of satisfiability and unsatisfiability by structural renaming.

For any sentence $S_0 \doteq E$ we have $v(S_0) = v(E)$ for every interpretation $(\mathcal{D}, v)$. Therefore, we can replace S_0 on the right hand side of a terminological or assertional sentence in Γ by E without affecting its satisfiability or unsatisfiability. The acyclicity of descriptive knowledge bases ensures that by performing these replacements for all symbols in $\mathrm{D}_{\doteq}(\Gamma)$ we eventually obtain a transformed knowledge base Γ_1. No symbol in $\mathrm{D}_{\doteq}(\Gamma)$ occurs on the right hand side of any terminological or assertional sentence of Γ_1. Let $S_0 \doteq E$ be in Γ_1 with S is in $\mathrm{D}_{\doteq}(\Gamma)$. Suppose $\Gamma_2' = \Gamma_1 \setminus \{S_0 \doteq D\}$ where D is the expression resulting from unfolding concept and role definitions is satisfiable by $(\mathcal{D}, v_2')$. As Γ_2' contains no occurrence of the concept symbol S, $v(S)$ is either undefined or has no effect on the satisfiability of Γ_2'. Define an interpretation $(\mathcal{D}, v_1)$ for Γ_1 by $v_1(S_1) = v_2'(S_1)$ for all symbols $S_1 \neq S_0$ and $v_1(S_0) = v_2'(E)$. Thus, Γ is satisfiable if the knowledge base Γ_2 obtained from Γ_1 by eliminating all terminological axioms $S \doteq E$ such that S is in $\mathrm{D}_{\doteq}(\Gamma)$ are satisfiable.

In the transformation of Γ_1 to Γ_2 we may have eliminated two terminological sentences $S_0 \dot{\sqsubseteq} E$ and $S_1 \dot{\sqsubseteq} E$ with the same symbol on the right hand side. In the following we distinguish between occurrences of E introduced by unfolding with respect to $S_0 \dot{\sqsubseteq} E$ and those introduced by unfolding with respect to $S_1 \dot{\sqsubseteq} E$.

Consider a concept D introduced into the knowledge base by unfolding a terminological sentence $A \doteq C$ (D need not be identical to C). Using the same renaming techniques used for first-order formulae, we can replace all positive occurrences of D in the knowledge base by A and add a concept restriction $A \dot{\sqsubseteq} D$ to the knowledge base. Occurrences of $\neg D$ in Γ_2 introduced by unfolding $A \doteq C$ are replaced by a new concept symbol $\overline{A}$. We add a concept restriction $\overline{A} \dot{\sqsubseteq} \neg D$ to the knowledge base. We proceed until all (sub)concept introduced by the transformation of Γ_1 into Γ_2 have been replaced. In the case of role definitions $P \doteq R$ the procedure is slightly different. Consider a role S introduced into the knowledge base by unfolding a sentence $P \doteq R$. We replace all positive occurrences of S by P^d and add a sentence $P^d \dot{\sqsubseteq} R$ to the knowledge base. We replace all negative occurrences of S by P^u and add a sentence $R \dot{\sqsubseteq} P^u$. The resulting knowledge base is Γ_3. Again it is obvious that Γ_3 is satisfiable if and only if Γ_2 is satisfiable.

No element d of the domain of a potential interpretation $(\mathcal{D}, v_3)$ of Γ_3 can be both in $v_3(A)$ and $v_3(\overline{A})$, since $d \in v_3(A)$ implies $d \in v_3(D)$ and $d \in v_3(\overline{A})$ implies $d \in v_3(\neg D) = \mathcal{D} \setminus v_3(D)$. Thus, we may add the terminological sentence $A \dot{\sqsubseteq} \neg\overline{A}$ stating the disjointness of A and $\overline{A}$ to our knowledge base without affecting its satisfiability. Likewise, any pair (d, e) which is an element of $v_3(P^d)$ in a potential interpretation $(\mathcal{D}, v_3)$ of Γ_3 is also an element of $v_3(R)$ due to $P^d \dot{\sqsubseteq} R$, and therefore an element of $v_3(P^u)$ due to $R \dot{\sqsubseteq} P^u$. Thus, we may add the terminological sentence $P^d \dot{\sqsubseteq} P^u$ without affecting its satisfiability.

Finally, we can transform any concept and role occurring in the knowledge base to its negation normal form. The resulting knowledge base Γ_4 obtained coincides with $\overline{\Gamma}$. □

The translation of a knowledge base into first-order logic is as described in Section 4.2. The definitional form is produced by a variant $\overline{\Xi}$ of the transformation Ξ described in Section 4.2. First, $\overline{\Xi}$ uses *definitions* $\mathrm{Def}^+_\lambda(\varphi)$ and $\mathrm{Def}^-_\lambda(\varphi)$ depending on whether $\varphi|_\lambda$ occurs positively or negatively in φ. Second, only subformulae χ associated with non-atomic terms in $\overline{\Gamma}$, with the exception of non-atomic terms in terminological sentences introduced by step 4 of the transformation, are renamed. Third, the predicate symbol Q_λ used in $\mathrm{Def}^+_\lambda(\varphi)$ and $\mathrm{Def}^-_\lambda(\varphi)$ is uniquely associated with concepts and roles, not with occurrences of concepts and roles. Consequently, every newly introduced symbol Q_λ is associated with an expression E of the original language, and hence, we will denote Q_λ by p_E. Again, we assume that the variable ordering in a literal $Q_\lambda(x, y)$ introduced by $\overline{\Xi}$ follows the convention we have used in the definition of π.

Lemma 4.26. *$\overline{\Xi}\Pi(\overline{\Gamma})$ is satisfiable if and only if $\overline{\Gamma}$ is satisfiable if and only if Γ is satisfiable.*

Define a dependency relation $\succ^1_c$ on the predicate symbols by: $p_A \succ^1_c p_B$, if there is a definition $\phi \rightarrow \psi$ in $\overline{\Xi}\Pi(\overline{\Gamma})$ such that p_A occurs in ϕ and p_B occurs in ψ. Let $\succ_{\mathsf{S}}$ be an ordering on the predicate symbols in $\overline{\Xi}\Pi(\overline{\Gamma})$ which is compatible with the transitive closure $\succ^+_c$ of $\succ^1_c$. Due to the acyclicity of the terminology and due to fact that we split role definitions, it is possible to find such an ordering.

While an ordering $\succ_{\mathcal{TAB}}$ is optional, our selection function $S_{\mathcal{TAB}}$ selects the literal $\neg p_{\overline{A}}(x)$ in C if C is the clause $\{\neg p_{\overline{A}}(x), \neg p_A(x)\}$ originating from $A \dot{\sqsubseteq} \neg\overline{A}$. For all other clauses, let $\neg L$ be an occurrence of a negative literal in C with predicate symbol p_P. Then $\neg L$ is selected in C if and only if either p_P is the $\succ_{\mathsf{S}}$-maximal predicate symbol in C, or $\neg L$ is a literal of the form $\neg p_P(s, y)$, where s is a ground term and y is a variable.

Table 4.3 lists all possible forms of subformulae of $\overline{\Xi}\Pi(\overline{\Gamma})$ in the first column and the corresponding clauses in the second column. The selected literals are marked by a $\_{+}$. Observe that in all clauses containing a negative literal, one of the negative literals is selected. In particular, all clauses obtained from a terminological sentence or from a definition introduced by $\overline{\Xi}$ contain a negative literal. Consequently, none of these clauses can be used as premise in a factoring inference step or as positive premise in a resolution inference step.

Lemma 4.27. *Let Γ be a descriptive knowledge base without any assertional sentences. Then Γ is satisfiable. Furthermore, no inference steps are necessary to establish the satisfiability of the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$ and therefore of Γ.*

All clauses originating from the translation of the assertional sentences of the knowledge base are ground unit clauses. In all clauses except those of the form

$$\{\neg p_0(x)_+, (\neg)p_1(x, y)^*, p_2(y)\} \tag{4.1}$$

TBox concept definitions/restrictions and additional definitions	
$\forall x\colon p_0(x) \to (\forall y\colon p_1(x,y) \to p_2(y))$	$\{\{\neg p_0(x)_+, \neg p_1(x,y), p_2(y)\}\}$
$\forall x\colon p_0(x) \to (\exists y\colon p_1(x,y) \wedge p_2(y))$	$\{\{\neg p_0(x)_+, p_1(x,f(x))\}, \{\neg p_0(x)_+, p_2(f(x))\}\}$
$\forall x\colon p_0(x) \to (p_1(x) \wedge p_2(x))$	$\{\{\neg p_0(x)_+, p_1(x)\}, \{\neg p_0(x)_+, p_2(x)\}\}$
$\forall x\colon p_0(x) \to (p_1(x) \vee p_2(x))$	$\{\{\neg p_0(x)_+, p_1(x), p_2(x)\}\}$
$\forall x\colon p_0(x) \to \neg p_1(x)$	$\{\{\neg p_0(x)_+, \neg p_1(x)\}\}$
$\forall x\colon p_0(x) \to p_1(x)$	$\{\{\neg p_0(x)_+, p_1(x)\}\}$
$\forall x\colon p_0(x) \to \bot$	$\{\{\neg p_0(x)_+\}\}$
$\forall x\colon p_A(x) \to \neg p_{\overline{A}}(x)$	$\{\{\neg p_{\overline{A}}(x)_+, \neg p_A(x)\}\}$
TBox role definitions/restrictions and additional definitions	
$\forall x,y\colon p_0(x,y) \to (p_1(x,y) \wedge p_2(y))$	$\{\{\neg p_0(x,y)_+, p_1(x,y)\}, \{\neg p_0(x,y)_+, p_2(y)\}\}$
$\forall x,y\colon (p_1(x,y) \wedge p_2(y)) \to p_0(x,y)$	$\{\{\neg p_1(x,y)+, \neg p_2(y)_+, p_0(x,y)\}\}$
$\forall x,y\colon p_0(x,y) \to p_1(y,x)$	$\{\{\neg p_0(x,y)_+, p_1(y,x)\}\}$
$\forall x,y\colon p_0(x,y) \to (p_1(x,y) \vee p_2(x,y))$	$\{\{\neg p_0(x,y)_+, p_1(x,y), p_2(x,y)^*\}\}$
$\forall x,y\colon (p_1(x,y) \vee p_2(x,y)) \to p_0(x,y)$	$\{\{\neg p_1(x,y)_+, p_0(x,y)\}, \{\neg p_2(x,y)_+, p_0(x,y)\}\}$
$\forall x,y\colon p_0(x,y) \to (p_1(x,y) \wedge p_2(x,y))$	$\{\{\neg p_0(x,y)_+, p_1(x,y)\}, \{\neg p_0(x,y)_+, p_2(x,y)\}\}$
$\forall x,y\colon (p_1(x,y) \wedge p_2(x,y)) \to p_0(x,y)$	$\{\{\neg p_1(x,y)_+, \neg p_2(x,y)_+, p_0(x,y)\}\}$
$\forall x,y : p_0(x,y) \to p_1(x,y)$	$\{\{\neg p_0(x,y)_+, p_1(x,y)\}\}$
$\forall x\colon p_0(x,y) \to \bot$	$\{\{\neg p_0(x,y)_+\}\}$
ABox	
$A(a)$	$\{\{A(a)\}\}$
$P(a,b)$	$\{\{P(a,b)\}\}$

Table 4.3: Clausal form of formulas in definitional form

the selected literal contains all variables of the clause, and with exception of

$$\{\neg p_0(x)_+, p_1(x,f(x))^*\} \tag{4.2}$$

and

$$\{\neg p_0(x)_+, p_2(f(x))^*\}. \tag{4.3}$$

no variables occur as arguments of compound terms.

Lemma 4.28. *Let Γ be a descriptive knowledge base and N the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$. If N does not contain clauses of the form (4.1), (4.2), and (4.3) then any derivation by (ordered) resolution with selection and positive (ordered) factoring followed by condensing in which the "Delete" operation is applied eagerly terminates.*

Proof. All ground clauses in N have depth 2. Only (positive) ground (unit) clauses can be the positive premise in a resolution inference. As observed, the resolvent will be a ground clause with the same depth as the positive premise. Likewise only a ground clause can be the premise of a factoring inference, and again the factor will be a ground clause with the same depth as its premise. Since "Delete" is applied eagerly, all clauses are kept in condensed form. There are only finitely many condensed ground clauses over the given signature with depth 2. Therefore, the procedure will eventually terminate. $\square$

Lemma 4.29. *Let Γ be a descriptive knowledge base and N the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$. If N does not contain clauses of the form (4.1), then any conclusion of an inference step by (ordered) resolution with selection and (ordered) factoring will result in a ground clause.*

Proof. In the proof of Lemma 4.28 we have already established that when N contains no clauses of the form (4.1), (4.2), and (4.3), any conclusion of an inference step will be a ground clause. We have also observed that the selected literals in (4.2) and (4.3) contain the only variable of the respective clauses. Therefore, any inference step with a ground clause as positive premise results in a ground clause. □

Inferences with negative premise of the form (4.1) are problematic, since the resolvent may contain more free variables than the positive premise of the inference step. Suppose we have derived a clause of the form

$$\{p_1(a), p_2(a), p_3(a)\}$$

and $\overline{\Xi}\Pi(\overline{\Gamma})$ contains the clauses $\{\neg p_i(x)_+, \neg r_i(x, y), q_i(y)\}$, for $1 \leq i \leq 3$. Without taking further restrictions into account we can derive the clause

$$\{\neg r_1(a, x), q_1(x), \neg r_2(a, y), q_2(y), \neg r_3(a, z), q_3(z)\}.$$

It contains more variables than any clause in $\overline{\Xi}\Pi(\overline{\Gamma})$.

In general, the positive premise of a resolution inference step with a clause (4.1) is a ground clause $\{p_0(s)^*\} \cup D_1$ such that no literals in D_1 are selected. The conclusion of the inference step is a clause

$$C_1 = \{\neg p_1(s, y)_+, p_2(y)\} \cup D_1 \tag{4.4}$$

containing one free variable. However, the literal $\neg p_1(s, y)$ is selected by $S_{\mathcal{TAB}}$ and no inference steps are possible on D_1 (which still contains no selected literals). As a consequence of Lemma 4.29 the only clauses we can derive containing a positive literal with predicate symbol p_1 will be ground clauses, that is, clauses of the form $C_2 = \{p_1(s, t)^*\} \cup D_2$. The conclusion of an inference step between C_1 and C_2 is the ground clause $\{p_2(t)\} \cup D_1 \cup D_2$. Consequently, all clauses occurring in a derivation from the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$ contain at most two variables.

The problem with inferences involving negative premises of the form (4.2) and (4.3) is that resolvents may contain terms of greater depth than the positive premise of the inference. Nevertheless, we can still show that there is an upper bound on the depth of terms. We define a complexity measure μ_N on clauses occurring in a derivation from the clausal form N of $\overline{\Xi}\Pi(\overline{\Gamma})$ by

$$\mu_N(C) = \begin{cases} (p_1, p_1), \text{ if } C = \{p_1(t)\} \\ (p_1, p_2), \text{ if } C = \{p_2(s, t)\} \text{ or } C = \{\neg p_2(s, t)\} \cup C_1 \text{ and } C \text{ has a positive parent} \\ \qquad\qquad \text{clause } D \text{ with } \mu_N(D) = (p_1, p_3) \\ (p_2, p_2), \text{ if } C = \{p_2(s, t)\} \text{ and } C \text{ is in } N \end{cases}$$

That is, the complexity measure associated with a clause is a pair of predicate symbols. Complexity measures are compared by the lexicographic combination $\succ_{\mathsf{S}}^2 = (\succ_{\mathsf{S}}, \succ_{\mathsf{S}})$. Since $\succ_{\mathsf{S}}$ is well-founded, also $\succ_{\mathsf{S}}^2$ is well-founded.

It is straightforward to check that any inference step from a positive premise C by ordered resolution or ordered factoring will result in a clause D such that $\mu_N(C)$ is greater than $\mu_N(D)$ with respect to $\succ_{\mathsf{S}}^2$.

Theorem 4.30.
Let Γ be a descriptive knowledge base and N the clausal form of $\Xi\Pi(\overline{\Gamma})$. Then any derivation from N by (ordered) resolution with selection and (ordered) factoring based on (the ordering $\succ_{\mathcal{TAB}}$ and) the selection function $S_{\mathcal{TAB}}$ terminates.

Proof. We have shown that in any derivation only ground clauses will be used as positive premises of an inference step. We have shown that there is a bound on the depth of terms occurring in the clauses of the derivation. We have also shown that the clauses contain not more than two variables. Since clauses are always kept in condensed form, the number of distinct clauses with these properties is bounded. Thus, any derivation will eventually terminate. □

In tableaux-based decision procedures the satisfiability test for a descriptive knowledge base Γ for $\mathcal{ALC}$ is traditionally performed in four steps [65]:

1. Elimination of concept specialisations: Concept specialisations $A \mathrel{\dot{\sqsubseteq}} \top$ are eliminated from Γ. Any remaining concept specialisation $A \mathrel{\dot{\sqsubseteq}} C$ is replaced by $A \doteq C \sqcap A^*$ where A^* is a new concept symbol.

2. Elimination of concept definitions: If $A \doteq C$ is a concept definition, then any occurrence of A on the right hand side of a concept definition is replaced by C. This process is iterated until no defined concepts occurs on the right hand side of a definition. Since Γ contains no terminological cycle, this process terminates and the resulting set of terminological sentences is an *expanded TBox*. The expanded TBox may have exponential size with respect to the original TBox.

3. Elimination of the TBox: Every occurrence of a defined concept in an assertional sentence of Γ is replaced by its definition in the expanded TBox.

4. Satisfiability test of the ABox: After performing the first three steps, the ABox Δ consists of a set of expressions $a \in C$ and $(a, b) \in R$. Testing the satisfiability is done by applying the following completion rules:

   (a) $\Delta \Rightarrow_{\sqcap} \Delta \cup \{a \in C, a \in D\}$
   if $a \in (C \sqcap D)$ is in Δ, $a \in C$ and $a \in D$ are not both in Δ.

   (b) $\Delta \Rightarrow_{\sqcup} \Delta \cup \{a \in E\}$
   if $a \in (C \sqcup D)$ is in Δ, neither $a \in C$ nor $a \in D$ is in Δ and $E = C$ or $E = D$.

   (c) $\Delta \Rightarrow_{\exists} \Delta \cup \{(a, b) \in R, b \in C\}$
   if $a \in \exists R.C$ is in Δ, there is no d such that both $(a, d) \in R$ and $d \in C$ are in Δ, and b is a new object symbol with respect to Δ.

   (d) $\Delta \Rightarrow_{\forall} \Delta \cup \{b \in C\}$
   if $a \in \forall R.C$ and $(a, b) \in R$ are in Δ and $b \in C$ is not in Δ.

   (e) $\Delta \Rightarrow_{\perp} \Delta \cup \{a \in \perp\}$, if $a \in A$ and $a \in \neg A$ are in Δ, where A is a concept symbol.

   Let $\Rightarrow_{\mathcal{TAB}}$ be the transitive closure of the union of the transformation rules given above. An ABox Δ contains a clash if $a \in \perp$ is in Δ. An ABox Δ is satisfiable if there exists an ABox Δ' such that (i) $\Delta \Rightarrow_{\mathcal{TAB}} \Delta'$, (ii) no further applications of $\Rightarrow_{\mathcal{TAB}}$ to Δ' are possible, and (iii) Δ' is clash-free.

The reduction which eliminates the TBox before testing for satisfiability can be extended to reduce the satisfiability test for a knowledge base to a series of tests of the coherence of concepts. This is done by exhaustively applying the rules $\Rightarrow_\forall$, $\Rightarrow_\sqcap$, and $\Rightarrow_\sqcup$ to an ABox Δ. The resulting knowledge base is called the *pre-completion* of Δ. Let $\mathsf{C}(\Delta, a)$ denote the set $\{C \mid a \in C \text{ is in } \Delta\}$ and let $\bigsqcap \mathsf{C}(\Delta, a)$ denote the concept intersection of the concepts in $\mathsf{C}(\Delta, a)$. Then Δ is satisfiable if and only if for every object symbol a in the pre-completion Δ' of Δ the concept $\bigsqcap \mathsf{C}(\Delta', a)$ is coherent. Since the approach as described above has serious drawbacks concerning its efficiency, implementations usually realize some interleaved form of the four steps.

The correspondence between the tableaux-based decision procedure and the selection-based decision procedure is not difficult to see. First, note that for every concept C and every role R which may possibly occur in an ABox during a satisfiability test there exist corresponding predicate symbols p_C and p_R in the clausal form of $\overline{\overline{\Xi}}\Pi(\overline{\Gamma})$.

1. An application of the $\Rightarrow_\sqcap$ rule corresponds to a resolution inference step between a ground unit clause $\{p_{C\sqcap D}(a)\}$ and clauses $\{\neg p_{C\sqcap D}(x), p_C(x)\}$ and $\{\neg p_{C\sqcap D}(x), p_D(x)\}$, generating the resolvents $\{p_C(a)\}$ and $\{p_D(a)\}$.

2. An application of the $\Rightarrow_\sqcup$ rule corresponds to a resolution inference step between a ground unit clause $\{p_{C\sqcup D}(a)\}$ and the clause $\{\neg p_{C\sqcup D}(x), p_C(x), p_D(x)\}$. We then apply the "Split" expansion rule to the conclusion $\{p_C(a), p_D(a)\}$ which will generate two branches, one on which our set of clauses contains $\{p_C(a)\}$ and one on which it contains $\{p_D(a)\}$.

3. An application of the $\Rightarrow_\exists$ rule corresponds to a resolution inference step between a ground unit clause $\{p_{\exists R.C}(a)\}$ and the clauses $\{\neg p_{\exists R.C}(x), p_R(x, f(x))\}$ and $\{\neg p_{\exists R.C}(x), p_C(f(x))\}$. This will add $\{p_R(a, f(a))\}$ and $\{p_C(f(a))\}$ to the clause set. The term $f(a)$ corresponds to the new object symbol b introduced by the $\Rightarrow_\exists$ rule.

4. An application of the $\Rightarrow_\forall$ rule corresponds to two consecutive inference steps. Here, the set of clauses contains $\{p_{\forall R.C}(a)\}$ and $\{p^u_R(a, b)\}$ (to obtain $\{p^u_R(a, b)\}$ an inference step with a clause $\{\neg P^d_R(x, y), P^u_R(x, y)\}$ may be necessary). First, $\{p_{\forall R.C}(a)\}$ is resolved with $\{\neg p_{\forall R.C}(x), \neg p^u_R(x, y), p_C(y)\}$ to obtain $\{\neg p^u_R(a, y), p_C(y)\}$. Then the conclusion of the previous inference step is resolved with $\{p^u_R(a, b)\}$ to obtain $\{p_C(b)\}$.

5. For applications of the $\Rightarrow_\perp$ rule we distinguish two cases. If A is not in $\mathrm{D}_{\doteq}(\Gamma)$, then the set of clauses contains $\{p_A(t_a)\}$ and $\{p_{\neg A}(t_a)\}$. Two consecutive inference steps using these two clauses and $\{\neg p_{\neg A}(x)_+, \neg p_A(x)\}$, the definition of $\neg A$, produce the empty clause. Otherwise, A is in $\mathrm{D}_{\doteq}(\Gamma)$ and the set of clauses contains $\{p_A(t_a)\}$ and $\{p_{\overline{A}}(t_a)\}$. In this case the empty clause can be derived with $\{\neg p_{\overline{A}}(x)_+, \neg p_A(x)\}$.

Note that all these resolution inference steps strictly obey the restrictions enforced by the selection function $S_{\mathcal{TAB}}$. This proves:

Theorem 4.31.
Selection-based decision procedures can p-simulate tableaux-based decision procedures (for $\mathcal{ALC}$).

If we exclude factoring inference steps and redundancy elimination techniques from the consideration then the simulation result also holds in the reverse direction.

Theorem 4.32.
Tableaux-based decision procedures (for $\mathcal{ALC}$) can p-simulate selection-based decision procedures without factoring and redundancy elimination.

Theorem 4.31 also holds in the presence of role operators. However, tableaux-based procedures hide some of the inferential effort in the side-conditions of the completion rules which is explicit in our selection-based decision procedure. Consider the following assertional sentences involving an application of role intersection and its translation (in clausal form):

$$
\begin{array}{ll}
a \in \forall\, P \sqcap Q.B & \{q_1(a)\} \\
& \{\neg q_1(x)_+, \neg q_2(x,y), p_B(y)\} \\
& \{\neg p_P(x,y)_+, \neg p_Q(x,y)_+, q_2(x,y)\} \\
(a,b) \in P & \{p_P(a,b)\} \\
(a,b) \in Q & \{p_Q(a,b)\}
\end{array}
$$

The tableaux-based procedure makes use of the following modified version of $\Rightarrow_\forall$:

> $\Delta \Rightarrow_\forall \Delta \cup \{b \in C\}$ if $a \in \forall\, R.C$ is in Δ and $(a,b) \in R$ holds in Δ and $b \in C$ is not in Δ, where $(a,b) \in R_1 \sqcap \ldots \sqcap R_n$ *holds in* Δ if and only if $(a,b) \in R_i$ is in Δ for all i, $1 \leq i \leq n$.

Given the ABox above, the tableaux-based procedure needs only one inference step to conclude that $b \in B$ holds. The inferential effort to determine that $(a,b) \in (P \sqcap Q)$ is a consequence of Δ is hidden in the side condition of $\Rightarrow_\forall$.

By contrast, the selection-based decision procedure will first perform two resolution inference steps to derive $\{q_2(a,b)\}$ before two further inference steps lead to $\{p_B(b)\}$. Obviously, the inferential steps necessary to deduce that $(a,b) \in R$ holds in Δ for complex roles R have to be taken into account in the simulation. Then any tableaux inference step can be simulated by at most two inference steps in the selection-based decision procedure.

Why have we excluded role complement, role value maps, and the top role from consideration? Role complement, role value maps, and the top role share the common characteristic that they introduce non-ground clauses containing only positive literals. Negative occurrences of $\neg R$, as in $\forall \neg R.C$, introduce clauses of the form $\{p_0(x,y), p_R(x,y)\}$. Then, we can no longer assume that the positive premises of inferences are ground clauses. Negative occurrences of the top role and role value maps resulting in clauses $\{\neg p_0(x), p_1(y)\}$ and $\{p_R(x, f(x)), p_S(x, f(x)), p_0(x)\}$, cause the same problem.

However, positive occurrences of these operators will result in clauses containing at least one negative, embracing literal. The selection function $S_{\mathcal{TAB}}$ will restrict inferences from these clauses in the same way as for clauses (4.1) and (4.4). The decidability result still holds.

According to Table 4.2, negative occurrences of bindings $(\subseteq U\,V)$ and $(\supseteq U\,V)$ also introduce non-ground clauses containing only positive literals into our clause set. As Table 4.4 shows, positive occurrences of bindings $(\subseteq U\,V)$, $(\supseteq U\,V)$, and arbitrary occurrences of role composition, result in clauses containing negative literals. Although only for the clause obtained from positive occurrences of role composition the negative literal is embracing, all the variables of a clause occur in at least one of the negative literals of a clause. It is important to remember that the predicate symbols in Table 4.4 are not arbitrary. The symbol p_0 is always different from p_R, p_S, p_U and p_V, and there are no cyclic (inferential) dependencies. So, neither the clause we obtain

Positive occurrences of $R \circ S$	$\{\{\neg p_0(x,y)_+, p_R(x, f(x,y))\}, \{\neg p_0(x,y)_+, p_S(f(x,y),y)\}\}$
Negative occurrences of $R \circ S$	$\{\{\neg p_R(x,z)_+, \neg p_S(z,y)_+, p_0(x,y)\}\}$
Positive occurrences of $(\subseteq U\, V)$	$\{\{\neg p_0(x,y)_+, \neg p_U(x,z)_+, p_V(y,z)\}\}$
Positive occurrences of $(\supseteq U\, V)$	$\{\{\neg p_0(x,y)_+, \neg p_V(y,z)_+, p_U(x,z)\}\}$

Table 4.4: Clausal form role composition and binding forming operators

for negative occurrences of $R \circ S$ nor the clause we obtain for positive occurrences of $(\supseteq U\, V)$ and $(\subseteq U\, V)$ has the transitivity clause as an instance or logical consequence. This is sufficient to establish termination. Let $\mathcal{U}_{\mathsf{D}}^{-}$ be the extension of $\mathcal{ALB}_{\mathsf{D}}$ by role value maps, role complement, role composition, and bindings. We assume that the selection $S_{\mathcal{TAB}}$ as indicated in Table 4.4.

Theorem 4.33.
Let Γ be a descriptive knowledge base over $\mathcal{U}_{\mathsf{D}}^{-}$ such that expressions of the form $\neg R$ and $(R{=}S)$ occur only positively, and expressions of the form $(\supseteq U\, V)$ and $(\subseteq U\, V)$ occur only negatively. Let N be the clausal form of $\overline{\Xi}\Pi(\overline{\Gamma})$. Then any derivation from N by (ordered) resolution with selection and (ordered) factoring based on (the ordering $\succ_{\mathcal{TAB}}$ and) the selection function $S_{\mathcal{TAB}}$ terminates.

4.6 Conclusion

Finally, we discuss how the classes of clauses described in this chapter relate to other solvable classes. The class of DL-clauses is not comparable with the guarded fragment or the loosely guarded fragment. In the guarded fragments the conditional quantifiers may not include negations or disjunctions. On the other hand, the guarded fragments allow predicates of arbitrary arity. Recently it has been shown that the extension of the guarded fragment with two transitive relations and equality is undecidable [60]. However, basic modal logic plus transitivity is known to be decidable. Therefore, looking at more restricted classes than the guarded fragment may lead to better characterisations of the connection between modal logics and decidable subclasses of first-order logic [46].

The class of DL-clauses is more restrictive than the class One-Free, which stipulates that quantified subformulae have at most one free variable. But as noted in Section 4.4, it is possible to extend $\mathcal{ALB}$ by certain restricted forms of role composition (e.g., positive occurrences), for which the procedure described in Section 4.2 remains a decision procedure. The corresponding clausal class is distinct from the One-Free class. It is known from the literature on algebraic logic that arbitrary occurrences of composition in the presence of role negation leads to undecidability.

The resolution framework used here has a general notion of redundancy which does not have the drawback of [127], where certain standard deletion rules, e.g. tautology deletion, have to be restricted for completeness. Real world knowledge bases typically contain hundreds of concept definitions. The corresponding clauses can be used to derive an extensive number of tautologies.

In Chapter 6 we describe experiments with resolution theorem provers which show there are theorem provers which can serve as reasonable and efficient inference procedures for description logics.

Chapter 5

Modal logics

As in the research area of description logics, decidability issues and the development of decision procedures play a prominent role in the research area of extended modal logics. Although it is not difficult to see most logics under consideration can be translated to first-order logic, the exact relation to decidable subclasses of first-order logic and in particular to subclasses decidable by resolution is still under investigation. Andréka, van Benthem and Németi [3, 4] introduced the *guarded fragment* as an attempt to characterise a class of first-order formulae sharing properties like decidability, the finite model property, and the tree model property, with modal logics. Essentially, the guarded fragment is obtained by restricting quantifications to the form

$$\forall \mathbf{y}\colon r(\mathbf{x},\mathbf{y}) \rightarrow \varphi(\mathbf{y}) \qquad \text{and} \qquad \exists \mathbf{y}\colon r(\mathbf{x},\mathbf{y}) \wedge \varphi(\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are (disjoint) sequences of variables and $r(\mathbf{x},\mathbf{y})$ is an atomic formula. Ganzinger and de Nivelle [45] show that ordered resolution with paramodulation is a decision procedure for the guarded fragment and restricted extensions with equality.

However, while the standard relational translation embeds modal logics like the basic modal logic K and its extensions with the axiom schemata B, D, and T into the guarded fragment, it does not cover extensions of K with the axiom schema $\mathsf{4}$, which characterises Kripke frames with a transitive accessibility relation. Grädel [60] shows that the extension of the guarded fragment with three variables and transitive relations is undecidable. Recently, Ganzinger, Meyer and Veanes [47] have considered the class GF^2 consisting of all formulae in the intersection of the guarded fragment and the two-variable fragment of first-order logic and *monadic* GF^2 *with transitive relations* consisting of the formulae in GF^2 where all binary relations are assumed to be transitive.[1] They show that GF^2 and monadic GF^2 with transitive relations are undecidable.

Although the class $\overline{\mathrm{K}}$ presented in Chapter 3 is not intended to be a characterisation or generalisation of the fragment of first-order logic corresponding to modal logics, it covers the relational translation of modal formulae of basic modal logic as well as the translation of many axiom schemata, in particular, B, D, and T. Like the guarded fragment it also covers some of the relational operations of extended modal logics shown in Table 5.1. While both the class $\overline{\mathrm{K}}$ and the guarded fragment are able to accommodate (the translation of) modal formulae $[R \cap S]\varphi$ and $[R \cup S]\varphi$ but exclude modal formulae $[R \circ S]\varphi$ and $[R^*]\varphi$, only the class $\overline{\mathrm{K}}$ allows for the

[1] Transitivity of a binary relation can not be expressed in the guarded fragment itself, but has to be expressed on the meta level.

ψ	$\pi_r(\psi, x)$	ψ	$\pi_r(\psi, x)$
$[\neg R]\varphi$	$\forall y\colon \neg r(x,y) \to \pi_r(\varphi, y)$		
$[R \cap S]\varphi$	$\forall y\colon (r(x,y) \wedge s(x,y)) \to \pi_r(\varphi, y)$	$[R \circ S]\varphi$	$\forall y\colon (\exists z\colon r(x,z) \wedge s(z,y)) \to \pi_r(\varphi, y)$
$[R \cup S]\varphi$	$\forall y\colon (r(x,y) \vee s(x,y)) \to \pi_r(\varphi, y)$	$[R^*]\varphi$	no first-order equivalent formula

Table 5.1: Relational operations in extended modal logics

embedding of $[\neg R]\varphi$. So, the class $\overline{\mathrm{K}}$ actually allows to express more of the relational operations than the guarded fragment does.

It is interesting to note that the undecidability result of Ganzinger, Meyer and Veanes also applies to the class $\overline{\mathrm{K}}$, that is, the satisfiability problem for the restriction of $\overline{\mathrm{K}}$ to formulae with at most two variables with transitive relations is undecidable.

From the perspective of first-order logic, both fragments are incomparable. Consider the formulae

$$\forall x, y, z\colon \neg p(x,y,z) \vee q(y,z) \tag{5.1}$$

$$\forall x, y\colon p(x,y) \vee q(x,y). \tag{5.2}$$

Formula (5.1) belongs to the guarded fragment, but not to the class $\overline{\mathrm{K}}$, while formula (5.2) belongs to the class $\overline{\mathrm{K}}$, but not to the guarded fragment.

Therefore, the exact relation of modal logics to decidable subclasses of first-order logic is still an open problem. In this chapter we will take the following approach. Instead of starting from a generalisation of the class of formulae obtained by translating modal formulae to first-order logic, we will study decidability issues on classes of first-order formulae which most closely resemble translated modal formulae. We will consider various extensions of the basic modal logic K with axiom schemata like $\mathsf{4}$, $\mathsf{5}$, B, T, and D, and we will also consider variations of the translation morphism for mapping modal formulae to first-order logic. Using this approach we will not only present resolution-based decision procedures for modal logics already covered by the guarded fragment or the class $\overline{\mathrm{K}}$, but also for extensions of $\mathsf{K4}$.

5.1 Syntax and semantics of modal logics

The language of the propositional modal logic $\mathsf{K}\Sigma$ is that of propositional logic plus additional modal operators $\Box$ and $\Diamond$. By definition, a *formula* of $\mathsf{K}\Sigma$ is a Boolean combination of propositional and modal atoms. A *modal atom* is an expression of the form $\Box\psi$ or $\Diamond\psi$ where ψ is a formula of $\mathsf{K}\Sigma$. A *literal* is a propositional atom or its negation. In the following we assume that modal formulae are in negation normal form, containing no occurrences of the Boolean connectives $\to$ and $\leftrightarrow$. In general, Σ is a (possibly empty) set of additional axiom schemata. We will also consider frame properties given by first-order formulae which need not be definable by modal schemata.

There are two major approaches to providing a semantics for modal logic: an algebraic one and a model theoretic one. Although the algebraic approach has a long-standing tradition and has shown off many important results (confer Goldblatt [56, 57]), we restrict our attention to the model theoretic approach. Here we consider the Kripke semantics [89] and the functional semantics [105] for modal logics. To emphasise the main difference between these two definitions, that is, the relational description versus the functional description of accessibility in the models, we also use the term *relational semantics* for Kripke semantics. The relational semantics is given in terms of frames. A *frame* is pair $F = (W, R)$ where W is a non-empty set of *worlds* and R is a binary relation on W, called the *accessibility relation*. A *relational model*[2] is a pair $M = (F, v)$ consisting of a frame and a *valuation* v mapping propositional variables to subsets of W. M is said to be *based on the frame* F. Validity in a relational model M and a world $w \in W$ is defined by:

$$
\begin{array}{ll}
M, w \models p & \text{iff } w \in v(p) \\
M, w \models \top & \\
M, w \models \neg\varphi & \text{iff } M, w \not\models \varphi \\
M, w \models \varphi_1 \wedge \ldots \wedge \varphi_n & \text{iff } M, w \models \varphi_i, \text{ for every } i,\ 1 \leq i \leq n \\
M, w \models \varphi_1 \vee \ldots \vee \varphi_n & \text{iff } M, w \models \varphi_i, \text{ for some } i,\ 1 \leq i \leq n \\
M, w \models \Box\varphi & \text{iff for every } v \in W,\ (w, v) \in R \text{ implies } M, v \models \varphi \\
M, w \models \Diamond\varphi & \text{iff for some } v \in W,\ (w, v) \in R \text{ and } M, v \models \varphi.
\end{array}
$$

A modal formula is *valid in a frame* F iff it is valid in all models based on F.

A sound and complete axiom system for $\mathsf{K}\Sigma$ with respect to the relational semantics consists of the standard axioms for propositional logic, the axiom schema (K), all instances of the axiom schemata in Σ, the modus ponens inference rule (MP), and the necessitation inference rule (N).

(K) $$\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$$

(MP) $$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

(N) $$\frac{\varphi}{\Box\varphi}$$

A modal formula derivable in this axiom system is a *theorem* of $\mathsf{K}\Sigma$. A model M is a (relational) $\mathsf{K}\Sigma$-model if every theorem of $\mathsf{K}\Sigma$ is valid in M. A modal formula φ is satisfiable in $\mathsf{K}\Sigma$ if there is a $\mathsf{K}\Sigma$-model M and a world w in M such that $M, w \models \varphi$. A modal logic $\mathsf{K}\Sigma$ is *sound* with respect to a class of frames $\mathcal{F}$ iff every modal formula φ which is a theorem in $\mathsf{K}\Sigma$ is valid in every frame in $\mathcal{F}$. A modal logic $\mathsf{K}\Sigma$ is *complete* with respect to a class of frames $\mathcal{F}$ iff every modal formula φ which is valid in every frame in $\mathcal{F}$ is a theorem in $K\Sigma$. A modal logic is *determined* by a class of frames $\mathcal{F}$ iff it is sound and complete with respect to $\mathcal{F}$. A modal logic is complete if it is complete with respect to some class of frames.

The *functional semantics* of modal logic is based on the insight that any binary relation can be expressed as the union of a family of partial functions. To obtain a closer correspondence between the functional semantics and the optimised functional translation we use a presentation based on total functions. A *functional frame* F is a tuple $(W, \mathsf{def}, AF, [\_\_])$ where W is a non-empty set of worlds, def is a subset of W, AF is a non-empty set of total *accessibility functions* on W, and $[\_\_] : W \times AF \rightarrow W$ is the *application function*. A world $x \in W$ such that $x \notin \mathsf{def}$ is a *dead-end*. Now, in a functional model M functional frames replace relational frames and validity is defined

[2] From the viewpoint of first-order logic it would be more appropriate to speak of (relational) interpretations.

4	Transitivity	$\Box p \to \Box\Box p$	$\forall x, y, z\colon (r(x,y) \land r(y,z)) \to r(x,z)$
5	Euclideanness	$\Diamond p \to \Box\Diamond p$	$\forall x, y, z\colon (r(x,y) \land r(x,z)) \to r(y,z)$
B	Symmetry	$p \to \Box\Diamond p$	$\forall x, y\colon r(x,y) \to r(y,x)$
D	Seriality	$\Box p \to \Diamond p$	$\forall x\, \exists y\colon r(x,y)$
G	Confluence	$\Diamond\Box p \to \Box\Diamond p$	$\forall x, y, z\colon (r(x,y) \land r(x,z)) \to (\exists u\colon r(y,u) \land r(z,u))$
M	McKinsey's axiom	$\Box\Diamond p \to \Diamond\Box p$	
T	Reflexivity	$\Box p \to p$	$\forall x\colon r(x,x)$
	Weak density	$\Box\Box p \to \Box p$	$\forall x, y\colon r(x,y) \to (\exists z\colon r(x,z) \land r(z,y))$
	Irreflexivity		$\forall x\colon \neg r(x,y)$
	Universality		$\forall x, y\colon r(x,y)$

Table 5.2: Axiom schemata and relational frame properties

as follows.

$$\begin{array}{ll}
M, w \models p & \text{iff } w \in v(p) \\
M, w \models \top & \\
M, w \models \neg\varphi & \text{iff } M, w \not\models \varphi \\
M, w \models \varphi_1 \land \ldots \land \varphi_n & \text{iff } M, w \models \varphi_i, \text{ for every } i,\ 1 \leq i \leq n \\
M, w \models \varphi_1 \lor \ldots \lor \varphi_n & \text{iff } M, w \models \varphi_i, \text{ for some } i,\ 1 \leq i \leq n \\
M, w \models \Box\varphi & \text{iff } w \in \mathsf{def} \text{ implies that for every } \alpha \in AF,\ M, [w\alpha] \models \varphi \\
M, w \models \Diamond\varphi & \text{iff } w \in \mathsf{def} \text{ and for some } \alpha \in AF,\ M, [w\alpha] \models \varphi.
\end{array}$$

Saul Kripke [89] observed that certain axiom schemata correspond to certain properties of the accessibility relation in the relational semantics. That is, for certain axiom schemata and combinations Σ of axiom schemata we can characterise the class of frames $\mathcal{F}$ such that $\mathsf{K}\Sigma$ is determined by $\mathcal{F}$ by characterising properties of the accessibility relation in frames of $\mathcal{F}$ by means of first-order and second-order formulae. These formulae are called the *relational frame properties* of the modal logic under consideration. Table 5.2 presents some axiom schemata and their corresponding relational frame properties. A class of frames comprising all frames satisfying a set of first-order formulae is said to be an *elementary class*. A modal logic $\mathsf{K}\Sigma$ is *first-order definable* if it is sound and complete with respect to an elementary class. Table 5.2 shows that the standard modal logics considered in the literature, that is, extensions of K by 4, 5, B, D, T are first-order definable. However, the extension of K by McKinsey's axiom is not first-order definable. Such modal logics are called *essentially second-order*. The correspondence results of Table 5.2 are also helpful to determine which modal logics are identical, that is, have the same set of theorems. For example, KB4 and KB5 are identical since in the presence of symmetry, transitivity, and euclideanness are equivalent properties of a binary relation. Similarly, KT5, which is also called S5, is identical to KT45, KTB4, KDB4, KDB5, and the extension of K by universality. The modal logic KT4 is also called S4.

Correspondence results can also be established for the functional semantics. However, there are cases where a modal logic which is essentially second-order with respect to the relational semantics, is first-order definable with respect to the functional semantics, for example, McKinsey's axiom whose first-order equivalent formula is given in Table 5.3 together with the functional frame properties of the other axiom schemata of Table 5.2.

4	Transitivity	$\forall x\,\forall \alpha, \beta\,\exists \gamma\colon (x \in \mathsf{def} \wedge [x\alpha] \in \mathsf{def}) \rightarrow [x\alpha\beta] = [x\gamma]$
5	Euclideanness	$\forall x\,\forall \alpha, \beta\,\exists \gamma\colon (x \in \mathsf{def} \rightarrow [x\beta] \in \mathsf{def}) \wedge (x \in \mathsf{def} \rightarrow [x\alpha] = [x\beta\gamma])$
B	Symmetry	$\forall x\,\forall \alpha\,\exists \beta\colon (x \in \mathsf{def} \rightarrow [x\alpha] \in \mathsf{def}) \wedge (x \in \mathsf{def} \rightarrow x = [x\alpha\beta])$
D	Seriality	$\forall x\colon x \in \mathsf{def}$
G	Confluence	$\forall x\,\forall \alpha, \beta\,\exists \gamma, \delta\colon x \in \mathsf{def} \rightarrow ([x\alpha] \in \mathsf{def} \wedge [x\beta] \in \mathsf{def} \wedge [x\alpha\gamma] = [x\beta\delta])$
M	McKinsey's axiom	$\forall x\,\forall \beta\,\exists \alpha\,\forall \delta\,\exists \gamma\colon [x\alpha\beta] = [x\gamma\delta]$
T	Reflexivity	$\forall x\,\exists \alpha\colon x \in \mathsf{def} \wedge x = [x\alpha]$
	Weak density	$\forall x\,\forall \alpha\,\exists \beta, \gamma\colon x \in \mathsf{def} \rightarrow ([x\beta] \in \mathsf{def} \wedge [x\alpha] = [x\beta\gamma])$
	Irreflexivity	$\forall x\,\forall \alpha\colon x \in \mathsf{def} \rightarrow x \neq [x\alpha]$

Table 5.3: Axiom schemata and functional frame properties

The relational and functional semantics allows for the definition of semantics-based translation of modal logics into first-order logic. The three approaches we will consider are the *relational translation*, based on the relational semantics, the *optimised functional translation*, based on the functional semantics, and the *semi-functional translation*, based on a combination of the relational and functional semantics.

By definition, the relational translation operator, Π_r^Σ maps φ to

$$\mathrm{Ax}_r^\Sigma \rightarrow \forall x\colon \pi_r(\varphi, x),$$

where Ax_r^Σ is the conjunction of formulae of the relational frame properties corresponding to Σ. The morphism π_r is defined by

$$\begin{aligned}
\pi_r(p, x) &= P(x) \\
\pi_r(\neg\varphi, x) &= \neg\pi_r(\varphi, x) \\
\pi_r(\varphi_1 \wedge \ldots \wedge \varphi_n, x) &= \pi_r(\varphi_1, x) \wedge \ldots \wedge \pi_r(\varphi_n, x) \\
\pi_r(\varphi_1 \vee \ldots \vee \varphi_n, x) &= \pi_r(\varphi_1, x) \vee \ldots \vee \pi_r(\varphi_n, x) \\
\pi_r(\Box\varphi, x) &= \forall y\colon r(x, y) \rightarrow \pi_r(\varphi, y) \\
\pi_r(\Diamond\varphi, x) &= \exists y\colon r(x, y) \wedge \pi_r(\varphi, y).
\end{aligned}$$

p is a propositional variable and P is a unary predicate uniquely associated with p. The symbol r is a special binary predicate denoting the accessibility relation in the underlying Kripke semantics. As $\pi_r(\varphi, x)$ is in negation normal form, all non-atomic subformulae of $\pi_r(\varphi, x)$ have positive polarity. The relational translation is sound and complete for complete modal logics.

Theorem 5.1.
Let $\mathsf{K}\Sigma$ be a complete modal logic such that Ax_r^Σ is a first-order (second-order) formula. Then

1. *φ is a theorem in $\mathsf{K}\Sigma$ if and only if $\Pi_r^\Sigma(\varphi)$ is a first-order (second-order) theorem, and*

2. *φ is satisfiable in $\mathsf{K}\Sigma$ if and only if $\overline{\Pi}_r^\Sigma(\varphi) = \neg\Pi_r^\Sigma(\neg\varphi)$ is satisfiable.*

The optimised functional translation maps modal formulae into a logic, called *basic path logic*, which is a monadic fragment of sorted first-order logic with one binary function symbol $[\_\,\_]$. The sorts W and AF distinguish between worlds and accessibility functions. The unary predicate symbols uniquely associated with the propositional variables have sort W. Also the special unary predicate def representing the subset def of W in functional frames is of sort W. The binary function $[\_\,\_]$ has sort $W \times AF \to W$. In the following we use the convention that x, y, z, x_1, y_1, ... are variables of sort W, s, t, s_1, t_1, ... are terms of sort W, ϵ is a constant of sort W, and α, β, α_1, β_1, ... are variables of sort AF. We abbreviate $[[[x\alpha_1]\dots]\alpha_n]$ by $[x\alpha_1\dots\alpha_n]$.

The optimised functional translation [107] does a sequence of transformations. The first transformation Π_f^Σ maps a modal formula φ to its so-called functional translation defined by

$$\mathrm{Ax}_f^\Sigma \to \forall x\colon \pi_f(\varphi, x).$$

For the propositional connective π_f is a homomorphism analogous to π_r. For the remaining cases, π_f is defined by

$$\begin{aligned}\pi_f(p,s) &= P(s)\\ \pi_f(\Box\varphi,s) &= \mathsf{def}(s) \to \forall\alpha\colon \pi_f(\varphi,[s\alpha])\\ \pi_f(\Diamond\varphi,s) &= \mathsf{def}(s) \wedge \exists\alpha\colon \pi_f(\varphi,[s\alpha]).\end{aligned}$$

The second transformation applies the so-called quantifier exchange operator Υ to $\forall x\colon \pi_f(\varphi,x)$, which moves existential quantifiers inwards over universal quantifiers using the rule '$\exists\alpha\forall\beta\,\psi$ becomes $\forall\beta\exists\alpha\,\psi$'. The transformation by $\Upsilon\Pi_f$ is sound and complete for complete modal logics.

Theorem 5.2 (Ohlbach and Schmidt [107]).
Let $\mathsf{K}\Sigma$ be a complete modal logic such that Ax_f^Σ is a first-order (second-order) formula. Then

1. *φ is a theorem in $\mathsf{K}\Sigma$ if and only if $\Upsilon\Pi_f^\Sigma(\varphi)$ is a first-order (second-order) theorem, and*
2. *φ is satisfiable in $\mathsf{K}\Sigma$ if and only if $\overline{\Pi}_f^\Sigma(\varphi) = \neg\Upsilon\Pi_f^\Sigma(\neg\varphi)$ is satisfiable.*

The semi-functional translation approach [78, 102] tries to combine the advantages of the relational and functional translation approach and to avoid their disadvantages. For an elaboration of the considerations leading to the development of the semi-functional translation approach refer to Nonnengart [101].

The semi-functional translation maps modal formulae to many-sorted first-order formulae. Like in the case of the optimised functional translation we distinguish between the sorts W and AF for worlds and accessibility functions. Unary predicate symbols have sort W, the binary predicate symbol r associated with the accessibility relation has sort $W \times W$, the constant symbol ϵ has sort W, and the binary function $[\_\,\_]$ has sort $W \times AF \to W$. We use the same convention as for the optimised functional translation to name variables. In addition, u, u_1, and u_2 will denote either a variable with sort W or the constant ϵ. Π_{sf}^Σ maps a modal formula φ in negation normal form to

$$(\mathrm{Ax}_r^\Sigma \wedge \mathrm{Ax}_{sf}^{\mathsf{def}}) \to \forall x\colon \pi_{sf}(\varphi, x),$$

where π_{sf} is a homomorphism on the propositional connectives and is defined by

$$\begin{aligned}\pi_{sf}(p,s) &= P(s)\\ \pi_{sf}(\Box\varphi,s) &= \forall y\colon r(s,y) \to \pi_{sf}(\varphi,y)\\ \pi_{sf}(\Diamond\varphi,s) &= \mathsf{def}(s) \wedge \exists\alpha\colon \pi_f(\varphi,[s\alpha])\end{aligned}$$

in the remaining cases. Note that $\forall y$ quantifies over a variable of sort W while $\exists\alpha$ quantifies over a variable of sort AF. The expression $[s\alpha]$ is of sort W. Since the semi-functional translation

K4	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x) \to r(x,[x\alpha])$ $\wedge\,(\mathsf{def}(x) \wedge r(x,y)) \to r(x,[y\alpha])$	KD	$\forall x\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,[x\alpha])$
K45	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x) \to \mathsf{def}(y)$ $\wedge\,\mathsf{def}(y) \to r(x,[y\alpha])$	KD45	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,[y\alpha])$
KB	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x) \to \mathsf{def}(y)$ $\wedge\,\mathsf{def}(x) \to r(x,[x\alpha])$ $\wedge\,\mathsf{def}(x) \to r([x\alpha],x)$	KDB	$\forall x\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,[x\alpha])$ $\wedge\, r([x\alpha],x)$
KD4	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,[x\alpha])$ $\wedge\, r(x,y) \to r(x,[y\alpha])$	KT	$\forall x\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,x)$ $\wedge\, r(x,[x\alpha])$
S4	$\forall x, y\,\forall \alpha$:	$\mathsf{def}(x)$ $\wedge\, r(x,x)$ $\wedge\, r(x,y) \to r(x,[y\alpha])$	S5	$\forall x, y$:	$\mathsf{def}(x)$ $\wedge\, r(x,y)$
K5	$\forall x, y\,\forall \alpha,\beta$:	$\mathsf{def}(x) \to \mathsf{def}(y)$ $\wedge\,\mathsf{def}(\epsilon) \to r(\epsilon,[\epsilon\alpha])$ $\wedge\,(\mathsf{def}(x) \wedge \mathsf{def}(y)) \to r([x\alpha],[y\beta])$	KD5	$\forall x, y\,\forall \alpha,\beta$:	$\mathsf{def}(x)$ $\wedge\, r(\epsilon,[\epsilon\alpha])$ $\wedge\, r([x\alpha],[y\beta])$

Table 5.4: Axiom schemata and semi-functional frame properties

incorporates both the relational representation and the functional representation of the accessibility relation, it is necessary to relate the two representations by means of the following formula $\mathrm{Ax}_{sf}^{\mathsf{def}}$

$$\forall x, y\,\forall \alpha\colon (\mathsf{def}(x) \to r(x,[x\alpha])) \wedge (r(x,y) \to \mathsf{def}(x)).$$

The following theorem shows that the translation preserves the satisfiability of modal formulae.

Theorem 5.3 (Nonnengart [101, p. 34]).
Let $\mathsf{K}\Gamma$ be a complete modal logic with first-order definable frame properties $\mathrm{Ax}_{sf}^{\Sigma}$. A modal formula φ in negation normal form is satisfiable if and only if $\overline{\Pi}_{sf}^{\Sigma} = \neg\Pi_{sf}^{\Sigma}(\neg\varphi)$ is satisfiable.

The formulae Ax_r^{Σ} and $\mathrm{Ax}_{sf}^{\mathsf{def}}$ do not depend on the formula φ under consideration, but only on the modal logic $\mathsf{K}\Sigma$. Therefore, it makes sense to (partially) saturate $\mathrm{Ax}_r^{\Sigma} \wedge \mathrm{Ax}_{sf}^{\mathsf{def}}$ independently of the formula $\pi_{sf}(\varphi)$. Table 5.4 list the resulting formulae, which we will denote by $\mathrm{Ax}_{sf}^{\Sigma}$, for some combinations of axiom schemata.

5.2 The relational and optimised functional translation

In the following we consider decision procedures for the satisfiablity problem in modal logics based on the translation of modal formulae into decidable fragments of first-order logic.

First, we consider the relational translation of modal formulae. According to Theorem 5.1 a modal formula φ is satisfiable if and only if $\overline{\Pi}_r^{\Sigma}(\varphi) = \mathrm{Ax}_r^{\Sigma} \wedge \exists x\colon \pi_r(\neg\varphi, x)$ is satisfiable.

It is straightforward to see that modal formulae are a notational variant of concept terms introduced in Section 4.2, and vice versa. Also, the relational translation π_r of modal formulae is identical to the translation π of concept terms. Using the one-to-one correspondence between subformulae of a modal formula and subexpressions of a concept, we can define the structural transformation Ξ on π_r and $\overline{\Pi}_r$ in analogy to Ξ on π. Consequently, the results described in Sections 4.2 to 4.5 apply. In particular, the clausal form of $\exists x\colon \pi_r(\neg\varphi, x)$ consists only of static DL-clauses. Furthermore, the relational frame properties corresponding to the axiom schemata D, T, Irreflexivity, and Universality are static DL-clauses and the frame properties corresponding to B and Weak density are DL-clauses. As a consequence of Theorem 4.16 we obtain

Corollary 5.4. *Let Σ be any combination of the axiom schemata D, T, B, and the first-order formulae Irreflexivity, Weak density, and Universality. Let φ be a modal formula and N be the clausal form of $\Xi\overline{\Pi}_r(\varphi)$. Then any derivation from N by ordered resolution and (ordered) factoring based on $\succ_{\mathcal{COV}}$ terminates.*

The tableaux-based decision procedure described in Section 4.5 is a notational variant of a *prefix tableaux calculus* [42, 59, 98] for the satisfiability problem in the modal logic K: To test the satisfiability of a modal formula φ, we apply the transformation $\Rightarrow_{\mathcal{TAB}}$ to $\Delta = \{\epsilon \in \varphi\}$ where ϵ is an arbitrary (object) symbol. If we are able to derive a clash-free set Δ' from Δ, then φ is satisfiable and Δ' is a representation of a relational model for φ.

It follows from the considerations in Section 4.5 that the refinement of resolution based on the selection function $S_{\mathcal{TAB}}$ provides a decision procedure for the satisfiability problem in K and it simulates the prefix tableaux calculus. The termination and simulation result also holds for extensions of K by a combination of the axiom schemata D, T, and B. The clausal forms of these axiom schemata are $C_{\mathsf{D}} = \{r(x, f(x))\}$, $C_{\mathsf{T}} = \{r(x,x)\}$, and $C_{\mathsf{B}} = \{\neg r(x,y), r(y,x)\}$, respectively. In the presence of C_{B} we extend the selection function $S_{\mathcal{TAB}}$ to select the negative literal in C_{B}.

As a corollary of Theorem 4.30 we obtain:

Corollary 5.5. *Let Σ be any combination of the axiom schemata D, T, B. Let φ be a modal formula and N be the clausal form of $\Xi\overline{\Pi}_r(\varphi)$. Then any derivation from N by (ordered) resolution with selection and (ordered) factoring based on (the ordering $\succ_{\mathcal{TAB}}$ and) the selection function $S_{\mathcal{TAB}}$ terminates.*

Proof. C_{D} and C_{T} can only be used to derive $\{p(f(s))\}$ and $\{p(s)\}$, respectively, using a negative premise of the form $C_{\Box} = \{\neg r(s,y)_+, p(y)\}$. According to the complexity measure established in Section 4.5, the conclusion we obtain in each case is smaller than the premises, which ensures termination of the derivation.

To accommodate the termination proof for the clause C_{B}, we have to extend the complexity measure c_L by a third component d_L which is 1 if L is a monadic literal or a dyadic literal $r(s,t)$ such that $t \succ^s s$ and 0 otherwise. We compare complexity measures on ground literals by the ordering $\succ_c^{\text{lit}}$ given by the lexicographic combination of the ordering $\succ_{\mathsf{S}}$, the multiset extension $\succ_{mul}^s$ of the strict subterm ordering $\succ^s$, and the ordering $>$ on natural numbers. It is straightforward to check the only inference possible with C_{B} is the derivation of $\{r(t,s)\}$ using the positive premise $\{r(s,t)\}$ with $t \succ^s s$ or $t = s$. If s and t are equal, then this inference step does not add a new clause to the clause set. If s is a strict subterm of t, then $\{r(s,t)\}$ is greater than $\{r(t,s)\}$ with respect to $\succ_c^{\text{lit}}$. $\square$

The simulation of tableaux calculi by a refinement of resolution based on $S_{\mathcal{TAB}}$ also holds in the presence of the axiom schemata $\mathsf{4}$ and $\mathsf{5}$. Here we assume that in the clausal form $C_{\mathsf{4}} =$

$\{\neg r(x,y), \neg r(y,z), r(x,z)\}$ and $C_5 = \{\neg r(x,y), \neg r(x,z), r(y,z)\}$ of the axiom schemata and any clause derivable from C_4 and C_5 by a single resolution inference step, one of the negative r literals is selected by $S_{\mathcal{TAB}}$. As a corollary of Theorem 4.31, the simulation result for tableaux-based procedures for the satisfiability problem in $\mathcal{ALC}$, we obtain:

Corollary 5.6. *Let Σ be any combination of the axiom schemata* D, T, B, 4, *and* 5. *The refinement of resolution based on the selection function $S_{\mathcal{TAB}}$ p-simulates prefix tableaux calculi for* KΣ.

However, in the presence of 4 or 5 termination by the selection-based refinement is no longer guaranteed. For example, consider the formula $\Diamond q \wedge \Box\Diamond p$ in K4. The clausal form of $\Xi\overline{\Pi}_r^4(\Diamond q \wedge \Box\Diamond p)$ is

$$
\begin{array}{ll}
(3) & \{p_0(\epsilon)\} \\
(4) & \{\neg p_0(x)_+, r(x, f(x))\} \\
(5) & \{\neg p_0(x)_+, q(f(x))\} \\
(6) & \{\neg p_0(x)_+, \neg r(x,y), p_1(y)\} \\
(7) & \{\neg p_1(x)_+, r(x, g(x))\} \\
(8) & \{\neg p_1(x)_+, p(g(x))\} \\
(9) & \{\neg r(x,y)_+, \neg r(y,z)_+, r(x,y)\}
\end{array}
$$

In the selection-based refinement we obtain the following unbounded derivation:

$$
\begin{array}{lll}
[(3)1,\mathrm{R},(4)1] & (10) & \{r(\epsilon, f(\epsilon))\} \\
[(3)1,\mathrm{R},(6)1] & (11) & \{\neg r(\epsilon, y)_+, p_1(y)\} \\
[(10)1,\mathrm{R},(11)1] & (12) & \{p_1(f(\epsilon))\} \\
[(7)1,\mathrm{R},(12)1] & (13) & \{r(f(\epsilon), g(f(\epsilon)))\} \\
[(10)1,(13)1,\mathrm{R},(9)1,2] & (14) & \{r(\epsilon, g(f(\epsilon)))\} \\
[(11)1,\mathrm{R},(14)1] & (15) & \{p_1(g(f(\epsilon)))\} \\
[(7)1,\mathrm{R},(15)1] & (16) & \{r(g(f(\epsilon)), g(g(f(\epsilon))))\} \\
[(14)1,(16)1,\mathrm{R},(9)1,2] & (17) & \{r(\epsilon, g(g(f(\epsilon))))\} \\
 & \vdots &
\end{array}
$$

As any fair theorem proving derivation terminates on unsatisfiable clause sets, these considerations are only relevant for non-theorems. Note that without additional techniques like loop-checking also tableaux calculi do not terminate in general for extensions of K by the axiom schemata 4 or 5 [61, 64, 90]. Thus, neither the refinement of resolution based on the ordering $\succ_{\mathcal{COV}}$ nor the refinement based on the selection function $S_{\mathcal{TAB}}$ provides a decision procedure for extensions of K by 4 or 5. However, it is possible to obtain a decision procedure based on the ordered chaining calculus [9], a general resolution calculus designed for binary relations satisfying the general scheme $r_i \circ r_j \subseteq r_k$ (including equality) combining ideas from rewrite systems and resolution. Ganzinger, Hustadt, Meyer and Schmidt [46] show that this calculus may be used to obtain resolution decision procedures for the relational translation of a range of propositional modal logics including K4, KD4, and S4.

Second, we consider the optimised functional translation. According to Theorem 5.2, a modal formula φ is KΣ-satisfiable if and only if $\overline{\Pi}_f^{\Sigma}(\varphi) = \mathrm{Ax}_f^{\Sigma} \wedge \exists x\colon \neg\Upsilon\pi_f(\neg\varphi, x)$ is satisfiable. Remember, Υ moves existential quantifiers inwards over universal quantifiers. Taking into account that the resulting formula is negated, we obtain a $\exists^*\forall^*$ formula. Consequently, all Skolem functions in the clausal form of $\exists x\colon \neg\Upsilon\pi_f(\neg\varphi, x)$ are constants. Furthermore, the variables in the terms occurring in $\exists x\colon \neg\Upsilon\pi_f(\neg\varphi, x)$ are *prefix stable*, that is, for any variable α_{i+1} there exists a unique prefix

Regular clauses		Non-regular clauses
$\{p_0(\epsilon)\}$	$\{\mathsf{def}(x)\}$	$\{r(x,[y\alpha])\}$
$\{\neg p_0(x),(\neg)p_1(x),\ldots,(\neg)p_n(x)\}$	$\{\neg\mathsf{def}(x),\mathsf{def}(y)\}$	$\{\neg\mathsf{def}(y),r(x,[y\alpha])\}$
$\{\neg p_0(x),\neg r(x,y),p_1(y)\}$	$\{r(x,x)\}$	$\{r([x\alpha],[y\beta])\}$
$\{\neg p_0(x),p_1([xf(x)])\}$	$\{\neg r(x,x)\}$	$\{\neg\mathsf{def}(x),\neg\mathsf{def}(y),r([x\alpha],[y\beta])\}$
	$\{r(x,y)\}$	$\{\neg r(x,y),r(x,[y\alpha])\}$
	$\{\neg\mathsf{def}(x),r(x,[x\alpha])\}$	$\{\neg\mathsf{def}(x),\neg r(x,y),r(x,[y\alpha])\}$
	$\{\neg\mathsf{def}(x),r([x\alpha],x)\}$	
	$\{\neg\mathsf{def}(\epsilon),r(\epsilon,[\epsilon\alpha])\}$	

Table 5.5: Clausal form of formulae in definitional form

$[x\alpha_1\ldots\alpha_i]$ such that every term containing α_{i+1} has the form $[x\alpha_1\ldots\alpha_i\alpha_{i+1}\ldots\alpha_n]$. Prefix stability holds independently of the transformation Υ. Note that none of the terms in $\exists x\colon\neg\Upsilon\pi_f(\neg\varphi,x)$ contains a variable of sort W. These properties allow for a restrictive, syntactical characterisation of the clausal form of $\overline{\Pi}^{\Sigma}_f(\varphi)$, in so-called *path logics* [120, 121]. As in general the functional frame properties contain equations, *theory resolution* with normalisation has been proposed as a decision procedure for certain extensions of K. Schmidt [120, 121] proves that theory resolution with normalisation and condensing is a decision procedure for the satisfiability of finite sets of clauses in the basic path logic, if (i) a bound on the depth of derived clauses exists, (ii) unification with respect to the functional frame properties is decidable, and (iii) an effective normalisation function exists which returns basic path clauses. For the modal logics K, KD, KT, and $\mathsf{S5}$ it can be shown that the maximal term depth in the conclusion of an inference steps by theory factoring or theory resolution will not exceed the maximal term depth in the premises. Termination of unrefined theory resolution and decidability immediately follows. For the modal logics $\mathsf{KD4}$ and $\mathsf{S4}$ it is possible to compute an a priori depth bound based on the number of occurrences of the modal operators $\Box$ and $\Diamond$ in φ. Prohibiting the generation of clauses exceeding this a priori depth bound, we obtain a decision procedure for $\mathsf{KD4}$ and $\mathsf{S4}$ by unrefined resolution. Note that enforcing a bound on the depth of derived clauses will not guarantee termination for the relational translation approach using unrefined resolution, since the number of variables in derived clauses would still grow unboundedly, but it would for the refinement of resolution based on the selection function $S_{\mathcal{TAB}}$.

5.3 The semi-functional translation

Now, we turn to the semi-functional translation. According to Theorem 5.3, a modal formula φ is $\mathsf{K}\Sigma$-satisfiable if and only if $\overline{\Pi}^{\Sigma}_{sf}(\varphi) = \mathrm{Ax}^{\Sigma}_{sf}\wedge\exists x\colon\pi_{sf}(\mathrm{nnf}(\neg\varphi))$ is satisfiable. Table 5.5 lists the form of clauses in the clausal form of $\Xi\overline{\Pi}^{\Sigma}_{sf}(\varphi)$. The left column lists clauses from $\exists x\colon\pi_{sf}(\mathrm{nnf}(\neg\varphi))$. The clauses in the middle column stem from the semi-functional frame properties for K and its extension by B, D, T, Irreflexivity, and Universality. Note that the semi-functional frame property for the modal logic $\mathsf{S5}$ is identical to Universality. Every clause in the first two columns is regular, and in addition, strongly CDV-free. Consequently, it is not necessary to apply the transformation $\Rightarrow_{\mathcal{M}}$ to these clauses. As a corollary of Theorem 3.33 we obtain:

Theorem 5.7.
Let Σ be any combination of the axiom schemata D, T, B, *and the first-order formulae Irreflexivity and Universality. Let φ be a modal formula and N be the clausal form of $\Xi\overline{\Pi}^{\Sigma}_{sf}(\varphi)$. Let $\succ$ be an atom ordering satisfying Condition (3.3) defined on page 56. Then any derivation from N by ordered resolution and (ordered) factoring based on $\succ$ terminates.*

For the remainder of this section we consider clause sets containing one of the clauses in the right column of Table 5.5. These originate from the semi-functional frame properties of 4 and 5. Since these clauses are non-regular, the results of Chapter 3 do not apply.

The clause set $\mathrm{Cls}\Xi\overline{\Pi}^{\Sigma}_{sf}(\varphi)$ contains only predicate symbols of maximal arity 2 such that all arguments have to be of sort W, one constant symbol ϵ of sort W, unary function symbols of sort $W \to AF$, and one binary function symbol of sort $[\_\_] : W \times AF \to W$. This signature is called an *SF-signature*. It is important to note that:

Lemma 5.8. *For any syntactical most general unifier σ of two well-sorted terms t_1 and t_2, $t_1\sigma = t_2\sigma$ is again well-sorted.*

The same holds for atoms and literals. Consequently, sorts will play no role in the following considerations.

We proceed by defining a class of condensed clauses generalising the clauses present in $\mathrm{Cls}\Xi\overline{\Pi}^{\Sigma}_{sf}(\varphi)$, which is finitely bounded whenever the signature is finite. To this end, we introduce some more notation to abbreviate certain more general forms of clauses. Subsequently we assume that:

$$\begin{aligned}
\neg r(\bar{u}_n, t) &\quad \text{expands to} \quad \bigcup_{1\leq i\leq n} \{\neg r(u_i, t)\},\\
\neg r(t, \bar{u}_n) &\quad \text{expands to} \quad \bigcup_{1\leq i\leq n} \{\neg r(t, u_i)\},\\
\mathcal{P}(\bar{u}_n) &\quad \text{expands to} \quad \bigcup_{1\leq i\leq n} \{\mathcal{P}(u_i)\}, \quad \text{and}\\
\mathcal{P}(t) &\quad \text{expands to} \quad (\neg)p_1(t) \vee \ldots \vee (\neg)p_m(t),
\end{aligned}$$

where t is a term and $\bar{u}_n$ denotes a vector of distinct variables. If the number of variables is not important we write $\bar{u}$ instead of $\bar{u}_n$. Any of the disjunctions may be empty. The p_i in $\mathcal{P}(t)$ are pairwise distinct monadic predicates applied to the same term t. Different occurrences of $\mathcal{P}$ within a clause may involve different sets of predicates. For example, let $\bar{x}_2$ be the vector of two variables, x_1 and x_2, and assume that there are two monadic predicates p and q. Then $\mathcal{P}(\bar{x}_2) \cup \mathcal{P}(a)$ may expand to a clause $\{p(x_1), \neg p(x_2), q(x_1), q(a)\}$, but not to $\{p(x_1), p(x_1), q(a)\}$.

Definition 5.9 (SF-regular term).
A well-sorted, regular term over an SF-signature is called an *SF-regular term*.

Note that well-sortedness does not restrict the maximal depth of literals and clauses. However, a term like $[x\alpha_1\alpha_2] = [[x\alpha_1]\alpha_2]$ is not regular, since $[[x\alpha_1]\alpha_2]$ does not dominate $[x\alpha_1]$. It is straightforward to see, that every well-sorted, regular term is of the form u, $[u\alpha]$, or $[uf(u)]$ where u is either a variable of sort W or the constant ϵ, and α is a variable of sort AF.

Lemma 5.10. *Let t_1 and t_2 be SF-regular terms and let σ be the most general unifier of t_1 and t_2. Then $t_1\sigma = t_2\sigma$ is a SF-regular term and* $\mathrm{dp}(t_1\sigma) = \max(\mathrm{dp}(t_1), \mathrm{dp}(t_2))$.

Proof. By a straightforward case analysis of all possible forms of t_1 and t_2. $\square$

Definition 5.11 (SF-regular clause).
A clause C is an *SF-regular clause* if C is a well-sorted, strongly CDV-free, regular clause over an SF-signature such that (i) there are no occurrences of negative, dyadic literals, (ii) there is at most one occurrence of a positive, dyadic literal L, (iii) the first argument of a dyadic literal L in C is a subterm of the second argument of L, and (iv) if C contains a compound term t and a dyadic literal L, then t is identical to the second argument of L.

Definition 5.12 (Small SF-clause).
A clause C is a *small SF-clause* if one of the following is true.

1. C is a SF-regular clause,

2. C is in one of the following forms

$$
\begin{array}{ll}
(\mathcal{C}_\Box) & \mathcal{P}(\bar{x}_2) \cup \{\neg r(x_1, x_2)\}, \\
(\mathcal{C}_5) & \mathcal{P}(\bar{x}_2) \cup \{r([x_1\alpha_1], [x_2\alpha_2])\}, \\
(\mathcal{C}_{45}) & \mathcal{P}(\bar{x}_2) \cup \{r(x_1, [x_2\alpha_2])\},
\end{array}
$$

   where x_1 and x_2 are variables of sort W, and α_1 and α_2 are variables of sort AF.

By Theorem 3.25 there are only finitely many SF-regular clauses modulo variable renaming. Obviously, there are only finitely many condensed clauses of the form $\mathcal{C}_5$ and $\mathcal{C}_{45}$.

Theorem 5.13.
Let Σ be the axiom schema 5*, or its combination with* 4*,* D*, and* T*. Let φ be a modal formula in negation normal form. Every clause in* $\mathrm{Cls}\Xi\overline{\Pi}_{sf}^{\Sigma}(\varphi)$ *is a small SF-clause.*

Proof. Except for $\{\neg p_0(x), \neg r(x, y), p_1(y)\}$ the clausal form of $\exists x\colon \pi_{sf}(\mathrm{nnf}(\neg\varphi))$ contains no dyadic literals and the clauses are well-sorted, strongly CDV-free, and regular. Clauses of the form $\{\neg p_0(x), \neg r(x, y), p_1(y)\}$ are instances of $\mathcal{C}_\Box$.

The clauses $\{\mathsf{def}(x)\}$, $\{r(x, [x\alpha])\}$, and $\{r(x, x)\}$ associated with D and T are SF-regular clauses. The clauses corresponding to 5, and the combination 45 are instances of $\mathcal{C}_4$ and $\mathcal{C}_{45}$, respectively. $\square$

It remains to show that the class of small SF-clauses is closed under ordered resolution and ordered factoring given an appropriate ordering. Recall the definition of the ordering $\succ_{\mathcal{COV}}$ in Chapter 4: $\succ_{\mathcal{COV}}$ is any atom ordering compatible with c_L where c_L is the complexity measure given by the multiset of arguments of L and the multiset extension $\succ^s_{mul}$ of the strict subterm ordering $\succ^s$. To make use of the results of Chapter 3 we have to prove that the atom ordering $\succ_{\mathcal{COV}}$ satisfies Condition (3.3) on page 56, that is, $\succ_{\mathcal{COV}}$-maximality of a literal implies $\succ_Z$-maximality.

Lemma 5.14. *The atom ordering $\succ_{\mathcal{COV}}$ satisfies Condition (3.3) on indecomposable, SF-regular clauses.*

Proof. Let C be an indecomposable, SF-regular clause. Let L_1 be a $\succ_{\mathcal{COV}}$-maximal literal in C.

Suppose $L_1 = r(s_1, t_1)$ is a dyadic literal. By Condition (ii), L_1 is the only dyadic literal in C. So, let L_2 be a monadic literal in C with argument t_2. If t_2 is a variable, then t_2 has to occur in L_1, otherwise C would be decomposable. However, if t_2 occurs in L_2, then it will be a subterm of s_1 or t_1. Thus, $L_1 \gtrsim_Z L_2$. Note also, that $L_1 \succ_{\mathcal{COV}} L_2$, that is, L_2 is not $\succ_{\mathcal{COV}}$-maximal. If t_1 is a compound term, then by Condition (iv), it is equal to t_2. Again, $L_1 \gtrsim_Z L_2$ and L_2 is not $\succ_{\mathcal{COV}}$-maximal. Hence, L_1 is $\succ_Z$-maximal.

Suppose L_1 is a monadic literal with argument t_1. This implies all literals in C are monadic. Since C is regular, it contains a dominating literal. If L_1 itself is a dominating literal, then L_1 is $\succ_Z$-maximal. Otherwise, there is a literal $L_3 \neq L_1$ with $L_3 \gtrsim_Z L_1$ and $L_1 \not\gtrsim_Z L_3$. That means, $t_3 \gtrsim_Z t_1$ and $t_1 \not\gtrsim_Z t_3$. By a case analysis of the syntactical form of t_1 and t_3 it follows that t_1 is a strict subterm of t_3. However, this contradicts the assumption that L_1 is $\succ_{\mathcal{COV}}$-maximal in C. So, L_1 is a dominating literal in C and therefore $\succ_Z$-maximal. □

Corollary 5.15. *Let $\{L\} \cup C$ be an indecomposable, SF-regular clause with dyadic literal L and let σ be a substitution such that $L\sigma$ is well-sorted and regular. Then L and $L\sigma$ are $\succ_{\mathcal{COV}}$-maximal with respect to C and $C\sigma$, respectively.*

Recall that the results of Chapter 3 allow term depth growth during a theorem proving derivation on regular clauses. We will now show that this is not the case for SF-regular clauses.

Lemma 5.16. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint, indecomposable, SF-regular clauses such that A_1 and A_2 are unifiable with most general unifier σ, and let $A_1\sigma$ and $\neg A_2\sigma$ be $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$ and $D_2\sigma$, respectively. Then every split component E of $(D_1 \cup D_2)\sigma$ is an SF-regular clause and* $\mathrm{dp}(E) \leq \max(\mathrm{dp}(C_1), \mathrm{dp}(C_2))$.

Proof. The clause E is regular due to Lemma 3.19 and strongly CDV-free due to Lemma 3.30. It remains to show that E has properties (i) to (iv) of Definition 5.22. E does not contain a negative, dyadic literal, since neither C_1 nor C_2 contains one. This also implies that $\neg A_2$ is not a dyadic literal. Therefore, both A_1 and $\neg A_2$ are monadic literal. By Corollary 5.15, we can only resolve on a monadic literal in an SF-clause if it does not contain a dyadic literal. So, E contains no dyadic literal, since neither C_1 nor C_2 contains one.

It remains to show that the depth of E will be less than or equal to the maximal depth of its parent clauses. The argument t_i, $1 \leq i \leq 2$, of A_i is an SF-regular term and the argument of any literal in D_i is a subterm of t_i. By Lemma 5.10, $\mathrm{dp}(t_1\sigma) = \mathrm{dp}(t_2\sigma) = \max(\mathrm{dp}(t_1), \mathrm{dp}(t_2))$. Thus, $\mathrm{dp}(E) \leq \max(\mathrm{dp}(C_1), \mathrm{dp}(C_2))$. □

Lemma 5.17. *Let $C_1 = \{L_1, L_2\} \cup D_1$ be an indecomposable, SF-regular clause such that L_1 and L_2 are unifiable with most general unifier σ and $L_1\sigma$ is $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$. Then σ is the identity substitution, the factor $(\{L_1\} \cup D_1)\sigma$ is an indecomposable, SF-regular clause of the same depth as C_1.*

Proof. Since there is at most one dyadic literal in C_1, neither L_1 nor L_2 are dyadic. The argument t_1 of L_1 is of the form u_1, $[u_1\alpha_1]$ or $[u_1 f(u_1)]$ where u_1 is a either a variable or the constant ϵ. Now consider the argument t_2 of L_2. If t_2 is a variable distinct from u_1 or the constant ϵ, then C_1 is not indecomposable. If t_2 is identical to u_1, but t_1 is a compound term, then t_1 and t_2 are not unifiable. Similarly, if t_2 is a compound term and t_1 is a variable or constant.

Suppose t_2 is of the form $[u_2\alpha_2]$. If t_1 is of the form $[u_1 f(u_1)]$, then C_1 is not regular. Similarly, if t_1 is of the form $[u_1\alpha_1]$, but either u_2 is distinct from u_1 or α_2 is distinct from α_1.

In the remaining cases the argument is similar. Thus, t_1 and t_2 are identical and their most general unifier is the identity substitution. Since $(\{L_1\} \cup D_1)\sigma$ is a subset of C_1, the rest follows. □

Consequently, condensation is the only form of factoring that is possible on indecomposable, SF-regular clauses.

Lemma 5.18. *Let $C_1 = \{L_1\} \cup D_1$ be clauses of the form $\mathcal{C}_\Box$, $\mathcal{C}_5$, or $\mathcal{C}_{45}$. Let σ be a substitution such that $C_1\sigma$ is well-sorted and contains no non-regular term. If $L_1\sigma$ is $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$, then L_1 is a dyadic literal.*

Proof. Straightforward. □

As a consequence of Lemma 5.15 and Lemma 5.18 we obtain:

Lemma 5.19. *Let C_1 be a clause of the form $\mathcal{C}_5$ or $\mathcal{C}_{45}$. Let C_2 be an SF-regular clause, a clause of the form $\mathcal{C}_5$, or a clause of the form $\mathcal{C}_{45}$.*

1. *No inference step by ordered resolution based on $\succ_{\mathcal{COV}}$ is possible with premises C_1 and C_2.*
2. *No inference step by ordered factoring based on $\succ_{\mathcal{COV}}$ is possible with premise C_1.*

Proof. A resolution inference step on the monadic literals in C_1 and C_2, or a factoring inference step on the monadic literals in C_1, violates the ordering constraints. Since C_1 and C_2 contain exactly one positive, dyadic literal no other inference steps are possible. □

Similarly, resolution inference steps or factoring inference steps with premises of the form $\mathcal{C}_\Box$ are not possible. It remains to consider resolution inference steps with a negative premise of the form $\mathcal{C}_\Box$.

Lemma 5.20. *Let $C_1 = \{A_1\} \cup D_1$ be an SF-regular clause, a clause of the form $\mathcal{C}_5$, or of the form $\mathcal{C}_{45}$. Let $C_2 = \{\neg A_2\} \cup D_2$ be a clause of the form $\mathcal{C}_\Box$. Let σ be the most general unifier of A_1 and A_2, and let $A_1\sigma$ and $\neg A_2\sigma$ be $\succ_{\mathcal{COV}}$-maximal with respect to $D_1\sigma$ and $D_2\sigma$. Then $(D_1 \cup D_2)\sigma$ is an SF-regular clause.*

Proof. By Lemma 5.18, $\neg A_2$ is the dyadic literal of the form in C_2, that is, $\neg A_2 = \neg r(x_1, x_2)$ and $D_1 = \mathcal{P}(x_1) \cup \mathcal{P}(x_2)$. So, A_1 is a dyadic literal as well. Without loss of generality, we assume that the most general unifier maps x_1 and x_2 to the arguments t_1 and t_2 of A_1.

If C_1 is a clause of the form $\mathcal{C}_5$ or $\mathcal{C}_{45}$, t_1 and t_2 are well-sorted regular terms which share no variables. Let y_1 and y_2 be the variables of sort W in t_1 and t_2, respectively. The conclusion $(D_1 \cup D_2)\sigma$ of the inference step has the form $\mathcal{P}(y_1) \cup \mathcal{P}(t_1) \cup \mathcal{P}(y_2) \cup \mathcal{P}(t_2)$ where $\mathcal{P}(y_1) \cup \mathcal{P}(t_1)$ and $\mathcal{P}(y_2) \cup \mathcal{P}(t_2)$ are the split components of $(D_1 \cup D_2)\sigma$. Obviously, they are SF-regular clauses.

If C_1 is a SF-regular clause, then $D_2\sigma = \mathcal{P}(t_1) \cup \mathcal{P}(t_2)$. It is straightforward to see that $(D_1 \cup D_2)\sigma$ is SF-regular. □

Theorem 5.21.
Let Σ be a combination of the axiom schema $\mathsf{4}$, D, and T plus the axiom schema $\mathsf{5}$. Let φ be a modal formula in negation normal form and let N be the set of clauses obtained by applying $\mathrm{Cls}\Xi\overline{\Pi}^{\Sigma}_{sf}$ to φ. Any derivation from N by ordered resolution and ordered factoring based on the ordering $\succ_{\mathcal{COV}}$ terminates.

Proof. By Theorem 5.13 every clause in N is a small SF-clause. By the Lemmata 5.16 5.17, 5.19, and 5.20 the class of small SF-regular clauses is closed under inference steps by ordered resolution and ordered factoring based on $\succ_{\mathcal{COV}}$. Since the number of small SF-regular clauses is finitely bounded modulo variable renaming, any derivation from N terminates. □

Finally, we consider extensions of K4 by the axiom schemata D and T. We start by characterising an appropriate class of clauses.

Definition 5.22 (SF-clause).
A clause C is an *SF-clause* if one of the following is true.

1. C is an SF-regular clause,
2. C is a clause of the form

   $$(\mathcal{C}_{inv}) \qquad \mathcal{P}(\bar{u}) \cup \neg r(\bar{u}, v) \cup \mathcal{P}(v) \cup \mathcal{P}(\bar{w}) \cup \neg r(\bar{w}, t) \cup \mathcal{P}(t).$$

   where v is either a variable of sort W or the constant ϵ, $\bar{u}$ and $\bar{w}$ are vectors of variables and constants of sort W, $t = [v\alpha]$ for some variable α of sort AF or $t = [vf(v)]$ for some unary function symbol f, such that, additionally, if u and w are variables occurring in a monadic atom in C, then there is at most one negative r literal in which this variable occurs.
3. C is of the form

   $$(\mathcal{C}_4) \qquad \mathcal{P}(\bar{x}_2) \cup \{\neg r(x_1, x_2), r(x_1, [x_2\alpha])\},$$

   where x_1 and x_2 are variables of sort W, and α is a variable of sort AF.

For clauses of the form ($\mathcal{C}_{inv}$) we shall also write $C = C[v]$ to emphasise the special role of v as the only variable or constant of sort W that may occur on the right side of r literals in C, if there are any such literals. In that case, $C[v']$ will denote the clause in which v is replaced by v'. We will write $C = C[t]$ to emphasise the term t occurring in C.

Note that if a variable x in $\bar{u}$ or $\bar{w}$ occurs in a monadic literal of a clause C in the form ($\mathcal{C}_{inv}$), but not in a dyadic literal in C, then C is either decomposable or it can be condensed. We will therefore assume in the following that these variables occur in exactly one r-literal in C or that C does not contain any r-literals at all.

Theorem 5.23.
Let Σ be any combination of the axiom schemata 4, D, and T. Let φ be a modal formula. Every clause in $\mathrm{Cls}\Xi\overline{\Pi}^{\Sigma}_{sf}(\varphi)$ is an SF-clause.

Proof. Except for $\{\neg p_0(x), \neg r(x, y), p_1(y)\}$ the clausal form of $\exists x\colon \pi_{sf}(\mathrm{nnf}(\neg\varphi))$ contains no dyadic literals and the clauses are well-sorted, strongly CDV-free, and regular. Clauses of the form $\{\neg p_0(x), \neg r(x, y), p_1(y)\}$ are instances of $\mathcal{C}_{inv}$. The clauses $\{\mathsf{def}(x)\}$, $\{r(x, [x\alpha])\}$, and $\{r(x, x)\}$ associated with D and T are also well-sorted, strongly CDV-free, regular clauses. The clause corresponding to 4 is an instance of $\mathcal{C}_4$. □

Lemma 5.24. *Over a finite signature there are only a finitely bounded number of condensed SF-clauses (modulo variable renaming).*

Proof. Obviously, there are only finitely many condensed clauses of the form $\mathcal{C}_4$ and by Theorem 3.25 there are only finitely many SF-regular clauses modulo variable renaming.

It remains to consider clauses of the form $\mathcal{C}_{inv}$. Here we may view the terms t and v as a global parameter. Then the dyadic literals of the form $\neg r(u, v)$ and $\neg r(w, t)$ can be viewed as monadic literals $\neg r_v(u)$ and $\neg r_t(w)$. A condensed clause consisting of monadic literals only, can contain at most exponentially many variable-disjoint subclauses, each containing at most exponentially many literals. It follows that there are only finitely many SF-clauses modulo variable renaming. □

Next we define an ordering and a selection function with respect to which the class of SF-clauses is closed under ordered resolution and ordered factoring.

Let $\succ$ be any total reduction ordering on ground terms in which the constant ϵ is the minimal term. For every ground literal L, let

$$c'_L = (\max_L, \mathrm{ar}_L, \mathrm{pol}_L, s_L)$$

where (i) $\max_L$ is the maximal argument of L with respect to $\succ$, (ii) ar_L is the arity of L, (iii) pol_L is 1, if L is negative, and 0 otherwise, and (iv) s_L is 1, if L is a dyadic literal $\neg r(s,t)$ and $s \succ t$, and 0 otherwise, The ordering $\succ_c$ over the complexity measure is then the lexicographic combination of $\succ$, $>_{\mathbb{N}}$, $>_{\mathbb{N}}$, and $>_{\mathbb{N}}$.

For example, if $s \succ t$, then the complexity of $r(s,t)$ is $(s,2,0,1)$, whereas the complexity of $\neg r(t,s)$ is $(s,2,1,0)$. Observe that the maximal term is the main criterion, and a negative literal is considered more complex than a positive literal with the same maximal term.

Note that $\succ_c$ represents a strict partial and well-founded ordering on ground literals. Any total and well-founded extension (again denoted by $\succ$) of $\succ_c$ is an admissible ordering in the sense of [9]. Let us assume for the remainder of this section that $\succ_{\mathcal{ML}}$ denotes one specific but arbitrary such ordering. The ordering $\succ_{\mathcal{ML}}$is lifted to non-ground expression in a standard manner.

The selection function $S_{\mathcal{ML}}$ is defined as follows. If a ground clause C contains a negative dyadic literal of the form $\neg r(s,t)$ such that s is an occurrence of a $\succ$-maximal term in C, then S selects one such literal. No other literals are selected by $S_{\mathcal{ML}}$. A literal L is *selected* in a non-ground clause C, $L\sigma$ is selected in $C\sigma$, for all ground instances, by a substitution σ, of an inference with $C\sigma$ by ordered resolution or ordered factoring such that the ordering constraints are satisfied.

Lemma 5.25. *Let $C_1 = \{A_1\} \cup D_1$ and $C_2 = \{\neg A_2\} \cup D_2$ be variable-disjoint clauses of the form $\mathcal{C}_{inv}$. Let σ be the most general unifier of A_1 and A_2. Let $E = (D_1 \cup D_2)\sigma$ be the conclusion of ordered resolution with premises C_1 and C_2 on the literal A_1 and $\neg A_2$. Then E is of the form $\mathcal{C}_{inv}$.*

Proof. Since there are no positive occurrences of dyadic literals in clauses of the form $\mathcal{C}_{inv}$, neither A_1 nor A_2 is a dyadic literal. If one of the negative, dyadic literals in either C_1 or C_2 is selected, then no inference step is possible on a monadic literal. Hence we may subsequently assume that no literal is selected in C_1 and C_2.

1. Suppose that A_1 is of the form $p(u_1)$. Consequently, u_1 represents the maximal term in $C_1[v_1]$. In this case no compound term $t[u_1]$ can occur in C_1. The term u_1 cannot occur as the first argument of an r literal as otherwise this literal would be selected. Therefore u_1 occurs as the right argument of an r atom, that is, $u_1 = v_1$, if there are r literals in C.

   Let $\neg A_2$ in $C_2[v_2]$ be of the form $\neg p(u_2)$. By a similar reasoning we infer that no compound terms occur in C_2 and $u_2 = v_2$, if there are r literals in C_2. The substitution σ unifies v_1 and v_2 and the conclusion E of the inference step is a clause of the form $\mathcal{C}_{inv}$.

   If $\neg A_2$ is of the form $\neg p(t_2)$ for a compound term t_2, then v_1 is a variable and the most general unifier σ is of the form $\{v_1/t_2\}$. Thus, $D_2\sigma = D_2$ and every occurrence of the variable v_1 in D_1 is replaced by t_2. The conclusion E is again a clause of the form $\mathcal{C}_{inv}$.

2. Suppose that A_1 is of the form $p(t_1)$ for a compound term t_1. The term t_1 is the maximal term in $C_1[t_1]$.

If $\neg A_2$ is of the form $\neg p(u_2)$, then this case is symmetrical to the previous case with $A_1 = p(u_1)$ and $\neg A_2 = \neg p(t_2)$. Suppose $\neg A_2$ is of the form $\neg p(t_2)$ for a compound term t_2. If both terms contain unary function symbols, these function symbols are identical. So, the term t_1 is either of the form $[v_1\alpha_1]$ or $[v_1 f(v_1)]$, while t_2 is either of the form $[v_2\alpha_2]$ or $[v_2 f(v_2)]$. Obviously, $t_1\sigma = t_2\sigma$ is again a term of this form. Consequently, the conclusion E of the inference step is again of the form $\mathcal{C}_{inv}$ and its depth does not exceed the maximal depth of its premises. □

Lemma 5.26. *Let $C_1 = \{L_1, L_2\} \cup D_1$ be a condensed clause of the form $\mathcal{C}_{inv}$ such that L_1 and L_2 are unifiable with most general unifier σ. Let $E = (\{L_1\} \cup D_1)\sigma$ be the ordered factor of C_1. Then E is of the form $\mathcal{C}_{inv}$.*

Proof. We distinguish the following cases:

1. Suppose that L_1 is of the form $\neg r(w_1, t)$ where t is a compound term with strict subterm v. Then L_2 is neither a monadic literal, nor a literal of the form $\neg r(u_2, v)$, since v and t are not unifiable. So, L_2 is of the form $\neg r(w_2, t)$ and then E is obviously of the form $\mathcal{C}_{inv}$. The case that L_1 is of the form $\neg r(u_1, v)$ is symmetrical.

2. Suppose L_1 is of the form $(\neg)p(t)$. The literal L_2 is neither identical to L_2, since C is condensed, nor of the form $(\neg)p(v)$, since v and t are not unifiable. So, L_2 is either of the form $(\neg)p(u_2)$ or of the form $(\neg)p(w_2)$ where u_2 and w_2 are variables and the unifier σ maps u_2, respectively w_2, to the term t. If C contains an r literal, then it contains a literal $\neg r(u_2, v)$ and $\neg r(w_2, t)$, respectively. The complexity measure associated with $\neg r(u_2, v)\sigma = \neg r(t, v)$ is $(t, 2, 1, 1)$ and the complexity measure associated with $\neg r(w_2, t)\sigma = \neg r(t, t)$ is $(t, 2, 1, 0)$. In both cases, these complexity measures are greater than the complexity measure $c'_{L_1\sigma} = c'_{L_2\sigma} = (t, 1, \mathrm{pol}_{L_1}, 0)$, that is, $L_1\sigma$ is not $\succ_{\mathcal{ML}}$-maximal in C and an inference step by ordered factoring is not possible. Thus, C contains no r literals and E is again of the form $\mathcal{C}_{inv}$.

3. Suppose L_1 is of the form $(\neg)p(v)$. The case that L_2 is of the form $(\neg)p(t)$ is symmetrical to the previous one, so it remains to consider that L_2 is either of the form $(\neg)p(w_2)$ or $(\neg)p(u_2)$. Without loss of generality, we assume that $L_1\sigma = L_2\sigma = L_1$. Obviously, in the presence of a monadic or dyadic literal in C with argument t, $L_1\sigma$ is not $\succ_{\mathcal{ML}}$-maximal. Thus, we can assume C contains no such literals. In the case of $L_2 = (\neg)p(w_2)$ this excludes the presence of an r literal with w_2 (and t) as argument. So, E is of the form $\mathcal{C}_{inv}$ again. If $L_2 = (\neg)p(u_2)$, then the complexity measure associated with $L_1\sigma = L_2\sigma$ is $(v, 1, \mathrm{pol}_{L_1}, 0)$. In the presence of r literals in C, there is one of the form $\neg r(u_2, v)$. So, the complexity measure associated with $\neg r(u_2, v)\sigma = \neg r(v, v)$ is $(v, 2, 1, 0)$ and $L_1\sigma$ is not $\succ_{\mathcal{ML}}$-maximal. Therefore, C does not contain r literals and E is of the form $\mathcal{C}_{inv}$.

4. Suppose L_1 is of the form $(\neg)p(u_1)$. We have already considered the case where L_2 is of the form $(\neg)p(t)$ and $(\neg)p(v)$. It remains to consider that L_2 is of the form $(\neg)p(u_2)$ or $(\neg)p(w_2)$. In the first case, the result trivially holds. In the second case, we observe that C does not contain the compound term t and no r literal $\neg r(w_2, t)$. It follows that E is of the form $\mathcal{C}_{inv}$. □

To make use of the results we obtained for extensions of K5 we show that $\succ_{\mathcal{ML}}$-maximality implies $\succ_{\mathcal{COV}}$-maximality on indecomposable, SF-regular clauses. Since $\succ_{\mathcal{ML}}$ and $\succ_{\mathcal{COV}}$ are based on the orderings $\succ_c$ and $\succ^s_{mul}$ on the complexity measures c_L and c'_L, respectively, we show the following.

Lemma 5.27. *Let $C_1 = \{L_1\} \cup D_1$ be an indecomposable, SF-regular clause, and let σ be a substitution such that $C_1\sigma$ is well-sorted and SF-regular. If $L_1\sigma$ is $\succ_c$-maximal with respect to $D_1\sigma$, then $L_1\sigma$ is $\succ^s_{mul}$-maximal with respect to $D_1\sigma$.*

Proof. Suppose C_1 contains a dyadic literal $L_2 = r(s_2, t_2)$ and let L_3 be a monadic literal in C with argument term t_3. We know that t_3 is a subterm of t_2, so $t_3\sigma$ is a subterm of $t_2\sigma$. Hence, $c'_{L_2\sigma} \succ_c c'_{L_3\sigma}$. Consequently, $L_1 = L_2$ and by Corollary 5.15, $L_2\sigma$ is also $\succ^s_{mul}$-maximal with respect to $C_1\sigma$.

Suppose C_1 contains no dyadic literal. So, L_1 is a monadic literal with argument term t_1. Let $L_3 \neq L_1$ be a literal in C_1 with argument term t_3. Since C_1 is indecomposable, neither t_1 nor t_3 is a ground term. In the proof of Lemma 5.14 we saw that L_1 is a dominating literal in C_1, that is, $t_1 \gtrsim_Z t_3$. If t_3 is a strict subterm of t_1, then it is straightforward to see that $c'_{L_1\sigma} \succ_c c'_{L_3\sigma}$ and $c_{L_1\sigma} \succ^s_{mul} c_{L_3\sigma}$ holds. Suppose t_3 is not a strict subterm of t_1. A case analysis of the syntactical form of t_1 and t_3 reveals that either $t_1 = t_3$, or $t_1 = [xf(x)]$ and $t_3 = [xg(x)]$ for a variable x and distinct function symbols f and g. In both cases L_1 and L_3 are $\succ^s_{mul}$-maximal. Hence, the result holds trivially. $\square$

Corollary 5.28. *Let $\{L\} \cup C$ be an indecomposable, SF-regular clause with dyadic literal L and let σ be a substitution such that $L\sigma$ is well-sorted and regular. Then L and $L\sigma$ are $\succ_{\mathcal{ML}}$-maximal with respect to C and $C\sigma$, respectively.*

As corollary of Lemma 5.16 and Lemma 5.17 we obtain:

Corollary 5.29. *Let C_1 and C_2 be indecomposable, SF-regular clauses. Then:*

1. *Every split component of the conclusion of an inference step by ordered resolution based on $\succ_{\mathcal{ML}}$ is an SF-regular clause.*
2. *The conclusion of an inference step by ordered factoring based on $\succ_{\mathcal{ML}}$ is an SF-regular clause.*

Lemma 5.30. *Let C_1 be an SF-regular clause and C_2 be a clause of the form $\mathcal{C}_4$. No inference step by ordered resolution based on $\succ_{\mathcal{ML}}$ is possible with premises C_1 and C_2.*

Proof. An inference step with positive premise C_2 and negative premise C_1 is not possible, since C_1 contains only monadic literals which are negative and no monadic literal in C_2 is $\succ_{\mathcal{ML}}$-maximal or selected.

Similarly, an inference step with negative premise C_2 and positive premise C_1 using the most general unifier σ can only be performed on dyadic literals. To this end, the negative literal $\neg r(x_1, x_2)$ either has to be selected in C_2, or the instance $\neg r(x_1, x_2)\sigma$ has to be maximal in C_2. Consider the positive, dyadic literal $r(t_1, t_2)$ in C_1. Without loss of generality we can assume that $\sigma = \{x_1/t_1, x_2/t_2\}$. By Condition (iii), t_1 is a subterm of t_2. Consequently, $r(x_1, [x_2\alpha])\sigma = r(t_1, [t_2\alpha]) \succ_{\mathcal{ML}} \neg r(t_1, t_2)$ and in no ground instance $C_2\theta$ of C_2 will $t_1\theta$ be the maximal term of $C_2\theta$. Thus, neither is $\neg r(x_1, x_2)$ selected nor is $\neg r(x_1, x_2)\sigma$ maximal, which renders the inference step impossible. $\square$

Lemma 5.31. *Let C_1 and C_2 be clauses of the form $\mathcal{C}_4$.*

1. *No inference step by ordered resolution based on $\succ_{\mathcal{ML}}$ is possible with premises C_1 and C_2.*
2. *No inference step by ordered factoring based on $\succ_{\mathcal{ML}}$ is possible with premise C_1.*

Proof. A resolution inference step on the monadic literals in C_1 and C_2 violates the ordering constraints. The only remaining possibility is a resolution inference step upon the literal $\neg r(x_1, x_2)$ in the negative premise C_1. Consider the positive premise $C_2 = \mathcal{P}(x_3) \cup \mathcal{P}(x_4) \cup \{\neg r(x_3, x_4), r(x_3, [x_4\alpha_2])\}$. Without loss of generality we assume that the most general unifier σ maps x_1 to x_3 and x_2 to $x_4\alpha_2$. The inference step is only admissible if the ordering constraints are satisfied. That is, $r(x_3, [x_4\alpha_2])$ has to be strictly $\succ_{\mathcal{ML}}$-maximal with respect to $\mathcal{P}(x_3) \cup \mathcal{P}(x_4) \cup \{\neg r(x_3, x_4)\}$. Consequently, $[x_4\alpha_2]$ represents the $\succ$-maximal term in $C_2\sigma$. However, this means, $[x_4\alpha_2\alpha_1]$ represents the $\succ$-maximal term in $C_1\sigma = \mathcal{P}(x_3) \cup \mathcal{P}([x_4\alpha_2]) \cup \{\neg r(x_3, [x_4\alpha_2]), r(x_3, [x_4\alpha_2\alpha_1])\}$. Thus, neither is $\neg r(x_1, x_3)$ selected in C_1 nor is $\neg r(x_3, [x_4\alpha_2])$ maximal with respect to $\mathcal{P}(x_3) \cup \mathcal{P}([x_4\alpha_2]) \cup \{r(x_3, [x_4\alpha_2\alpha_1])\}$.

Because the clause C_1 does not contain two positive or two negative dyadic literals, factoring steps are only possible on the monadic literals. However, since C_1 contains a dyadic literal, such an inference step violates the ordering constraints. □

Lemma 5.32. *Let C_1 be an indecomposable, SF-regular clause and C_2 be an indecomposable, condensed clause of form $\mathcal{C}_{inv}$. The conclusion of any inference by ordered resolution with selection based on $\succ_{\mathcal{ML}}$ and $S_{\mathcal{ML}}$ from C_1 and C_2 will be an SF-clause.*

Proof. Suppose $C_1 = \{\neg A_1\} \cup D_1$ is the negative premise in an inference step by ordered resolution. Since C_1 contains no negative, dyadic literal, A_1 is monadic and C_1 does not contain any dyadic literals. Consider the positive premise $C_2 = \{A_2\} \cup D_2$ of the form $\mathcal{C}_{inv}$. Suppose A_2 is an atom $p(u_2)$ in $\mathcal{P}(\bar{u})$. Then u_2 represents the maximal term in C_2 and the literal $\neg r(u_2, v)$ in C_2 is selected. This prevents an inference step on $p(u_2)$. Similarly, it prevents an inference step on literals in $\mathcal{P}(\bar{w})$. Suppose A_2 is an atom $p(v)$ in $\mathcal{P}(v)$. Then C_2 does not contain any compound term t and no negative, dyadic literals. Consequently, C_2 is also a SF-regular clause and by Lemma 5.16, the conclusion of an inference step by ordered resolution will be SF-regular. Finally, suppose A_2 is an atom $p(t_2)$ where t_2 is either identical to $[v\alpha_2]$ or to $[vf(v)]$. A_1 is an atom $p(t_1)$ where t_1 is an SF-regular term different from ϵ, since t_1 and t_2 are unifiable. Any term occurring in C_1 is a subterm of t_1. Thus, the conclusion of the inference step is a clause $C = C[t_1\sigma]$ of the form $\mathcal{C}_{inv}$ where $C_1\sigma$ is a subclause of $\mathcal{P}(t_1\sigma)$.

Suppose $C_1 = \{A_1\} \cup D_1$ is the positive premise in an inference step by ordered resolution. If A_1 is a monadic atom, then we proceed as in the previous case. Suppose that A_1 is a dyadic literal $r(s_1, t_1)$. The terms s_1 and t_1 are SF-regular, and s_1 is a subterm of t_1. Assume $C_2 = \{\neg A_2\} \cup D_2$. $\neg A_2$ is either a dyadic literal $\neg r(u_2, v_2)$ or a dyadic literal $\neg r(w_2, t_2)$ where v_2 is a strict subterm of t_2.

Suppose $\neg A_2 = \neg r(u_2, v_2)$. If t_1 is a compound term and C_2 contains a compound term t_2 where v_2 is a strict subterm of t_2, then the conclusion of the resolution inference step is no longer of the form $\mathcal{C}_{inv}$. So, we have to show that C_2 does not contain such a compound term. Assume the opposite. Then, v_2 is not a $\succ$-maximal term in C_2. To be able to resolve upon $\neg A_2$, u_2 has to represent a maximal term in C_2, that is, $u_2 \succeq t_2 \succ v_2$. However, we have $t_1 \succeq s_1$, which renders the resolution step impossible. We conclude that C_2 does not contain a compound term. It is straightforward to check that the conclusion of the inference step is a clause of the form $\mathcal{C}_{inv}$.

Suppose $\neg A_2 = \neg r(w_2, t_2)$. By Condition (iv) all compound terms in C_1 are equal to t_2. By Lemma 5.8 we know that $t_2\sigma$ is well-sorted and by Lemma 5.10, $\mathrm{dp}(t_2\sigma) = \max(\mathrm{dp}(t_1), \mathrm{dp}(t_2))$. So, $t_1\sigma = t_2\sigma$ is a term of the appropriate form. Furthermore $w_2\sigma = s_1\sigma$ is either identical to ϵ, w_2, v_2, or t_2. In the first two case, $(\mathcal{P}(w_2) \cup \mathcal{P}(s_1))\sigma$ is a split component of the conclusion while the rest of the conclusion is of the form $\mathcal{C}_{inv}$. In the remaining cases, we obtain a conclusion of the form $\mathcal{C}_{inv}$. □

Lemma 5.33. *Let C_1 be a clause of the form $\mathcal{C}_4$ and let C_2 be a clause of the form $\mathcal{C}_{inv}$. The conclusion of any inference by ordered resolution based on $\succ_{\mathcal{ML}}$ from C_1 and C_2 will be an SF-clause.*

Proof. In a clause of the form $\mathcal{C}_4$ the dyadic literal $r(x, [y\alpha])$ is the only literal on which resolution steps may be performed. Thus, C_1 will be the positive premise of the inference step and C_2 the negative premise.

Assume $C_2 = \{\neg A_2\} \cup D_2$. $\neg A_2$ is either a dyadic literal $\neg r(u_2, v_2)$ or a dyadic literal $\neg r(w_2, t_2)$ where v_2 is a strict subterm of t_2. In the first case, a resolution inference step on $\neg r(u_2, v_2)$ is only permissible if no compound terms occur in D_2. If C_2 contains non-empty subclauses $\mathcal{P}(u_2)$ and $\mathcal{P}(w_2)$, then the dyadic literals $\neg r(u_2, v_2)$ and $\neg r(w_2, t_2)$, respectively, are the only literals in which u_2 and w_2 occur together with some other term and prevent the application of condensation or splitting.

Suppose we resolve with a clause of form $\mathcal{C}_4$. We can assume that $u_2\sigma = x$, and $v_2\sigma = [x_2\alpha]$. If $D_2 = \mathcal{P}(\bar{u}) \cup \{\neg r(\bar{u}, v)\} \cup \mathcal{P}(v)$, then we obtain the conclusion:

$$\begin{aligned} C &= \mathcal{P}(u_2) \cup \{\neg r(u_2, x_2)\} \cup \mathcal{P}(x_2) \cup D_2\sigma \\ &= \mathcal{P}(u_2) \cup \{\neg r(u_2, x_2)\} \cup \mathcal{P}(x_2) \cup \mathcal{P}(\bar{u}) \cup \neg r(\bar{u}, [x_2\alpha]) \cup \mathcal{P}([x_2, \alpha]). \end{aligned}$$

The term $[x_2\alpha]$ is the only compound term in the conclusion and the $\succ_{\mathcal{ML}}$-maximal literal will be among those literals containing $[x_2\alpha]$. So, we have derived a clause of the form $\mathcal{C}_{inv}$. Note that if the term v_2 only occurs in the dyadic literal $\neg r(u_2, v_2)$, then the derived clause C is (a variant of) a subclause of C_2.

If we resolve upon $\neg r(w_2, t_2)$ in C_2 and a clause of the form $\mathcal{C}_4$, then we can assume that $x_1\sigma = w_2$, $x_2\sigma = v_2$, and $[x_2\alpha]\sigma = t_2$ (note that v_2 may be a constant and t_2 may be of the form $[v_2 f(v_2)]$). The conclusion of the inference step is:

$$C = \mathcal{P}(w_2) \cup \{\neg r(w_2, v_2)\} \cup \mathcal{P}(v_2) \cup D_2.$$

Obviously, C is of the form $\mathcal{C}_{inv}$. □

Theorem 5.34.
Let Σ be any combination of the axiom schemata 4, D, *and* T. *Let φ be a modal formula in negation normal form and let N be the set of clauses obtained by applying $\mathrm{Cls}\Xi\overline{\Pi}^{\Sigma}_{sf}$ to φ. Any derivation from N by ordered resolution and ordered factoring with selection based on the ordering $\succ_{\mathcal{ML}}$ and the selection function $S_{\mathcal{ML}}$, terminates.*

Proof. By Theorem 5.23 every clause in N is an SF-clause. By Corollary 5.29 and Lemmata 5.30, 5.31, 5.32, and 5.33 the class of SF-clauses is closed under inference steps by ordered resolution and ordered factoring based on $\succ_{\mathcal{ML}}$ and $S_{\mathcal{ML}}$. By Theorem 5.24 the class of SF-clauses is finitely bounded. $\square$

Let us consider an example. The formula

$$\varphi_1 = \Box(p_1 \vee p_2) \wedge \Diamond(\Box(\neg p_1 \vee p_2) \wedge \Diamond\Diamond\neg p_2)$$

is unsatisfiable in $\mathsf{K4}$. The clausal form of $\Xi\overline{\Pi}^{4}_{sf}(\varphi_1)$ includes among others the following clauses.

$$\begin{array}{ll}
(18) & \{\neg\mathsf{def}(x), r(x,[x\alpha])\} \\
(19) & \{\neg\mathsf{def}(x), \neg r(x,y), r(x,[y\alpha])\} \\
(20) & \{\neg q_1(x), \mathsf{def}(x)\} \\
(21) & \{\neg q_1(x), \neg p_2([xf_1(x)])\} \\
(22) & \{\neg q_2(x), \mathsf{def}(x)\} \\
(23) & \{\neg q_2(x), q_1([xf_2(x)])\} \\
(24) & \{\neg q_3(x), \neg p_1(x), p_2(x)\} \\
(25) & \{\neg q_4(x), \neg r(x,y), q_3(y)\} \\
(26) & \{\neg q_5(x), q_2(x)\} \\
(27) & \{\neg q_5(x), q_4(x)\} \\
(28) & \{\neg q_6(x), \mathsf{def}(x)\} \\
(29) & \{\neg q_6(x), q_5([xf_3(x)])\} \\
(30) & \{\neg q_7(x), p_1(x), p_2(x)\} \\
(31) & \{\neg q_8(x), \neg r(x,y), q_7(y)\} \\
(32) & \{\neg q_9(x), q_6(x)\} \\
(33) & \{\neg q_9(x), q_8(x)\} \\
(34) & \{\ \ q_9(\epsilon)\}.
\end{array}$$

Note that $q_8(x)$ can be interpreted as '$\Box(p_1 \vee p_2)$ holds at world x'. The literals $q_4(x)$, $q_2(x)$, and $q_1(x)$ have an analogous meaning for the subformulae $\Box(\neg p_1 \vee p_2)$, $\Diamond\Diamond\neg p_2$, and $\Diamond\neg p_2$, respectively. Recall that condensation is performed implicitly in the "Deduce" expansion rule.

$$\begin{array}{lll}
[(24)2,\mathrm{R},(30)2] & (35) & \{\neg q_3(x), \neg q_7(x), p_2(x)\} \\
[(35)3,\mathrm{R},(21)2] & (36) & \{\neg q_3([yf_1(y)]), \neg q_7([yf_1(y)]), \neg q_1(y)\} \\
[(25)2,\mathrm{R},(19)3] & (37) & \{\neg\mathsf{def}(x), \neg q_4(x), \neg r(x,y), q_3([y\alpha])\} \\
[(31)2,\mathrm{R},(19)3] & (38) & \{\neg\mathsf{def}(x), \neg q_8(x), \neg r(x,y), q_7([y\alpha])\} \\
[(37)4,\mathrm{R},(36)1] & (39) & \{\neg\mathsf{def}(x), \neg q_4(x), \neg q_1(y), \neg r(x,y), \neg q_7([yf_1(y)])\} \\
[(39)5,\mathrm{R},(38)4] & (40) & \{\neg\mathsf{def}(x), \neg q_4(x), \neg r(x,y), \neg\mathsf{def}(z), \neg q_8(z), \neg r(z,y), \neg q_1(y)\}.
\end{array}$$

Clause (40) expresses that if $\Box(\neg p_1 \vee p_2)$ holds at a world x, $\Box(p_1 \vee p_2)$ holds at world z, the worlds x and z are not dead-ends, and there is a world y which is accessible from both x and z, then $\neg\Diamond\neg p_2$, that is $\Box p_2$, holds in y. No assumptions are made as to whether x is accessible from z, or vice versa. This property cannot be expressed without the object language containing explicit representations of (universally quantified) worlds and the accessibility relation. This is one of the main factors which enables us to maintain all the information which needs to be derived in the restricted form of $\mathcal{C}_{inv}$. The remainder of the refutation is as follows.

[(40)3,R,(18)2]	(41)	$\{\neg\mathsf{def}(x), \neg q_4(x), \neg\mathsf{def}(z), \neg q_8(z), \neg r(z, [x\alpha]), \neg q_1([x\alpha])\}$
[(41)6,R,(23)2]	(42)	$\{\neg\mathsf{def}(x), \neg q_4(x), \neg q_2(x), \neg\mathsf{def}(z), \neg q_8(z), \neg r(z, [xf_2(x)])\}$
[(42)4,R,(19)3]	(43)	$\{\neg\mathsf{def}(x), \neg q_4(x), \neg q_2(x), \neg\mathsf{def}(z), \neg q_8(z), \neg r(z, x)\}$
[(43)6,R,(18)2]	(44)	$\{\neg\mathsf{def}([z\alpha]), \neg q_4([z\alpha]), \neg q_2([z\alpha]), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(44)1,R,(22)2]	(45)	$\{\neg q_4([z\alpha]), \neg q_2([z\alpha]), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(45)2,R,(28)2]	(46)	$\{\neg q_4([z\alpha]), \neg q_5([z\alpha]), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(46)2,R,(29)2]	(47)	$\{\neg q_4([zf_3(z)]), \neg q_6(z), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(47)1,R,(27)2]	(48)	$\{\neg q_5([zf_3(z)]), \neg q_6(z), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(48)1,R,(29)2]	(49)	$\{\neg q_6(z), \neg\mathsf{def}(z), \neg q_8(z)\}$
[(49)2,R,(28)2]	(50)	$\{\neg q_6(z), \neg q_8(z)\}$
[(50)1,R,(32)2]	(51)	$\{\neg q_9(z), \neg q_8(z)\}$
[(51)2,R,(33)2]	(52)	$\{\neg q_9(z)\}$
[(52)1,R,(34)1]	(53)	$\bot$.

It is interesting to note the close corresponds to the example derivation in [46].

5.4 Conclusion

In this chapter we have considered various embeddings of modal logics into fragments of first-order logic. We have been able to derive decidability results for the relational translation and semi-functional translation method. Our results for the latter include also extensions with the axiom schema 4 which are of particular interest. For the relational translation, a decision procedure based on ordered chaining is presented in Ganzinger et al. [46]. Derivations by these decision procedures do not resemble derivations by tableaux-based decision procedures for K4, as is illustrated by the example at the end of Section 5.3.

Based on the consideration in Section 4.5 we have been able to describe a decision procedure using a refinement of resolution based solely on a particular selection function which is able to simulate tableaux-based decision procedures for extension of K.

All the results in Section 5.2 extend to multi-modal logics with one important exception: The independent join of S5 with other modal logics. In this case it is no longer sound to use Universality as the relational or semi-functional frame property corresponding to the combination of the axiom schemata T and 5. A resolution-based decision procedure for the independent join of S5 with other modal logics is not only interesting to close a final gap in the range of modal logics the approach presented in the chapter is able to cover. It is also closely related to the problem of obtaining resolution-based decision procedures for temporal logics of knowledge and belief. Currently, a resolution-based decision procedure for the combination of propositional linear temporal logic with a single S5 modality exists [31]. This procedure does not use an embedding into first-order logic. It is open how to obtain a decision procedure in the presence of multiple S5 modalities.

As stated in the introduction, the class $\overline{\mathrm{K}}$ and the guarded fragment can be considered to be generalisations of a range of modal logics not including K4. A topic for future research is to look at generalisations of K4 by relational operations or by allowing for predicates of arbitrary arity. First results in this directions have been obtained by Ganzinger, Meyer and Veanes [47].

Chapter 6

Performance evaluation

Besides the resolution-based decision procedures described in Chapter 5 there are various other procedures for establishing the theoremhood and satisfiability of modal formulae. To name just a few: Basin, Matthews, and Viganò [17] present an approach based on natural deduction, Fitting [42] and Baader and Hollunder [5] make use of tableaux calculi, Giunchiglia and Sebastiani [51] extend the DPLL algorithm [25, 24] to multi-modal logic $\mathsf{K}_{(m)}$.

The simulation results of chapter 4 and 5 use an analytical approach to shed some light on the relative performance of resolution-based decision procedures compared to tableaux-based decision procedures, In this chapter we compare various implementations of the approaches mentioned above on an empirical bases. Shorter versions of this chapter are [79, 80, 82]. Related work by Hustadt, Schmidt and Weidenbach also appears in [81, 85]. Other related experiments have been done by Baader, Hollunder, Nebel, Profitlich, and Franconi [6], E. Giunchiglia, F. Giunchiglia, Sebastiani and Tachella [51, 52, 50], Heuerding and Schwendimann [63], Horrocks and Patel-Schneider [70, 71], and Paramasivam and Plaisted [110].

6.1 Analytical versus empirical performance studies

There are two basic approaches for studying the performance of algorithms and their implementations: An analytical one and an empirical one. The following discussion of the pros and cons of the two approaches elaborates considerations by Hooker [67, 68].

During the last decades the analytical approach has matured into a well-developed science. One of the major contributions is the insight that there exist a wide range of problem classes which have an inherent difficulty that we cannot overcome by any clever algorithm. For example, the satisfiability problem of boolean formulae is **NP**-complete and the satisfiability problem of modal logic formulae in the modal logic K is **PSPACE**-complete. Besides the rather fine grained hierarchy of complexity classes, there is the coarse division into tractable and intractable class: A class is said to be *tractable* if it can be solved in polynomial time, otherwise it is *intractable*. So, in general problems from the classes **NP** and **PSPACE** are intractable. However, underlying these characterisations are reflections on the worst-case behaviour of algorithms on a given problem class.

It is natural to consider the average-case behaviour of an algorithm on a given problem class instead. The average-case analysis considers random instances of a problem class, that is, it considers a problem class together with a probability density function μ which assigns probabilities to instances of the problem class. The first problem that arises is caused by the sensitivity of an average-case analysis to the choice of μ. If the density function μ decreases faster than $2^{-|x|}$ where $|x|$ is the length of the instance x, then all **NP**-complete problems are solvable in polynomial time on μ-average [124]. Even if this is not the case, there are a wide range of density functions which are unreasonable. The most famous example is the density function underlying the result of Goldberg [53, 54, 55]. Goldberg has shown that the satisfiability problem of boolean formulae can be solved by the DPLL procedure in polynomial time in the average-case. Subsequently, Franco and Paull [43] proved that based on the density function of Goldberg even a fixed number of guesses will reveal the satisfiability of a boolean formula with probability 1. So the good result obtained by Goldberg is due to the density function he assumed and not a feature of the DPLL algorithm. In addition, Franco and Paull have shown that for a more reasonable density function on the class of boolean formulae, a variant of the DPLL procedure needs exponential time in the average case. However, the paper of Franco and Paull also shows the limitations of the analytical approach: The variant of the DPLL algorithm they have analysed utilises a rather simple heuristic for the application of the splitting rule and does not use the pure literal rule. So, they had to simplify the DPLL algorithm to provide grounds for an analytical study. In this light, it seems to be rather unlikely that resolution-based decision procedures for subclasses of first-order logic are a suitable subject for an analytical study on the basis of the techniques currently available for such an enterprise.

At first sight, an empirical analysis sidesteps these problems. Commonly, such an analysis requires collecting a set of benchmark problems and comparing the performances of algorithms on them. This task seems straightforward. Taking a closer look we identify several difficulties.

First, an empirical analysis based on a set of benchmark problems is of comparative and competitive nature. It does not make much sense to report the performance of a single algorithm or its implementation on some set of benchmark problems, since this kind of report does not provide an evaluation of the quality of the algorithm. Instead we need empirical results for more than one algorithm.

Second, we cannot obtain empirical results for the algorithm directly, but we need some implementation of it. This raises the question to which level of sophistication we have to drive the coding of the algorithms. The problem is worse since we have to compare several algorithms. So we need implementations for every algorithm under examination. It seems hardly possible to afford implementing all the algorithms on our own. On the other hand, if the implementations are contributed by several researchers one cannot expect that they follow the same design principles and design goals. It is likely that different programming languages have been used, that the software design is vastly different, and we cannot even assume that providing the best performance possible has been the major design goal. It goes without saying that this divergence has serious effects on the performance provided by the implementations.

Third, selecting a set of benchmark problems is a non-trivial task. By definition a set of benchmark problems should be representative for the problems occurring in the 'real world'. Imagine that many 'real world problems' exist. Although this seems to be an ideal situation for selecting an appropriate set of benchmark problems there is almost certainly a catch to it. There already exist well-developed (commercial) products solving these problems. An empirical analysis has to be comparative. We have to compare our new algorithm (more specifically its implementation) with existing ones. Such a comparison is often discouraging. For example, Lustig, Marsten, and Shanno [93] note that in the field of linear programming "CPLEX and IBM's OSL Release 2 simplex code, represent such a major improvement in simplex technology that if the original interior point implementations had been tested against these codes, it might well have discouraged further development of interior point technology." In such a situation the existing code is tuned for the benchmark problems. A quick implementation of a new algorithm will not be competitive.

We are not much better off if not enough 'real world problems' exist to set up a benchmark suite. In this situation one usually seeks and acquires problems which have been used by other researchers for benchmark purposes. Besides the obvious drawback that these problems need not resemble 'real world problems', there is an additional disadvantage. Since somebody has reported the performance of an algorithm on a collection of problems, the result must have been encouraging, that is the algorithm performed very well on the majority of the problems. Furthermore, it is common habit to omit results for those problems where an algorithm has failed to perform well enough. If we construct a benchmark suite from publications, the suite will be tuned for the existing algorithms. Again it will be hard to outperform the existing code. This kind of 'tuning' can become even worse, if we are working in an area where the specific formulation of a problem can drastically influence the performance of an algorithm. So we have to decide whether we change the formulation of the benchmark problem to make them better suited for the new algorithm.

An alternative way to set up a benchmark suite is to randomly generate test problems. One of the first examples of this approach is Hooker's empirical comparison of a resolution-based method and a cutting plane inference algorithm for propositional logic [66]. The test problems were generated randomly according to a density function similar to the one chosen by Goldberg for his analysis of the average-case complexity of the DPLL algorithm. The problems with this approach are apparent. Again we cannot expect that the randomly generated test problems resemble 'real world problems'. Furthermore, not every density function μ gives rise to a suitable benchmark suite. In particular, the result by Franco and Paull [43] indicates that the instance distribution used in Hooker's comparison [66] might not be appropriate.

Last but not least, benchmarking rarely improves our comprehension of the algorithms we develop. Benchmarking may tell us that a particular algorithm performs better than other ones, but it does not reveal why.

6.2 Scientific testing and scientific benchmarking

Hooker [68] proposes the following experimental design to overcome the problems of empirical analysis: We should start by setting up a list of hypotheses about factors that could affect the performance of the algorithm. Some of these factors can be related to features of the problems we like to deal with, e.g. size of the problems, a specific structure of the problems, etc. Other factors can be related to the algorithm itself, e.g. its inference rules, its heuristics, its redundancy checks, etc. When formulating a hypothesis we should use some abstract measure to describe the effect of a factor, for example, the number of nodes in a tableau or the length of a refutation using resolution, instead of simply relying on running time. For each factor we set up a benchmark suite suitable for verifying our hypothesis about the influence of the factor on the performance of the algorithm. The benchmark problems can be purely artificial and need not be related to any 'real world' problems. So randomly generated problems are well suited for this purpose. Of course, we still have to be careful that the generated problems possesses the problem characteristics needed for the test. Since we do not intend to perform a competitive test, it is not necessary to have an efficient implementation of our algorithm at hand. The only necessary prerequisite is that we are able to alter the implementation to test our hypotheses about the algorithm itself, for example, we need to be able to turn off specific inference rules and adjust specific heuristics.

Hooker calls this kind of experimental design *scientific testing*. He claims that scientific testing solves or alleviates all the problems of empirical algorithm analysis described above. However, the scope of the scientific testing methodology is rather limited: It is not suitable for testing hypotheses concerning fundamentally different algorithms. It is an implicit assumption of the scientific testing methodology that we restrict ourselves to variations of one basic algorithm. For example, we might test the effect of various heuristics for choosing the next variable for extending the partial truth assignment on the performance of a DPLL algorithm or we might test the influence of the **KE**-rule on propositional tableaux calculi. For an elaborated example of scientific testing see Gent and Walsh [49].

In contrast, consider that we want to compare a tableaux-based algorithm to a resolution-based algorithm. The simulation results of chapter 4 and 5 shed some light on the relative performance we can expect of resolution-based algorithms compared to tableaux-based algorithms provided that the resolution-based algorithm follows a particular strategy matching the one used by the tableaux-based algorithm. If this is not the case, and the algorithms follow unrelated strategies, the analytical results do not predict the relative performance we will measure in an experiment. Furthermore, for full-fledged algorithms including redundancy elimination techniques it becomes difficult to establish a common abstract measure of the performance. The computational effort for an inference step by the tableaux-based algorithm does no longer correspond to the computational effort for an inference step by the resolution-based algorithm. However, without an abstract performance measure, the actual implementation is getting important again.

Since it is our main objective in this chapter to compare decision procedures for modal logic based on a variety of different calculi and the procedures make use of various optimisation techniques, the approach of scientific testing is not directly applicable. To compare fundamentally different algorithms we have to use the methods involved in classical benchmarking, that is, we have to compare the performance of implementations of the algorithms we are interested in. However, the aim of benchmarking should not be to implicate the superiority of a specific approach. Instead benchmarking should improve our understanding of the influence various elements of a decision procedure and their interdependencies have on the overall performance of the procedure. The benchmark suite has to be designed in a way that enables us to form and to test hypotheses about the design factors influencing the performance of an algorithm. An empirical comparison of algorithms following this approach will be called *scientific benchmarking*. The next section presents such a comparison and shows some of the pitfalls one might encounter.

6.3 Theorem provers for the modal logic $\mathsf{K}_{(m)}$

The language of the multi-modal logic $\mathsf{K}_{(m)}$ is that of propositional logic plus m additional modal operators $\Box_i$, $1{\leq}i{\leq}m$. A *formula* of $\mathsf{K}_{(m)}$ is a boolean combination of propositional and modal atoms. A *modal atom* is an expression of the form $\Box_i\psi$, $1{\leq}i{\leq}m$, and ψ is a formula of $\mathsf{K}_{(m)}$. In contrast to Section 5.1, we will consider $\Diamond_i\psi$ to be an abbreviation for $\neg\Box_i\neg\psi$.

This section describes the inference mechanisms of Ksat, $\mathcal{KRIS}$, the Logics Workbench and a variant of the optimised functional translation approach.

Ksat [51] extends the DPLL algorithm for testing the satisfiability of propositional formulae to $\mathsf{K}_{(m)}$. Its basic algorithm, called Ksat0, is based on the following two procedures:

KDP: Given a modal formula ϕ, this procedure generates a partial truth assignment μ for the propositional and modal atoms in ϕ which renders ϕ true propositionally. This is done using a decision procedure for propositional logic.

KM: Given a modal formula ϕ and an assignment μ computed by KDP, let $\Box_i\psi_{ij}$ denote any modal atom in ϕ which is assigned false by μ, that is, $\mu(\Box_i\psi_{ij}) = \bot$ and $\Box_i\phi_{ik}$ any modal atom that is assigned true by μ, that is, $\mu(\Box_i\phi_{ik}) = \top$. The procedure checks for each index i, $1 \leq i \leq m$, and each j whether the formula

$$\varphi_{ij} = \bigwedge_k \phi_{ik} \wedge \neg\psi_{ij}$$

is satisfiable. This is done with KDP. If at least one of the formulae φ_{ij} is not satisfiable, then KM *fails on* μ, otherwise it *succeeds*.

Ksat0 starts by generating a truth assignment μ for ϕ using KDP. If KM succeeds on μ, then ϕ is $\mathsf{K}_{(m)}$-satisfiable. If KM fails on μ, we have to generate a new truth assignment for ϕ using KDP. If no further truth assignment is found, then ϕ is $\mathsf{K}_{(m)}$-unsatisfiable.

The decision procedure KDP for propositional logic can be described by a set of transition rules on ordered pairs $P \triangleright S$ where P is a sequence of pairs $\langle\phi, \mu\rangle$ of a modal formula ϕ and a partial truth assignment μ, and S is a set of satisfying truth assignments.

`dp_sol`: $$\frac{\langle\top, \mu\rangle \mid P \triangleright S}{P \triangleright S \cup \{\mu\}}$$

`dp_clash`: $$\frac{\langle\bot, \mu\rangle \mid P \triangleright S}{P \triangleright S}$$

`dp_unit`: $$\frac{\langle\phi[c], \mu\rangle \mid P \triangleright S}{\langle\phi', \mu \cup \{c = \top\}\rangle \mid P \triangleright S}$$

if c is a unit clause in ϕ and ϕ' is the result of replacing all occurrences of c and $\overline{c}$ by $\top$ and $\bot$, respectively, followed by boolean simplification.

`dp_split`: $$\frac{\langle\phi[\psi], \mu\rangle \mid P \triangleright S}{\langle\phi[\psi] \wedge p, \mu\rangle \mid \langle\phi[\psi] \wedge \neg p, \mu\rangle \mid P \triangleright S}$$

if `dp_unit` cannot be applied to $\langle\phi[\psi], \mu\rangle$, ψ is a propositional or modal atom.

The symbol $\mid$ denotes concatenation of sequences. $\overline{\phi}$ denotes the complementary formula of ϕ, for example $\overline{\neg p} = p$ and $\overline{\Box_i p} = \Diamond_i \neg p$.

Starting with $\langle\phi, \emptyset\rangle \triangleright \emptyset$, applying the inference rules exhaustively will result in $\emptyset \triangleright S$ where S is a complete set of partial truth assignments making ϕ true. The crucial nondeterminism of the procedure is the selection of the splitting 'variable' ψ in the transition rule `dp_split`. Ksat employs the heuristic that selects an atom with a maximal number of occurrences in ϕ.

At any point of time the computation in KDP can be interrupted and KM can be called with the partial truth assignment μ constructed so far. If KM fails on μ, then is not necessary to continue the completion of μ by KDP. KSAT0 calls KM before every application of the `dp_split` rule.

It is important to note that not every propositional theorem prover can be the basis for KSAT0. Quite the contrary, completeness of KSAT0 can be lost easily, even if the underlying propositional theorem prover is complete. Suppose that we add the pure literal rule to the DPLL procedure described above. That is, whenever an atom ψ occurs only positively (respectively negatively) in ϕ, we can add $\{\psi = \top\}$ (respectively $\{\psi = \bot\}$) to the truth assignment and replace all occurrences of ψ by $\top$ (respectively $\bot$). The application of the pure literal rule preserves satisfiability and can be applied eagerly to ϕ. Now consider the formula

$$\begin{aligned}\phi_1 = {} & (p \vee q \vee \neg\Box_1(p \vee \neg p)) \wedge \\ & (\neg p \vee \neg q \vee \neg\Box_1(p \vee \neg p)).\end{aligned}$$

There is one pure literal in ϕ_1, namely $\Box_1(p \vee \neg p)$, which occurs only negatively in ϕ_1. So we assign $\bot$ to $\Box_1(p \vee \neg p)$ and replace all occurrences of $\Box_1(p \vee \neg p)$ by $\bot$. After simplifying the resulting formula we get the formula $\top$. We have arrived at a truth assignment rendering ϕ true. Due to the eager application of the pure literal rule, this is the only truth assignment our procedure computes. In a second step we have to check using KM that $\neg(p \vee \neg p)$ is satisfiable. This is obviously not the case. Since KDP with the pure literal rule does not produce any additional truth assignments for ϕ, KSAT concludes that ϕ is unsatisfiable. However, ϕ is satisfiable with the truth assignment $\{p = \top, q = \bot\}$. For similar reasons, ordered resolution (with selection) or particular refinements of semantic tableaux [1] are not suitable for combination with KM. So, legitimate optimisations of the decision procedure for propositional logic can render KSAT0 incomplete. That is, not every technique developed for such decision procedure carries over to modal logic.

We will illustrate the four approaches to satisfiability testing under consideration by way of one satisfiable formula, namely

$$\psi = \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q).$$

Example 6.1:
Figure 6.1 depicts the derivation tree of KSAT for the formula ψ. In the first step the procedure KDP applies the `dp_unit` rule to the unit clause $\neg\Box_1(p \vee r)$. All occurrences of $\neg\Box_1(p \vee r)$ are replaced by $\top$ while all occurrences of $\Box_1(p \vee r)$ are replaced by $\bot$. The resulting formula $\top \wedge (\Box_1 p \vee \Box_1 q)$ is simplified to $\Box_1 p \vee \Box_1 q$ to which only the `dp_split` rule of KDP is applicable. Before any application of the `dp_split` rule, KSAT calls the procedure KM with the current truth assignment. Here, KM is used to prove that $\mu_0 = \{\Box_1(p \vee r) = \bot\}$ is $\mathsf{K}_{(m)}$-satisfiable. To this end, KM shows that $\neg(p \vee r)$ is satisfiable. This is done by KDP with two applications of the `dp_unit` rule to $\neg(p \vee r)$. Only now, the `dp_split` rule is actually applied to $\Box_1 p \vee \Box_1 q$. We assume that $\Box_1 p$ is the split variable. So, we have to show that either $\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ or $\neg\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ is satisfiable. KDP will first consider the formula $\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$. Obviously, we can apply the `dp_unit` rule to propagate the unit clause $\Box_1 p$. This step immediately reveals that the formula is satisfiable. That is, one satisfying truth assignment is $\mu_1 = \{\Box_1(p \vee r) = \bot, \Box_1 p = \top\}$. KSAT proceeds with KM to show that $\neg\Box_1(p \vee r) \wedge \Box_1 p$ is $\mathsf{K}_{(m)}$-satisfiable. This is done by showing that $\neg(p \vee r) \wedge p$ is satisfiable. But KDP will reveal with an application of the `dp_unit` rule to

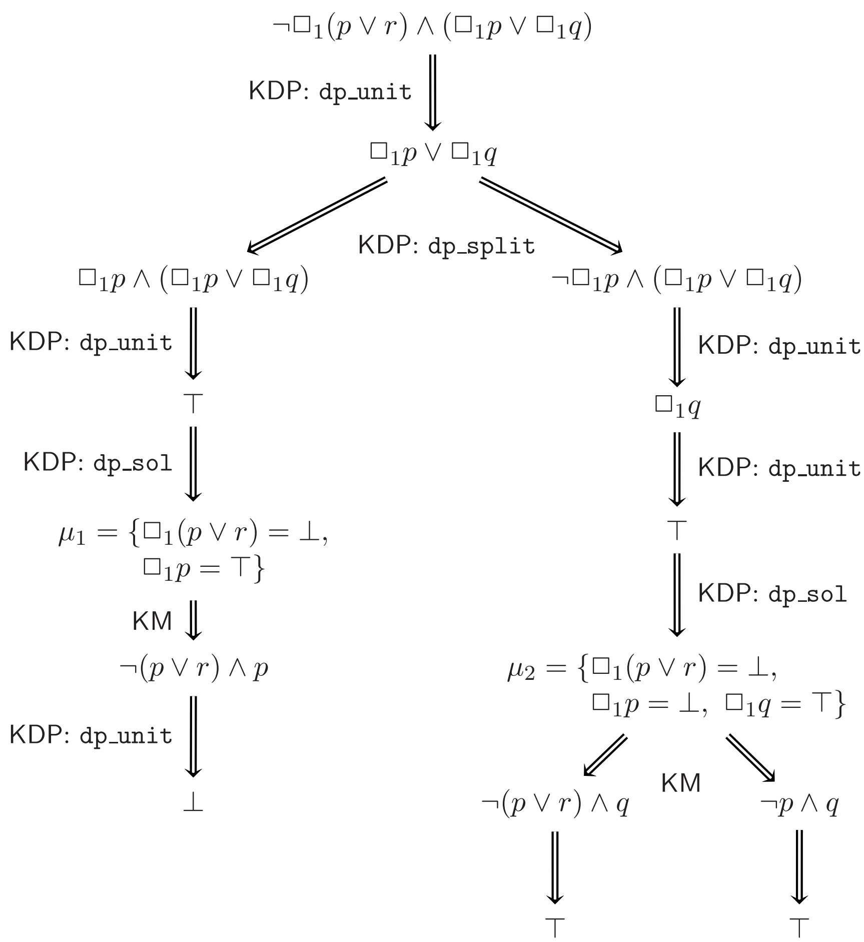


Figure 6.1: Sample derivation of Ksat

the unit clause p in $\neg(p \vee r) \wedge p$ that the formula is unsatisfiable. Thus, $\neg\Box_1(p \vee r) \wedge \Box_1 p$ is not $\mathsf{K}_{(m)}$-satisfiable. Consequently, KDP will continue with the second formula $\neg\Box_1 p \wedge (\Box_1 p \vee \Box_1 q)$ generated by the `dp_split` rule. Here two applications of the `dp_unit` rule to the unit clauses $\neg\Box_1 p$ and $\Box_1 q$ yield a second truth assignment $\mu_2 = \{\Box_1(p \vee r) = \bot, \Box_1 p = \bot, \Box_1 q = \top\}$. Again Ksat continues with KM. Note that μ_2 assigns $\bot$ to two modal atoms, namely $\Box_1(p \vee r)$ and $\Box_1 p$. Therefore, KM checks the satisfiability of two propositional formulae, that is, $\neg(p \vee r) \wedge q$ and $\neg p \wedge q$. For both formulae KDP immediately verifies their satisfiability. So, KM succeeds on μ_2 which completes the computation by Ksat. We conclude that $\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$ is satisfiable.

While Ksat abstracts from the modal part of formulae to employ decision procedures for propositional logic, $\mathcal{KRIS}$ manipulates modal formulae directly. More precisely, the inference rules of $\mathcal{KRIS}$ are relations on sequences of sets of labelled modal formulae of the form $w{:}\psi$

where w is a label chosen from a countably infinite set of labels Γ and ψ is modal formula. The set of inference rule contains an *elimination rule* for each of the operators $\top$, $\wedge$, $\vee$, and $\Diamond_i$. In addition, there are two elimination rules $\bot\texttt{C}$ and $\wedge\texttt{C}$ which remove obviously unsatisfiable sets of labelled formulae. For improved readability we write $w{:}\psi, C$ instead of $\{w{:}\psi\} \cup C$.

$$\bot\texttt{C}:\ \frac{w{:}\bot, C \ \mid\ S}{S}$$

$$\wedge\texttt{C}:\ \frac{w{:}\phi, w{:}\overline{\phi}, C \ \mid\ S}{S}$$

$$\top\texttt{E}:\ \frac{w{:}\top, C \ \mid\ S}{C \ \mid\ S}$$

$$\wedge\texttt{E}:\ \frac{w{:}\phi \wedge \psi, C \ \mid\ S}{w{:}\phi, w{:}\psi, C \ \mid\ S}$$

$$\vee\texttt{E}:\ \frac{w{:}\phi \vee \psi, C \ \mid\ S}{w{:}\phi, C \ \mid\ w{:}\psi, C \ \mid\ S}$$

if $w{:}\phi \vee \psi, C$ has been simplified by $\vee\texttt{S}_0$ and $\vee\texttt{S}_1$

$$\Diamond_i\texttt{E}:\ \frac{w{:}\Diamond_i\phi, D, C \ \mid\ S}{v{:}\phi \wedge \psi_1 \wedge \ldots \wedge \psi_n, D, C \ \mid\ S}$$

if $D = w{:}\Box_i\psi_1, \ldots, w{:}\Box_i\psi_n$, C does not contain any $w{:}\Box_i\psi$, and none of the other rules can be applied to C, and v is a new label from Γ.

Given a modal formula ϕ, the input sequence for $\mathcal{KRIS}$ is the singleton set $w_0{:}\phi'$, where w_0 is a label chosen from a countably infinite set of labels Γ and ϕ' is the modal negation normal form of ϕ. If $\mathcal{KRIS}$ arrives at a sequence $C \ \mid\ S$ such that no transformation rule can be applied to C, then the original formula ϕ is satisfiable. Otherwise the transformation rules will eventually reduce $w_0{:}\phi'$ to the empty sequence and ϕ is unsatisfiable. The rules $\bot\texttt{C}$, $\wedge\texttt{C}$, $\top\texttt{E}$, $\wedge\texttt{E}$, are applied exhaustively before any application of one of the elimination rules for $\vee$ and $\Diamond_i$. The $\top\texttt{E}$ rule is not necessary for the completeness of the set of rules. It is straightforward to see that $\mathcal{KRIS}$ is a variant of the tableaux-based decision procedure described in Section 4.5.

In addition to the inference rules, $\mathcal{KRIS}$ has two simplification rules, namely

$$\vee\texttt{S}_0:\qquad w{:}\phi \vee \psi, w{:}\phi, C \rightarrow w{:}\phi, C$$

$$\vee\texttt{S}_1:\qquad w{:}\phi \vee \psi, w{:}\overline{\phi}, C \rightarrow w{:}\psi, w{:}\overline{\phi}, C$$

These are applied only immediately before an application of the $\vee\texttt{E}$ rule and then they are applied only to the labelled formula $w{:}\phi \vee \psi$ to which we want to apply the $\vee\texttt{E}$ rule.

As far as the application of the $\vee\texttt{E}$ rule is concerned, $\mathcal{KRIS}$ actually considers the sets of labelled formulae as sequences and chooses the first disjunction in this sequence. To give a simple example, consider the formula ϕ_2 given by $(p \wedge \neg p) \vee \top$. Since ϕ_2 is in negation normal form, we start with the initial sequence

$$w_0{:}(p \wedge \neg p) \vee \top.$$

The only rule applicable is $\vee\,\mathtt{E}$ which generates the structure

$$w_0{:}(p \wedge \neg p) \;\mid\; w_0{:}\top.$$

For the reason that sequences are always processed from left to right, $w_0{:}(p \wedge \neg p)$ will be considered first. Only $\wedge\,\mathtt{E}$ is applicable transforming the sequence to

$$w_0{:}p, w_0{:}\neg p \;\mid\; w_0{:}\top.$$

Now we can apply the $\wedge\,\mathtt{C}$ rule to eliminate the first set of labelled formulae and get

$$w_0{:}\top.$$

A final application of the $\top\,\mathtt{E}$ rule reveals the sequence containing the empty set. No further rule can be applied. Since we have not arrived at the empty sequence, ϕ is satisfiable.

As the formula $(p \wedge \neg p) \vee \top$ is logically equivalent to $\top$, its satisfiability can be shown by a single application of the $\top$-elimination rule. However, $\mathcal{KRIS}$ has no simplification rules beside $\vee\,\mathtt{S}_0$ and $\vee\,\mathtt{S}_1$. In particular, $\mathcal{KRIS}$ does not simplify boolean expressions using the simplification rules of the preprocessing procedure that Giunchiglia and Sebastiani use in conjunction with KSAT which we discuss later (see Table 6.2 on page 132).

The condition that the $\Diamond_i\,\mathtt{E}$ rule can be applied only if none of the other rules can be applied to the set of labelled formulae under consideration is necessary for the completeness of the system. To illustrate the reason, consider the formula $\phi_3 = \neg q \wedge \Diamond_1 \neg p \wedge (\Box_1 p \vee q)$. Starting with

$$w_0{:}\neg q \wedge \Diamond_1 \neg p \wedge (\Box_1 p \vee q)$$

a sequence of applications of the $\wedge$-elimination rule will derive

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \vee q.$$

Suppose we apply the $\Diamond_1$-elimination rule before eliminating the occurrence of the $\vee$-operator in $w_0{:}\Box_1 p \vee q$. The resulting system is

$$w_0{:}\neg q, w_1{:}\neg p, w_0{:}\Box_1 p \vee q.$$

The application of $\vee$-elimination rule is still possible and we get

$$w_0{:}\neg q, w_1{:}\neg p, w_0{:}\Box_1 p \;\mid\; w_0{:}\neg q, w_1{:}\neg p, w_0{:}q.$$

Now, no further application of any inference rule is possible. Since, we have not derived the empty sequence, we would conclude that ϕ_3 is satisfiable. But, it is not. If we apply the $\vee$-elimination rule to

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \vee q$$

the resulting sequence contains two sets of labelled formulae

$$w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}\Box_1 p \;\mid\; w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}q.$$

The only rule applicable to the first system is the $\Diamond_1$-elimination rule. The rule will replace the occurrence of $w_0{:}\Diamond_1 \neg p$ with $w_1{:}\neg p \wedge p$. We have now derived the sequence

$$w_0{:}\neg q, w_1{:}\neg p \wedge p, w_0{:}\Box_1 p \;\mid\; w_0{:}\neg q, w_0{:}\Diamond_1 \neg p, w_0{:}q.$$

After an application of the $\wedge$-elimination rule we arrive at

$$w_0{:}\neg q, w_1{:}\neg p, w_1{:}p, w_0{:}\Box_1 p \;\mid\; w_0{:}\neg q, w_0{:}\Diamond_1\neg p, w_0{:}q.$$

It is straightforward to see that we can apply the $\wedge\,\mathtt{C}$ rule to both sets of labelled formulae. We end up with the empty sequence. Thus, ϕ_3 is unsatisfiable.

However, delaying the application of $\Diamond_1$-elimination to the end can also be a disadvantage. Consider, the structure

$$w_0{:}\Diamond_1\neg p, w_0{:}\Box_1 p, w_0{:}p \vee \Box_1 q.$$

Adding $w_1{:}\neg p \wedge p$ to the set of labelled formulae followed by an application of the $\wedge$-elimination and $\wedge\,\mathtt{C}$ rule allows the derivation of the empty sequence although we have not eliminated the disjunction in $p \vee \Box_1 q$ first. This test makes a difference computationally if the set of labelled formulae contains a large number of disjunctive formulae which are irrelevant with regards its satisfiability. It is possible to add the following $\Diamond_i\,\mathtt{T}$ inference rule to the system without loosing completeness.

$$\Diamond_i\,\mathtt{T}{:}\quad \frac{w{:}\Diamond_i\phi, D, C \;\mid\; S}{v{:}\phi \wedge \psi_1 \wedge \ldots \wedge \psi_n, w{:}\Diamond_i\phi, D, C \;\mid\; S}$$

if $D = w{:}\Box_i\psi_1, \ldots, w{:}\Box_i\psi_n$, and v is a new label chosen from Γ.

Furthermore, if we ensure that the rule is applied only finitely many times before we eventually eliminate $w{:}\Diamond_i\phi$ by the $\Diamond_i$-elimination rule, the inference system remains terminating. Note that the application of the $\Diamond_i\,\mathtt{T}$ rule closely resembles the intermediate calls of the KM procedure during a computation of KDP by KSAT.

We end our description of the system $\mathcal{KRIS}$ with a sample derivation.

Example 6.2:
Again, we consider the satisfiable modal formula $\psi = \neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. First, it transforms the formula ψ to its negation normal form ψ' which is $\psi' = \Diamond_1(\neg p \wedge \neg r) \wedge (\Box_1 p \vee \Box_1 q)$. Figure 6.2 shows how $\mathcal{KRIS}$ proceeds to prove the satisfiability of ψ'. First, $\mathcal{KRIS}$ eliminates the occurrence of the $\wedge$-operator in ψ'. Then it uses the $\vee\,\mathtt{E}$ rule to split the disjunctive formula $(\Box_1 p \vee \Box_1 q)$. Now we have to deal with two sets of labelled formulae. $\mathcal{KRIS}$ continues with the left set $w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p$. The only rule applicable to this set is $\Diamond_1\,\mathtt{E}$. The application of the $\Diamond_1\,\mathtt{E}$

$$\begin{array}{c}
w_0{:}\Diamond_1(\neg p \wedge \neg r) \wedge (\Box_1 p \vee \Box_1 q) \\
\hline
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p \vee \Box_1 q \quad \wedge\,\mathtt{E} \\
\hline
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 p \;\mid\; w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \quad \vee\,\mathtt{E} \\
\hline
w_1{:}(\neg p \wedge \neg r) \wedge p, w_0{:}\Box_1 p \;\mid\; w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \quad \Diamond\,\mathtt{E} \\
\hline
w_1{:}\neg p, w_1{:}\neg r, w_1{:}p, w_0{:}\Box_1 p \;\mid\; w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \quad \wedge\,\mathtt{E} \\
\hline
w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q \quad \wedge\,\mathtt{C} \\
\hline
w_1{:}(\neg p \wedge \neg r) \wedge q, w_0{:}\Box_1 q \quad \Diamond\,\mathtt{E} \\
\hline
w_1{:}\neg p, w_1{:}\neg r, w_1{:}q, w_0{:}\Box_1 q \quad \wedge\,\mathtt{E}
\end{array}$$

Figure 6.2: Sample derivation of $\mathcal{KRIS}$

Axioms: $\phi, \Gamma \Rightarrow \phi, \Delta \qquad \Gamma \Rightarrow \top, \Delta \qquad \bot, \Gamma \Rightarrow \Delta$

Rules:

$$\frac{\phi, \psi, \Gamma \Rightarrow \Delta}{\phi \wedge \psi, \Gamma \Rightarrow \Delta}\ (l\wedge) \qquad \frac{\Gamma \Rightarrow \phi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \phi \wedge \psi, \Delta}\ (r\wedge)$$

$$\frac{\phi, \Gamma \Rightarrow \Delta \qquad \psi, \Gamma \Rightarrow \Delta}{\phi \vee \psi, \Gamma \Rightarrow \Delta}\ (l\vee) \qquad \frac{\Gamma \Rightarrow \phi, \psi, \Delta}{\Gamma \Rightarrow \phi \vee \psi, \Delta}\ (r\vee)$$

$$\frac{\Gamma \Rightarrow \phi, \Delta}{\neg\phi, \Gamma \Rightarrow \Delta}\ (l\neg) \qquad \frac{\phi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg\phi, \Delta}\ (r\neg)$$

$$\frac{\phi, \Gamma \Rightarrow \Delta}{\Diamond_i\phi, \Box_i\Gamma, \Sigma \Rightarrow \Diamond_i\Delta, \Pi}\ (l\Diamond_i) \qquad \frac{\Gamma \Rightarrow \phi, \Delta}{\Box_i\Gamma, \Sigma \Rightarrow \Box_i\phi, \Diamond_i\Delta, \Pi}\ (r\Box_i)$$

Figure 6.3: Axioms and rules of the Logics Workbench

rule eliminates the labelled formula $w_0{:}\Diamond_1(\neg p \wedge \neg r)$ from our set and adds $w_1{:}\neg p \wedge \neg r \wedge p$. Applying the $\wedge$E rule to this labelled formula reveals that our set of labelled formulae contains both $w_1{:}\neg p$ and $w_1{:}p$. This is a contradiction and the $\wedge$C rule eliminates this set of labelled formula from the sequence. The remaining set of labelled formulae, namely $w_0{:}\Diamond_1(\neg p \wedge \neg r), w_0{:}\Box_1 q$, is the second set generated by the $\vee$E rule. Again, the only applicable rule is $\Diamond_1$E. This adds the formula $w_1{:}\neg p \wedge \neg r \wedge q$ to the set while removing $w_0{:}\Diamond_1(\neg p \wedge \neg r)$. A sequence of applications of the $\wedge$E rule results in a set of labelled formulae to which no further rule applies. Thus, $\mathcal{KRIS}$ has shown that ψ' and ψ are satisfiable.

The Logics Workbench (LWB) is an interactive system providing inference mechanisms for a variety of logical formalisms including basic modal logic. The decision procedure for $\mathsf{K}_{(m)}$ is based on the sequent calculus presented in Figure 6.3 (of which some axioms and rules are eliminable) [61, 62]. A modal formula ϕ is derivable using the axioms and rules of the sequent calculus if and only if ϕ is true in all Kripke models. Since we are interested in satisfiability not provability, we exploit that a given formula ϕ is unsatisfiable if and only if $\neg\phi$ is provable.

Unlike $\mathcal{KRIS}$, the Logics Workbench has no simplification rules. For example, a sequent proof of the satisfiability of the formula $\neg p \wedge (p \vee q)$ is:

$$\cfrac{\cfrac{\cfrac{\cfrac{p \Rightarrow p \qquad \cfrac{\text{Failure}}{q \Rightarrow p}}{(p \vee q) \Rightarrow p}\ (l\vee)}{\neg p, (p \vee q) \Rightarrow}\ (l\neg)}{\neg p \wedge (p \vee q) \Rightarrow}\ (l\wedge)}{\Rightarrow \neg(\neg p \wedge (p \vee q))}\ (r\neg)$$

Starting with the sequent $\Rightarrow \neg(\neg p \wedge (p \vee q))$, the Logics Workbench conducts a backwards proof search. That is, the inference rules presented in Figure 6.3 are applied bottom up. The $(r\neg)$-rule moves the formula $\neg p \wedge (p \vee q)$ to the left side of the sequent. Then we eliminate the occurrence of the conjunctive operator using the $(l\wedge)$-rule. The left hand side of the sequent now consist of two formulae, namely $\neg p$ and $(p \vee q)$. It uses the $(l\neg)$-rule to move $\neg p$ to the right-hand side of the sequent. Now the $(l\vee)$-rule is the only rule applicable to the sequent $(p \vee q) \Rightarrow p$ we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \Rightarrow p, r}{p \Rightarrow p \vee r}\,(r\vee)}{\Box_1 p \Rightarrow \Box_1(p \vee r)}\,(r\Box_1) \qquad \dfrac{\dfrac{\dfrac{\text{Failure}}{q \Rightarrow p, r}}{q \Rightarrow p \vee r}\,(r\vee)}{\Box_1 q \Rightarrow \Box_1(p \vee r)}\,(r\Box_1)}{\Box_1 p \vee \Box_1 q \Rightarrow \Box_1(p \vee r)}\,(l\vee)}{\neg\Box_1(p \vee r), \Box_1 p \vee \Box_1 q \Rightarrow}\,(l\neg)}{\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q) \Rightarrow}\,(l\wedge)}{\Rightarrow \neg(\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q))}\,(r\neg)$$

Figure 6.4: Sample derivation of the Logics Workbench

arrived at. We get two sequents, namely $p \Rightarrow p$ and $q \Rightarrow p$. Only the first one is an axiom. The sequent $q \Rightarrow p$ is neither an axiom nor can we apply any further rules of the calculus. We have failed to construct a proof of $\Rightarrow \neg(\neg p \wedge (p \vee q))$. Therefore $\neg p \wedge (p \vee q)$ is satisfiable.

There are two points worth noting. An application of the $(l\vee)$-rule creates two branches into our backwards proof search. If one of the branches fails, the whole proof attempt fails. We could directly derive the sequent $\neg p, q \Rightarrow$ from $\neg p, (p \vee q) \Rightarrow$ using the equivalent of the $\vee \mathtt{S}_1$ rule for sequents. This would eliminate the need to apply the $(l\vee)$-rule in the example. But, as mentioned before, the Logics Workbench has no equivalents of the $\vee$-simplification rules.

However, the Logics Workbench uses the following form of branch pruning. Provided in a backwards application of the $(l\vee)$-rule the formula ϕ is not used in the proof of $\phi, \Gamma \Rightarrow \Delta$, that is, $\Gamma \Rightarrow \Delta$ holds, then it is not necessary to consider the branch $\psi, \Gamma \Rightarrow \Delta$. Similarly, branch pruning is applied to the $(r\wedge)$-rule.

The Logics Workbench applies the $(l\wedge)$-rule, $(l\neg)$-rule, $(r\neg)$-rule and $(r\neg)$-rule exhaustively before any application of the remaining rules. The selection of the disjunctive and conjunctive formulae for applications of the $(l\vee)$-rule and $(r\wedge)$-rule, respectively, is determined by the order of formulae in the left-hand side and right-hand side of the sequent, respectively. The $(l\Diamond_i)$-rule and $(r\Box_i)$-rule are applied only after no application of the other rules is possible.

Example 6.3:
Figure 6.4 gives the derivation produced by the Logics Workbench of the satisfiability of $\psi = \neg\Box_1(p\vee r)\wedge(\Box_1 p\vee\Box_1 q)$. Starting from $\Rightarrow \neg(\neg\Box_1(p\vee r)\wedge(\Box_1 p\vee\Box_1 q))$ the backwards applications of the $(r\vee)$-rule, $(l\wedge)$-rule and $(l\neg)$-rule lead to the sequent $\Box_1 p\vee\Box_1 q \Rightarrow \Box_1(p\vee r)$. The backwards application of the $(l\vee)$-rule generates two sequents $\Box_1 p \Rightarrow \Box_1(p \vee r)$ and $\Box_1 q \Rightarrow \Box_1(p \vee r)$. The Logics Workbench first considers the sequent $\Box_1 p \Rightarrow \Box_1(p\vee r)$. Here we have to apply the $(r\Box_1)$-rule, for which we have to select a formula of the form $\Box\phi$ on the right-hand side of the sequent. Since in the sequent under consideration only one $\Box$-formula occurs on the right-hand side of the sequent, the choice is deterministic. The application of the $(r\Box_1)$-rule yields the sequent $p \Rightarrow p \vee r$. With a final application of the $(r\vee)$-rule we arrive at the axiom $p \Rightarrow p, r$. Now the Logics Workbench turns to the second alternative $\Box_1 q \Rightarrow \Box_1(p \vee r)$. Here the application of the $(r\Box_1)$-rule produces $q \Rightarrow p \vee r$. An application of the $(r\vee)$-rule renders $q \Rightarrow p, r$. Since no more rules apply and $q \Rightarrow p, r$ is not an axiom, our attempt to construct a proof fails. No other proof attempts are possible. So ψ is satisfiable.

Observe the near correspondence between the proof search of $\mathcal{KRIS}$ and that of the Logics

Workbench. As a procedure for testing the satisfiability of a modal formulae, the inference rules of the Logics Workbench can be seen to be a notational variant of the inference rules of $\mathcal{KRIS}$. We can directly translate the deduction steps in the tableaux-calculus of $\mathcal{KRIS}$ into the sequent calculus of the Logics Workbench. The differences between the two system are the absence of simplification rules in the Logics Workbench, the presence of branch pruning in the Logics Workbench, and the conversion to negation normal form by $\mathcal{KRIS}$.

The fourth system we will include in the comparison is based on the optimised functional translation described in Section 5.1. For a modal formula ϕ in $\mathsf{K}_{(m)}$ additional transformations of the clause set we obtain from $\Xi\overline{\Pi}_f(\phi)$ are possible. First, we replace all occurrences of literals $P(s)$ where s is a path of the form $[x\alpha^1_{i_1}\alpha^2_{i_2}\ldots\alpha^n_{i_n}]$ with length $n{+}1$ where $\alpha^j_{i_j}$ is a variable or constant of sort AF_{i_j}, for $1 \leq j \leq n$, by $P_{n+1}(x, \alpha^1_{i_1}, \ldots, \alpha^n_{i_n})$ where P_{n+1} is an $(n{+}1)$-ary predicate symbol uniquely associated with P and n. Second, the sort information associated with the variables and constants occurring in the literals in the clause set can be encoded in the predicate symbols of the literals. So, we can replace all occurrences of literals $P_{n+1}(x, \alpha^1_{i_1}, \ldots, \alpha^n_{i_n})$ by $P_{i_1\ldots i_n}(x, \alpha^1, \ldots, \alpha^n)$ where $P_{i_1\ldots i_n}$ is a predicate symbol uniquely associated with the predicate symbol P_{n+1} and the sorts $AF_{i_1}, \ldots, AF_{i_n}$. The variables and constants $\alpha^1, \ldots, \alpha^n$ no longer carry any sort information. Finally, we observe that all literals in the transformed clause set share the first argument x, which we can eliminate safely. This sequence of three transformations can be combined in one:

$$P([x\alpha^1_{i_1}\alpha^2_{i_2}\ldots\alpha^n_{i_n}]) \quad \text{becomes} \quad P_{i_1\ldots i_n}(\alpha^1, \ldots, \alpha^n).$$

Example 6.4:
We consider our example formula ψ given by $\neg\Box_1(p \vee r) \wedge (\Box_1 p \vee \Box_1 q)$. The result of $\mathrm{Cls}\Xi\overline{\Pi}_f(\psi)$ is a set of four clauses:

(6.1) def_1

(6.2) $\neg P_1(\underline{a})$

(6.3) $\neg R_1(\underline{a})$

(6.4) $\neg\mathsf{def}_1 \vee \neg\mathsf{def}_1 \vee P_1(x) \vee Q_1(y)$

Two resolution steps are possible: Resolving clauses (6.1) and (6.4) yields $P_1(x) \vee Q_1(y)$. The derived clause subsumes the clause (6.4). Resolving $P_1(x) \vee Q_1(y)$ with clause (6.2) yields the unit clause $Q_1(y)$, that subsumes the clause $P_1(x) \vee Q_1(y)$. Subsumption leaves the following clause set on which no further inference steps are possible.

$$\begin{array}{c} \mathsf{def}_1 \\ \neg P_1(\underline{a}) \\ \neg R_1(\underline{a}) \\ Q_1(y) \end{array}$$

Since the final clause set does not contain the empty clause, the original clause set, and consequently, the modal formula ϕ is satisfiable.

For theorem proving we use FLOTTER and SPASS Version 0.55 developed by Weidenbach *et al.* [134]. FLOTTER is a system that computes the clausal normal form of a given first-order formula. It performs the following steps.

1. Rename subformulae of the input formula in order to obtain a clause set containing a minimal number of clauses. Here an improved variant of the technique developed by Boy de la Tour [19] is used.

2. Remove implications and equivalences using the appropriate transformation rules.

3. Compute the negation normal form.

4. Eliminate existential quantifiers by Skolemisation.

5. Compute the clausal normal form.

6. Test the resulting clause set for redundancy by subsumption, tautology removal and condensing.

The theorem prover SPASS is based on the superposition calculus of Bachmair and Ganzinger [8] extended with the sort techniques of Weidenbach [133].

We opted to use SPASS and not other well-known theorem provers (like OTTER) for the following reasons:

1. SPASS uses ordered resolution and ordered factoring based on an extended Knuth-Bendix ordering [112].

2. It supports splitting and branch condensing. Splitting amounts to case analysis while branch condensing resembles branch pruning in the Logics Workbench. The splitting rule of SPASS is not identical to the "Split" expansion rule described in Section 1.2.

3. It has an elaborated set of reduction rules including tautology deletion, subsumption, and condensing.

4. It supports dynamic sort theories by additional inference rules including sort generation and sort resolution and additional reduction rules like sort simplification and clause deletion.

Ordered inference rules and splitting are of particular importance when treating satisfiable formulae. Also, SPASS supports dynamic sort theories by additional inference rules including sort generation and sort resolution and additional reduction rules like sort simplification and clause deletion. It considers every unary predicate symbol as a sort (not to be confused with the sorts of the translation morphism). The translation of random 3CNF formulae will result in first-order formulae which contain a great number of such symbols.

6.4 A benchmark suite for scientific benchmarking

A good starting point for setting up a suitable benchmark suite for the system described in Section 6.3 is the recent work by Giunchiglia and Sebastiani [51, 52]. The evaluation method adopted by Giunchiglia and Sebastiani follows the approach of Mitchell, Selman, and Levesque [100]. Mitchell et al. have used propositional formulae generated using the fixed clause-length model to set up a benchmark suite for theorem provers for propositional logic. Giunchiglia and Sebastiani [52] provide an modification of this approach suitable for the modal logic $\mathsf{K}_{(m)}$.

	N	M	K	D	P		N	M	K	D	P
PS0	5	1	3	2	0.5	**PS5**	4	1	3	2	0.5
PS1	3	1	3	5	0.5	**PS6**	4	2	3	2	0.5
PS2	3	1	3	4	0.5	**PS7**	4	5	3	2	0.5
PS3	3	1	3	3	0.5	**PS8**	4	10	3	2	0.5
PS4	3	1	3	2	0.5	**PS9**	4	20	3	2	0.5

Table 6.1: Parameter settings

There are five parameters: the number of propositional variables N, the number of modalities M, the number of modal subformulae per disjunction K, the number of modal subformulae per conjunction L, the modal degree D, and the probability P. Based on a given choice of parameters random modal KCNF formulae are defined inductively as follows. A *random (modal) atom* of degree 0 is a variable randomly chosen from the set of N propositional variables. A *random modal atom* of degree D, $D>0$, is with probability P a random modal atom of degree 0 or an expression of the form $\Box_i\phi$, otherwise, where $\Box_i$ is a modality randomly chosen form the set of M modalities and ϕ is a random modal KCNF clause of modal degree $D-1$ (defined below). A *random modal literal* (of degree D) is with probability 0.5 a random modal atom (of degree D) or its negation, otherwise. A *random modal KCNF clause* (of degree D) is a disjunction of K random modal literals (of degree D). Now, a *random modal KCNF formula* (of degree D) is a conjunction of L random modal KCNF clauses (of degree D).

Like Giunchiglia and Sebastiani we proceed as follows to compare the performance of two theorem provers for modal logic. We fix all parameters except L, the number of clauses in a formula. For example, we choose N=3, M=1, K=3, D=5, and P=0.5. The parameter L ranges from N to $40N$. For each value of the ratio L/N a set of 100 random modal KCNF formulae of degree D is generated. We will see that for small L the generated formulae are most likely to be satisfiable and for larger L the generated formulae are most likely to be unsatisfiable. For each generated formula ϕ we measure the time needed by one of the decision procedures to determine the satisfiability of ϕ. Since checking a single formula can take arbitrarily long in the worst case, there is an upper limit for the CPU time consumed. As soon as the upper limit is reached, the computation for ϕ is stopped. The median CPU runtime over the ratio L/N is the measure on which the relative performance of the systems is compared.

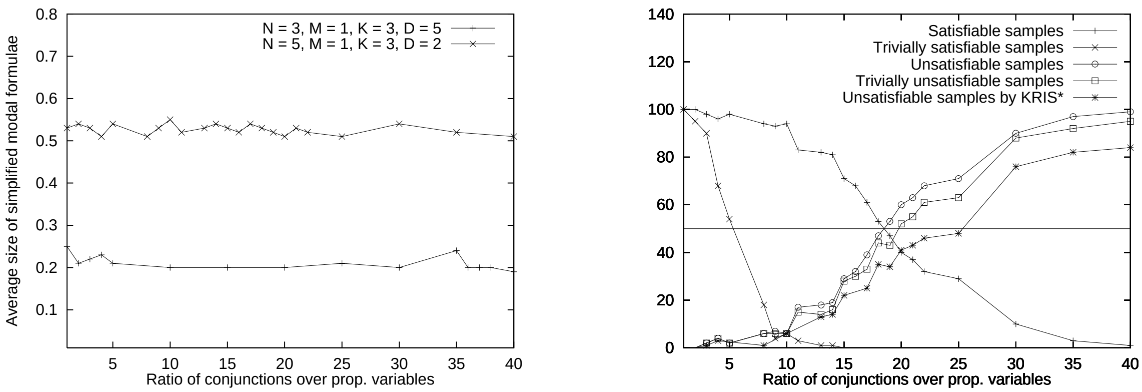
Selecting good test instances is crucial when evaluating and comparing the performances of algorithms empirically. For the random formula generator described above this means, we have to determine sets of parameter settings which are suitable for generating appropriate test instances for the systems under consideration. One step in this process is to determine the characteristics of the test instances before starting a performance comparison. This is particularly important when we set up a completely new collection of test instances. Giunchiglia and Sebastiani [52] have chosen the parameter settings of Table 6.1 for their comparison of KSAT and $\mathcal{KRIS}$.

We address the question which characteristics the formulae produced by the random generator and the parameter settings chosen by Giunchiglia and Sebastiani have and how they influence the theorem provers under consideration.

It is important to note that for D=0 and K=3 random modal KCNF formulae do *not* coincide with random 3SAT formulae. Generating a clause of a random 3SAT formula means randomly generating a *set* of three propositional variables and then negating each member of the set with

$\neg\phi \vee \phi \to \top$	$\phi \vee \top \to \top$	$\phi \vee \bot \to \phi$	$\phi \vee \phi \to \phi$
$\neg\phi \wedge \phi \to \bot$	$\phi \wedge \top \to \phi$	$\phi \wedge \bot \to \bot$	$\phi \wedge \phi \to \phi$
$\neg\bot \to \top$	$\neg\top \to \bot$	$\Box_i \top \to \top$	

Table 6.2: Simplification rules for modal formulae



(a) The effect of simplification for **PS0** and **PS1**

(b) Percentages of trivial samples for **PS0**

Figure 6.5: The quality of the test sets

probability 0.5. In contrast, generating a random modal 3CNF clause of degree 0 means randomly generating a *multiset* of three propositional variables and negating each member of the multiset with probability 0.5. For example, $p \vee q \vee \neg r$ is a 3SAT clause and also a modal 3CNF clause of degree 0. The clauses $p \vee \neg p \vee p$ and $p \vee p \vee q$ are not random 3SAT clauses, but both are random modal 3CNF clauses of degree 0. In random modal 3CNF formulae of higher degree, such clauses occur within the scope of a modal operator. For example, contradictory expressions like $\neg\Box_1(p \vee \neg p \vee p)$ may occur.

Thus, random modal KCNF formulae contain tautological and contradictory subformulae. It is easy to remove these subformulae without affecting satisfiability, for example, by use of the simplification rules of Table 6.2. The graphs of Figure 6.5(a) reflect the average ratio of the size of the simplified random modal 3CNF formulae over the size of the original formulae for the parameter settings **PS0** and **PS1**. For the random modal 3CNF formulae generated using three propositional variables, on average, the size of a simplified formula is only 1/4 of the size of the original formula. For the second parameter setting we observe a reduction to 1/2 of the original size. In other words, one half to three quarters of the random modal 3CNF formulae is "logical garbage" that can be eliminated at little cost.

A second criterion for the quality of the formulae under consideration is whether there are computationally inexpensive tests, that is, tests which can be performed in polynomial time, which can determine the satisfiability or unsatisfiability of a formulae. Suppose that we want to test a random modal 3CNF formula ϕ with N propositional variables for satisfiability in a Kripke

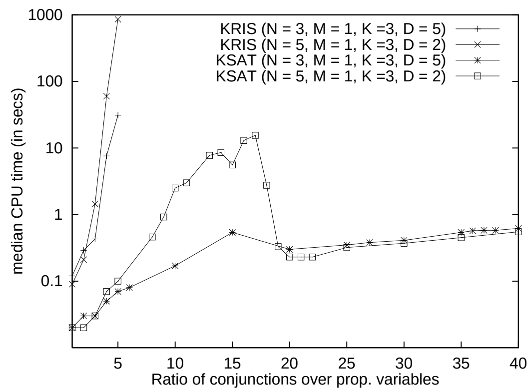


Figure 6.6: The performance of $\mathcal{KRIS}$ and KSAT for **PS0** and **PS1**

model with only one world. We have to test at most 2^N truth assignments to the propositional variables. Since $N \leq 5$ for the modal formulae under consideration, this is a trivial task, even by the truth table method. We say a random modal 3CNF formula ϕ is *trivially satisfiable* if ϕ is satisfiable in a Kripke model with only one world. We also say a random modal 3CNF formula ϕ is *trivially unsatisfiable* if the conjunction of the purely propositional clauses of ϕ is unsatisfiable. Again, testing whether ϕ is trivially unsatisfiable requires only the consideration of 2^N truth assignments.

The graphs of Figure 6.5(b) show the percentage of satisfiable, trivially satisfiable, unsatisfiable, trivially unsatisfiable, and unsatisfiable formulae in the samples detected by $\mathcal{KRIS}$* of the set of test formulae generated for **PS0**. We see that almost all unsatisfiable test formulae are trivially unsatisfiable. This holds also for all the other parameter settings used by Giunchiglia and Sebastiani. This indicates, none of the parameter settings is suited to generate challenging unsatisfiable modal formulae. Only for ratios L/N between 7 and 20 can we expect that the benchmark suite contains a sufficient number of non-trivial formulae.

6.5 Evaluation of theorem provers and a benchmark suite

Based on our findings in the previous section one might expect that there is little deviation between the performance of the theorem provers described in Section 6.3 on the parameter settings **PS0** to **PS9**. The opposite is true.

Figure 6.6 shows the median CPU time consumption of $\mathcal{KRIS}$ and KSAT on the parameter settings **PS0** and **PS1** which have been produced on a Sun Ultra 1/170E with 196MB main memory using a time-limit of 1000 CPU seconds. The gaps in the graphs (for example for $\mathcal{KRIS}$ above $L/N = 5$) indicate that more than 50 out of 100 formulae of given ratio L/N had to be abandoned. While the performance of KSAT coincides with our expectations, that is, only for ratios L/N between 7 and 20 for the parameter setting **PS0** the benchmark suite contains a sufficient number of problems which are able to challenge KSAT, $\mathcal{KRIS}$ fails to solve more than 50% of the sample formulae for any ratio L/N greater than 5.

Our observations concerning the amount of tautological and contradictory subformulae in the random formulae provides the key for understanding this phenomenon. KSAT utilises a form of preprocessing that removes duplicate and contradictory subformulae of an input formula, by ap-

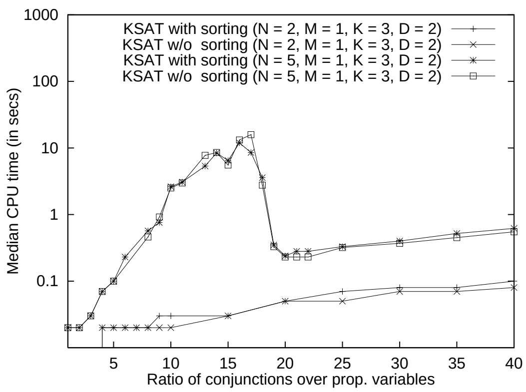


Figure 6.7: The performance of Ksat with and without sorting

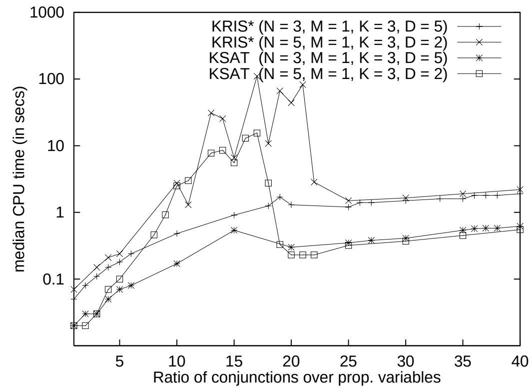


Figure 6.8: The performance of Ksat and $\mathcal{KRIS}$* for **PS0** and **PS1**

plying the simplification rules presented in Table 6.2. Neither $\mathcal{KRIS}$ nor the Logics Workbench, on the other hand, perform a similar form of preprocessing. Simplification of the random modal formulae is reasonable, so we added the preprocessing function of Ksat also to the other theorem provers that we consider. The modified versions of $\mathcal{KRIS}$, the Logics Workbench and the translation approach with preprocessing will be denoted by $\mathcal{KRIS}$*, LWB* and TA*, respectively.

While the simplification rules of Table 6.2 replace $p \vee q \vee p$ by $p \vee q$ and $\Box_1(p \vee q) \wedge \neg\Box_1(p \vee q)$ by $\bot$, they will not reduce $\Box_1(p \vee q) \wedge \neg\Box_1(q \vee p)$ to $\top$, since $\Box_1(p \vee q)$ is not syntactically equal to $\Box_1(q \vee p)$. Ksat also sorts disjunctions lexicographically, for example, $\Box_1(q \vee p)$ will be replaced by $\Box_1(p \vee q)$. This allows for additional applications of the simplification rules. However, in all our experiments we have chosen to disable the reordering inside Ksat. For the median CPU runtime considerations of this section, reordering has no significant effect as Figure 6.7 shows. Likewise the other approaches take no advantage of reordering as performed in Ksat. But reordering is an interesting technique that deserves further investigation, for all procedures. In particular, generalising the notion of reordering as implemented inside Ksat to a notion of reordering of conjunctions of clauses, will have a positive effect on the Logics Workbench and $\mathcal{KRIS}$.

The graphs in Figure 6.8 show the performances of Ksat and $\mathcal{KRIS}$*. Although the performance of Ksat is still better than that of $\mathcal{KRIS}$*, the picture is completely different than that of Figure 6.6. To explain the remaining difference we study the quality of the random modal 3CNF formulae. If we consider Figure 6.5(b) and 6.8 together, for ratios L/N between 19 and 21 and N=5 we observe the graph of $\mathcal{KRIS}$* (in Figure 6.8) deviates a lot (by a factor of more than 100) from the graph of Ksat. This is the area near the crossover point where the percentage of trivially unsatisfiable formulae rises above 50%, however, the percentage of unsatisfiable formulae detected by $\mathcal{KRIS}$* is still below 50% in this area. $\mathcal{KRIS}$* does not detect all trivially unsatisfiable formulae within the time-limit which explains the deviation in performance from Ksat. The reason for $\mathcal{KRIS}$* not detecting all trivially unsatisfiable formulae within the time limit, can be illustrated by the following example.

Example 6.5:
Let ϕ_4 be a simplified modal 3CNF formula

$$\begin{aligned} p \wedge q \wedge (\psi_{11} \vee \psi_{12} \vee \psi_{13}) \\ \dots \\ \wedge (\psi_{k1} \vee \psi_{k2} \vee \psi_{k3}) \wedge (\neg p \vee \neg q) \end{aligned}$$

where the ψ_{ij}, with $1 \le i \le k$, $1 \le j \le 3$, are modal literals different from p, q, $\neg p$, and $\neg q$. Evidently, ϕ_4 is trivially unsatisfiable. KSAT does the following: Since p and q are unit clauses in ϕ_4, it applies the rule `dp_unit` twice to ϕ. The rule replaces the occurrences of p and q by $\top$, it replaces the occurrences of $\neg p$ and $\neg q$ by $\bot$, and it simplifies the formula. The resulting formula is $\bot$. At this point only the rule `dp_clash` is applicable and KSAT detects that ϕ_4 is unsatisfiable. In contrast, $\mathcal{KRIS}^*$ proceeds as follows. First it applies the $\wedge$E rule $k+2$ times, eliminating all occurrences of the $\wedge$ operator. Then it applies the $\vee$E rule to all disjunctions, starting with $\psi_{11} \vee \psi_{12} \vee \psi_{13}$ and ending with $\psi_{k1} \vee \psi_{k2} \vee \psi_{k3}$. This generates 3^k subproblems. Each of these subproblems contains the literals p and q and the disjunction $\neg p \vee \neg q$. The simplification rule $\vee \mathtt{S}_1$ eliminates the disjunction $\neg p \vee \neg q$ and a final application of the $\wedge$C rule exhibits the unsatisfiability of each subproblem. Obviously, for k large enough, $\mathcal{KRIS}^*$ will not be able to finish this computation within the time-limit.

In the Logics Workbench branch pruning avoids this kind of computation. Starting from the sequent $\Rightarrow \neg\phi_4$ it first applies the $(r\neg)$-rule followed by applications of the $(l\wedge)$-rule until all outer conjunction operators are eliminated. A sequence of $k+1$ applications of the $(l\vee)$-rule follows generating 2^{k+1} (potential) branches. On the first and second branch the sequents

$$p, q, \psi_{11}, \dots, \psi_{k1}, \neg p \Rightarrow \qquad \text{and} \qquad p, q, \psi_{11}, \dots, \psi_{k1}, \neg q \Rightarrow$$

are considered which are both provable. As neither proof requires the use of one of the literals $\psi_{11}, \dots, \psi_{k1}$, the Logics Workbench prunes all remaining branches and detects the unsatisfiability of ϕ_4.

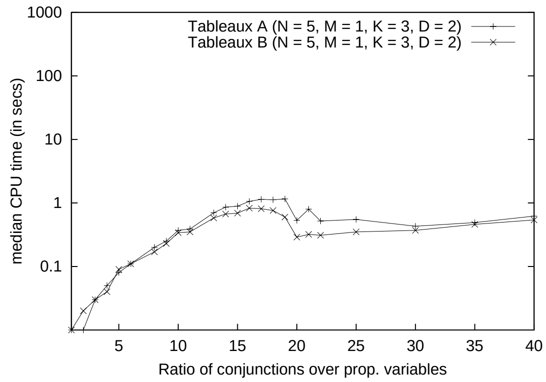
Giunchiglia and Sebastiani [52] come to a different conclusion concerning the cause for the fact that KSAT outperforms $\mathcal{KRIS}$. Their analysis is based on a result by D'Agostino [23], who shows that in the worst case, algorithms using the $\vee$E rule cannot simulate truth tables in polynomial time. Instead one has to use the following modified form of $\vee$E:

$$\vee\mathtt{E'}: \quad \frac{w{:}\phi \vee \psi, C \;\mid\; S}{w{:}\phi, C \;\mid\; w{:}\psi, w{:}\overline{\phi}, C \;\mid\; S}$$

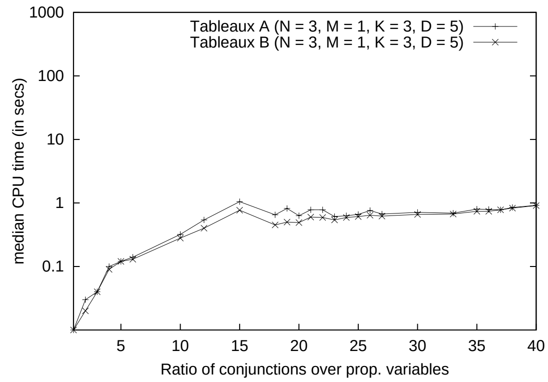
This rule ensures that the two subproblems $w{:}\phi, C$ and $w{:}\psi, w{:}\overline{\phi}, C$ generated by the elimination of the disjunction $\phi \vee \psi$ are mutually exclusive.

While the use of $\vee$E instead of $\vee$E' is an obvious advantage for propositional formulae in conjunctive normal form, this is not evident for modal formulae. In the propositional case, $\overline{\phi}$ is a literal which will not be subject to any of the other elimination rules. However, in the modal case, $\overline{\phi}$ can be a complex formula to which the elimination rules have to be applied, causing additional computational effort compared to an application of the $\vee$E rule which does not introduce $\overline{\phi}$ on one of the branches. In particular, in combination with the $\diamond_i$-elimination rule, the introduction of $\overline{\phi}$ can increase the size of the search space considerably.

Note that this is the kind of question for which scientific testing is the ideal approach. For the purpose of performing the test we implemented two tableaux-based procedures **A** and **B** which use

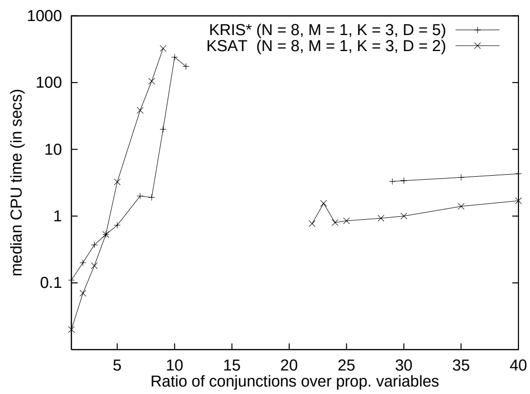


(a) **PS0** (N=5, M=1, K=3, D=2, P=0.5)

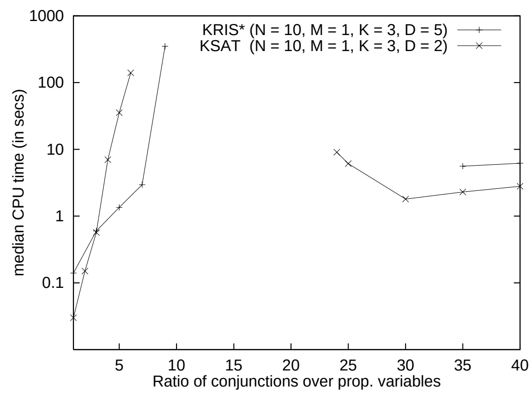


(b) **PS1** (N=3, M=1, K=3, D=5, P=0.5)

Figure 6.9: Impact of $\vee$`E` versus $\vee$`E'`



(a) **PS10** (N=8, M=1, K=3, D=2, P=0.5)



(b) **PS11** (N=10, M=1, K=3, D=2, P=0.5)

Figure 6.10: The performance of KSAT and $\mathcal{KRIS}$*

the rules $\vee$`E` and $\vee$`E'`, respectively, but are identical in all other aspects. Figures 6.9(a) and 6.9(b) show the median CPU time graphs of the two procedures for **PS0** and **PS1**, respectively.

We see that there is virtually no difference between them. This result can be reproduced for all the other parameter settings under consideration. Thus, it seems unlikely that the use of $\vee$`E` instead of $\vee$`E'` is a major factor concerning the performance of tableaux-based procedure on **PS0** to **PS9**.

We can gain further evidence by turning to parameter settings using values of N greater than 5. Figure 6.10(a) shows the performance of KSAT and $\mathcal{KRIS}$* on the parameter setting **PS10** (N=8, M=1, K=3, D=2, P=0.5), while Figure 6.10(b) shows the performance on the parameter setting **PS11** (N=10, M=1, K=3, D=2, P=0.5). We see that the performance of $\mathcal{KRIS}$* for a ratio L/N between 4 and 11 on **PS10** and for a ratio L/N between 3 and 9 on **PS11** is better than the performance of KSAT. For increased numbers of propositional variables, the `dp_unit` rule

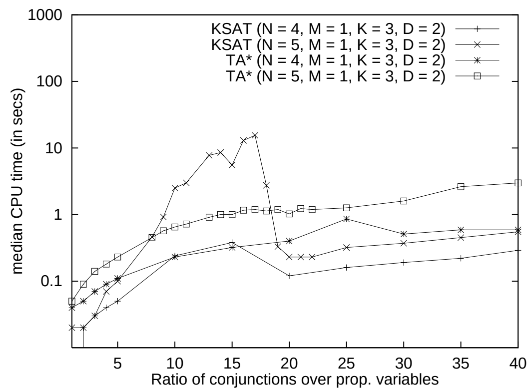


Figure 6.11: The performance of KSAT and TA*

and exhaustive boolean simplification of KSAT is of no particular importance for modal formulae which are likely satisfiable. And, the intermediate calls to KM before each application of the `dp_split` have a deteriorating effect on the performance.

$\mathcal{KRIS}$* applies the ∨-elimination rule to every disjunction in the modal formula and continues on the first branch. As the number of propositional variables and modal atoms is large, the $\wedge$`C` rule is less likely to close a branch and the second branch need not be treated. After all occurrences of the ∨-operator are eliminated, $\mathcal{KRIS}$* performs all possible applications of the $\Diamond_i$`E` rule. Each application is likely to succeed.

By contrast, KSAT uses `dp_split` to generate two possible extensions of the current truth assignment. Like $\mathcal{KRIS}$*, it rarely has to consider the second extension at all. However, before every application of the `dp_split` rule the procedure KM is called. This has the following effect: The `dp_split` rule needs to be applied more often before reaching a satisfying truth assignment, since the number of different propositional variables and modal atoms has become larger. This also holds for the recursive calls of KDP by KM. There is an increased number of intermediate calls to the procedure KM and each call is more expensive than for simpler formulae. The effect is strengthened by the following inefficiency of the intermediate calls to KM. Suppose we have just checked the $\mathsf{K}_{(m)}$-satisfiability of the truth assignment $\mu_1 = \{\Box_1\psi_1 = \bot, \Box_1\phi_{11} = \top, \ldots, \Box_1\phi_{1n} = \top\}$ and extend μ_1 by $\{\Box_1\psi_2 = \bot\}$. By the next call to KM, KSAT will not only test whether $\neg\psi_2 \wedge \phi_{11} \wedge \ldots \wedge \phi_{1n}$ is satisfiable, but it will repeat the test whether $\neg\psi_1 \wedge \phi_{11} \wedge \ldots \wedge \phi_{1n}$ is satisfiable. So, KSAT performs the same tests over and over again without need.

There is no intrinsic reason that a tableaux-based system cannot outperform KSAT and the tableaux-based system **A** which uses the $\vee$`E` elimination rule is evidence for this (Figure 6.9(a)). Although the difference between the rules $\vee$`E` and $\vee$`E'` is fundamental from a theoretical point of view, it is irrelevant on the randomly generated modal formulae under consideration. The reason for $\mathcal{KRIS}$* having worse performance than KSAT is that $\vee\mathtt{S}_0$ and $\vee\mathtt{S}_1$ are not applied exhaustively before any applications of the branching rule $\vee$`E`.

According to Giunchiglia and Sebastiani [52] there is partial evidence of an easy-hard-easy pattern on randomly generated modal formulae independent of all the parameters of evaluation considered. This claim is supported by Figure 6.6 where the median CPU time consumption of KSAT decreases drastically at the ratio $L/N = 17.5$ for the second sample. This is almost the point, where 50% of the sample formulae are satisfiable. This decline seems to resemble the behaviour of propositional SAT decision procedures on randomly generated 3SAT problems.

There is no doubt that there exist classes of randomly generated modal formulae on which we will observe an easy-hard-easy pattern independent of the theorem prover we use, since these patterns exist for random propositional 3CNF formulae. However, concerning the parameter settings **PS0** to **PS9** we can exclude the existence of such an intrinsic easy-hard-easy pattern. Figure 6.11 compares the performance of KSAT with the performance of the translation approach on two parameter settings, where the easy-hard-easy pattern is most visible for KSAT. The translation approach does not show the peaking behaviour of KSAT. The median CPU time grows monotonically with the size of modal formulae. Also the two tableaux-based procedures in Figure 6.9(a) do not exhibit an easy-hard-easy pattern. Thus, the phase transition visible in Figure 6.6 is a phenomenon of KSAT (and $\mathcal{KRIS}$), and not an intrinsic property of the generated modal formulae.

Observe that the peaking behaviour occurs in the area where the number of trivially satisfiable sample formulae approaches zero. The following example tries to explain this.

Example 6.6:
Let ϕ_5 be a simplified modal 3CNF formula of the form

$$\begin{aligned}\neg\Box_1 s \wedge \Box_1(p \vee r) &\wedge (\Box_1\neg r \vee \Box_1 q)\\ &\wedge (\neg\Box_1 p \vee \Box_1 r)\\ &\wedge (\psi_{11} \vee \psi_{12} \vee \psi_{13})\\ &\ldots\\ &\wedge (\psi_{n1} \vee \psi_{n2} \vee \psi_{n3})\end{aligned}$$

where the ψ_{ij}, with $1{\leq}i{\leq}n$, $1{\leq}j{\leq}3$, are modal literals different from the modal literals in the first three conjunctions of ϕ_5. Let us assume that ϕ_5 is satisfiable. Observe:

1. $\Box_1\neg r$ is false in any model of ϕ_5, since $\Box_1\neg r$ and $\neg\Box_1 s \wedge (\neg\Box_1 p \vee \Box_1 r)$ imply $\neg\Box_1 p$, and $\Box_1(p \vee r) \wedge \Box_1\neg r \wedge \neg\Box_1 p$ is not $\mathsf{K}_{(m)}$-satisfiable.

2. As a consequence, any truth assignment μ such that $\mu(\Box_1\neg r) = \top$ is not $\mathsf{K}_{(m)}$-satisfiable.

3. A unit propagation step by KDP, replacing $\Box_1\neg r$ by $\top$, does not affect the literal $\Box_1 r$.

KSAT starts by assigning $\top$ to $\neg\Box_1 s$ and $\Box_1(p{\vee}r)$. Then it will apply a sequence of applications of the `dp_split` and `dp_unit` rules to ϕ_5. Let us assume that the first split variable is $\Box_1\neg r$, followed by k modal literals $\psi_1, \ldots, \psi_k$ chosen from $\psi_{11}, \ldots, \psi_{n3}$, and finally $\neg\Box_1 p$. Before any further applications of the `dp_split` rule, KSAT calls the procedure KM to test the $\mathsf{K}_{(m)}$-satisfiability of the current truth assignment μ. Since μ assigns $\top$ to $\Box_1\neg r$, KM will fail. However, KSAT has no means to detect the primary cause of the failure. KSAT continues by considering all other cases generated by the application of `dp_split` to $\neg\Box_1 p$, ψ_k, $\psi_{k-1}, \ldots, \psi_1$. It will fail to generate a satisfying truth assignment in all these cases. Finally, it considers the case that $\Box_1\neg r$ is false. Eventually, KSAT finds a satisfying truth assignment to ϕ_5. However, KSAT has considered at least 2^{k+1} cases unnecessarily without finding a satisfying truth assignment. This explains the bad behaviour of KSAT on those sample formulae where satisfiability tests in the non-propositional context are essential. $\mathcal{KRIS}^*$ behaves even worse since it delays the application of the $\Diamond_i\,\mathtt{E}$ until no other rule can be applied.

By contrast, the Logics Workbench takes advantage of its branch pruning. Starting from the sequent $\Rightarrow \neg\phi_5$ it first applies the $(r\neg)$-rule followed by applications of the $(l\wedge)$-rule until all outer conjunction operators are eliminated. A sequence of applications of the $(l\vee)$-rule follows. Let us assume the disjunctions are considered in this order: $(\Box_1\neg r \vee \Box_1 q)$, $(\psi_{11} \vee \psi_{12} \vee \psi_{13})$, $\ldots$, $(\psi_{n1} \vee \psi_{n2} \vee \psi_{n3})$, and finally $(\neg\Box_1 p \vee \Box_1 r)$. Like for KSAT and $\mathcal{KRIS}$ this generates 2^{n+2} (possible) branches. On the first branch the sequent Γ_1

$$\Box_1(p \vee r), \Box_1\neg r, \psi_{11}, \ldots, \psi_{n1} \Rightarrow \Box_1 p, \Box_1 s$$

is considered. This sequent is provable. So, the Logics Workbench considers the second branch generated by the application of the $(l\vee)$-rule to $(\neg\Box_1 p \vee \Box_1 r)$. Again, the sequent Γ_2

$$\Box_1(p \vee r), \Box_1\neg r, \psi_{11}, \ldots, \psi_{n1}, \Box_1 r \Rightarrow \Box_1 s$$

is provable. As neither the proof of Γ_1 nor of Γ_2 makes use of any of the literals $\psi_{11}, \ldots, \psi_{n1}$, the Logics Workbench does branch pruning. It will jump back directly to the point where the $(l\vee)$-rule is applied to $(\Box_1\neg r \vee \Box_1 q)$ and considers the branch in which $\Box_1 q$ is added to the left-hand side of the sequent. Thus, the search space is reduced considerably.

The translation approach proceeds as follows. It generates a clause set for ϕ_5 containing the five clauses

$$\begin{array}{l} \ \mathsf{def}_1 \\ \neg S(\underline{a}) \\ \neg\mathsf{def}_1 \vee P_1(x) \vee R_1(x), \\ \neg\mathsf{def}_1 \vee \neg R_1(x) \vee \neg\mathsf{def}_1 \vee Q_1(y), \\ \neg P_1(\underline{b}) \vee \neg\mathsf{def}_1 \vee R_1(x) \end{array}$$

where $\underline{a}$ and $\underline{b}$ denote Skolem constants associated with the two occurrences of $\neg\Box_1$ and x and y are variables. Unit propagation of the first clause followed by subsumption replaces the original clause set by the following one:

$$\begin{array}{l} \ \mathsf{def}_1 \\ \neg S_1(\underline{a}) \\ \ P_1(x) \vee R_1(x), \\ \neg R_1(x) \vee Q_1(y), \\ \neg P_1(\underline{b}) \vee R_1(x) \end{array}$$

Three resolvents can be derived from these clauses: $P_1(x) \vee Q_1(y)$, $\neg P_1(\underline{b}) \vee Q_1(y)$, and $R_1(\underline{b}) \vee R_1(x)$. Factoring on the last resolvent yields the unit clause $R(\underline{b})$. At this point, the translation approach has detected that $\Box_1\neg r$ is not satisfiable in any model of ϕ_5. An additional inference step computes the unit clause $Q_1(y)$. No further inference is possible on this subset.

Using the splitting rule of SPASS it is also possible to construct a derivation which resembles closely the one of the Logics Workbench. Instead of computing the three resolvents we can start by splitting the clause $\neg R_1(x) \vee Q_1(y)$ into its variable-disjoint subclauses, $\neg R_1(x)$ and $Q_1(y)$. Let us first consider the branch on which we add the clause $\neg R_1(x)$ to the clause set. This corresponds to assigning true to $\Box_1\neg r$. Let us assume that the translation of the disjunctions $(\psi_{11} \vee \psi_{12} \vee \psi_{13})$ to $(\psi_{n1} \vee \psi_{n2} \vee \psi_{n3})$ (indicated by a * below) generates clauses to which we can apply the splitting rule as well. Finally, apply the splitting rule to $\neg P_1(\underline{b}) \vee R_1(x)$. On the first

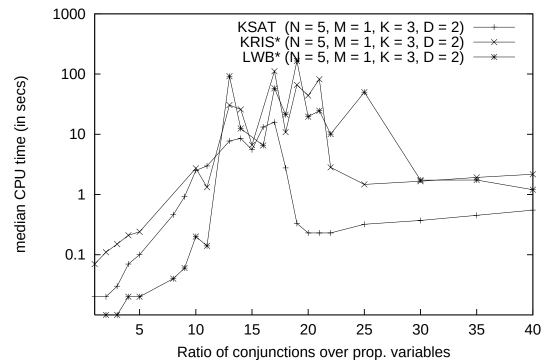


Figure 6.12: The performance of the Logics Workbench

branch we consider the clause set $\mathcal{S}_1$

$$\begin{array}{l} \;\;\mathsf{def}_1 \\ \neg S_1(\underline{a}) \\ \;\;P_1(x) \vee R_1(x), \\ \neg R_1(x), \\ \;\;\psi^*_{11}, \\ \;\;\ldots, \\ \;\;\psi^*_{n1}, \\ \neg P_1(\underline{b}). \end{array}$$

The clauses $\neg P_1(\underline{b})$, $P_1(x) \vee R_1(x)$, and $\neg R_1(x)$ yield a contradiction. Since the clause introduced by the last application of the splitting rule is involved in the derivation of the empty clause, we have to consider the clause set $\mathcal{S}_2$

$$\begin{array}{l} \;\;\mathsf{def}_1 \\ \neg S_1(\underline{a}) \\ \;\;P_1(x) \vee R_1(x), \\ \neg R_1(x), \\ \;\;\psi^*_{11}, \\ \;\;\ldots, \\ \;\;\psi^*_{n1}, \\ \;\;R_1(x). \end{array}$$

Here, $\neg R_1(x)$ and $R_1(x)$ produce a contradiction. Since none of the clauses $\psi^*_{11}, \ldots, \psi^*_{n1}$ have been used in the refutation of $\mathcal{S}_1$ and $\mathcal{S}_2$, branch condensing will prevent the consideration of any of the alternative branches that exist for these clauses. SPASS proceeds directly by considering the branch where the clause $Q_1(y)$ belongs to the set of clauses.

Examples 6.5 and 6.6 illustrate how the branch pruning technique of the Logics Workbench can avoid two pitfalls in which $\mathcal{KRIS}^*$ and KSAT can be caught. Figure 6.12 shows however that brunch pruning alone does not lead to an improved median CPU time consumption for all formulae. The following example illustrates what happens.

Example 6.7:
Consider the formula ϕ_6

$$\begin{aligned} & p \vee \Box_1 q \\ \wedge\ & \neg\Box_1(p \vee r \vee q) \vee p \\ \wedge\ & \Box_1(p \vee q) \vee \Box_1(q \vee p) \\ \wedge\ & \Box_1 p \vee \Box_1 q \vee p. \end{aligned}$$

$\mathcal{KRIS}^*$ will easily detect the satisfiability of ϕ_6. After exhaustive application of the conjunction elimination rule, it applies $\vee\mathtt{E}$ to the first disjunction $p \vee \Box_1 q$, $\vee\mathtt{S}_0$ to the second disjunction, $\vee\mathtt{E}$ to the third disjunction and $\vee\mathtt{S}_0$ to the fourth disjunction. $\mathcal{KRIS}^*$ obtains the set of labelled formulae $\{w_0{:}p, w_0{:}\Box_1(p \vee q)\}$ to which no further rules can be applied.

The Logics Workbench has no equivalent to $\vee\mathtt{S}_0$ and deals with the second and fourth disjunction by means of the $(l\vee)$-rule. Furthermore, it will consider the left branch introduced by an (backwards) application of the $(l\vee)$-rule, first. So, it considers the sequent Γ_1

$$p, \neg\Box_1(p \vee r \vee q), \Box_1(p \vee q) \vee \Box_1(q \vee p), \Box_1 p \vee \Box_1 q \vee p \Rightarrow$$

before Γ_2

$$p, p, \Box_1(p \vee q) \vee \Box_1(q \vee p), \Box_1 p \vee \Box_1 q \vee p \Rightarrow$$

which are both obtained from $\Rightarrow \neg\phi_6$, by applications of the $(r\neg)$-, $(l\wedge)-$, and $(l\vee)$-rules. After further applications of the $(l\vee)$- and $(r\Box_i)$-rule, the Logics Workbench discovers that Γ_1 is provable and turns to Γ_2. Only then it detects that ϕ_6 is satisfiable.

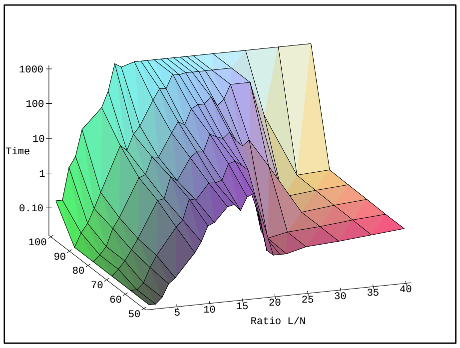
So, the Logics Workbench spends a serious amount of computational effort considering obviously useless branches introduced by the $(l\vee)$-rule. Figure 6.12 seems to indicate that this overwhelms the gain of branch pruning. It is worth noting that the behaviour of the Logics Workbench on KCNF formulae can be improved either by adding simplification rules or by employing better criteria for selecting the branches introduced by the $(l\vee)$-rule.

Example 6.7 also illustrates that is important to first assign a truth value to the propositional variables in a random formula since this allows to reduce the number of further assignments.

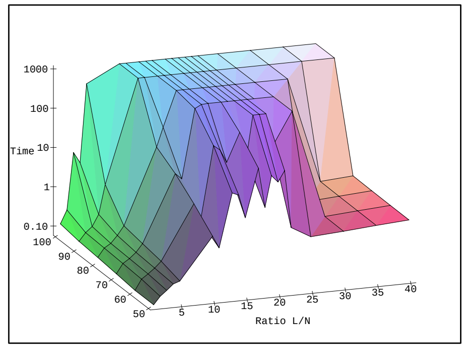
6.6 Broadening the evaluation

The graphs of the previous sections and of the papers of Giunchiglia and Sebastiani are 50% percentile graphs as each point presents the median CPU time consumption for 100 formulae with ratio L/N. This means that the graphs merely reflect the performance for the easier half of the formulae set. More informative are the collections of 50%, 60%, ... , 100% percentile graphs we present in Figures 6.13(a), 6.13(b), 6.13(c) and 6.13(d). Formally, the Q%-*percentile* of a set of data is the value V such that Q% of the data is smaller or equal to V and $(100 - Q)$% of the data is greater than V. The median of a set coincided with its 50%-percentile. The Figures 6.13(a), 6.13(b), 6.13(c) and 6.13(d) respectively show the percentile graphs for KSAT, $\mathcal{KRIS}^*$, LWB* and the translation approach on the parameter setting **PS0** (N=5, M=1, K=3, D=2). The difference in shape for KSAT, $\mathcal{KRIS}^*$, and the Logics Workbench as opposed to that for the translation approach is striking.

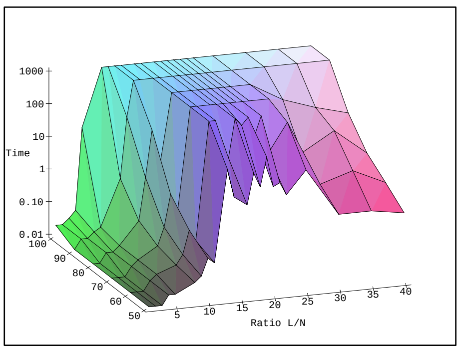
For the translation approach the difference between the 50%-percentile and the 90%-percentile of the CPU time consumption is marginal. We see the same monotonic increase of the CPU time



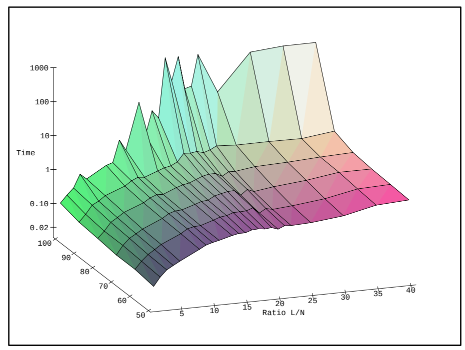
(a) KSAT



(b) $\mathcal{KRIS}$*



(c) LWB*



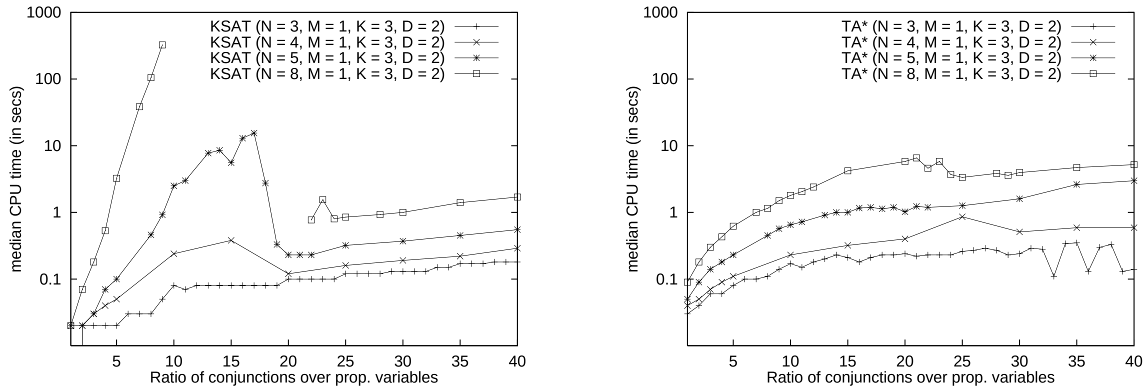
(d) Translation approach

Figure 6.13: The percentile graphs on **PS0**

consumption with increasing ratio L/N for all percentiles. Only the 100%-percentile reaches the time-limit of 1000 CPU seconds at some points. This means, there are some hard random 3CNF formulae in the collection, but for each ratio L/N their number does not exceed 10. This again supports our view that the problems generated using the parameter settings **PS0** are easier than the computational behaviour of KSAT and the other methods except the translation approach indicates.

The contrast to $\mathcal{KRIS}$ and the Logics Workbench is most extreme. While the Logics Workbench shows a good uniform behaviour where the ratio L/N is smaller than 10, we see a dramatic breakdown for ratios L/N greater than 10. As the percentage of trivially satisfiable samples reaches zero, the Logics Workbench can hardly complete 60% of the sample formulae within the time-limit. Even at ratios L/N above 30 where the percentage of trivially unsatisfiable formulae is greater than 90%, the Logics Workbench fails on 10% of the formulae. Similarly, for $\mathcal{KRIS}$. The absence of simplification rules in the Logics Workbench explains the less prominent 'valley' for ratios L/N above 30.

The percentage of sample formulae on which a decision procedure fails to complete its computation within a given time-limit (of reasonable size) may be regarded as a kind of risk for the user of that decision procedure. We call this the *failure risk*. The failure risk for each procedure



(a) Ksat

(b) Translation approach

Figure 6.14: Varying the parameter N

is reflected in Figures 6.13(a) to 6.13(d) by the size of the plateau at the time-limit of 1000 CPU seconds. The risk of failure for the parameter setting under examination is highest for the Logics Workbench and $\mathcal{KRIS}$*, and lowest for the translation approach.

We call the percentage of sample formulae on which a decision procedure terminates its computation within a given time-limit the *success chance* of a decision procedure. The notions of success chance and failure risk are complementary. The success chance will be regarded as an additional measure of the quality of a decision procedure. The weighting of the two quality measures, the success chance and median CPU time consumption, depends on the preferences of the user.

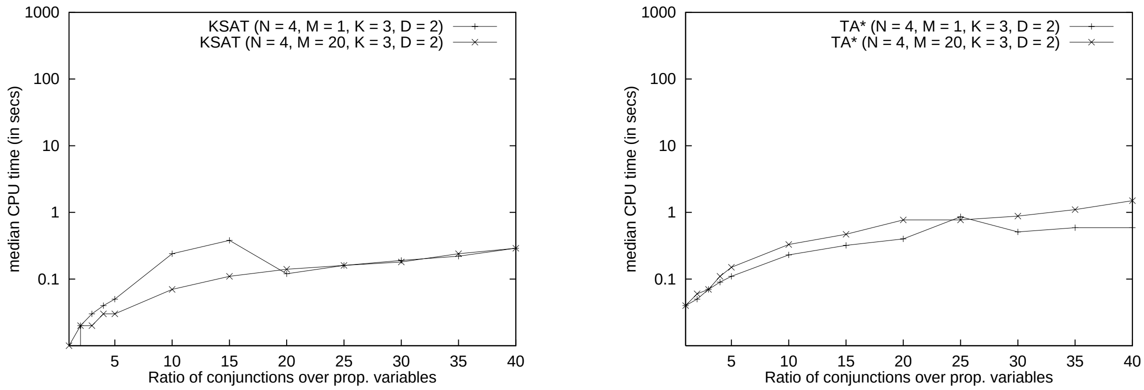
The percentile graphs are more informative and provide a better framework for comparison than the median curves. We can say Ksat performs better than $\mathcal{KRIS}$* and has a higher chance of success on the entire range of ratios L/N for the parameter setting **PS0**. The Logics Workbench is unbeatable for ratios L/N below 7.

We believe the graphs indicate a qualitative difference in the performance of the translation approach as opposed to the other three approaches.

6.7 Where are the hard problems?

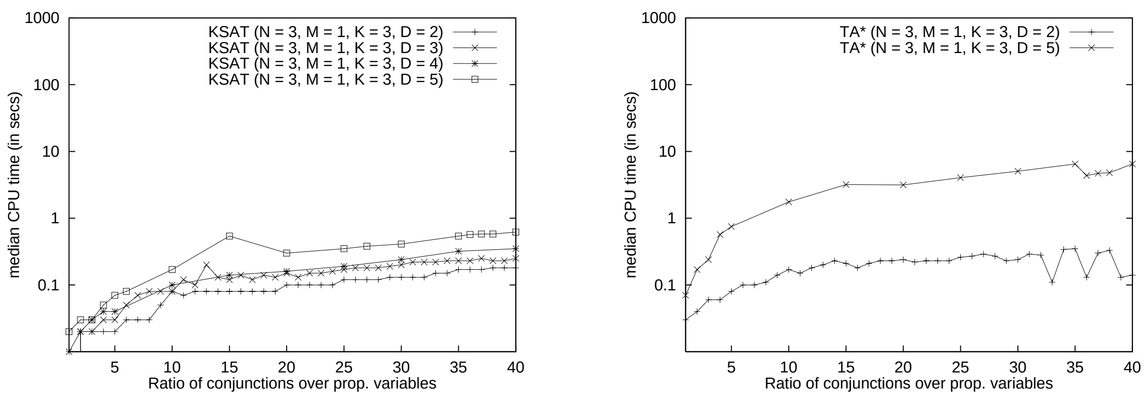
This section considers the question of how the parameter settings and random formula generator can be modified to provide better (more difficult) test samples.

The parameter setting **PS0** provides the most challenging collection of random 3CNF formulae among all the parameter settings used by Giunchiglia and Sebastiani. The Figures 6.14(a) and 6.14(b) show the influence of the parameter N, that is, the number of propositional variables, on the median CPU time consumption of Ksat and the translation approach. We see an increasing median CPU time consumption over the range of the ratio L/N with increasing value N. Thus increasing the number of propositional variables involved in the random generation of modal 3CNF formula provides more challenging test samples.



(a) Ksat (b) Translation approach

Figure 6.15: Varying the parameter M



(a) Ksat (b) Translation approach

Figure 6.16: Varying the parameter D

The Figures 6.15(a) and 6.15(b) provide an indication of the influence of the parameter M, that is, the number of modalities, on the median CPU time consumption of Ksat and the translation approach. The influence on the translation approach can be considered as being insignificant. Likewise we see that for a ratio L/N greater than 20, the median CPU time consumption of Ksat on the two parameter settings are identical. This can be explained by our observation that almost all unsatisfiable formulae are trivially unsatisfiable. The modal subformulae in trivially unsatisfiable formulae are irrelevant. Therefore, increasing the number of modalities is also irrelevant for unsatisfiable formulae. Below a ratio L/N of 20, the modal formulae generated using only one modality seem to be slightly more challenging than the modal formulae generated using twenty different modalities. This is due to the fact that the procedure KM is less likely to fail for twenty modalities than for just one modality [52]. The small divergence in the behaviour of Ksat on **PS5** (N=4, M=1, D=2, P=0.5) and **PS9** (N=4, M=20, D=2, P=0.5) is due to a smaller number of contradictions between modal literals for **PS9**. We illustrate this observation by the following example.

Example 6.8:
The formula ϕ_7 given by

$$(\Box_1(p \vee q) \vee \Box_1(r \vee q)) \wedge \neg\Box_1(q \vee p \vee s)$$

is satisfiable. KSAT will first apply the `dp_unit` rule replacing $\Box_1(q \vee p \vee s)$ by $\bot$. The first conjunct of ϕ_7 is left unchanged and KSAT has to apply the `dp_split` rule. Suppose it chooses $\Box_1(p\vee q)$ as split 'variable'. Replacing $\Box_1(p\vee q)$ by $\top$ renders ϕ_7 true propositionally, but checking the satisfiability of $\neg(q \vee p \vee s) \wedge (p \vee q)$ reveals that this truth assignment is not $\mathsf{K}_{(m)}$-satisfiable. So we have to continue with $\Box_1(r\vee q)$, the second case generated by the `dp_split` rule. Replacing the last remaining modal atom by $\top$ again renders the formula true propositionally. Finally, we have to check the satisfiability of $\neg(q \vee p \vee s) \wedge (r \vee q)$ which succeeds.

In contrast consider the formula ϕ_8 given by

$$(\Box_2(p \vee q) \vee \Box_1(r \vee q)) \wedge \neg\Box_1(q \vee p \vee s),$$

which is like ϕ_7 except the first occurrence of a $\Box_1$ is replaced by $\Box_2$. KSAT proceeds in the same way as for ϕ_7. It replaces $\Box_1(q \vee p \vee s)$ by $\bot$ and chooses $\Box_2(p \vee q)$ as split 'variable'. Replacing $\Box_2(p \vee q)$ by $\top$ renders ϕ true propositionally. But now instead of checking the satisfiability of $\neg(q \vee p \vee s) \wedge (p \vee q)$ we just have to check that $\neg(q \vee p \vee s)$ is satisfiable, because $p \vee q$ occurs below a different modality. Since this check succeeds ϕ_8 is satisfiable. Evidently, the computation for ϕ_8 is easier than for ϕ_7.

Now we vary the parameter D, the modal depth of the randomly generated modal 3CNF formulae. The situation for the parameter D is slightly more complicated than for the parameters N and M. By the definition of modal 3CNF formulae, increasing the modal depth increases the size of the formulae. The size, however, is an important factor influencing the performances of the procedures under consideration. Although the graphs in Figures 6.16(b) and 6.16(a) seem to indicate that increasing the modal depth of the sample formulae also increases the median CPU time consumption of the decision procedures, the increase parallels the increase of the median size of the modal formulae shown in Figure 6.17. A closer look at the graphs reveals that increasing the modal depth of the randomly generated modal 3CNF formulae actually makes the satisfiability problem easier. While the median formula size increases by a factor of five between modal depth 2 and modal depth 5, the median CPU time consumption of KSAT only increases by a factor of three.

Based on these observations we identify three guidelines for generating more challenging problems.

1. We have to avoid generating trivially unsatisfiable modal formulae. A straightforward solution is to require that all literals of a 3CNF clause of modal degree 1 are expressions of the form $\Box_1\phi$ or $\neg\Box_1\phi$ where ϕ is a random modal 3CNF clause of propositional variables. This amounts to setting the parameter P to zero.

2. For all occurrences of $\Box_1\phi$ in a random modal 3CNF formula of degree 1, ϕ has to be a non-tautologous clause containing exactly three differing literals.

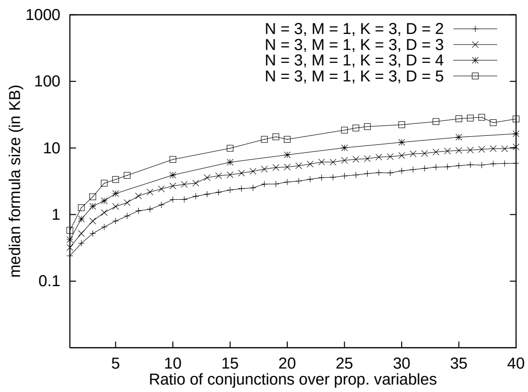


Figure 6.17: The influence of the parameter D on the formula size

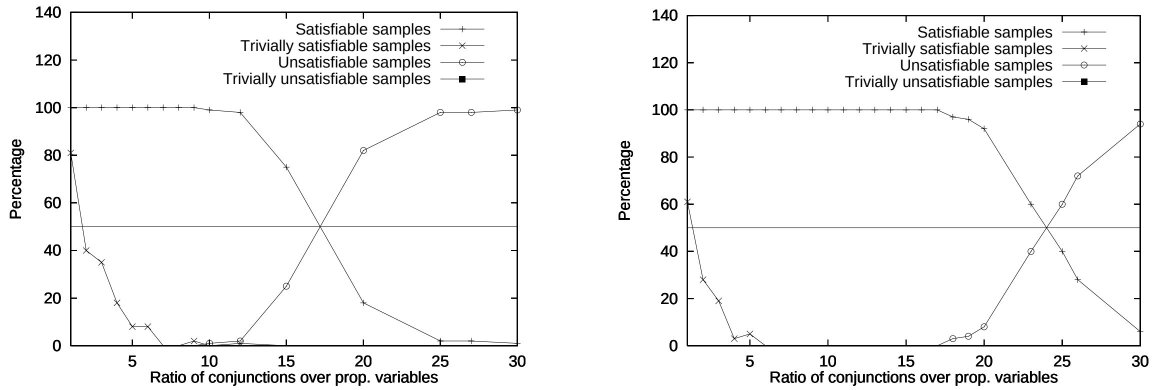
3. Parameters that have no significant influence on the "difficulty" of the randomly generated formulae should be set to the smallest possible value. This applies to the parameter M. As far as the parameter settings **PS0** to **PS9** are concerned it applies also to the parameter D. However, for formulae generated according to the first two guidelines increasing the parameter D leads to test formulae which are considerably more difficult[1]. In fact, they are too difficult to be suitable for an empirical comparison of existing modal theorem provers. Therefore, we restrict our attention to random modal 3CNF formulae of degree one using only one modality.

In line with the first guideline one may consider excluding also trivially satisfiable modal formulae. However, this amounts to doing preliminary satisfiability checks of the generated modal formulae in order to identify and reject the trivially satisfiable ones. For the moment, we do not perform these checks. Note that according to the second guideline, the generation of $\Box_1(p \vee \neg p \vee q)$ should be avoided. However, the generation of $\Box_1\varphi \vee \neg\Box_1\varphi' \vee \Box_1\psi$, where φ and φ' are either equal or identical modulo the associativity and commutativity of $\vee$, is permissible.

The restriction to random modal 3CNF formulae of degree one is somewhat surprising if one takes into account that if we bound D then the worst-case complexity of the satisfiability problem in basic modal logic is no longer **PSPACE**-complete, but **NP**-complete. This is a point that deserves further investigation. How can difficult modal formulae with increased modal degree be generated automatically? Some difficult examples of higher degree which have been constructed by hand can be found in the benchmark collection of the Logics Workbench [63].

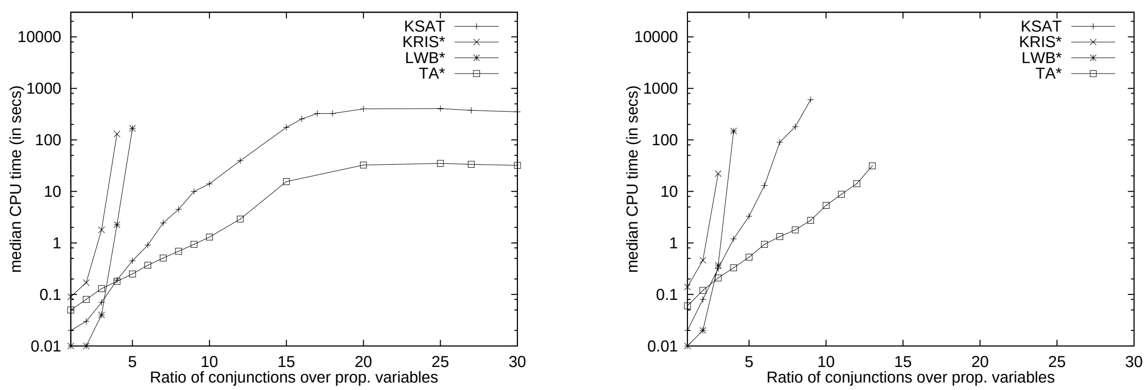
The parameters not fixed by the three guidelines are the number N of propositional variables and the number K of literals in any clause. We choose to fix $K{=}3$ in two parameter settings **PS12** ($N{=}4$, $M{=}1$, $K{=}3$, $D{=}1$, $P{=}0$) and **PS13** ($N{=}6$, $M{=}1$, $K{=}3$, $D{=}1$, $P{=}0$). Figures 6.18(a) and 6.18(b) reflect the quality of the parameter settings **PS12** and **PS13** by the percentage of satisfiable, unsatisfiable, trivially satisfiable, and trivially unsatisfiable modal formulae in the sample sets we generated. Compared to Figure 6.5(b) (on page 132) for the parameter setting **PS0**, the percentage of trivially satisfiable formulae has decreased significantly. As expected, the percentage of trivially unsatisfiable formulae is zero. Furthermore, the region in which the transition from almost always satisfiable formulae to almost always unsatisfiable formulae occurs is smaller. Already for a ratio L/N of 25 for **PS12** and a ration L/N of 30 for **PS13** there are

[1] Thanks to Ian Horrocks for pointing this out.



(a) **PS12** (N=4, M=1, K=3, D=1, P=0.0)   (b) **PS13** (N=6, M=1, K=3, D=1, P=0.0)

Figure 6.18: The quality of the test sets

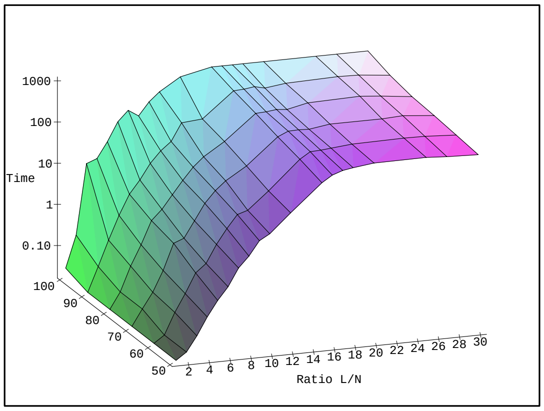


(a) **PS12** (N=4, M=1, K=3, D=1, P=0.0)   (b) **PS13** (N=6, M=1, K=3, D=1, P=0.0)

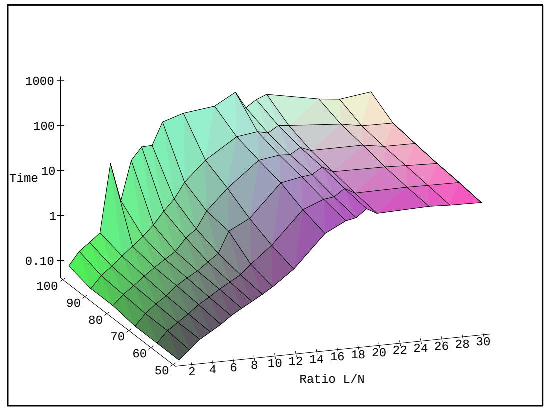
Figure 6.19: The median performances

almost no satisfiable formulae. For this reason, the experiments consider only the sample sets with ratio L/N between 1 and 30.

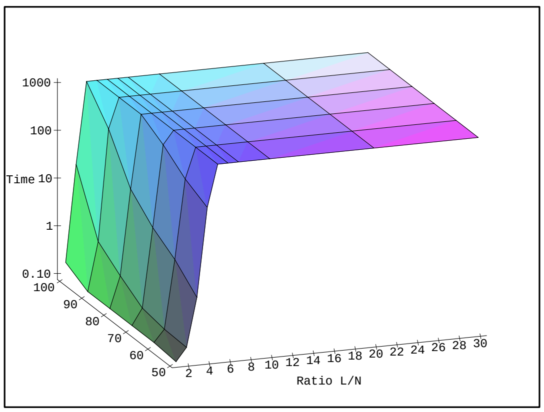
The percentile graphs of Ksat, $\mathcal{KRIS}$*, LWB* and the translation approach on the settings **PS12** and **PS13** are given in Figures 6.19(a) and 6.19(b). Figures 6.20(a) to 6.21(d) present the corresponding percentile graphs. Again, we observe that Ksat outperforms $\mathcal{KRIS}$* and the Logics Workbench, while the translation approach does best. More important, the formulae generated by the new parameter settings and the modified random generator are much harder than any of the formula samples generated for the settings **PS0** to **PS9** by the original generator. Figures 6.21(a) to 6.21(b) show the percentile graphs on **PS13**. We see that even the translation approach fails to decide within the given time-limit the satisfiability of half of the input formulae for ratios L/N greater than 13.



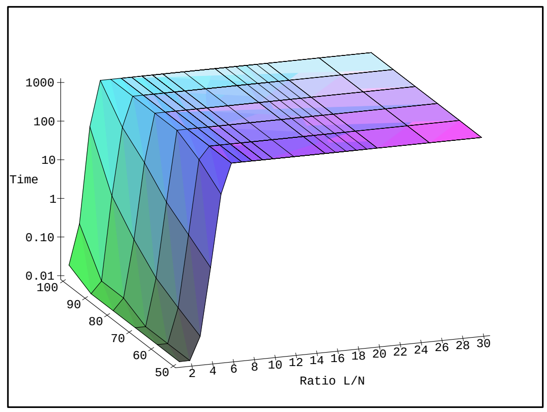
(a) KSAT



(b) Translation approach



(c) $\mathcal{KRIS}$*



(d) Logics Workbench LWB*

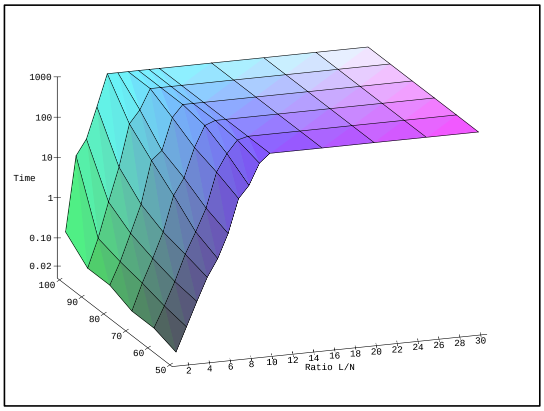
Figure 6.20: The percentile graphs on **PS12**

6.8 Conclusion

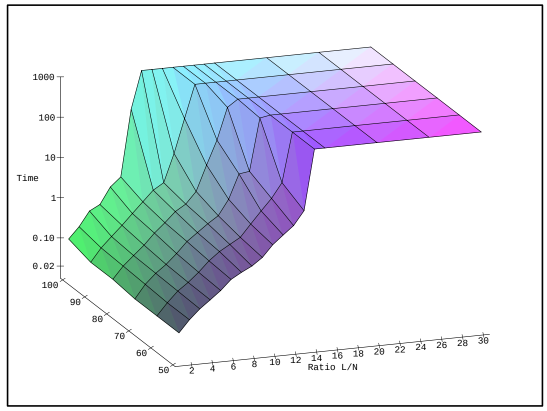
It should be stressed that it is not the aim of scientific benchmarking to find or declare the best-performing system. Instead the focus is on different techniques, strategies, and heuristics used in different theorem prover for improved performance on particular problem sets.

We have pointed out a number of problems with evaluating the performance of different algorithms for modal reasoning. A crucial factor is the quality of the randomly generated formulae. Even for propositional theorem proving defining adequate random formula generators for performance evaluation is hard [20]. We have shown that the random generator and parameter settings used in [51, 52] produce formulae with particular characteristics (like redundant subformulae and almost no non-trivially unsatisfiable formulae within the test sets) which have to be carefully taken into account in an empirical study. We have proposed guidelines for modifying the generator.

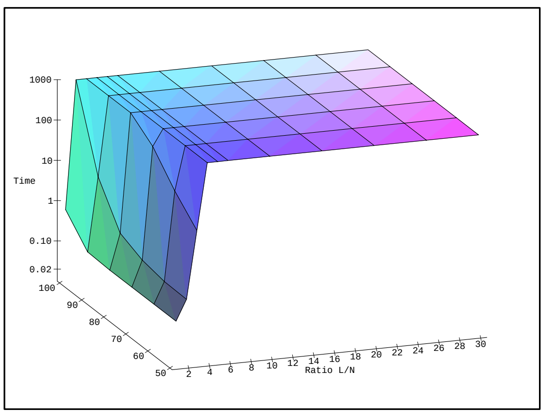
The basic algorithm of KSAT is an instance of a **KE**-tableaux algorithm augmented by simpli-



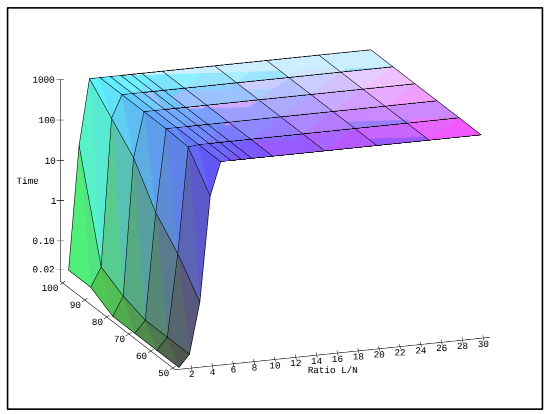
(a) Ksat



(b) Translation approach



(c) $\mathcal{KRIS}$*



(d) Logics Workbench LWB*

Figure 6.21: The percentile graphs on **PS13**

fication rules [99]. On modal KCNF formulae it is an instance of a **KE**-tableaux algorithm. Thus, the only fundamental difference between Ksat, $\mathcal{KRIS}$, and the Logics Workbench on formulae in conjunctive normal form is the inference rule for disjunction elimination.

The differences essential for the improved performance of Ksat as compared to $\mathcal{KRIS}$ and the Logics Workbench are:

1. Ksat utilises an elaborated set of simplification rules for boolean and modal formulae. These are the `dp_unit` inference rule of the procedure KDP and the rules in Table 6.2. These rules are applied whenever possible throughout the computation. By contrast, $\mathcal{KRIS}$ has only a very limited set of simplification rules, namely $\vee \mathsf{S}_0$ and $\vee \mathsf{S}_1$, which are applied occasionally. The Logics Workbench uses no simplification rules at all.

2. Ksat utilises a heuristic for selecting the particular disjunction for the application of disjunction elimination (namely, applying `dp_split` to a modal atom with maximal number of occurrences). By contrast, $\mathcal{KRIS}$ and the Logics Workbench process disjunctions in a

fixed order determined by the ordering of the disjunctions in the input formula.

3. Ksat performs intermediate checks of $\mathsf{K}_{(m)}$-satisfiability of the current truth assignment before every application of the `dp_split` rule. This corresponds to an application of our proposed $\Diamond_i \mathtt{T}$ inference rule for tableaux-based systems. $\mathcal{KRIS}$ has no equivalent of the $\Diamond_i \mathtt{T}$ rule.

   The Logics Workbench has a similar strategy as $\mathcal{KRIS}$. It delays the application of the $(l\Diamond_i)$- and the $(r\Box_i)$-rules until no further applications of the other rules are possible.

4. The Logics Workbench utilises branch pruning which the other systems do not.

Based on our performance evaluation and the insights we have gained by inspecting the code of the various systems under examination, our assessment of the relevance of these differences between the theorem provers concerning their performance is the following:

1. The presence of simplification rules and their exhaustive application is vital for any theorem prover, particularly for the class of formulae we have been considering. It is surprising that there are theorem provers like $\mathcal{KRIS}$ and the Logics Workbench making very little use of simplification.

2. Further investigations will have to answer whether elaborated heuristics for the selection of split 'variables' in the application of the `dp_split` rule or disjunctions in the application of the $\vee$-elimination rule lead to improved performance of Ksat for the entire range of generated sample sets.

3. The introduction of intermediate calls to the KM procedure to check the $\mathsf{K}_{(m)}$-satisfiability of the current truth assignment is valuable. It makes a difference to the performance of Ksat. However, in its present form Ksat cannot make optimal use of the information provided by a failure of an intermediate call to KM (Example 6.6).

   We envisage that more redundancy can be eliminated by delaying the application of rules dealing with modal operators and using branch pruning to backtrack to an appropriate state of the search space, like the Logics Workbench does.

Further improvements of the SAT-based procedure Ksat are possible and further investigations are needed to evaluate the usefulness of the various techniques. A first step in this direction is [50] with a new implementation of Ksat which is able to outperform the translation approach on **PS12** and **PS13**. Another example is the FaCT system delevoped by Horrocks [69, 70, 72] which combines techniques from Ksat with the branch pruning technique of the Logics Workbench.

All the techniques can be transferred to tableaux-based systems like $\mathcal{KRIS}$ and sequent calculus-based systems like the Logics Workbench. Likewise the techniques employed in $\mathcal{KRIS}$ and the Logics Workbench can be transferred to Ksat.

Our experiments show the suitability of the translation approach in combination with the theorem prover SPASS for modal theorem proving on all samples of randomly generated modal 3CNF formulae we have considered, except for the samples of very small or easily solvable formulae. This is due to the initial overhead of the transformation to clausal form. Hustadt, Schmidt, and Weidenbach [85] shows that this also extends to the carefully constructed benchmarking formulae of the Logics Workbench. It is open which resolution inference rules and search strategies perform best for basic modal logic and its extensions. We emphasise the positive results obtained for the combination of the translation approach and SPASS can most probably not be obtained with less sophisticated theorem provers (without splitting and branch condensing).

Conclusion

In this thesis we have considered various closely related solvable subclasses of first-order logic and have provided decision procedures for each of them in the resolution framework of Bachmair and Ganzinger [8]. We have shown that ordering refinements of resolution are able to decide the classes $\mathbf{E}^+$, $\overline{\mathrm{K}}$, $\overline{\mathrm{DK}}$, the class of DL-clauses which is related to description logics, and the class of SF-clauses which is related to modal logics.

In the case of description logics and modal logics we have given alternative decision procedures using a selection refinement of resolution and we have shown that these decision procedures are able to polynomially simulate standard tableaux-based decision procedures for these logics. In fact, the relation between the two approaches is so close that one can consider them as notational variants as long as we do not take the more powerful redundancy elimination techniques of the resolution-based approach into account.

There are closely related results which are worth mentioning. Ganzinger and de Nivelle [45] show that the guarded fragment with equality is solvable using a decision procedure based on resolution with superposition. Here a refinement based on an ordering and a selection function is used.

Fermüller and Salzer [40] show that ordered paramodulation and resolution provide a decision procedure for an extension of the Ackermann class with equality. They use a modified version of paramodulation as defined by Hsiang and Rusinowitch [73]. Fermüller and Leitsch [41] present a decision procedure for an extension $\mathcal{PVD}_g^=$ of the class $\mathcal{PVD}_+$ by ground equations based on a version of the derivations rules of [73] which removes the ordering restrictions on factoring and resolution and internalises factoring into the other rules. The decidability of $\mathcal{PVD}$ and $\mathcal{PVD}_+$ by hyperresolution is shown in [38]. These results can be reformulated in the framework described in this thesis.

There are two interesting solvable classes for which no resolution-based decision procedure has been found as yet. The first one is the fragment of first-order logic related to the independent joint of the modal logic $\mathsf{S5}$ with other modal logics. It is possible to use the refinement of resolution based on the selection function $S_{\mathcal{TAB}}$ together with a term depth bound to provide a decision procedure for this class. However, we do not regard this as a practical and compelling solution.

The second interesting logic is *fluted logic* [115, 116]. Fluted logic can be regarded as yet another alternative generalisation of the fragment of first-order logic corresponding to modal logics. It is defined in terms of a fixed ordering of quantifiers and variable occurrences in atomic subformulae. Fluted logic includes formulae of the form

$$\forall x_1, x_2 \colon p(x_1, x_2) \vee (\forall x_3 \colon q(x_1, x_2, x_3) \wedge \exists x_4 \colon p(x_2, x_4))$$

which are neither in the class $\overline{\mathrm{K}}$ nor in the guarded fragment. We conjecture that fluted logic can be decided by a resolution refinement making use of the dynamic renaming technique exemplified in Chapter 2.

Appendix A

Properties of regular terms and literals

To provide a self-contained presentation of the completeness proof, this section presents those lemmata and proofs by Zamov [39, chapter 6, pages 130–150] which remain mostly unchanged by the modifications in Section 3.3.

Lemma A.1. *The relation $\succsim_Z$ is transitive.*

Proof. Let s, t, and u be terms such that $s \succsim_Z t \succsim_Z u$. To show that $s \succsim_Z u$ holds, we consider the following cases:

1. $s = t$ or $t = u$ holds. Then we have $s \succsim_Z u$ trivially.
2. Let $s = f(s_1, \ldots, s_k)$, for some $k \geq 0$. If t is a variable and $t = s_i$ for some $1 \leq i \leq k$, then u is equal to t, since a variable can only dominate itself. Thus we have a reduction to the previous case.

   Let $t = g(t_1, \ldots, t_m)$, for some $k \geq m \geq 0$. We have to distinguish two cases: Either u is a variable and $u = t_i$ for some $1 \leq i \leq m$. Since $s \succsim_Z t$ holds, we have $s_i = t_i = u$. Therefore, $s \succsim_Z u$ holds.

   Finally, consider $u = h(u_1, \ldots, u_n)$, for some $m \geq n \geq 0$. Since $k \geq m \geq n$, and $s_i = t_i$ for $1 \leq i \leq m$ and $t_i = u_i$ for $1 \leq i \leq n$ holds, we have $s \succsim_Z u$. □

Corollary A.2. *The relation $\succsim_Z$ is a quasi-ordering on terms.*

Lemma A.3. *Let s and t be compound terms. Let σ be a substitution. If $s \succsim_Z t$, then $s\sigma \succsim_Z t\sigma$.*

Proof. We distinguish two cases:

1. The relation $s \succsim_Z t$ holds, because $s = t$ holds. Then $s\sigma = t\sigma$ and therefore, $s\sigma \succsim_Z t\sigma$.
2. The relation $s \succsim_Z t$ holds, because $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, $m \geq n \geq 0$, and $s_i = t_i$, for all $1 \leq i \leq n$.

   Then $s_i = t_i$, for all $1 \leq i \leq n$, implies $s_i\sigma = t_i\sigma$, for all $1 \leq i \leq n$. Since $s\sigma = f(s_1\sigma, \ldots, s_n\sigma)$ and $t\sigma = g(t_1\sigma, \ldots, t_m\sigma)$, we have $s\sigma \succsim_Z t\sigma$. □

Lemma A.4. *Let t be a compound, regular term $f(t_1, \ldots, t_n)$. Then all variables occurring in t are arguments of t as well. Furthermore, if t_i is a compound term, then all variables occurring in t_i occur in $\{t_1 \ldots, t_{i-1}\}$.*

Proof. The proof is by contradiction. Let t be of the form $f(t_1, \ldots, t_n)$. Let i, $1 \leq i \leq n$, be the smallest number such that t_i is a compound term $g(s_1, \ldots, s_m)$ containing a variable x which is not equal to an argument t_j of t for some j, $1 \leq j \leq n$. Since t dominates t_i, we have $m \leq n$, $m < i$, and $s_k = t_i$ for all k, $1 \leq k \leq m$. Because t_i contains x, one of its arguments s_l, $1 \leq l \leq m$, contains x. But $s_l = t_l$, which implies that s_l contains x, and $l < m < i$. Thus, i is not the smallest such index which is a contradiction. □

Lemma A.5. *A term t is regular iff it dominates each of its compound arguments.*

Proof. The "if" part follows from the fact that every compound term dominates every constant and every variable argument. The "only if" part is evident. □

Lemma A.6. *If t is a regular term and t dominates a term s then s is regular too.*

Proof. If the term s is not compound, then s is trivially regular. Let s be a compound term of the form $g(s_1, \ldots, s_m)$. Since t dominates s, it follows that t is a compound term of the kind $f(s_1, \ldots, s_m, \ldots, s_n)$, where f and g are function symbols, s_1, ... , s_n are terms, $n \geq m > 0$. By Lemma A.5 it is sufficient to prove that s dominates each of its compound arguments. Let $s_i = h(t_1, \ldots, t_k)$ be some argument of the term s, $i \leq m$. Since t is regular, t dominates s_i and it follows that $t_1 = s_1$, ... , $t_k = s_k$. Moreover we have $k < i$, since otherwise t_i and s_i would be equal, that is, s_i is equal to one of its proper subterms. Since $k < i$ and $i \leq m$ we have $k < m$ and each argument of the term s_i is equal to a corresponding argument of the term s. Therefore s dominates s_i. □

Lemma A.7. *Let t be a regular term and σ a substitution such that the codomain of σ contains only constants and variables. Then $t\sigma$ is a regular term.*

Proof. The substitution σ preserves the dominating relation for terms occurring in t as well as the depth of terms. □

Lemma A.8. *Let the regular term $t = f(t_1, \ldots, t_n)$ dominate the term $s = g(t_1, \ldots, t_m)$, $n \geq m \geq 1$, and let σ be a substitution such that all variables in the domain of σ occur in s or do not occur in t. If $s\sigma$ is regular, then $t\sigma$ is regular too.*

Proof. It is sufficient to prove that $t\sigma$ dominates each of its compound arguments (by Lemma A.5). Let $t_j\sigma$ be a compund term, $1 \leq j \leq n$. t_j cannot be a constant, because no instance of a constant can be a compound term. The remaining two cases are:

1. t_j is a variable. Then t_j is a variable in the domain of the substitution σ, since otherwise $t_j\sigma = t_j$ which is impossible, because $t_j\sigma$ is a compound term. By the assumption of the lemma, t_j occurs in s. The term $t_j\sigma$ is an argument of the term $s\sigma$. Therefore, $s\sigma$ dominates $t_j\sigma$, since $s\sigma$ is regular. Due to Lemma 3.7 $t\sigma$ dominates $s\sigma$. Because the dominating relation is transitive, $t\sigma$ dominates $t_j\sigma$.

2. t_j is a compound term. It follows that t dominates t_j, since t is regular. Since the dominating relation is preserved under substitution in compound terms, the term $t\sigma$ dominates $t_j\sigma$.

So we have shown that $t\sigma$ dominates each of its compound arguments and thus is regular. □

Lemma A.9. *Let $t = f(t_1, \dots, t_n)$ be a regular term which dominates the term $s = g(t_1, \dots, t_m)$, z be a variable which does not occur in s and let σ be the substitution $\{z/s\}$. Then $t\sigma$ is regular.*

Proof. Let u be an argument of t and let $u\sigma$ be a compound term. There are the following cases:

1. u is a variable different from z. Then $u\sigma = u$ which is impossible, since $u\sigma$ is a compound term.

2. u is equal to z. Since s does not contain z, the terms $t_1, \dots, t_m$ do not contain z as well. Therefore

$$\begin{aligned} t\sigma &= f(t_1, \dots, t_m, t_{m+1}, \dots, t_n)\sigma \\ &= f(t_1\sigma, \dots, t_m\sigma, t_{m+1}\sigma, \dots, t_n\sigma) \\ &= f(t_1, \dots, t_m, t_{m+1}\sigma, \dots, t_n\sigma). \end{aligned}$$

   Thus $u = t_j$ for some j, $m + 1 \leq j \leq n$. So, $u\sigma = z\sigma = s = t_j\sigma$.

$$\begin{aligned} t\sigma &= f(t_1, \dots, t_m, \dots, t_j\sigma, \dots) \\ &= f(t_1, \dots, t_m, \dots, s, \dots). \end{aligned}$$

   It follows that $t\sigma$ dominates $u\sigma$.

3. u is a compound term or a constant. Since t is regular, it dominates the term u, therefore $t\sigma$ dominates $u\sigma$. □

Lemma A.10. *If s and t are regular terms and σ is a most general unifier of $\{t, s\}$ then $t\sigma$ is regular.*

Proof. If s and t are syntactically equal, then the most general unifier is the identity substitution and $t\sigma$ is regular. This includes the case where s and t are constants. Otherwise, we consider the following cases:

1. One of the terms (s, for example) is a variable. Then t does not contain s, $\sigma = \{s/t\}$ and $t\sigma = t$ which is regular.

2. Both s and t are compound terms of the kind $f(s_1, \dots, s_n)$ and $f(t_1, \dots, t_n)$, respectively. We will prove the regularity of $t\sigma$ by induction on $n - i$, where i satisfies the following conditions:

   (A.1) $$t_1 = s_1, \dots, t_i = s_i, \quad (1 \leq i \leq n)$$

   (A.2) $$t_{i+1} \neq s_{i+1}.$$

   Since s and t are unifiable, the terms s_{i+1} and t_{i+1} are unifiable.

   The induction base is trivial.

   Let $n - i > 0$. Let us first demonstrate that in this case one of the terms t_{i+1}, s_{i+1} is a variable. Assume the contrary and consider the following subcases:

(a) Neither term is compound. In this case terms t_{i+1} and s_{i+1} are constants. They are different by Condition (A.2). But then they are not unifiable which contradicts the unifiability of t and s.

(b) The t_{i+1} is a compound term and s_{i+1} is a constant. Again, these terms are not unifiable.

(c) Both terms are compound. Assume $t_{i+1} = g(v_1, \ldots, v_m)$, $s_{i+1} = g(u_1, \ldots, u_m)$. Since t is regular, it dominates the term t_{i+1}, therefore

$$t_1 = v_1, \ldots, t_m = v_m \tag{A.3}$$

Since no term contains itself as argument, it follows that $t_{i+1} \neq v_k$ for each k, $1 \leq k \leq m$. Therefore $i + 1 > m$. Similarly, we can show that

$$s_1 = u_1, \ldots, s_m = u_m \tag{A.4}$$

We conclude from (A.1), (A.3), and (A.4) that

$$v_k = t_k = s_k = u_k$$

for every k, $1 \leq k \leq m$. It follows that $t_{i+1} = s_{i+1}$ which contradicts Condition (A.2).

We have proved that at least one of the terms t_{i+1} or s_{i+1} is a variable. Without loss of generality, we can assume that t_{i+1} is a variable. Let σ_{i+1} be the substitution $\{t_{i+1}/s_{i+1}\}$.

If s_{i+1} is a variable or a constant, then both $t\sigma$ and $s\sigma$ are regular by Lemma A.7.

Let s_{i+1} be a compound term. Since s is regular, it dominates the term s_{i+1}. Since the first i arguments of terms s and t are equal by Condition (A.1), t dominates s_{i+1}. By Lemma A.9 the terms $t\sigma_{i+1}$ and $s\sigma_{i+1}$ are regular. Consequently, the following equations for the arguments of $t\sigma_{i+1}$ and $s\sigma_{i+1}$ are valid:

$$\begin{aligned} t_1\sigma_{i+1} &= s_1\sigma_{i+1} \\ &\vdots \\ t_i\sigma_{i+1} &= s_i\sigma_{i+1} \\ t_{i+1}\sigma_{i+1} &= s_{i+1}\sigma_{i+1} \\ &\vdots \end{aligned}$$

By the induction hypothesis $t\sigma_{i+1}\theta$ and $s\sigma_{i+1}\theta$ are regular terms, where θ is a most general unifier for $t\sigma_{i+1}$ and $s\sigma_{i+1}$. □

Lemma A.11. *Let L be a deep, regular literal, let the term t be a dominating term of A and let σ be a substitution such that $t\sigma$ is regular. Then $L\sigma$ is a regular literal.*

Proof. Let us show that $t\sigma$ dominates each argument of $L\sigma$. Assume that s is some argument of L. There are the following cases:

1. s is a variable. In this case s is an argument of the dominating term t since L is a regular literal. It follows that $s\sigma$ is an argument of $t\sigma$. Since $t\sigma$ is a regular term, it dominates each of its arguments, particularly the term $s\sigma$.

2. s is a constant or a compound term. Then t dominates s. In both cases the $\gtrsim_Z$ is preserved under substitution. Therefore $t\sigma$ dominates the term $s\sigma$. $\square$

Lemma A.12. *Let $L_1 = (\neg)p(s_1, \ldots, s_n)$ and $L_2 = (\neg)p(t_1, \ldots, t_n)$ be unifiable, deep, regular literals. If s_i is a dominating term of A_1, then also t_i must be a dominating term of A_2.*

Proof. Since L_1 is a deep literal, its dominating term s_i has maximal arity among all arguments of this literal and is a compound term. Let t_j be a dominating term of the literal L_2, that is, t_j is a compound term whose arity is maximal among arities of all arguments of L_2. Let σ be the most general unifier of L_1 and L_2. We distinguish the following cases:

1. The terms t_i and s_j are variables. Since L_1 is a deep literal, the variable s_j is an argument of the dominating term s_i. For the same reason t_i is an argument of the term t_j. The relation "to be an argument of" is preserved under substitution, therefore the following property holds:

   $$s_j\sigma \text{ is an argument of } s_i\sigma \text{ and } t_i\sigma \text{ is an argument of } t_j\sigma \tag{A.5}$$

   From the unifiability of L_1 and L_2 we conclude that

   $$s_i\sigma = t_i\sigma \text{ and } s_j\sigma = t_j\sigma \tag{A.6}$$

   It follows from (A.5) and (A.6) that $s_j\sigma$ is an argument of $s_j\sigma$ (and thus a proper subterm of itself), which is impossible.

2. One of the terms s_j or t_i (say, s_j) is a variable and the other one is a constant or a compound term. Then s_j is an argument s_i and t_j dominates the term t_i, therefore $\text{arity}(t_i) \leq \text{arity}(t_j)$. Since for non-variable terms the dominating relation is preserved under substitution, the following statements hold:

   $$s_j\sigma \text{ is an argument of } s_i\sigma \text{ and } t_j\sigma \text{ dominates } t_i\sigma \tag{A.7}$$

   From (A.6) and (A.7) we conclude that $s_j\sigma$ is an argument of some term which is dominated by $t_j\sigma$. Therefore, $s_j\sigma$ is an argument of $t_j\sigma$ which is impossible.

3. Neither t_i nor s_j is a variable. Then the following inequalities hold for the arities of the terms under consideration:

   $$\begin{aligned} \text{arity}(s_i) &\geq \text{arity}(s_j) && \text{since } s_i \text{ is a dominating term for } L_1, \\ \text{arity}(t_j) &\geq \text{arity}(t_i) && \text{since } t_j \text{ is a dominating term for } L_2, \\ \text{arity}(s_i) &= \text{arity}(t_i) && \text{since } s_i \text{ and } t_i \text{ are unifiable}, \\ \text{arity}(s_j) &= \text{arity}(t_j) && \text{since } s_j \text{ and } t_j \text{ are unifiable.} \end{aligned}$$

   These inequalities imply that $\text{arity}(s_i) = \text{arity}(s_j) = \text{arity}(t_i) = \text{arity}(t_j)$. It follows that t_i is a dominating term for L_2 as well. $\square$

To illustrate the previous lemma, take a look at the following examples:

$$L_1 = p(f(x,y), y) \qquad L_2 = p(f(x,y), g(x,y,z)) \tag{A.8}$$

The dominating term of L_1 is the first argument of L_1, that is, $f(x, y)$. The dominating term of L_2 is the second argument of L_2, that is, $g(x, y, z)$. Thus the dominating terms do not occur at the same argument position. It is straightforward to check, that these literals are not unifiable. So, one of the preconditions of Lemma A.7 is not satisfied.

$$L_1 = p(f(x, y), y) \qquad L_2 = p(f(x', y'), g(x')). \tag{A.9}$$

These literals are unifiable with most general unifier $\{x'/x, y/g(x), y'/g(x)\}$. The dominating term of L_1 is the first argument of L_1, that is, $f(x, y)$. The dominating term of L_2 is the first argument of L_2, that is, $f(x', y')$. The dominating terms are at the same argument position in L_1 and L_2, respectively. Finally, we consider

$$L_1 = p(f(x, y), y) \qquad L_2 = p(f(x', y'), g(u, v, w)). \tag{A.10}$$

These literals are unifiable with most general unifier $\{x'/x, y/g(u, v, w), y'/g(u, v, w)\}$. The dominating term of L_1 is the first argument, that is, $f(x, y)$. The term with the maximal arity in L_2 is the second argument, that is, $g(u, v, w)$. But $g(u, v, w)$ does not dominate $f(x', y')$. Nor does $f(x', y')$ dominate $g(u, v, w)$. L_2 is not a regular literal.

Lemma A.13. *Let $L_1 = (\neg)p(s_1, \dots, s_n)$ and $L_2 = (\neg)p(t_1, \dots, t_n)$ be regular literals and let σ be the most general unifier of L_1 and L_2. Then $L_1\sigma$ is regular.*

Proof. We consider the following cases:

1. L_1 and L_2 are both deep literals. Assume that s_i is a dominating term for L_1. By Lemma A.12 the dominating term for L_2 is the term t_i. Let θ be the most general unifier of s_i and t_i. By Lemma A.10 the terms $s_i\theta$ and $t_i\theta$ are regular. By Lemma A.11 the literals $L_1\theta$ and $L_2\theta$ are regular and their dominating terms $s_i\theta$ and $t_i\theta$ are equal.

   We use induction on the number of non-equal arguments in $L_1\theta$ and $L_2\theta$ to prove the lemma.

   The induction base is trivial. Let $s_j\theta$ differ from $t_j\theta$ for some j, $1 \leq j \leq n$, and let θ_j be a most general unifier of these terms. Since $s_i\theta$ is regular and dominates the term $s_j\theta$, this last term is regular by Lemma A.6. For the same reason $t_j\theta$ is regular. Therefore, the literal $(L_1\theta)\theta_j$ is regular by Lemma A.11. Similarly, we can prove that the literal $(L_2\theta)\theta_j$ is regular. By the induction hypothesis the lemma holds for the literals $(L_1\theta)\theta_j$ and $(L_2\theta)\theta_j$.

2. One of the literals, L_2 for example, contains no compound terms and the other one is a deep literal with a dominating term s_i. In this case we can prove the lemma by induction on the number of pairs of terms (s_k, s_l) such that $s_k \neq s_l$ and $t_k = t_l$, for $1 \leq k < l \leq n$.

   In the base case, the number of such pairs is zero. That means $t_k = t_l$ if and only if $s_k = s_l$ for all k, l, $1 \leq k < l \leq n$. For all variables x in the domain of σ which occur in L_1, $x\sigma$ is either a variable or a constant. By Lemma A.7 $s_i\sigma$ is regular and by Lemma A.11 $L_1\sigma$ is regular.

   In the induction step, we consider some arbitrary pair (s_k, s_l) such that $t_k = t_l$ and $s_k \neq s_l$. Since L_1 and L_2 are unifiable, s_k and s_l are unifiable too. Let θ be a most general unifier of s_k and s_l. Since s_k and s_l are regular terms, $s_k\theta = s_l\theta$ is a regular term according to Lemma A.10. The term s_i dominates s_k and s_l and all variables in the domain of θ occur in either s_k or s_l. By Lemma A.8 the term $s_i\theta$ is regular. By Lemma A.11 the literal $L_1\theta$ is regular. For the literals $L_1\theta$ and $L_2\theta$ the lemma holds by the induction hypothesis.

3. The literals L_1 and L_2 are both shallow. The proof is evident since $L_1\sigma$ and $L_2\sigma$ do not contain compound terms and therefore are regular. □

The proof presented here differs from Zamov's proof in the following aspect. In the base case Zamov assumes that the most general unifier is a match, that is, $L_2\sigma = L_1 = L_1\sigma$. However, in the presence of constants this need not be true. Consider $L_1 = p(y, f(x, y))$ and $L_2 = p(a, z)$. The most general unifier $\{y/a, z/f(x, a)\}$ is not a match.

Lemma A.14. *Let C be a regular clause, L_1 be a dominating literal for C and t_1 be a dominating term for L_1. Then t_1 dominates each argument of each literal in C.*

Proof. Let t_2 be some non-constant argument of some literal L_2 from C. Since A_1 is a dominating literal, it dominates L_2, that is, there exists some term in L_1 which dominates t_2. Since L_1 is regular, t_1 dominates each argument of L_1, therefore it dominates t_2.

Suppose t_2 is a constant, then t_1 trivially dominates t_2. □

Lemma A.15. *If a regular term t dominates the term s and σ is a substitution such that $t\sigma$ is regular, then $t\sigma$ dominates $s\sigma$.*

Proof. According to Lemma 3.7 we only have to consider the case where at least one of the terms is not compound. In addition, the cases where $s = t$ holds and where s is a constant are trivial.

Suppose s is a variable. Since $s \neq t$, t is a compound term such that s is an argument of t. Then $s\sigma$ is again an argument of $t\sigma$. Since $t\sigma$ is regular, it dominates each of its arguments. Therefore, $t\sigma$ dominates $s\sigma$. □

Corollary A.16. *If a regular term t dominates the term s and σ is a substitution such that $t\sigma$ is regular, then $s\sigma$ is regular.*

Proof. By Lemma A.15 and A.6. □

Lemma A.17. *Let C be a k-regular clause and t be a regular term in some literal in C. Neither of the first k arguments of the term t are compound.*

Proof. Let $t = f(t_1, \dots, t_n)$, $n \geq k$. Assume that t_i is compound for some i, $1 \leq i \leq k$, that is, $t_i = g(u_1, \dots, u_m)$ for some m, $k \leq m \leq n$. Since the term t is regular, it dominates t_i, therefore $u_j = t_j$ for every j, $1 \leq j \leq m$. It follows that $t_i = g(t_1, \dots, t_m)$ for some i, $1 \leq i \leq k \leq m$, which is impossible. □

Appendix B

Glossary of solvable classes

The following sections contain a brief summary of the definitions of the classes which have been introduced and for which terminating resolution procedures are presented in this thesis.

B.1 The classes $\mathbf{E}^+$ and $\mathbf{E}_1$

A compound term t is *covering* if for every compound subterm s of t the sets of variables of s and t are identical, that is, $\mathcal{V}(s) = \mathcal{V}(t)$. A compound term t is *weakly covering* if for every non-ground, compound subterm s of t $\mathcal{V}(s) = \mathcal{V}(t)$ holds.

An atom or literal L is *covering* if each argument of L is either a constant, a variable, or a covering term t with $\mathcal{V}(t) = \mathcal{V}(L)$. An atom or literal L is *weakly covering* if each argument of L is either a ground term, a variable, or a weakly covering term t with $\mathcal{V}(t) = \mathcal{V}(L)$. A clause C is *variable uniform* if (i) every literal in C is weakly covering, and (ii) for each literal L_1 and L_2 in C either $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ or $\mathcal{V}(L_1) \cap \mathcal{V}(L_2) = \emptyset$ holds.

Definition B.1 (The class $\mathbf{E}^+$).
A clause set N belongs to the class $\mathbf{E}^+$ iff all clauses C in N are variable uniform.

Definition B.2 (The class $\mathbf{E}_1$).
A clause C belongs to $\mathbf{E}_1$ if (i) every literal in C is covering, and (ii) for each literal L_1 and L_2 in C either $\mathcal{V}(L_1) = \mathcal{V}(L_2)$ or $\mathcal{V}(L_1) \cap \mathcal{V}(L_2) = \emptyset$ holds.

B.2 The classes $\overline{\mathrm{K}}$, $\overline{\mathrm{DK}}$, $\overline{\mathrm{KC}}$, and $\overline{\mathrm{DKC}}$

Let φ be a schema in negation normal form and ψ a subformula of φ. The *φ-prefix* of the formula ψ is a sequence of quantifiers of the schema φ which bind the free variables of ψ.

If a φ-prefix is of the form $\exists y_1 \ldots \exists y_m \forall x_1 Q_1 z_1 \ldots Q_n z_n$, where $m \geq 0$, $n \geq 0$, $Q_i \in \{\exists, \forall\}$ for all i, $1 \leq i \leq n$, then $\forall x_1 Q_1 z_1 \ldots Q_n z_n$ is the *terminal φ-prefix*. For a φ-prefix $\exists y_1 \ldots \exists y_m$ the terminal φ-prefix is the empty sequence of quantifiers.

Definition B.3 (The class $\overline{\mathrm{K}}$).
The schema φ in negation normal form belongs to the class $\overline{\mathrm{K}}$ if there are k quantifiers $\forall x_1$, $\ldots$, $\forall x_k$, $k \geq 0$, in φ not interspersed with existential quantifiers, such that for every atomic subformula ψ of φ the terminal φ-prefix of ψ, (i) either is of length less than or equal to 1, or (ii)

ends with an existential quantifier, or (iii) is of the form $\forall x_1 \forall x_2 \ldots \forall x_k$.

We say the variables $x_1, \ldots, x_k$, $k \geq 0$, are the *fixed universally quantified variables of* φ and φ is of *grade* k, indicating the number of fixed universally quantified variables.

Definition B.4 (The class $\overline{\mathrm{DK}}$).
Let $\varphi_1, \ldots, \varphi_n$ be formulae in the class $\overline{\mathrm{K}}$. Then $\varphi_1 \wedge \ldots \wedge \varphi_n$ is a formula in the class $\overline{\mathrm{DK}}$.

Definition B.5 (The class $\overline{\mathrm{KC}}$).
Without loss of generality we can restrict ourselves to formulae in prenex form whose matrix is in conjunctive normal form, that is, schemas in $\overline{\mathrm{K}}$ have the form

$$\exists y_1 \ldots \exists y_m \forall x_1 \ldots \forall x_k Q_1 z_1 \ldots Q z_l \bigwedge_{i=1,\ldots,n} \bigvee_{j=1,\ldots,m_i} L_{i,j} \tag{B.1}$$

where $m \geq 0$, $k \geq 0$, $l \geq 0$, $n > 0$, $m_i > 0$, and $L_{i,j}$ are literals. We assume that outer Skolemisation is used in the process of transforming (B.1) to clausal form.

Definition B.6 (The class $\overline{\mathrm{DKC}}$).
If φ is a formula in $\overline{\mathrm{KC}}$, then the clausal form of φ (using outer Skolemisation) is in the class $\overline{\mathrm{DKC}}$.

The term t *dominates* the term s, denoted by $t \succsim_Z s$, if (i) $t = s$, or (ii) $t = f(t_1, \ldots, t_n)$, s is a variable and $s = t_i$ for some i, $1 \leq i \leq n$, or (iii) $t = f(t_1, \ldots, t_n)$, $s = g(t_1, \ldots, t_m)$, $n \geq m \geq 0$. The set T_1 of terms *dominates* the set T_2 of terms if for every term t_2 in T_2 there exists a term t_1 in T_1 such that t_1 dominates t_2. Two terms s and t are *similar* if s dominates t and t dominates s. The literal L_1 *dominates* the literal L_2, denoted by $L_1 \succsim_Z L_2$, if the set of non-constant arguments of L_1 dominates the set of non-constant arguments of L_2. Two literals L_1 and L_2 are *similar* if the set of non-constant arguments of L_1 dominates the set of non-constant arguments of L_2, and vice versa.

A term is called *regular* if it dominates all its arguments A set of terms is called *regular* if it contains no compound term or it contains some regular compound term which dominates all terms of this set. A literal is called *regular* if the set of its arguments is regular.

A literal L is *singular* if it contains no compound term and $\mathcal{V}(L)$ is a singleton, otherwise it is *non-singular*. A regular literal containing a compound term is *deep*, otherwise it is *shallow*.

Definition B.7 (Regular clause).
A clause C of literals is k*-regular* if (i) C contains regular literals only, (ii) k is a non-negative integer not greater than the minimal arity of the non-constant function symbols occurring in C (if C does not contain compound terms, then k is arbitrary), (iii) C contains some literal which dominates every literal in the set C, (iv) iff L_1 and L_2 are non-singular, shallow literals in C, then L_1 and L_2 are similar, (v) if L_1 is a non-singular, shallow literal in C, then for all compound terms t occurring in any literal in C, $\arg_{set}(L_1) \setminus \mathsf{F}_0 \sim_Z \arg_{set}^{1\ldots k}(t) \setminus \mathsf{F}_0$ holds.

A clause is *regular* if it is k-regular for some $k \geq 0$. A clause is again called *quasi-regular* if all of its indecomposable components are regular.

B.3 DL-clauses and fluted DL-clauses

Let C be a clause and t be compound term in C. The term t is *(variable) embracing* if for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(t) \neq \emptyset$ implies $\mathcal{V}(L) \subseteq \mathcal{V}(t)$. A literal L in C is *(variable) embracing* if (i) for every L' in C, $\mathcal{V}(L') \cap \mathcal{V}(L) \neq \emptyset$ implies $\mathcal{V}(L') \subseteq \mathcal{V}(L)$ (that is, embracing literals contain all variables occurring in their split component of the clause), and (ii) if L contains a compound term t, then t is embracing.

A literal L is a *DL-literal* iff (i) L is regular, (ii) L is either monadic or dyadic and contains at most 2 variables, (iii) L is ground whenever L contains a constant symbol, and (iv) the maximal arity of function symbols in L is 1.

Definition B.8 (DL-clause).
A clause C is a *DL-clause* iff (i) if C contains a compound term t, then t is embracing, (ii) C is ground whenever C contains a constant symbol, (iii) all literals in C are DL-literals, and (iv) the argument multisets of all flat, dyadic literals coincide.

A literal L is a *fluted DL-literal* iff (i) L is regular, (ii) L is either monadic or dyadic and contains at most 2 variables, (iii) L is ground whenever L contains a constant symbol, (iv) the maximal arity of functions symbols in L is 1, and (v) there is at most one compound term t in L and t can only occur in the last argument position of L.

Definition B.9 (Fluted DL-clause).
A clause C is a *fluted DL-clause* iff (i) C is a 1-regular clause of grade k where $k \leq 2$ holds, (ii) C is ground whenever C contains a constant symbol, (iii) all literals in C are fluted DL-literals, and (iv) there exist distinct variables x and y such that all flat, dyadic literals in C are of the form $p(x, y)$ for some predicate symbol p.

B.4 Small SF-clauses and SF-clauses

A signature containing only predicate symbols of maximal arity 2 such that all arguments have to be of sort W, one constant symbol ϵ of sort W, unary function symbols of sort $W \to AF$, and one binary function symbol of sort $[\_\,\_] : W \times AF \to W$ is called an *SF-signature*. A well-sorted, regular term over an SF-signature is called an *SF-regular term*.

A clause C is an *SF-regular clause* iff C is a well-sorted, strongly CDV-free, regular clause over an SF-signature such that (i) there are no occurrences of negative, dyadic literals, (ii) there is at most one occurrence of a positive, dyadic literal L, (iii) the first argument of a dyadic literal L in C is a subterm of the second argument of L, and (iv) if C contains a compound term t and a dyadic literal L, then t is identical to the second argument of L.

Definition B.10 (Small SF-clause).
A clause C is a *small SF-clause* if (i) C is a SF-regular clause, or (ii) C is in one of the following forms

$$(\mathcal{C}_\Box) \qquad \mathcal{P}(\bar{x}_2) \cup \{\neg r(x_1, x_2)\},$$
$$(\mathcal{C}_5) \qquad \mathcal{P}(\bar{x}_2) \cup \{r([x_1\alpha_1], [x_2\alpha_2])\},$$
$$(\mathcal{C}_{45}) \qquad \mathcal{P}(\bar{x}_2) \cup \{r(x_1, [x_2\alpha_2])\},$$

where x_1 and x_2 are variables of sort W, and α_1 and α_2 are variables of sort AF.

Definition B.11 (SF-clause).
A clause C is an *SF-clause* iff (i) C is an SF-regular clause, or (ii) C is a clause of the form

$$(\mathcal{C}_{inv}) \qquad \mathcal{P}(\bar{u}) \cup \neg r(\bar{u}, v) \cup \mathcal{P}(v) \cup \mathcal{P}(\bar{w}) \cup \neg r(\bar{w}, t) \cup \mathcal{P}(t).$$

where v is either a variable of sort W or the constant ϵ, $\bar{u}$ and $\bar{w}$ are vectors of variables and constants of sort W, $t = [v\alpha]$ for some variable α of sort AF or $t = [vf(v)]$ for some unary function symbol f, such that, additionally, if u and w are variables occurring in a monadic atom in C, then there is at most one negative r literal in which this variable occurs, or (iii) C is of the form

$$(\mathcal{C}_4) \qquad \mathcal{P}(\bar{x}_2) \cup \{\neg r(x_1, x_2), r(x_1, [x_2\alpha])\},$$

where x_1 and x_2 are variables of sort W, and α is a variable of sort AF.

For clauses of the form ($\mathcal{C}_{inv}$) we shall also write $C = C[v]$ to emphasise the special role of v as the only variable or constant of sort W that may occur on the right side of r literals in C, if there are any such literals. In that case, $C[v']$ will denote the clause in which v is replaced by v'. We will write $C = C[t]$ to emphasise the term t occurring in C.

Bibliography

[1] G. Aguilera, I. P. de Guzmán, and M. Ojeda. Increasing the efficiency of automated theorem proving. *Journal of Applied Non-Classical Logics*, 5(1):9–29, 1995.

[2] H. Andréka, I. Németi, and I. Sain. Some new landmarks on the roadmap of two dimensional logics. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, Foundations of Computing, Foundations of Computing, pages 163–169. MIT Press, 1994.

[3] H. Andréka, J. van Benthem, and I. Németi. Back and forth between modal logic and classical logic. *Bulletin of the IGPL*, 3(5):685–720, 1995.

[4] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. ILLC scientific publication series for mathematical logic and foundation ML-1996-03, Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands, 1996.

[5] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In H. Boley and M. M. Richter, editors, *Proceedings of the International Workshop on Processing Declarative Knowledge (PDK'91)*, LNAI 567, pages 67–86. Springer, 1991.

[6] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "Making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, 1994.

[7] M. Baaz, C. Fermüller, and A. Leitsch. A non-elementary speed-up in proof length by structural clause form transformation. In *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS'94)*, pages 213–219. IEEE Computer Society Press, 1994.

[8] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[9] L. Bachmair and H. Ganzinger. Ordered chaining calculi for first-order theories of binary relations. Research report MPI-I-95-2-009, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Oct. 1995. To appear in the *Journal of the ACM*.

[10] L. Bachmair and H. Ganzinger. A theory of resolution. Research report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Apr. 1997. To appear in J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*.

[11] L. Bachmair and H. Ganzinger. Equational reasoning in saturation-based theorem proving. In W. Bibel and P. Schmitt, editors, *Automated Deduction: A Basis for Applications, Vol. I*, chapter 11, pages 353–397. Kluwer, 1998.

[12] L. Bachmair and H. Ganzinger. Strict basic superposition. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, LNAI 1421, pages 160–174. Springer, 1998.

[13] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation. *Information and Computation*, 121(2):172–192, 1995.

[14] L. Bachmair, H. Ganzinger, and A. Voronkov. Elimination of equality via transformation with ordering constraints. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, LNAI 1421, pages 175–190. Springer, 1998.

[15] L. Bachmair, H. Ganzinger, and U. Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational Logic and Proof Theory, Third Kurt Gödel Colloquium*, LNCS 713, pages 83–96. Springer, 1993. Earlier version: Technical Report MPI-I-93-204, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Feb. 1993.

[16] D. Basin and H. Ganzinger. Complexity analysis based on ordered resolution. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 456–465. IEEE Computer Society Press, 1996.

[17] D. Basin, S. Matthews, and L. Viganò. Natural deduction for non-classical logics. Research report MPI-I-96-2-006, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1996.

[18] M. P. Bonacina and J. Hsiang. On the modelling of search in theorem proving: Towards a theory of strategy analysis. *Information and Computation*, 147:171–208, 1998.

[19] T. Boy de la Tour. An optimality result for clause form translation. *Journal of Symbolic Computation*, 14:283–301, 1992.

[20] B. Cha and K. Iwama. Performance test of local search algorithms using new types of random CNF formulas. In C. S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 304–311. Morgan Kaufmann, 1995.

[21] C.-L. Chang and R. C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[22] S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[23] M. D'Agostino. Are tableaux an improvement on truth-tables? *Journal of Logic, Language, and Information*, 1:235–252, 1992.

[24] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[25] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[26] H. de Nivelle. Resolution games and non-liftable resolution orderings. In L. Pacholski and J. Tiuryn, editors, *Selected Papers from the 8th Workshop on Computer Science Logic (CSL '94)*, LNCS 933, pages 279–293. Springer, 1995.

[27] H. de Nivelle. *Ordering refinements of resolution*. PhD thesis, Technische Universiteit Delft, The Netherlands, 1996.

[28] H. de Nivelle. Deciding the E^+-class by an a posteriori, liftable order. ILLC scientific publication series for mathematical logic and foundation ML-98-03, Institute for Logic, Language, and Computation, University of Amsterdam, The Netherlands, 1998.

[29] H. de Nivelle. Resolution decided the guarded fragment. ILLC prepublication series for logic, semantics and philosophy of language CT-98-01, Institute for Logic, Language, and Computation, University of Amsterdam, The Netherlands, 1998.

[30] H. de Nivelle. A resolution decision procedure for the guarded fragment. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, LNAI 1421, pages 191–204. Springer, 1998.

[31] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for temporal logics of knowledge. *Journal of Logic and Computaton*, 8(3):345–372, 1998.

[32] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 151–162. Morgan Kaufmann, 1991.

[33] B. Dreben and W. D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.

[34] E. Eder. *The Relative Complexities of First Order Calculi*. Vieweg, 1992.

[35] N. Eisinger and H. J. Ohlbach. Deduction systems based on resolution. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. I, pages 184–271. Oxford University Press, 1993.

[36] N. Eisinger, H. J. Ohlbach, and A. Präcklein. Reduction rules for resolution-based systems. *Artificial Intelligence*, 50(2):141–181, 1991.

[37] D. Fehrer, U. Hustadt, M. Jaeger, A. Nonnengart, H. J. Ohlbach, R. A. Schmidt, C. Weidenbach, and E. Weydert. Description logics for natural language processing. In F. Baader, M. Lenzerini, W. Nutt, and P. F. Patel-Schneider, editors, *International Workshop on Description Logics '94*, Document D-94-10, pages 80–84. DFKI, 1994.

[38] C. Fermüller and A. Leitsch. Hyperresolution and automated model building. *Journal of Logic and Computation*, 6(2):173–230, 1996.

[39] C. Fermüller, A. Leitsch, T. Tammet, and N. Zamov. *Resolution Method for the Decicion Problem*, LNCS 679. Springer, 1993.

[40] C. Fermüller and G. Salzer. Ordered paramodulation and resolution as decision procedures. In A. Voronkov, editor, *Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning (LPAR '93)*, LNAI 698, pages 122–133. Springer, 1993.

[41] C. G. Fermüller and A. Leitsch. Decision procedures and model building in equational clause logic. *Logic Journal of the IGPL*, 6(1), 1998.

[42] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*, Synthese Library, Studies in Epistemology, Logic, Methodology, and Philosophy of Science 169. D. Reidel, 1983.

[43] J. Franco and M. Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 5:77–87, 1983.

[44] D. Gabbay and H. J. Ohlbach. Quantifier elimination in second-order predicate logic. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 425–438. Morgan Kaufmann, 1992.

[45] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*. IEEE Computer Society Press, 1999.

[46] H. Ganzinger, U. Hustadt, C. Meyer, and R. A. Schmidt. A resolution-based decision procedure for extensions of K4. In M. Zakharyaschev, K. Segerberg, M. de Rijke, and H. Wansing, editors, *Advances in Modal Logic, Volume 2*, pages 243–263. CSLI Publications, Stanford, 2000.

[47] H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 24–34. IEEE Computer Society Press, 1999.

[48] H. Ganzinger, C. Meyer, and C. Weidenbach. Soft typing for ordered resolution. In W. McCune, editor, *Proceedings of the 14th International Conference on Automated Deduction (CADE-14)*, LNAI 1249, pages 321–335. Springer, 1997.

[49] I. P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 28–33. AAAI Press, 1993.

[50] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tachella. More evaluation of decision procedures for modal logics. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 626–635. Morgan Kaufmann, 1998.

[51] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures: Case study of modal K. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 583–597. Springer, 1996.

[52] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for alc. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 304–314. Morgan Kaufmann, 1996.

[53] A. Goldberg. Average case complexity of the satisfiability problem. In *Proceedings of the 4th Workshop on Automated Deduction*, pages 1–6, 1979.

[54] A. Goldberg, P. Purdom, and C. Brown. Average time analyses of simplified Davis-Putnam procedures. *Information Processing Letters*, 15(2):72–75, 1982.

[55] A. Goldberg, P. Purdom, and C. Brown. Corrigendum: "Average time analyses of simplified Davis-Putnam procedures", Information Processing Letters 15(2):72–75. *Information Processing Letters*, 16(4):213–213, 1983.

[56] R. I. Goldblatt. Metamathematics of modal logic. *Reports on Mathemtical logic*, 6:47–78, 1976.

[57] R. I. Goldblatt. Metamathematics of modal logic. *Reports on Mathemtical logic*, 7:21–52, 1976.

[58] R. Goré. Cut-free sequent and tableu systems for propositional diodorean modal logics. Technical Report UMCS-93-8-3, Department of Computer Science, University of Manchester, England, 1993.

[59] R. Goré. Tableau methods for modal and temporal logics. Technical Report TR-ARP-15-95, Automated Reasoning Project, Australian National University, Nov. 1995.

[60] E. Grädel. On the restraining power of guards. Manuscript. Submitted to the *Journal of Symbolic Logic*, 1998.

[61] A. Heuerding. *Sequent Calculi for Proof Search in Some Modal Logics.* PhD thesis, Universität Bern, Switzerland, 1996.

[62] A. Heuerding, G. Jäger, S. Schwendimann, and M. Seyfried. The Logics Workbench LWB: A snapshot. *Euromath Bulletin*, 2(1):177–186, 1996.

[63] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. Technical Report IAM-96-015, University of Bern, Switzerland, 1996.

[64] A. Heuerding, M. Seyfried, and H. Zimmermann. Efficient loop-check for backward proof search in some non-classical propositional logics. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Proceedings of the 5th International Conference on Theorem Proving with Analytic Tableaux and Related Methods (TABLEAUX '96)*, LNAI 1071, pages 210–225. Springer, 1996.

[65] B. Hollunder. *Algorithmic Foundations of Terminological Knowledge Representation Systems.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1994.

[66] J. N. Hooker. Resolution vs. cutting place solution of inference problems: Some computational experience. *Operations Research Letters*, 7(1):1–7, 1988.

[67] J. N. Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42:201–212, 1994.

[68] J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1:33–42, 1996.

[69] I. Horrocks. Optimisation techniques for expressive description logics. Technical Report Series UMCS-97-2-1, Department of Computer Science, University of Manchester, England, Feb. 1997.

[70] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, England, 1997.

[71] 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, 1998.

[72] I. R. Horrocks. Using an expressive description logic: FaCT or Fiction. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647. Morgan Kaufmann, 1998.

[73] J. Hsiang and M. Rusinowith. Proving refutation completeness of theorem-proving strategies: The transfinite semantic tree method. *Journal of the ACM*, 38(3):559–587, 1991.

[74] U. Hustadt. Automated support for the development of non-classical logics. In H.-J. Bürckert and W. Nutt, editors, *Modeling Epistemic Propositions: Workshop during the 17th German Conference on Artificial Intelligence (KI'93)*, Document D-93-25. DFKI, 1993.

[75] U. Hustadt. Do we need the closed-world assumption in knowledge representation. In F. Baader, M. Buchheit, M. A. Jeusfeld, and W. Nutt, editors, *Working Notes of the KI'94 Workshop: Reasoning about Structured Objects: Knowledge Representation Meets Databases (KRDB'94)*, Document D-94-11, pages 24–26. DFKI, 1994.

[76] U. Hustadt. A multi-modal logic for stereotyping. In *Proceedings of the Fourth International Conference on User Modeling UM94*, pages 87–92. The MITRE Corporation, 1994.

[77] U. Hustadt. Introducing epistemic operators into a description logic. In A. Laux and H. Wansing, editors, *Knowledge and Belief in Philosophy and Artificial Intelligence*, chapter 3, pages 65–85. Akademie Verlag, 1995.

[78] U. Hustadt and A. Nonnengart. Modalities in knowledge representation. In C. Rowles, H. Liu, and N. Foo, editors, *Proceedings of the 6th Australian Joint Conference on Artificial Intelligence (AI'93)*, pages 249–254. World Scientific, 1993.

[79] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. Research report MPI-I-97-2-003, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1997.

[80] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. In M. E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Vol. 1, pages 202–207. Morgan Kaufmann, 1997.

[81] U. Hustadt and R. A. Schmidt. Simplification and backjumping in modal tableau. In H. de Swart, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, LNAI 1397, pages 187–201. Springer, 1998.

[82] U. Hustadt and R. A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999.

[83] U. Hustadt and R. A. Schmidt. Maslov's class K revisited. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, LNAI 1632, pages 172–186. Springer, 1999.

[84] U. Hustadt and R. A. Schmidt. On the relation of resolution and tableaux proof systems for description logics. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 110–115. Morgan Kaufmann, 1999.

[85] U. Hustadt, R. A. Schmidt, and C. Weidenbach. Optimised functional translation and resolution. In H. de Swart, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, LNAI 1397, pages 36–37. Springer, 1998.

[86] W. H. Joyner Jr. Resolution strategies as decision procedures. *Journal of the ACM*, 23(3):398–417, 1976.

[87] B. Kallick. A decision procedure based on the resolution method. In A. J. H. Morrell, editor, *Information Processing 68: Proceedings of IFIP Congress 1968. Volume 1: Mathematics, Software*, pages 269–275. North-Holland, 1969.

[88] C. Kirchner. A new unification method: A generalization of martelli-montanari's algorithm. In R. E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction (CADE-7)*, LNCS 170, pages 224–247. Springer, 1984.

[89] S. A. Kripke. Semantical considerations on modal logic. *Acta Philisophica Fennica*, 16:83–94, 1963.

[90] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computation*, 6(3):467–480, 1977.

[91] A. Leitsch. *The Resolution Calculus*. Springer, 1997.

[92] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation. *Computational Intelligence*, 3:78–93, 1987.

[93] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994.

[94] M. Manzano. Introduction to many-sorted logic. In K. Meinke and J. V. Tucker, editors, *Many-Sorted Logic and its Applications*, chapter 1, pages 3–86. John Wiley & Sons, 1993.

[95] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–292, 1982.

[96] S. J. Maslov. An inverse method for establishing deducibility in the classical predicate calculus. *Dokl.Akad.Nauk SSSR*, 159:1420–1424, 1964.

[97] S. J. Maslov. The inverse method for establishing deducibility for logical calculi. In V. P. Orevkov, editor, *The Calculi of Symbolic Logic I: Proceedings of the Steklov Institute of Mathematics edited by I.G. Petrovskiĭ and S. M. Nikol'skiĭ, number 98 (1968)*, pages 25–96. American Mathematical Society, 1971.

[98] F. Massacci. Strongly analytic tableaux for normal modal logics. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction (CADE-12)*, LNAI 814, pages 723–737. Springer, 1994.

[99] F. Massacci. Simplification: A general constraint propagation technique for propositional and modal tableaux. In H. de Swart, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, LNAI 1397, pages 217–231. Springer, 1998.

[100] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In W. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465. MIT Press, 1992.

[101] A. Nonnengart. *A Resolution-Based Calculus For Temporal Logics*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1995.

[102] A. Nonnengart. Resolution-based calculi for modal and temporal logics. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-96)*, LNAI 1104, pages 598–612. Springer, July 30–Aug. 3 1996.

[103] A. Nonnengart. Strong skolemization. Research Report MPI-I-96-2-010, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1996.

[104] A. Nonnengart, G. Rock, and C. Weidenbach. On generating small clause normal forms. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, LNAI 1421, pages 397–411. Springer, 1998.

[105] H. J. Ohlbach. Semantics-based translation methods for modal logics. *Journal of Logic and Computation*, 1(5):691–746, 1991.

[106] H. J. Ohlbach. Combining Hilbert style and semantic reasoning in a resolution framework. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction (CADE-15)*, LNAI 1421, pages 205–219. Springer, 1998.

[107] H. J. Ohlbach and R. A. Schmidt. Functional translation and second-order frame properties of modal logics. *Journal of Logic and Computation*, 7(5):581–603, 1997. Earlier version: Technical Report MPI-I-95-2-002, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Jan. 1995.

[108] H. J. Ohlbach, R. A. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. In H. Wansing, editor, *Proof Theory of Modal Logic*, Applied Logic Series 2, pages 253–291. Kluwer, 1996. Also available as Technical Report MPI-I-95-2-008, Max-Planck-Institut für Informatik, Saarbrücken, Germany (May 1995).

[109] H. J. Ohlbach and C. Weidenbach. A note on assumptions about Skolem functions. *Journal of Automated Reasoning*, 15(2):267–275, 1995.

[110] M. Paramasivam and D. A. Plaisted. Automated deduction techniques for classification in description logic systems. *Journal of Automated Reasoning*, 20(3):337–364, 1998.

[111] P. F. Patel-Schneider. *Decidable, Logic-Based Knowledge Representation*. PhD thesis, University of Toronto, Ontario, Canada, 1987.

[112] G. E. Peterson. A technique for establishing completeness results in theorem proving with equaility. *SIAM Journal of Computation*, 12(1):82–100, 1983.

[113] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2:293–304, 1986.

[114] D. A. Plaisted and Y. Zhu. *The Efficiency of Theorem Proving Strategies*. Vieweg, 1997.

[115] W. C. Purdy. Fluted formluas and the limits of decidability. *Journal of Symbolic Logic*, 61(2):608–620, 1996.

[116] V. Rantala. Constituents. In R. J. Bogdan, editor, *Jaako Hintikka*, pages 43–76. D. Reidel, 1987.

[117] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965. Reprinted in [125, pp. 397–415].

[118] K. Schild. A correspondence theory for terminological logics: Preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 466–471. Morgan Kaufmann, 1991.

[119] K. Schild. Undecidability of subsumption in $\mathcal{U}$. KIT-Report 67, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, 1988.

[120] R. A. Schmidt. *Optimised Modal Translation and Resolution*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1997.

[121] R. A. Schmidt. Decidability by resolution for propositional modal logics. *Journal of Automated Reasoning*, 22(4):379–396, 1999.

[122] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, pages 421–431. Morgan Kaufmann, 1989.

[123] M. Schmidt-Schauß and G. Smolka. Attributive concept description with complements. *Artifical Intelligence*, 48:1–26, 1991.

[124] R. Schuler and T. Yamakami. Structural average case complexity. Ulmer Informatik-Bericht 95-04, Universität Ulm, Germany, 1995.

[125] J. Siekmann and G. Wrightson. *Automation of Reasoning 1: Classical Papers on Computational Logic; 1957-1966*. Springer, 1983.

[126] J. Steinbach. Extensions and comparisons of simplification orderings. In N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications (RTA-89)*, LNCS 355, pages 434–448. Springer, 1989.

[127] T. Tammet. Using resolution for extending KL-ONE-type languages. In N. Pissinou, A. Silberschatz, E. K. Park, and K. Makki, editors, *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95)*. ACM Press, 1995.

[128] T. Tammet. The resolution program, able to decide some solvable classes. In P. Martin-Löf and G. Mints, editors, *Proceedings of the International Conference on Computer Logic (COLOG-88)*, LNCS 417, pages 300–312. Springer, 1990.

[129] T. Tammet. *Resolution methods for decision problems and finite-model building*. PhD thesis, Chalmers University of Technology, University of Göteborg, Sweden, 1991.

[130] T. Tammet. Using resolution for deciding solvable classes and building finite models. In J. Bārzdiņš and D. Bjørner, editors, *Baltic Computer Science*, LNCS 502, pages 33–64. Springer, 1991.

[131] A. Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[132] C. Weidenbach. Minimal resolution. Technical report MPI-I-94-227, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1994.

[133] C. Weidenbach. *Computational Aspects of a First-Order Logic with Sorts*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[134] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER version 0.42. In M. McRobbie and J. Slaney, editors, *Proceedings of the 13th International Conference on Automated Deduction (CADE-13)*, LNAI 1104, pages 141–145. Springer, 1996.

List of figures

6.1 Sample derivation of KSAT . . . 123
6.2 Sample derivation of $\mathcal{KRIS}$ . . . 126
6.3 Axioms and rules of the Logics Workbench . . . 127
6.4 Sample derivation of the Logics Workbench . . . 128
6.5 The quality of the test sets . . . 132
6.6 The performance of $\mathcal{KRIS}$ and KSAT for **PS0** and **PS1** . . . 133
6.7 The performance of KSAT with and without sorting . . . 134
6.8 The performance of KSAT and $\mathcal{KRIS}$* for **PS0** and **PS1** . . . 134
6.9 Impact of $\vee$`E` versus $\vee$`E'` . . . 136
6.10 The performance of KSAT and $\mathcal{KRIS}$* . . . 136
6.11 The performance of KSAT and TA* . . . 137
6.12 The performance of the Logics Workbench . . . 140
6.13 The percentile graphs on **PS0** . . . 142
6.14 Varying the parameter N . . . 143
6.15 Varying the parameter M . . . 144
6.16 Varying the parameter D . . . 144
6.17 The influence of the parameter D on the formula size . . . 146
6.18 The quality of the test sets . . . 147
6.19 The median performances . . . 147
6.20 The percentile graphs on **PS12** . . . 148
6.21 The percentile graphs on **PS13** . . . 149

List of tables

4.1 Clausal form of formulas in definitional form 74
4.2 Clausal form role composition and binding forming operators 83
4.3 Clausal form of formulas in definitional form 87
4.4 Clausal form role composition and binding forming operators 92

5.1 Relational operations in extended modal logics 96
5.2 Axiom schemata and relational frame properties 98
5.3 Axiom schemata and functional frame properties 99
5.4 Axiom schemata and semi-functional frame properties 101
5.5 Clausal form of formulae in definitional form 104

6.1 Parameter settings 131
6.2 Simplification rules for modal formulae 132

Index of subjects

A-ordering . . . **10**
ABox . . . **68**, 70, 74, 87, 89–91
Accessibility
 function . . . **97**
 relation . . . 95, **97**
Ackermann class
 initially extended ∼ . . . 1, **16**, 65
 with equality . . . 153
AE predicate logic . . . 1
$\mathcal{ALB}$ . . . 4, **70**, 71–79
$\mathcal{ALB}_\mathsf{D}$ . . . 4, **84**, 84–92
$\mathcal{ALC}$ . . . 4, 67, **70**
Antisymmetric relation . . . **9**
Application
 function . . . **97**
 of a substitution . . . **9**
Arity
 of function symbols . . . **7**
 of predicate symbols . . . **7**
Assertional sentence . . . **68**, 70, 85, 86, 89, 91
Asymmetric relation . . . **9**
Atom . . . **7**
 covering ∼ . . . **20**
 modal ∼ . . . **96**
 ordering . . . **10**
 weakly covering ∼ . . . **20**
Atomic
 concept . . . **67**
 role . . . **67**
Axiom schema
 4 . . . 4, 96, **98**, 98–116
 5 . . . 4, 96, **98**, 98–116
 B . . . 95, 96, **98**, 98–105
 D . . . 4, 95, 96, **98**, 98–114
 G . . . **98**, 99
 M . . . **98**, 99
 T . . . 4, 95, 96, **98**, 98–116

Basic concept . . . **71**
Basic path logic . . . 67, 99
Basic role . . . **71**
Bernays-Schönfinkel class . . . 2, 16, 67
Binding . . . **68**, 68, 82, 84, 91, 92
Boolean algebra . . . 70
Bottom
 concept . . . **68**, 70
 role . . . **68**
Branch pruning . . . 135, 139, 140, 150

Cardinality . . . 8
CDV
 clause . . . **53**
 literal . . . **53**
CDV-free
 clause . . . **53**
 literal . . . **53**
Class
 $\mathcal{ALB}$ . . . 4, **70**, 71–79
 $\mathcal{ALB}_\mathsf{D}$ . . . 4, **84**, 84–92
 $\mathcal{ALC}$ . . . **70**
 basic path logic . . . 104
 Bernays-Schönfinkel ∼ . . . 2, 16, 67
 $\overline{\text{DK}}$ . . . 3, **59**, 59–66, 153
 $\overline{\text{DKC}}$ . . . **59**, 59–66
 DL-clauses . . . 4, **73**, 73–79, 92, 153
 $\mathbf{E}_1$ . . . 2, **37**
 $\mathbf{E}^+$ . . . 2, 3, **20**, 19–37, 65, 153, **163**
 fluted DL-clauses . . . 4, **80**, 80–82
 fluted logic . . . **17**, 37, 153
 guarded fragment 2, **16**, 67, 92, 95, 153
 initially extended Ackermann ∼ . 1, **16**, 65
 initially extended Gödel ∼ . . . 1, **16**, 65
 initially extended Skolem ∼ . . 1, **16**, 65
 K . . . 1–3, 39, 62
 k-regular clauses . . . **43**, 43–66

$\overline{\mathrm{K}}$ 3, 4, 39, **40**, 43, 57, 95, 153
$\overline{\mathrm{K}}_Z$ **62**
$\overline{\mathrm{KC}}$ **42**, 42–43
$\mathsf{K}\Sigma$ 95, **96**, 96–116
loosely guarded fragment **16**, 92
monadic ∼ 1, **16**, 65
$\mathbf{M}^+$ 2, **17**
$\mathcal{OCCI}$ 2, **17**
One-Free 2, **17**, 65, 67, 75, 92
propositional modal logic 4, 95, **96**, 96–116
$\mathcal{PVD}$ 2, **17**
$\mathcal{PVD}_g^=$ 153
$\mathcal{PVD}_+$ **17**, 153
quasi-regular clauses **43**, 43–66
SF-clauses 153
SF-regular clauses **106**, 106–113
small SF-clauses 4, **106**, 106–109
solvable ∼ **15**
$\mathcal{S}^+$ **17**, 65
two-variable fragment 2, **16**, 65, 95
$\mathcal{U}^-$ **84**
$\mathcal{U}_\mathsf{D}^-$ **92**
variable uniform clauses **20**, 19–37, 65, **163**
Classification of a TBox 70
Clause **8**
depth distorted ∼ **25**, 25–37
depth undistorted ∼ **25**, 28, 30
CDV ∼ **53**
CDV-free ∼ **53**
condensed ∼ **11**
definition 26, 56
depth **8**
DL-∼ 72, **73**, 73–79, 92, 153
fluted DL-∼ **80**, 80–82
Horn ∼ **8**
indecomposable ∼ **11**
inhabited ∼ **60**, 61
k-regular ∼ **43**, 43–66
quasi-regular ∼ **43**, 43–66
redundant ∼ **12**
regular ∼ **43**, 43–66
SF-∼ 4, **109**, 109–115, 153
SF-regular ∼ **106**, 106–113
small SF-∼ 4, **106**, 106–109
split component **11**
strongly CDV-free ∼ **53**, 53–59
strongly k-regular ∼ **60**, 60–61
variable uniform ∼ **20**, 20–37, 163
Clean literal **73**
Closure
role ∼ **68**
Codomain of a substitution **9**
Coherent concept **70**
Complement
concept ∼ **68**, 70
literal ∼ **8**
role ∼ **68**, 70, 84, 91
Completeness
of a modal logic **97**
Complexity measure 102
Composition
of substitutions **9**
role ∼ **68**, 70, 83, 84, 91, 92
Compound term **7**
Concept **67**, 68
atomic ∼ **67**
basic ∼ **71**
bottom ∼ **68**, 70
coherent ∼ **70**
complement **68**, 70
defined ∼ **68**, 89
definition **68**, 85, 89, 92
existential restriction **68**, 70, 89
incoherent ∼ **70**
intersection **68**, 70, 89
negation normal form **71**, 84–86
number restriction **68**
qualified number restriction **68**
role value map **68**, 70, 84, 91
satisfiable ∼ **70**
specialisation **68**, 89
subsumption **70**, 70
symbol **67**
term **67**
top ∼ **68**, 70
union **68**, 70, 89
universal restriction **68**, 70, 89
unsatisfiable ∼ **70**
Condensed clause **11**
Condensing **11**, 67, 104, 108

Confluence . . . 98, 99
Conjunct . . . **8**
Conjunctive normal form . . . **8**
Constant . . . **7**
Converse
  role ∼ . . . **68**, 70, 80, 82
Correspondence theory . . . 98
Covering
  atom . . . **20**
  literal . . . **20**
  term . . . **20**

Dead-end . . . **97**
Decision procedure . . . 1
Deep literal . . . **43**
Definition
  of a concept . . . **68**, 85, 89, 92
  of a predicate . . . 86
  of a role . . . **68**, 85, 86
Definitional
  form . . . **14**, 72, 86
Depth
  of a clause . . . **8**
  of a literal . . . **8**
  of a term . . . **7**
Derivation . . . 14
  fair theorem proving ∼ . . . **13**
  theorem proving ∼ . . . **12**
Description logic . . . 65, 67
Descriptive knowledge base . . . **68**, 84–91
Determination
  of a modal logic . . . **97**
Disjunct . . . **8**
$\overline{\text{DK}}$ . . . 3, **59**, 59–66, 153
$\overline{\text{DKC}}$ . . . **59**, 66
$\overline{\text{K}}$ . . . 59
DL-clause . . . 4, 72, **73**, 73–79, 92, 153
  fluted ∼ . . . 4, **80**, 80–82
DL-literal . . . **73**, 79
  fluted ∼ . . . **80**, 80
Domain
  of a substitution . . . **9**
  restriction . . . **68**, 70
Dominating
  literal . . . **42**
  set of terms . . . **42**
  term . . . **42**

Element
  maximal ∼ . . . **10**
  strictly maximal ∼ . . . **10**
Elementary class . . . **98**
Embracing
  literal . . . **72**, 83, 91
  term . . . **72**, 73, 75–77, 91
$\mathbf{E}_1$ . . . 2, **37**
$\mathbf{E}^+$ . . . 2, 3, **20**, 19–37, 65, 153, **163**
Equality . . . 16
Equivalence
  of concepts . . . **70**
  relation . . . **9**
Essentially second-order modal logic . . . **98**
Euclideanness . . . 41, 98, 99, 103, 105
Existential restriction . . . **68**, 70, 89
Expanded TBox . . . 89
Expression
  identical modulo (variable) renaming . **9**
  instance . . . **9**
  most general unifier . . . **9**
  terminological ∼ . . . **68**
  unifier . . . **9**
  variant ∼ . . . **9**

Factoring . . . 108
Finite model property . . . 2, 95
First-order definable modal logic . . . **98**
Fixed universally quantified variables . . . **40**
Flat literal . . . **73**
FLOTTER . . . 129
Fluted
  DL-clause . . . **80**, 81–82
  DL-literal . . . **80**
  logic . . . **17**, 37
Fluted logic . . . 153
Formula
  atomic ∼ . . . **7**
  conjunctive ∼ . . . **8**
  conjunctive normal form . . . **8**
  disjunctive ∼ . . . **8**
  first-order ∼ . . . **8**
  modal . . . **96**
  negation normal form . . . **8**
  of grade k . . . **40**
  prenex form . . . **8**
  rectified ∼ . . . **8**

Frame
- functional ∼ **97**
- relational **97**

Frame property
- confluence **98**, 99
- euclideanness **98**, 98, 99, 103, 105
- functional **98**
- irreflexivity **98**, 99, 102, 104
- McKinsey's axiom **98**, 99
- reflexivity 95, **98**, 99, 109
- relational **98**
- semi-functional **101**
- seriality 95, **98**, 99, 109
- symmetry 95, **98**, 98, 99
- transitivity 95, **98**, 98, 99, 103, 105, 109
- universality **98**, 102, 104
- weak density **98**, 99

Function symbol **7**
- lexicographic status **10**
- multiset status **10**
- precedence **10**
- status **10**

Functional
- frame **97**
- model **97**
- semantics **97**, 99

Gödel class
- initially extended 1
- initially extended ∼ **16**, 65

GF^2 . . . . **95**
- monadic 95

Grade
- of a formula **40**

Ground term **7**

Guarded fragment 2, **16**, 67, 92, 95
- loosely ∼ **16**, 92
- with equality 153

Identity
- modulo (variable) renaming **9**
- role **68**, 70, 83
- substitution **9**

Incoherent concept **70**

Inhabited clause 60, 61

Instance **9**
- checking 70

Interpretation
- terminological **68**

Intersection
- concept ∼ **68**, 70, 89
- role ∼ **68**, 70

Inverse method 1, 62

Irreflexivity 98, 99, 102, 104

K 1–3, 39, 62

$\overline{\mathrm{K}}$ 3, 4, 39, **40**, 43, 57, 95, 153

$\overline{\mathrm{K}}_Z$ **62**

K4 . . . . 101, 103, 116

K45 101

K5 101

$\overline{\mathrm{KC}}$ **42**, 42–43

KD 101, 104

KD4 . . . . 103, 104

KD45 101

KD5 101

KDB 101

Knowledge base **68**
- descriptive ∼ **68**, 84–91
- entailment 69
- model **69**
- satisfiability **69**

Knowledge representation 2

Kripke Semantics **97**

$\mathcal{KRIS}$ 5, 121, **123**, 128, 134, 140–150

Ksat 5, **121**, 121, 133, 140–150

Ksat0 . . . . **121**, 122

KT 104

Literal **7**
- depth dominated ∼ **25**, 27
- CDV ∼ **53**
- CDV-free ∼ **53**
- clean ∼ **73**
- complement **8**
- covering ∼ **20**
- deep ∼ **43**
- depth **8**
- DL-∼ **73**, 79
- dominating ∼ **42**
- embracing ∼ **72**, 83, 91
- flat ∼ **73**
- fluted DL-∼ **80**, 80
- negative ∼ **8**

norm **8**
ordering **10**
positive ∼ **7**
regular ∼ **43**, 73
selected ∼ **11**
shallow ∼ **43**
similar ∼s **43**, 44–48, 54, 55, 63, 65
singular ∼ **43**
$\succ_d$-distorting ∼ **25**, 26–28, 33–37
weakly covering ∼ **20**, 20, 21, 28–30
Logics Workbench 5, 121, **127**, 127–130, 134, 135, 139–141, 143, 146–150
Loop-checking 103
Loosely guarded fragment **16**, 92

Maximal element **10**
McKinsey's axiom 98, 99
Modal atom **96**
Modal formula **96**
Modal logic 65, 67
essentially second-order **98**
extended ∼ 95
first-order definable **98**
K 95, **96**, 96–116
$\mathsf{K}_{(m)}$ 2, 117, **120**, 127, 130
K4 . . . . 101, 103, 116
K45 101
K5 101
KD 101, 104
KD4 . . . . 103, 104
KD45 101
KD5 101
KDB 101
KΣ **96**, 96–116
KT 104
propositional ∼ 4, 95, **96**, 96–116
S4 . . . . **98**, 101, 103, 104
S5 . . . . **98**, 101, 104, 116, 153
Model
functional ∼ **97**
of a knowledge base **69**
relational ∼ **97**
terminological ∼ **69**
Monadic
class 1, **16**, 65
GF^2 95
Most general unifier **9**

$\mathbf{M}^+$ 2, **17**
Multiset **8**

Negation normal form
of a concept **71**, 84–86
of a role **71**, 84–86
Negative literal **8**
Norm
of a literal **8**
Normalisation 104
Number restriction **68**, 83

Object symbol **67**
$\mathcal{OCCI}$ 2, **17**
One-Free 2, **17**, 65, 67, 75, 92
Optimised functional translation . 5, 67, **99**, 99, **100**, 129
decidability results 103–104
soundness and completeness 100
Ordered chaining calculus 103, 116
Ordering
a posteriori application **12**
a priori application **12**
A-∼ **10**
atom ∼ **10**
liftable ∼ **10**
literal ∼ **10**
partial ∼ **10**
quasi-∼ **10**
recursive path ∼ **10**
refinement **10**
stable under contexts **9**
stable under substitutions **10**
strict partial ∼ **10**
strict subterm ∼ **76**
$\succ_{\mathcal{COV}}$ 72, **76**, 76–84, 102–112
$\succ_d$ **23**, 25, 27, 36, 37
$\succ'_d$ **28**
$\succ_{\mathcal{ML}}$ **110**, 110–115
$\succ_n$ **23**, 24, 36
$\succ_{\mathsf{P}}$ **28**
$\succ_{\mathcal{TAB}}$ **86**, 89, 92, 102
$\succ_v$ **23**, 24, 25, 36
$\succ'_v$ **23**, 24
total ∼ **10**

φ-prefix 40

terminal ∼ 40
Partial ordering **10**
Path logic 104
basic ∼ 67
Polarity 70
negative ∼ **8**
positive ∼ **8**
zero ∼ **8**
Position **9**
Positive literal **7**
Predicate symbol **7**
Prefix stability 103
Prenex form **8**
Proof system 14
Propositional
logic 1
variable **7**
$\mathcal{PVD}$ 2, **17**
$\mathcal{PVD}_g^=$ 153
$\mathcal{PVD}_+$ **17**, 153

Qualified number restriction **68**
Quasi-ordering **10**
strict part **10**
well-founded ∼ **10**
Quasi-regular clause **43**, 43–66

Range restriction **68**, 70
Realization 70
Rectified formula **8**
Recursive path ordering **10**
Redundancy
elimination 3, 12, 25, 153
saturated up to ∼ **12**
with respect to a clause set **12**
Refinement
of an ordering **10**
Reflexivity 95, 98, 99, 109
Refutation **12**
Regular
literal **43**
set of terms **43**
term **43**
Regular literal 73
Relation
antisymmetric ∼ **9**
asymmetric ∼ **9**
equivalence ∼ **9**
liftable ∼ **10**
reflexive ∼ **9**
stable under contexts **9**
stable under substitutions **10**
symmetric ∼ **9**
transitive ∼ **9**
Relational
frame properties **98**
model **97**
model based on a frame **97**
semantics **97**, 99
translation **99**, 99
Relational operation 95
composition 95
intersection 95
transitive closure 95
union 95
Relational translation 4
decidability results 101–103
Renaming
dynamic ∼ 27, 37, 154
sign ∼ **17**
Resolution 1, 67
Restriction of a substitution **9**
Retrieval 70
Rewrite relation **10**
Role **67**, 68
atomic ∼ **67**
basic ∼ **71**
bottom ∼ **68**
closure **68**
complement **68**, 70, 84, 91
composition **68**, 70, 83, 84, 91, 92
converse **68**, 70, 80, 82
defined ∼ **68**
definition **68**, 85, 86
domain restriction **68**, 70
identity ∼ **68**, 70, 83
intersection **68**, 70
negation normal form **71**, 84–86
range restriction **68**, 70
specialisation **68**
symbol **67**
term **67**
top ∼ **68**, 70, 84, 91

union . . . **68**, 70
value map . . . **68**, 70, 84, 91

S4 . . . 101, 103, 104
S5 . . . **98**, 101, 104, 116, 153
Satisfiability
of a concept . . . **70**, 70
of a knowledge base . . . **69**
of a modal formula . . . **97**
of a terminological sentences . . . **69**
Schema . . . **8**
Scientific benchmarking . . . 5
Search space . . . 15
Selection function . . . **11**, 11
$S_{\mathcal{KC}}$ . . . **59**
$S_{\mathcal{ML}}$ . . . **110**, 110, 113–115
$S_{\mathcal{TAB}}$ . . . **86**, 88–92, 102–104, 153
S_v . . . **35**, 36
Semantics
functional ∼ . . . **97**, 98, 99
Kripke ∼ . . . **97**
relational ∼ . . . **97**, 99
Semantics-based translation . . . 99
Semi-functional translation . 4, 99, **100**, 100
decidability results . . . 104–116
soundness and completeness . . . 101
Sentence . . . **68**
assertional ∼ . . . **68**, 70, 85, 86, 89, 91
terminological ∼ . **68**, 68, 70, 72, 86, 89
Seriality . . . 95, 98, 99, 109
SF-clause . . . 4, **109**, 109–115, 153
small ∼ . . . 4, **106**, 106–109
SF-regular
clause . . . **106**, 106–113
term . . . **105**
SF-signature . . . **105**, 105, 106
Shallow literal . . . **43**
Sign renaming . . . **17**
Signature . . . **7**
of a terminological logic . . . **67**
Similar
literals . . . **43**, 44–48, 54, 55, 63, 65
terms . . . **42**
Simplification . . . 27, 124–142, 149, 150
of $\mathcal{ALB}$ expressions . . . **71**
Simulation . . . 3, 19, 35, 36
polynomial ∼ . . . **15**, 153

step-wise ∼ . . . **15**
Singular literal . . . **43**
Skolem class
initially extended ∼ . . . 1, **16**, 65
Skolem function
k-originated ∼ . . . **59**
Skolemisation
outer ∼ . . . 14, 42
strong ∼ . . . 14, 45
Solvable class . . . **15**
Soundness
of a modal logic . . . **97**
SPASS . . . 5, 129, 130, 139, 140, 150
Specialisation
concept ∼ . . . **68**, 89
role ∼ . . . **68**
$\mathcal{S}^+$ . . . **17**, 65
Strict
subclause . . . **8**
subterm . . . **7**
Strict part
of a quasi-ordering . . . **10**
Strict partial ordering . . . **10**
total ∼ . . . **10**
Strict subterm ordering . . . **76**
Strictly maximal element . . . **10**
Strongly k-regular clause . . . **60**, 60–61
Subclause . . . **8**
strict . . . **8**
Substitution . . . **9**
application . . . **9**
codomain . . . **9**
composition . . . **9**
domain . . . **9**
idempotent ∼ . . . **9**
identity ∼ . . . **9**
restriction . . . **9**
variable renaming . . . **9**
Subsumption
of concepts . . . **70**, 70
Subterm . . . **7**
strict ∼ . . . **7**
Symmetry . . . 95, 98, 99

Tableaux calculus
for modal logics . . . 102, 103, 116
for terminological logics . . . 84, 89–91

prefix ∼ . . . 102
TBox . . . **68**, 68, 70, 74, 87, 89, 90
expanded ∼ . . . 89
Term . . . **7**
compound ∼ . . . **7**
congruent ∼ . . . **17**
covering ∼ . . . **20**
depth . . . **7**
dominating ∼ . . . **42**
embracing ∼ . . . **72**, 73, 75–77, 91
ground ∼ . . . **7**
interpretation . . . 69
regular ∼ . . . **43**
SF-regular ∼ . . . **105**
similar ∼s . . . **42**
subterm . . . **7**
terminological ∼ . . . **68**
weakly covering ∼ . . . **20**, 21
Terminal φ-prefix . . . 40
Terminological
cycle . . . **68**
expression . . . **68**
interpretation . . . **68**
model . . . **69**
sentence . . . **68**, 68, 70, 72, 86, 89
term . . . **68**
Terminological inference services
classification of a TBox . . . **70**
equivalence of concepts . . . **70**
instance checking . . . **70**
realization . . . **70**
retrieval . . . **70**
satisfiability of a concept . . . **70**
subsumption . . . **70**
Terminological sentence
entailment . . . 69
Theorem
of the modal logic $\mathsf{K}\Sigma$ . . . **97**
Theorem proving derivation . . . **12**
fair ∼ . . . **13**
Theory resolution . . . 104
Top
concept . . . **68**, 70
role . . . **68**, 70, 84, 91
Transitivity . **9**, 41, 92, 95, 98, 99, 103, 105, 109
Translation
of $\mathcal{ALB}$ expressions . . . 71
of $\mathcal{ALB}$ sentences . . . 71
optimised functional ∼ . . . 5, 67, **99**, 99, **100**, 100, 103–104, 129
relational ∼ . . . 4, **99**, 99, 101–103
semantics-based ∼ . . . 99
semi-functional ∼ . 4, 99, **100**, 100, 101, 104–116
Tree model property . . . 2, 95
Two-variable fragment . . . 2, **16**, 65, 95

$\mathcal{U}$ . . . **67**, 70, 82, 84
$\mathcal{U}^-$ . . . **84**
$\mathcal{U}^-_{\mathsf{D}}$ . . . **92**
Unification . . . **9**
Unifier . . . **9**
most general . . . **9**
Union
concept ∼ . . . **68**, 70, 89
role ∼ . . . **68**, 70
Unique name assumption . . . 68, 72, 83
Universal
restriction . . . **68**, 70, 89
terminological logic . . . **67**, 70, 82, 84
Universality . . . 98, 102, 104
Unsatisfiable concept . . . **70**

Validity
in a frame . . . **97**
in a functional model . . . **97**
in a relational model . . . **97**
Valuation . . . **97**
Variable . . . **7**
fixed universally quantified ∼s . . . **40**
renaming . . . **9**
uniform clause . . . **20**, 20–37, 163
Variant expression . . . **9**

Weak density . . . 98, 99
Weakly covering
atom . . . **20**
literal . . . **20**, 20, 21, 28–30
term . . . **20**, 21
Well-founded
quasi-ordering . . . **10**
World . . . **97**

Index of symbols

$\perp$ . . . **68**, 71
$\top$ . . . **68**, 71
$\sqcap$ . . . **68**
$\sqcup$ . . . **68**
$\triangle$ . . . **68**, 71
$\triangledown$ . . . **68**, 70, 71
$(\subseteq \_\_)$ . . . **68**, 84, 91, 92
$(\supseteq \_\_)$ . . . **68**, 84, 91, 92
$\text{-}\!\downarrow_{\mathcal{D}}$ . . . **26**
$\Rightarrow_{\mathcal{D}}$ . . . **26**
$\Rightarrow_{\mathcal{M}}$ . . . **56**
$\Rightarrow_{\mathcal{TAB}}$ . . . **89**
$[\_\_]$ . . . **100**, 100, 105, 165
$|\_|$ . . . **8**
$|\ \_\ |$ . . . **8**
$\geq_G$ . . . **21**
$\succ_{\mathcal{COV}}$ . . . 72, **76**, 76–84, 102–112
$\succ_d$ . . . **23**, 25, 27, 36, 37
$\succ'_d$ . . . **28**
$\succ_{\mathcal{ML}}$ . . . **110**
$\succ^s_{mul}$ . . . **76**
$\succ_n$ . . . **23**, 24, 36
$\succ_{\mathsf{P}}$ . . . **28**
$\succ_S$ . . . **57**, 57
$\succ_\Sigma$ . . . **56**
$\succ_{\mathcal{TAB}}$ . . . **86**, 89, 92, 102
$\succ_v$ . . . **23**, 24, 25, 36
$\succ'_v$ . . . **23**, 24
$\succ_Z$ . . . **42**, 52, 56, 59
$\succsim_Z$ . . . **42**, 42–44
$\sim_Z$ . . . **42**

AF . . . **99**
$\arg_{mul}$ . . . **8**
$\arg_{set}$ . . . **8**

$\mathrm{card}^{\mathsf{V}}_{\max}$ . . . **20**, **36**

$\mathrm{d}_\downarrow$ . . . **26**
def . . . **100**
$\mathrm{dp}^{\mathrm{GT}}_{\max}$ . . . **20**
$\mathrm{dp}^{\mathsf{V}}_{\max}$ . . . **20**
$\mathrm{dp}^{x}_{\max}$ . . . **20**

ϵ . . . **100**

F . . . **7**

Γ_N . . . **20**
GT . . . **20**
$\mathsf{GT}(\mathsf{F})$ . . . **7**

id . . . **68**

$L*$ . . . **20**

$\mathbb{N}$ . . . **8**

P . . . **7**
π_f . . . **100**
π_r . . . **99**
π_{sf} . . . **100**
Π^Σ_f . . . **100**
Π^Σ_r . . . **99**
Π^Σ_{sf} . . . **100**

$S_{\mathcal{KC}}$ . . . **59**
$S_{\mathcal{ML}}$ . . . **110**, 110, 113–115
$S_{\mathcal{TAB}}$ . . . **86**, 88–92, 102–104
S_v . . . **35**, 36

$\mathsf{T}(\mathsf{F},\mathsf{V})$ . . . **7**

Υ . . . **100**

V . . . **7**

W . . . **99**

Ξ . **72**, 73, 75, 79, 82–84, 102–105, 108, 109, 114, 115
$\overline{\Xi}$ . . . **86**, 86, 88–90, 92