Soft Typing for Clausal Inference Systems

Dissertation

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

Christoph Meyer

Saarbrücken
1999

Tag des Kolloquiums:	5. Oktober 1999
Dekan:	Prof. Dr. Wolfgang Paul
Berichterstatter:	Prof. Dr. Harald Ganzinger
	Prof. Dr. Jörg Siekmann

Abstract

The purpose of this thesis is to study methods for the *semantical* control of automated theorem proving systems for first-order logic (with equality) based on certain *effective* abstraction techniques. We propose a general framework called *soft typing for clausal inference systems* to incorporate the validity/satisfiability of clauses with respect to certain model hypotheses into the control of inference systems. The effectiveness of the method is obtained by inferring *automatically* abstractions that result in approximations of validity/satisfiability of clauses with respect to these models. For an effective use of soft typing for ordered resolution and superposition, we propose the use of sets of Horn clauses (with equality) to represent approximations of model hypotheses for sets of clauses. We show that certain decidable fragments of first-order logic without equality can be generalized to non-trivial decidable *equational theories.* Our abstraction techniques, applied to arbitrary sets of clauses (with equality), result exactly in the decidable (equational) fragments. The satisfiability of clauses with respect to the approximations implies the satisfiability in the original model hypotheses.

Zusammenfassung

Wir untersuchen Methoden zur *semantischen* Steuerung von automatischen Beweissystemen für Logik erster Stufe (mit Gleichheit) mit Hilfe von bestimmten *berechenbaren* Abstraktionstechniken. Wir schlagen ein allgemeines Konzept vor, dass die Berücksichtigung der Gültigkeit beziehungsweise der Erfüllbarkeit von Klauseln bezüglich bestimmter Modellhypothesen in der Steuerung von Inferenzsystemen ermöglicht. Dieses Konzept nennen wir *soft typing for clausal inference systems.* Die Berechenbarkeit dieser Methoden wird durch automatische Abstraktionstechniken zur Approximation von Gültigkeit beziehungsweise von Erfüllbarkeit von Klauseln gewährleistet. Zur Darstellung der Approximationen von Modellhypothesen über Klauselmengen schlagen wir die Verwendung von Mengen von Hornklauseln (mit Gleichheit) vor, um Soft Typing von geordneter Resolution beziehungsweise Superposition zu ermöglichen. Wir zeigen, dass bestimmte entscheidbare Fragmente von Logik erster Stufe ohne Gleichheit zu nicht-trivialen entscheidbaren Gleichheitstheorien verallgemeinert werden können. Die Anwendung unserer Abstraktionstechniken auf beliebige Klauselmengen (mit Gleichheit) resultiert in diesen entscheidbaren Fragmenten (Gleichheitstheorien) wobei die Erfüllbarkeit von Klauseln bezüglich der Approximationen die Erfüllbarkeit in den ursprünglichen Modellhypothesen impliziert.

Acknowledgements

I thank Professor Dr. Harald Ganzinger for accepting me as a PhD student at the Max-Planck-Institute for Computer Science. The last three years in his programming logics group were a pleasant and challenging time. I am very grateful for his continuous support and encouragement.

The exhaustive elaboration of Harald's idea of soft typing for ordered resolution is one of the main contributions of this thesis. In this context, Dr. Florent Jacquemard, Dr. Christoph Weidenbach, and myself studied certain equational theories as an application of soft typing. This cooperation was very fruitful due to our diverse viewpoints on this topic, thanks to both of them. I am also grateful to Dr. Witold Charatonik and Priv. Doz. Dr. Andreas Podelski for the discussions on tree automata and logic programs.

During my last year as a PhD student I was involved in the investigation of decidable first-order fragments which are obtained by translation procedures from multi-modal propositional logics. Many valuable discussions with Dr. Renate Schmidt and Ullrich Hustadt, followed by the pleasant and productive cooperation with Dr. Margus Veanes, have been particularly important for my academic progress.

I express my gratitude to Gunnar Klau, Dr. Andreas Nonnengart, Jürgen Stuber, Dr. Jean-Marc Talbot, and Dr. Margus Veanes for reading draft versions of this thesis and for their comments and suggestions.

Special thanks to my roommate Friedrich Eisenbrand and to Dr. Sergei Vorobyov for all the interesting discussions on various topics of computer science. Collective thanks go to my colleagues Dr. Giorgio Delzanno, Solofo Ramangalahy, Dr. Leon van der Torre, and René Weiskircher for the great time at the MPI.

Contents

Extended Abstract

The purpose of this thesis is to obtain a comprehensive understanding of *semantically* guided automated theorem proving and to provide decidable fragments of first-order logic (with equality) which are suitable for an *effective* approximation of proof guidance by semantics. Semantically guided theorem proving is the attempt to establish proofs in which the meaning of the objects of concern may control the theorem proving process. The meaning of an object is associated with its truth value with respect to some model hypothesis. Suppose a model hypothesis satisfies, for example, the axiomatization of a mathematical theory. Then any (sound) inference from axioms of the theory only yields true statements with respect to the model hypothesis. Conversely, in some refutational theorem proving process, so-called reduction inferences from counterexamples of the model hypothesis may contribute to the derivation of a contradiction. A counterexample could be, for instance, a negated conjecture.

We propose a general framework called *soft typing for clausal inference systems* to incorporate semantics into automated theorem proving systems. The goal of soft typing is to minimize the derivation of non-reduction inferences by imposing semantical constraints on the inferences. Soft typing controls theorem proving derivations from clausal inference systems by a blocking mechanism according to the truth value of the involved clauses with respect to certain model hypotheses. We show that any clausal inference system which enjoys the so-called (strong) reduction property is compatible with this concept. The strong reduction property also implies the compatibility with a general concept of simplification based on a certain notion of redundancy (Bachmair & Ganzinger 1994). Simplification techniques are involved in all modern automated theorem proving systems.

A clausal inference system has the reduction property (Bachmair & Ganzinger 1997, Bachmair & Ganzinger 1998*b*) with respect to some model hypothesis I^N, if for all sets N of clauses and all minimal counterexamples C for I^N in N, there exists a (reduction) inference in the system that reduces C to a smaller counterexample than C with respect to some well-founded ordering. By I^N, we indicate that the model hypothesis may be different for distinct sets N of clauses. Intuitively, an inference system which enjoys the reduction property always contains the necessary inferences to eventually derive a contradiction from an unsatisfiable clause set. Soft typing relies on the observation that the derivation of reduction inferences is not only necessary but also sufficient while any non-reduction inference may at least be temporarily blocked. Soft typing enhances inference systems by an additional parameter that distinguishes reduction and non-reduction inferences.

Soft typing is shown to be a suitable concept for semantically guided theorem proving in terms of (refinements of) resolution and superposition (with selection) (Bachmair & Ganzinger 1994). Soft typing for semantic resolution explains the refutational completeness of semantic resolution (Slagle 1967) and clarifies, together with soft typing for ordered resolution, the theoretical background of other prominent concepts of semantically guided theorem proving methods.

The effectiveness of soft typing for ordered resolution and superposition requires decidable approximations of certain model hypotheses. We propose the use of sets of Horn clauses (with equality) to represent approximations of model hypotheses for sets of clauses in the spirit of Frühwirth, Shapiro, Vardi & Yardeni (1991). The approximation is obtained by inferring *automatically* abstractions from sets of clauses that result in a set $\mathcal{H}$ of Horn clauses. The satisfiability of clauses with respect to the minimal model of $\mathcal{H}$ implies then the satisfiability in the original model hypothesis.

We demonstrate, by a termination analysis of certain refinements of superposition, the decidability of the satisfiability problem of the class of so-called monadic equational types with respect to the following equational theories. Monadic equational types are existentially closed conjunctions of monadic atoms and equations.

- *semi-linear sorted equational theories* (Jacquemard, Meyer & Weidenbach 1998*a*) which strictly embed the (non-linear) shallow equational theories in (Comon, Haberstrau & Jouannaud 1994),
- *semi-standard sorted equational theories* (Jacquemard, Meyer & Weidenbach 1998*b*) which strictly embed the standard (equational) theories in (Nieuwenhuis 1996), and
- *linear shallow typed equational theories* which extend the monadic Horn theories in (Weidenbach 1999) by equality.

All theories are syntactically characterized as certain sets of Horn clauses (with equality) which allows the automatic abstraction of arbitrary sets of clauses into the theories. The satisfiability problem is shown to be EXPTIME-complete for linear (semi-standard) sorted equational theories and linear shallow typed equational theories and EXPTIME-hard for (non-linear) semi-linear and semi-standard sorted equational theories. Note that the E-unifiability problem is an instance of the satisfiability problem of (monadic) equational types. In contrast to the decidability of sorted unification in pseudo-linear *sort* theories (Weidenbach 1996*b*), we show that the word problem is undecidable already in pseudo-linear *equational* theories.

We also report on the complexity of the satisfiability problem of monadic types with respect to sort theories and so-called type theories. In particular, we show the EXPTIME-completeness for alternating linear (shallow) sort theories and linear shallow type theories and the EXPTIME-completeness of the non-emptiness problem of semi-linear sort theories. Moreover, the satisfiability problem of monadic types with respect to alternating non-linear shallow and semi-linear sort theories is shown to be EXPTIME-hard whereas the inclusion in EXPTIME remains open.

Ausführliche Zusammenfassung

Das Ziel dieser Arbeit ist ein umfassendes Verständnis des automatischen Beweisens mit Hilfe von Semantik zu erhalten und entscheidbare Fragmente der Logik erster Stufe (mit Gleichheit) zu identifizieren, die eine berechenbare Approximation der Beweissteuerung durch Semantik ermöglicht. Automatisches Beweisen mittels Semantik ist der Versuch die Bedeutung von Aussagen im Beweis zur Steuerung des Beweisablaufs heranzuziehen. Die Bedeutung einer Aussage wird mit ihrem Wahrheitswert bezüglich einer Modellhypothese assoziiert. Nehmen wir zum Beispiel an, dass die Axiomatisierung einer mathematischen Theorie durch eine Modellhypothese erfüllt wird. Jede (korrekte) Inferenz zwischen Axiomen der Theorie wird dann nur Aussagen produzieren, die bezüglich der Modellhypothese ebenfalls wahr sind. In einem Widerlegungsbeweis können dagegen sogenannte Reduktionsinferenzen, die von Gegenbeispielen der Modellhypothese ausgehen, zur Ableitung eines Widerspruchs beitragen. Die gegenteilige Annahme einer zu beweisenden Vermutung könnte ein solches Gegenbeispiel sein.

Wir schlagen ein allgemeines Konzept zur Steuerung von automatischen Beweissystemen mittels Semantik vor, das wir *soft typing for clausal inference systems* nennen. Das Ziel von Soft Typing ist die Minimierung der Ableitung von Inferenzen, die keine Reduktionsinferenzen sind, mit Hilfe von semantischen Restriktionen. Soft Typing steuert die Herleitung von Beweisen durch klausale Inferenzsysteme indem Inferenzen entsprechend dem Wahrheitswert der involvierten Klauseln bezüglich von bestimmten Modellhypothesen blockiert werden. Wir zeigen, dass ein klausales Inferenzsystem, das die sogenannte (starke) Reduktionseigenschaft besitzt, mit Soft Typing verträglich ist. Die starke Reduktionseigenschaft impliziert darüberhinaus die Verträglichkeit mit einem allgemeinen Konzept von Simplifikation, das auf einem bestimmten Redundanzbegriff (Bachmair & Ganzinger 1994) beruht. Simplifikationstechniken spielen eine wesentliche Rolle in allen modernen automatischen Beweissystemen.

Ein klausales Inferenzsystem besitzt die Reduktionseigenschaft (Bachmair & Ganzinger 1997, Bachmair & Ganzinger 1998*b*) bezüglich einer Modellhypothese I^N, wenn es für alle Mengen N von Klauseln and alle minimalen Gegenbeispiele C von I^N in N eine (Reduktions-) Inferenz in dem System gibt, die C zu einem kleineren Gegenbeispiel als C bezüglich einer wohlfundierten Ordnung reduziert. Wir weisen darauf hin, dass die Modellhypothese I^N für unterschiedliche Mengen N von Klauseln variieren kann. Ein Inferenzsystem, das die Reduktionseigenschaft besitzt, enthält also immer die notwendigen Inferenzen um einen Widerspruch innerhalb einer unerfüllbaren Klauselmenge herleiten zu können. Soft Typing basiert auf der Beobachtung, dass die Ableitung von Reduktionsinferenzen nicht nur notwendig ist, sondern auch hinreichend wobei jede Inferenz, die keine Reduktionsinferenz ist, mindestens temporär blockiert werden kann. Soft Typing erweitert Inferenzsysteme mit einem zusätzlichen Parameter, der eine Unterscheidung von Reduktionsinferenzen und anderen Inferenzen ermöglicht.

Wir zeigen, dass (bestimmte Verfeinerungen von) Resolution und Superposition (mit Selektion) (Bachmair & Ganzinger 1994) mit Soft Typing verträglich sind. Soft Typing für semantische Resolution bestätigt nicht nur die Widerlegungsvollständigkeit von semantischer Resolution (Slagle 1967), sondern klärt auch, genauso wie Soft Typing für geordnete Resolution, den theoretischen Hintergrund anderer bekannter Verfahren des automatischen Beweisens mittels Semantik.

Um Soft Typing für geordnete Resolution beziehungsweise Superposition einsetzen zu können, sind entscheidbare Approximationen bestimmter Modellhypothesen notwendig. Zur Darstellung der Approximationen von Modellhypothesen über Klauselmengen schlagen wir die Verwendung von Mengen von Hornklauseln (mit Gleichheit) vor im Sinne einer Verallgemeinerung des Vorschlags von Frühwirth et al. (1991). Die Approximation erhält man durch die Ableitung automatischer Abstraktionen von Klauselmengen, die in einer Menge $\mathcal{H}$ von Hornklauseln resultieren. Die Erfüllbarkeit von Klauseln bezüglich des minimalen Modells von $\mathcal{H}$ impliziert dann die Erfüllbarkeit in der ursprünglichen Modellhypothese.

Wir demonstrieren mit Hilfe einer Analyse der Termination von bestimmten Verfeinerungen von Superposition die Entscheidbarkeit des Erfüllbarkeitsproblems der Klasse der sogenannten monadischen Gleichheitstypen bezüglich folgender Gleichheitstheorien. Monadische Gleichheitstypen sind existenziell quantifizierte Konjunktionen von monadischen Atomen und Gleichungen.

- *semi-lineare sortierte Gleichheitstheorien* (Jacquemard et al. 1998*a*), die die (nicht-linearen) flachen Gleichheitstheorien in (Comon et al. 1994) beinhalten,

- *semi-standard sortierte Gleichheitstheorien* (Jacquemard et al. 1998*b*), die die sogenannten Standardgleichheitstheorien in (Nieuwenhuis 1996) beinhalten, und

- *lineare flache typisierte Gleichheitstheorien*, die eine Erweiterung der monadischen Horntheorien in (Weidenbach 1999) mit Gleichheit darstellen.

Alle Theorien können rein syntaktisch als bestimmte Mengen von Hornklauseln (mit Gleichheit) beschrieben werden, so dass eine automatische Abstraktion von beliebigen Klauselmengen in diese Theorien möglich ist. Wir demonstrieren, dass das Erfüllbarkeitsproblem bezüglich linearer (semi-standard) sortierter Gleichheitstheorien und bezüglich linearer flacher typisierter Gleichheitstheorien EXPTIME-vollständig ist. Das gleiche Problem bezüglich (nicht-linearer) semi-linearer und semi-standard sortierter Gleichheitstheorien wird als EXPTIME-hart gezeigt. Wir weisen darauf hin, dass das Unifikationsproblem ein Teilproblem des Erfüllbarkeitsproblems von (monadischen) Gleichheitstypen ist. Im

Gegensatz zur Entscheidbarkeit von sortierter Unifikation in pseudo-linearen Sortentheorien (Weidenbach 1996*b*), können wir zeigen, dass das Wortproblem bereits in pseudo-linearen Gleichheitstheorien unentscheidbar ist.

Weiterhin untersuchen wir die Komplexität des Erfüllbarkeitsproblems von monadischen Typen bezüglich Sortentheorien und sogenannten Typtheorien. Wir zeigen die EXPTIME-Vollständigkeit für alternierende lineare (flache) Sortentheorien und lineare flache Typtheorien ebenso wie die EXPTIME-Vollständigkeit des Leerheitsproblems von semi-linearen Sortentheorien. Das Erfüllbarkeitsproblem von monadischen Typen bezüglich alternierender nicht-linearer flacher und semi-linearer Sortentheorien wird als EXPTIME-hart gekennzeichnet während das Enthaltensein in EXPTIME ein offenes Problem bleibt.

Chapter 1

Introduction

The central motivation of this thesis is to study methods for the *semantical* control of automated theorem proving systems for first-order logic (with equality) based on certain *effective* abstraction techniques. We propose a general framework called *soft typing for clausal inference systems*, e.g., for (refinements of) resolution and superposition, as a concept to incorporate the validity/satisfiability of clauses with respect to certain model hypotheses into the control of the inference process. The effectiveness of the method is obtained by inferring *automatically* abstractions that result in approximations of validity/satisfiability of clauses with respect to these models. We introduce soft typing as a suitable method to address the problem of semantically guided theorem proving by demonstrating, in particular, that

- soft typing is compatible with clausal inference systems that enjoy the so-called (*strong*) *reduction property* and, conversely, that clausal inference systems with the strong reduction property are compatible with a general concept of *simplification* based on a certain notion of *redundancy.*

- soft typing can be established by a suitable notion of (temporarily) *blocked* clauses and inferences which reflects the dynamic nature of theorem proving with respect to semantical considerations.

- soft typing in combination with the (strong) reduction property clarifies the theoretical background of other prominent concepts of semantically guided theorem proving methods.

For an effective use of soft typing for ordered resolution and superposition, we propose the use of sets of Horn clauses to represent (approximations of) model hypotheses for sets of clauses and, in particular, we show that

- certain decidable fragments of first-order logic can be generalized to non-trivial decidable *equational theories* within the same complexity class.

- certain automatic abstractions of arbitrary sets of clauses result exactly in the decidable fragments. The satisfiability of clauses with respect to the approximation implies the satisfiability in the original model hypothesis.

In the following, we give a brief introduction to semantically guided theorem proving and discuss the main aspects of soft typing for clausal inference systems. We then describe the idea of automatic abstractions and explain the general concept to obtain the various decidability results of this work. We conclude this chapter by an overview of complexity results on the satisfiability problem of certain classes of first-order formulae (with equality) with respect to the decidable (equational) theories.

1.1 Soft Typing

Automated theorem proving is the mechanization of the process of proving mathematical theorems. Throughout this work we consider, as the formal language for this purpose, first-order logic (with equality). A common property among the objects which are potentially involved in a proof attempt is that the derived objects are logical consequences of at least those objects that actually motivated the proof step. An *inference system* or *logic calculus*, based on a formal language, approximates the logical consequence relation such that the computation of *theorem proving derivations* is effective using the (syntactic) inference rules of the system. A syntactic inference rule usually describes an inherently *local* form of theorem proving since the application of the rule does not involve the particular context of the premises. The *search space* induced by such inference rules is therefore eminently non-deterministic which motivates the investigation of *search strategies*. In this thesis we are concerned with the investigation of methods to prune the search space of inference systems by semantical considerations.

Semantically guided theorem proving is the attempt to establish proofs in which the meaning of the objects of concern may trigger the next step in a theorem proving derivation. From a human-oriented point of view, this concept seems to be the natural choice of theorem proving whereas, from a computational point of view, it is obviously difficult to capture this particular search strategy. For a formal analysis of semantically guided theorem proving we reach two fundamental design decisions: (i) the meaning of an object is identified with the validity/satisfiability of the object with respect to some model hypothesis and (ii) we restrict our attention to refutational theorem proving by *clausal inference systems* which are based on the particular data structure of clauses. The well-known *resolution calculus* (Robinson 1965) is a clausal inference system which plays a major role throughout this work.

The goal of *soft typing for clausal inference systems* is to minimize the derivation of superfluous information by imposing semantical constraints on the inferences. We shall explain soft typing by analyzing fundamental properties of refutationally complete inference systems. A sufficient criterion to establish the refutational completeness of a clausal inference system is the so-called *reduction property* (Bachmair & Ganzinger 1997, Bachmair & Ganzinger 1998*b*). A clausal inference system has the reduction property with respect to some model hypothesis I^N, if for all sets N of clauses and all minimal counterexamples C for I^N in N, there exists a (reduction) inference in the system that reduces C to a smaller counterexample than C with respect to some well-founded ordering. By I^N, we indicate that the model hypothesis may be different for distinct sets N of clauses. We may also call I^N the candidate model for N.

Intuitively, an inference system which enjoys the reduction property always contains the necessary inferences to eventually derive a contradiction from an unsatisfiable clause set where the presence of the empty clause, as the smallest object in the well-founded ordering, indicates the contradiction. Soft typing relies on the observation that the derivation of reduction inferences is not only necessary but also sufficient while any non-reduction inference may at least be temporarily *blocked.* Soft typing for clausal inference systems enhances the inference systems by an additional parameter that distinguishes reduction and non-reduction inferences.

Example 1.1.1
The following table contains a simple inconsistent set N of clauses. We assume that soft typing for resolution is applied where A and B are true in the candidate model I^N. Resolution inferences are restricted to reduction inferences with respect to I^N.

$\sharp$	Clause C	Remarks
1	$\rightarrow A$	true in I^N, positive clause
2	$A \rightarrow B$	true in I^N
3	$B \rightarrow A$	true in I^N
4	$B \rightarrow$	false in I^N, negative clause

There are two resolution inferences from (1) and (2) as well as (2) and (4) where only the latter is a reduction inference. Resolution from (2) and (4) yields the unit clause $A \rightarrow$. The new clause set looks as follows:

$\sharp$	Clause C	Remarks
1	$\rightarrow A$	true in I^N, positive clause
2	$A \rightarrow B$	true in I^N
3	$B \rightarrow A$	true in I^N
4	$B \rightarrow$	false in I^N, negative clause
5	$A \rightarrow$	false in I^N, resolvent of (2) and (4)

There are two resolution inferences from (1) and (2) as well as (1) and (5) where again only the latter is a reduction inference. Resolution from (1) and (5) yields the empty clause. The example shows that soft typing may control a theorem proving derivation such that the derivation even becomes a completely deterministic process.

Consider Figure 1.1 which depicts the operational concept of soft typing. At each step in a theorem proving derivation an inference may be classified as a reduction or non-reduction inference with respect to the candidate model I^N for the current set N of clauses. In general, however, blocking is not effective for (i) arbitrary candidate models

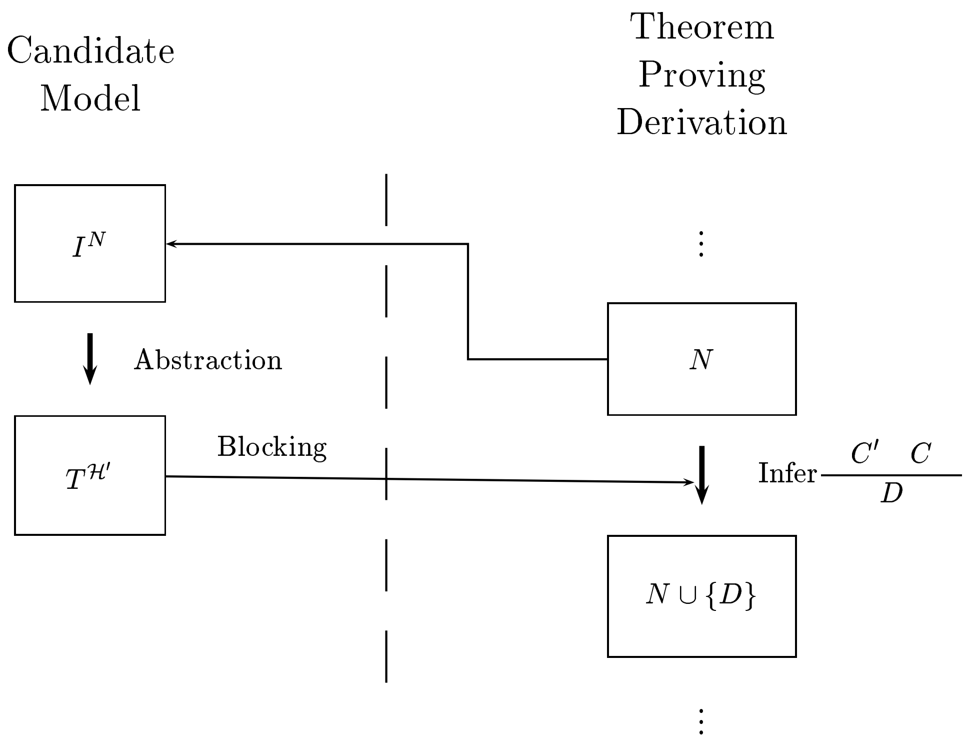

Figure 1.1: Soft typing for clausal inference systems

and (ii) arbitrary forms of clauses since the computation of counterexamples is in general undecidable. For an effective use of soft typing we therefore propose (i) certain *abstraction* techniques and (ii) so-called *typing functions*. The computation of the truth value of at least some part of an arbitrary clause C may become decidable with respect to sufficiently abstracted candidate models. A typing function identifies which part of a clause is subject to an evaluation. We call this part the *type of* C. For instance, we may identify the negative occurrences of monadic atoms in C as the type of C and compute abstract candidate models for which emptiness of monadic predicates is decidable. Whenever the emptiness of a monadic predicate implies the emptiness with respect to the original candidate model, we may at least effectively show that C cannot be a counterexample. Note that we discuss the abstraction concept in the next section. In particular, we will explain the abstraction step from I^N to $T^{\mathcal{H}'}$ in Figure 1.1 in more details as depicted in Figure 1.2.

We give a brief background of semantically guided theorem proving methods. Semantics in automated theorem proving have already been investigated by Gelernter (1960) who showed by his *geometry theorem prover* that already a limited understanding of the problem set may have an impact on syntactic proof procedures. Slagle (1967) generalized resolution to *semantic resolution* and marked in this way the beginning of semantically guided resolution-based theorem proving with respect to arbitrary first-order model domains. He observed that resolution inferences may be restricted to clauses with distinct truth value with respect to an arbitrary model hypothesis and thus generalized also the well-known *set-of-support* strategy.

We demonstrate, based on an appropriate construction of the candidate model (for semantic resolution), that *soft typing for semantic resolution* explains the refutational completeness of semantic resolution. Surprisingly, soft typing for semantic resolution also strictly embeds the so-called *ordered semantic hyper-linking* method (Plaisted 1994, Plaisted & Zhu 1997). More precisely, we show that the saturation criterion of ordered semantic hyper-linking is a particular instance of the saturation criterion of soft typing for semantic resolution. Saturation refers to the exhaustive application of inference rules. Hyper-linking distinguishes the approximation of sets of ground instances by unification and the verification of their unsatisfiability as it is implicitly combined in the resolution rule. That is, hyper-linking computes the most general unifier of potential hyper-resolution inferences and adds the premises instantiated by the unifier to the clause set rather than adding the conclusion. A subsequent enumeration of ground instances in combination with a decision procedure for propositional logic completes the picture of clause-linking.

Another example of semantically guided theorem proving is the prototypical theorem prover Satchmo (Manthey & Bry 1988) and its efficient counterpart, the *model generation theorem prover* Mgtp (Hasegawa, Fujita & Koshimura 1997), which has been implemented within the fifth generation project in Japan. It turns out, given an appropriate candidate model, that ordered resolution is compatible with soft typing and, moreover, that Satchmo's (Mgtp's) saturation criterion is an instance of the saturation criterion of soft typing for ordered resolution (Ganzinger, Meyer & Weidenbach 1997).

In the context of first-order theorem proving with equality, we study the compatibility of the superposition calculus (Bachmair & Ganzinger 1994) with soft typing. The superposition calculus includes a powerful redundancy concept which covers many of today's redundancy elimination methods. Independently from superposition, we refine the reduction property to the *strong reduction property* and argue that any clausal inference system which enjoys the strong reduction property is compatible with soft typing and *simplification* techniques based on the redundancy concept of Bachmair & Ganzinger (1994).

1.2 Abstractions

For an effective use of soft typing, at least for (refinements of) ordered resolution and superposition, we propose to use sets of Horn clauses to represent decidable approximations of candidate models of sets of clauses. The idea is inspired by Frühwirth et al. (1991) who use (abstracted) logic programs to infer type information for logic programs. This concept is related to methods which have been called *soft typing* (Cartwright & Fagan 1991, Cartwright & Felleisen 1996) in the programming language area. We generalize this approach to sets of clauses with equality.

Consider Figure 1.2 and suppose that the candidate model for a set N of clauses is given by I^N. We (i) transform the clauses N into a set of Horn clauses $\mathcal{H}$ and (ii) abstract the clauses in $\mathcal{H}$ further to a set of Horn clauses $\mathcal{H}'$ for which the satisfiability problem of

$$\begin{tikzcd}
\boxed{\text{Candidate Model } I^N} \arrow[r, phantom, "\subseteq"] & \boxed{\text{Minimal Model } T^{\mathcal{H}}} \arrow[r, phantom, "\subseteq"] & \boxed{\text{Minimal Model } T^{\mathcal{H}'}} \\
\boxed{\text{Clause Set } N} \arrow[u] \arrow[r, Rightarrow, "\uparrow\ \text{Distribution}"'] & \boxed{\text{Horn Theory } \mathcal{H}} \arrow[u] \arrow[r, Rightarrow, "\uparrow\ \text{Abstraction}"'] & \boxed{\text{Decidable Theory } \mathcal{H}'} \arrow[u]
\end{tikzcd}$$

Figure 1.2: Abstraction for soft typing

certain formulae with respect to the minimal model $T^{\mathcal{H}'}$ of $\mathcal{H}'$ is decidable. With $T^{\mathcal{H}'}$ we obtain a minimal model which is an *upper approximation* of the original candidate model.

Example 1.2.1
Suppose that $T^{\mathcal{H}'}$ is an upper approximation in which satisfiability of conjunctions of atoms is decidable. Consider a clause $C = A_1, \ldots, A_n \rightarrow \Delta$. If $T^{\mathcal{H}'} \not\models A_1 \wedge \ldots \wedge A_n$ then $I^N \not\models A_1 \wedge \ldots \wedge A_n$ which implies that C is true in I^N or, in other words, that C is not a counterexample of I^N.

The dual concept is also possible which then involves *lower approximations*. For the candidate models of ordered resolution and superposition, step (i) can be achieved by *distributing* disjunction into conjunctions, i.e. by transforming a non-Horn clause $\Gamma \rightarrow A_1, \ldots, A_n$ into n Horn clauses $\Gamma \rightarrow A_1$ to $\Gamma \rightarrow A_n$. In this way, we obtain arbitrary Horn theories which are, in general, undecidable. Further abstraction steps involve variable renaming and/or restrictions to certain predicates. The impact of the method seems to be determined by two aspects:

1. Does the abstracted theory describe a non-trivial structure?

2. If so, does this structure sufficiently reflect the structure of the original problem?

In the following, we introduce non-trivial decidable (equational) theories in order to address the first question. The theories are syntactically characterized which allows to infer automatically the decidable approximations. The problem of the second question, however, is not considered in this work. We rather discuss some informal aspects in the conclusion. In the following section, we will clarify the abstraction step from a Horn theory $\mathcal{H}$ to a decidable theory $\mathcal{H}'$ in Figure 1.2 and the subsequent decision procedure. The details of this process are depicted in Figure 1.3.

1.3 Decidable Approximations

We consider certain *monadic Horn theories* (with equality) as representations of decidable approximations. A monadic Horn theory is a finite set of monadic Horn clauses which only

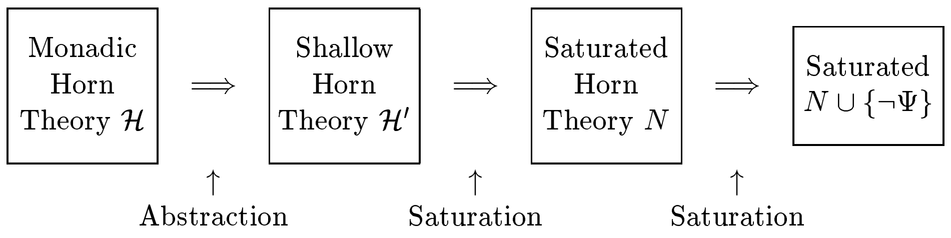
Monadic Horn Theory $\mathcal{H}$
Shallow Horn Theory $\mathcal{H}'$
Saturated Horn Theory N
Saturated $N \cup \{\neg\Psi\}$
Abstraction
Saturation
Saturation

Figure 1.3: Decision process for monadic Horn theories

contain monadic (unary) predicate symbols (and equality). The equality predicate may only occur in the positive literals of the Horn clauses. The clauses in a monadic Horn theory are represented by so-called *sorted clauses*. The data structure of sorted clauses simplifies technical matters but is not essential to obtain decidability.

Example 1.3.1
A clause C of the form $S(x), P(g(x,y)) \to T(f(x,y))$ may be represented by a sorted clause C' of the form $S(x) \,\|\, P(g(x,y)) \to T(f(x,y))$ where $S(x)$ is called the *sort constraint of* C'. The sort constraint distinguishes monadic atoms of the form $S(x)$ and monadic atoms which contain complex terms. The sorted clause C' is logically equivalent to the clause C.

We consider the satisfiability problem of the theory of monadic (equational) types with respect to monadic Horn theories. A monadic (equational) type is an existentially closed conjunction of monadic atoms (and equations). Suppose that a typing function selects the monadic (equational) type of each clause in a set N of clauses, i.e. each negative occurrence of a monadic atom (or equation) in a clause is subject to blocking. Then inferences may effectively be blocked from premises in N whose monadic (equational) type is unsatisfiable with respect to an upper approximation of some candidate model whenever the satisfiability problem is decidable with respect to the approximation.

Figure 1.3 shows the decision process for monadic Horn theories. In general, the satisfiability problem is undecidable with respect to arbitrary monadic Horn theories. However, given a monadic Horn theory $\mathcal{H}$, we may abstract $\mathcal{H}$ into a so-called *shallow* monadic Horn theory $\mathcal{H}'$. A shallow theory is obtained by "flattening" non-shallow terms. The original idea goes back to the work by Uribe (1992). Intuitively, a term is shallow whenever all variables in the term occur at most at depth one.

Example 1.3.2
Let $\mathcal{H}$ be a monadic Horn theory. Suppose there is a sorted clause C in $\mathcal{H}$ of the form $S(x), S(y) \,\|\, \to T(f(g(x), g(x), y))$ which contains a non-shallow term. We may represent C by its "flattened" version C' of the form $S_{g(x)}(z), S(y) \,\|\, \to T(f(z,z,y))$ and the new clause D of the form $S(x) \,\|\, \to S_{g(x)}(g(x))$. Let $\mathcal{H}'$ be the set $(\mathcal{H} \setminus \{C\}) \cup \{C', D\}$. In this case, the minimal model of $\mathcal{H}'$ is even equivalent to the minimal model of $\mathcal{H}$ up to monadic atoms which contain the new predicate symbols $S_{g(x)}$.

$$\begin{tikzcd} \boxed{\text{Sort Theory}} \arrow[r, Rightarrow, "\approx"] \arrow[d, Rightarrow, "\Theta"] & \boxed{\text{Sorted Equational Theory}} \arrow[d, Rightarrow, "\Theta"] \\ \boxed{\text{Type Theory}} \arrow[r, Rightarrow, "\approx"] & \boxed{\text{Typed Equational Theory}} \end{tikzcd}$$

Figure 1.4: Classes of monadic Horn theories

The next step in the decision process, which has been called *type inference* in (Frühwirth et al. 1991), involves the (finite) saturation of the shallow theory under specialized versions of ordered resolution and superposition. In particular, we employ variants of *sort resolution* and *sorted superposition/paramodulation* (Weidenbach 1996*a*) which have been designed for sorted clauses. Finally, the satisfiability of a monadic (equational) type Ψ with respect to a saturated set N can be decided by (finitely) saturating N and the negated type $\neg\Psi$. Frühwirth et al. (1991) call the last step *type checking.*

The undecidability of the satisfiability problem with respect to arbitrary monadic Horn theories motivates a classification of decidable theories depicted by Figure 1.4. A *sort theory* is a monadic Horn theory which contains sorted clauses of the form $\Psi \parallel \ \rightarrow S(t)$ where Ψ contains only monadic atoms of the form $T(x)$ with $x \in \mathit{vars}(t)$. A clause of a sort theory is called a *sort declaration.* The notation is inspired by the logical equivalence of the clausal representation and the standard representation of sort theories (Weidenbach 1996*a*).

We distinguish *shallow*, *linear*, *semi-linear*, and *pseudo-linear* sort theories depending on the form of t in the sort declarations. A term is semi-linear whenever the paths to the non-linear occurrences of each variable in the term are labeled by the same symbols. Pseudo-linearity is a relaxed form of semi-linearity where the non-linear occurrences of each variable must occur at the same depth throughout the term.

Example 1.3.3
The term $f(g(x), g(x), h(y, y))$ is semi-linear and pseudo-linear but neither linear nor shallow whereas the term $f(g(x), g(x), h(x, y))$ is pseudo-linear but not semi-linear. The term $f(x, x, y)$ is shallow, semi-linear, and pseudo-linear but not linear.

The class of *type theories* strictly embeds the class of sort theories. A type theory consists of generalizations of sort declarations called *type declarations* which are of the form $\Psi \parallel \Theta \rightarrow S(t)$ where Θ may contain arbitrary monadic atoms. The class of type theories corresponds to the class of monadic Horn clauses considered by Weidenbach (1999). The notation of type theory is not intended to indicate any relation to the research area of *type theory.*

The concept of *sorted equational theories* extends the class of sort theories by so-called *sorted equations* which are represented by sorted clauses of the form $\Psi \parallel \ \rightarrow s \approx t$ where Ψ contains monadic atoms of the form $T(x)$ with $x \in \mathit{vars}(s,t)$. The definition for shallow, linear, semi-linear, and pseudo-linear equations follows in a straightforward way from the representation of equations of the form $f(s_1,\ldots,s_n) \approx g(t_1,\ldots,t_m)$ and $f(s_1,\ldots,s_n) \approx x$ by the terms $h(s_1,\ldots,s_n,t_1,\ldots,t_m)$ and $h(s_1,\ldots,s_n,x)$, respectively. Semi-linear sorted equational theories strictly embed the (non-linear) shallow equational theories of Comon et al. (1994). We also consider so-called *semi-standard* sorted equational theories which strictly embed the standard (equational) theories of Nieuwenhuis (1996).

The definition of *typed equational theories* can be obtained in two ways, either from type theories by adding so-called *typed equations* of the form $\Psi \parallel \Theta \rightarrow s \approx t$ where Θ may contain arbitrary monadic atoms, or else from sorted equational theories by extending sort declarations and sorted equations to type declarations and typed equations, respectively.

By a termination analysis of saturation under sorted superposition/paramodulation, we obtain the following new decidability results of the satisfiability problem of the theory of monadic equational types with respect to the following equational theories:

- *semi-linear sorted equational theories* (Jacquemard et al. 1998*a*) which strictly embed the (non-linear) shallow equational theories in (Comon et al. 1994),
- *semi-standard sorted equational theories* (Jacquemard et al. 1998*b*) which strictly generalize the standard (equational) theories in (Nieuwenhuis 1996), and
- *linear shallow typed equational theories* which extend the monadic Horn theories in (Weidenbach 1999) by equality.

Sort theories and type theories can only be used to decide the satisfiability of monadic types without equality. The satisfiability problem of types which include equality is restricted to the equational theories. The E-unifiability problem is an instance of the satisfiability problem where the type contains only one equation. We draw the borderline of decidability by showing that the word problem of pseudo-linear equational theories is undecidable. In contrast, the satisfiability problem of monadic types with respect to pseudo-linear sort theories is decidable (Weidenbach 1996*b*).

1.4 Complexity

We derive several complexity results on the satisfiability problem from the relationship between sort theories and tree automata. In fact, an (alternating) linear shallow sort theory is shown to be essentially an (alternating) tree automaton up to a polynomial increase in the size of the theory. An alternating sort theory contains sort intersection problems in the sort constraint of the declarations. From an automata-theoretic point of

Sort Theory	Non-Emptiness	Reference
linear shallow	linear	e.g. Comon et al. (1997)
linear		Weidenbach (1996*a*)
non-linear shallow (deterministic)	EXPTIME-complete polynomial	Comon et al. (1997) Bogaert & Tison (1992)
semi-linear	EXPTIME-complete	4.4.12
pseudo-linear	decidable	Weidenbach (1996*b*)

Table 1.1: Complexity of (non-alternating) sort theories

Sort Theory	Monadic (Variable) Type	Reference
linear shallow	EXPTIME-complete	4.3.14
linear		4.4.13
non-linear shallow	EXPTIME-hard	4.3.13
semi-linear		4.4.11
pseudo-linear	decidable	Weidenbach (1996*b*)

Table 1.2: Complexity of alternating sort theories

view, alternation generalizes non-determinism by adding to the *existential states* of non-determinism the *universal states* of alternation. We also argue that an (alternating) non-linear shallow sort theory is essentially an (alternating) tree automaton with (syntactic) equality constraints between brother terms which have been introduced by Bogaert & Tison (1992).

Table 1.1 lists the complexity of the non-emptiness problem of non-alternating sort theories. The non-emptiness problem is an instance of the satisfiability problem of monadic types which contain only one monadic atom of the form $S(x)$. The EXPTIME-completeness for semi-linear sort theories follows from the fact that the transformation to a non-linear shallow sort theory can be done in polynomial time.

Table 1.2 contains the complexity of the satisfiability problem of monadic (variable) types with respect to alternating sort theories. Monadic variable types are conjunctions of monadic atoms of the form $S(x)$. Hence, the satisfiability problem of monadic variable types may express the intersection non-emptiness problem of (non-alternating) tree automata which is EXPTIME-complete (Frühwirth et al. 1991, Seidl 1994, Veanes 1997). Note that we consider the satisfiability problem with respect to monadic types which may include monadic atoms of the form $S(t)$ where t is an arbitrary term. In combination of the results given in Table 1.1, we observe that alternation as well as non-linearity result in an exponential increase in complexity. It is not clear whether the combination of both can still be solved in simply exponential time which implies that the EXPTIME-completeness for non-linear shallow and semi-linear alternating sort theories remains open.

Consider Table 1.3. The satisfiability problem of monadic (variable) types with respect

Type Theory	Monadic (Variable) Type	Reference
uniform program	EXPTIME-complete	5.2.9
type program		5.3.6
linear shallow		5.2.8

Table 1.3: Complexity of type theories

Sorted Equational Theory	M. Equational (Variable) Type	Reference
linear shallow	EXPTIME-complete	6.3.20
linear semi-shallow		6.7.8
linear		6.4.8
linear semi-standard		6.8.10
non-linear shallow	EXPTIME-hard	6.3.19
semi-linear		6.4.7
pseudo-linear	undecidable	6.5.5

Table 1.4: Complexity of sorted equational theories

to linear shallow type theories can be settled as EXPTIME-complete. In fact, we can show that the simultaneous saturation of a type theory and the negated monadic type can be done in simply exponential time. A subsequent transformation results in simply exponential time, however, with respect to the number of monadic predicates and not the size of the theory, in a non-alternating version of the saturated theory. A final non-emptiness test can be done in linear time. The problem is already EXPTIME-hard for so-called *uniform programs* which are an instance of linear shallow type theories. *Type programs* can be transformed in polynomial time into uniform programs.

The conclusion of Table 1.4 and 1.5 is that the presence of equality affects the complexity only by a polynomial increase compared to the same problems with respect to the theories without equality. Similar to the problem for type theories, we can show that the simultaneous saturation under sorted superposition/paramodulation of an equational theory and the negated monadic equational type can be done in simply exponential time. A subsequent transformation adds an exponential factor resulting in a non-alternating version of the saturated theory. A final non-emptiness test which does not involve equality can be done in linear time.

Type Theory	Monadic Equational (Variable) Type	Reference
linear shallow	EXPTIME-complete	7.1.5

Table 1.5: Complexity of typed equational theories

1.5 Outline

Chapter 2. We introduce the basic notions and notations which are used throughout this work. In particular, the syntactic characterizations of various (equational) theories are given in this chapter. More specific constructions are introduced in the according chapters.

Chapter 3. Soft typing is presented as a general framework to incorporate semantical knowledge into theorem proving processes by clausal inference systems. We discuss various instantiations of soft typing to semantic resolution, ordered resolution, and superposition. Since minimal models are used throughout this thesis we recall the basic concepts of the minimal model semantics of logic programs and slightly generalize them to treat equality appropriately. Formal definitions of abstractions, approximations, and theorem proving derivations (trees) as well as a brief discussion of the lifting problem complete the soft typing framework. We close the chapter with an overview of other work from the area of semantically guided theorem proving and demonstrate that soft typing strictly embeds other methods of this area.

Chapter 4. We introduce certain (non-linear) sort theories which are represented by a particular form of clauses. The sort theories essentially correspond to tree automata (with equality constraints). The purpose of this chapter is to introduce a general concept to obtain the various decidability results in the subsequent chapters rather than presenting new results, except for the complexity statements.

Chapter 5. Type theories are generalizations of sort theories. The notation of type theory is not intended to indicate any relation to the research area of *type theory*. We relate type theories to logic programs and obtain complexity results which are similar to the results for sort theories although the decidability results are not new. We discuss abstraction techniques which improve upon the known methods for this purpose. Similar to the chapter on sort theories, the presentation of type theories is also motivated to clarify fundamental techniques which are later generalized to work for type theories with equality.

Chapter 6. Sorted equational theories are the subject of this chapter. We may view sorted equational theories either as generalizations of sort theories to include equality or as equational theories in which the variables carry some sort information. We therefore relate these theories to tree automata as well as to so-called standard (equational) theories. We demonstrate the decidability of the E-unifiability problem for various theories and obtain complexity results for the more general satisfiability problem with respect to certain forms of first-order formulae (with equality). An undecidability proof of an extension of the decidable fragment reveals the borderline of computability. Finally, we demonstrate the decidability of the most general fragment in this chapter which strictly embeds the standard theories.

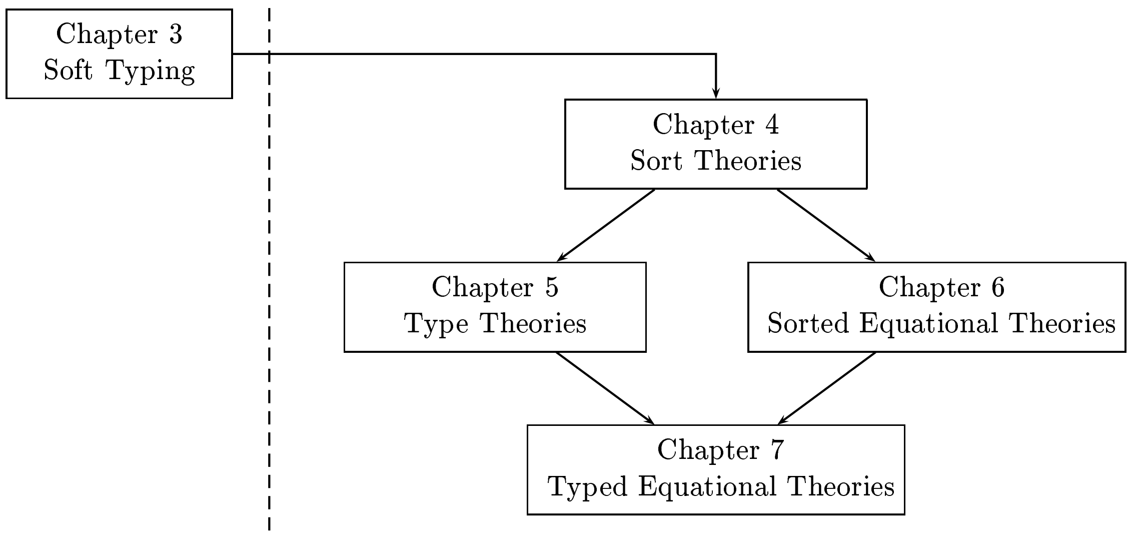

Figure 1.5: Dependencies between the chapters

Chapter 7. Typed equational theories are type theories with equality. We show the decidability of the satisfiability problem with respect to certain forms of first-order formulae (with equality). This problem is in the same complexity class as the similar problem for sorted equational theories.

Chapter 8. We summarize the main contributions of this thesis and discuss future work. We briefly report on experiments with an implementation of soft typing with respect to certain sort theories and motivate further work in this regard. Moreover, we discuss, as potential applications of soft typing, other decidable classes which involve constructor-based alphabets in equational theories or non-monadic predicates in decidable first-order fragments.

How to read the thesis. The dependencies between the chapters are illustrated in Figure 1.5. The thesis consists of two major parts, Chapter 3 on soft typing and Chapter 4 through 7 on decidability and complexity issues with respect to various (equational) theories. Both parts can be read independently although Chapter 3 motivates the work of the subsequent chapters. Conversely, the results of Chapter 4 through 7 have immediate applications in the general framework of soft typing.

Chapter 4 marks the entry point of the second part and is recommended as an introduction to Chapter 5 and 6. However, the main Chapter 6 of the second part can be read independently from Chapter 5. Finally, the ideas of Chapter 4 through 6 culminate in Chapter 7.

Chapter 2

Preliminaries

We define the fundamental notions of *orderings* and their extensions to *orderings on multisets.* Throughout this work, we assume that *first-order clauses* are represented by multisets of *literals. First-order terms* are defined as usual and form, in unordered pairs, *equations.* We introduce the standard notions of *positions* in terms and *substitutions.* Certain classes of terms are distinguished, by syntactic characterizations of their variable occurrences, in *shallow*, *linear*, *semi-linear*, and *pseudo-linear* categories. The classification plays an important role in the identification of decidable (equational) Horn theories. We use the standard notions of first-order languages and first-order models. At the end of this chapter, we also define the notion of *equational theories* and *term rewriting systems.* In particular, the definitions of the theories and rewrite systems origin in the survey of Dershowitz & Jouannaud (1990). Many other preliminary definitions are motivated by Baader & Nipkow (1998).

2.1 Orderings and Multisets

A *quasi-ordering* $\succeq$ on a set A is any reflexive and transitive binary relation on A. The intersection of $\succeq$ and the inverse $\preceq$ is the *associated equivalence relation* $\sim$ *of* $\succeq$, i.e. $a \sim b$ whenever $a \succeq b$ and $b \succeq a$ for all $a, b \in A$. An anti-symmetric quasi-ordering $\succeq$ on A is called a *partial ordering on* A where anti-symmetric means that, for all $a, b \in A$, if $a \succeq b$ and $b \succeq a$ then a is equal to b. Note that $\sim$ of a partial ordering corresponds to equality. The difference $\succ$ of a partial ordering $\succeq$ on A and $\sim$ is called a *strict partial ordering on* A, i.e. $a \succ b$ whenever $a \succeq b$ and not $a \sim b$ for all $a, b \in A$. The difference of a quasi-ordering $\succeq$ on A and $\sim$ is the *associated strict partial ordering* $\succ$ *of* $\succeq$ *on* A. Note that, in general, the associated strict partial ordering $\succ$ is not a strict partial ordering, since the quasi-ordering $\succeq$ does not have to be anti-symmetric.

Given a quasi-ordering $\succeq$ on A, an element $a \in A$ is called *maximal* (*minimal*) in A if, for all $b \in A$, $b \succeq a$ implies $a \succeq b$ ($b \preceq a$ implies $a \preceq b$). A quasi-ordering $\succeq$ (on A) is called *total on* A if, for all $a, b \in A$, either $a \succeq b$, or else $b \succeq a$. It is called *well-founded* if each subset of A has a minimal element, i.e. for all $B \subseteq A$ there is a $b \in B$ such that there is no $b' \in B$ with $b \succ b'$. It is called *Noetherian* if there is no infinite descending chain $a_0 \succ a_1 \succ a_2 \succ \ldots$ with $a_i \in A$ for all i. Note that, assuming the axiom of choice, both notions are equivalent.

A *multiset* on a set A is a function M from A to the natural numbers. The intended meaning of $M(a)$ with $a \in A$ is the number of occurrences of a in M. We say that a is an *element of* M if $M(a) > 0$. The inclusion $M \subseteq N$ holds whenever $M(a) \leq N(a)$ for all $a \in A$. A multiset M on A is empty, denoted by $M = \emptyset$, whenever $M(a) = 0$ for all $a \in A$. The union, intersection, and difference of multisets are defined by the identities $M_1(x) \cup M_2(x) = M_1(x) + M_2(x)$, $M_1(x) \cap M_2(x) = \min(M_1(x), M_2(x))$, and $M_1(x) \setminus M_2(x) = \max(0, M_1(x) - M_2(x))$, respectively. An element a of a multiset M on A is called *maximal* (*minimal*) with respect to M and a strict partial ordering $\succ$ on A if there is no $b \in A$ with $b \succ a$ ($a \succ b$). The element a is called *strictly maximal* (*strictly minimal*) with respect to M and $\succ$ if there is no $b \in A$ with $b \succeq a$ ($a \succeq b$). We extend a well-founded strict partial ordering $\succ$ on A to a well-founded *multiset-ordering* $\gg$ as follows. Given two multisets N and M on a set A, respectively, and a strict partial ordering $\succ$ on A, $M \gg N$ whenever there exist two multisets N' and M' on A, respectively, such that (i) $\emptyset \neq M' \subseteq M$, (ii) $N = (M \setminus M') \cup N'$, and (iii) for all $a \in N'$ there is a $b \in M'$ with $a \succ b$.

2.2 First-Order Terms and Equations

A *signature* Σ is a countably (infinite) set of *function symbols* where each $f \in \Sigma$ is associated with a non-negative integer n, the *arity* of f. We call a function symbol with arity 0 a *constant symbol*. For constant symbols we write lowercase letters a, b, and c whereas non-constant function symbols are denoted by lowercase letters f, g, and h. A function symbol with arity 1 is called *unary*. Throughout this work, we assume that any signature contains at least one constant symbol. Let X be countable infinite set of *variables* such that $\Sigma \cap X = \emptyset$. For variables we write lowercase letters x, y, and z.

We define (first-order) terms as usual built up from function symbols and variables. The set $\mathcal{T}(\Sigma, X)$ of all *terms* over Σ and X is inductively defined as (i) $X \subseteq \mathcal{T}(\Sigma, X)$ and (ii) for all $n \geq 0$ and all $f \in \Sigma$ where n is the arity of f, and all $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, X)$ we have that $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, X)$. Terms are denoted by lowercase letters s, t, u, v, and w. A term of the form $f(t_1, \ldots, t_n)$ with $n > 0$ is called *complex* whereas a term of the form a where a is a constant symbol is called *constant*.

The set of *positions* of the term t is a set *pos*(t) of strings over the alphabet of positive integers. The empty string is denoted by ϵ. We define *pos*(t) inductively as the smallest set such that (i) $\epsilon \in pos(t)$ and (ii) if t is of the form $f(t_1, \ldots, t_n)$ then $ip \in pos(t)$ for all i with $1 \leq i \leq n$ and for all $p \in pos(t_i)$. The function *size*:$\mathcal{T}(\Sigma, X) \rightarrow \mathbb{N}$ maps a term t to its *size* which is the cardinality of *pos*(t), i.e. $size(t) = |pos(t)|$. Given a position p, we define its *depth* (*length*) inductively by (i) $|\epsilon| = 0$ and (ii) $|p| = |q| + 1$ with $p = iq$ and i is a positive integer. The function *depth*:$\mathcal{T}(\Sigma, X) \rightarrow \mathbb{N}$ maps a term to its maximal depth, i.e. $depth(t) = \max(\{|p| \mid p \in pos(t)\})$.

Given a term t, for $p \in pos(t)$, the *subterm of t at position p*, denoted by $t|_p$, is defined inductively on the length of p by (i) $t|_\epsilon = t$ and (ii) $t|_{ip} = t_i|_p$ where t is of the form $f(t_1, \ldots, t_n)$ and $1 \leq i \leq n$. A subterm of t at position p is called *proper* if $p \neq \epsilon$. We write $t[u]_p$ if $t|_p = u$ and, ambiguously, by $t[v]_p$, we denote the term that is obtained from t by replacing the subterm u at position p by the term v. We may omit the explicit position p of u in t by writing $t[u]$ instead of $t[u]_p$. The function $vars{:}\mathcal{T}(\Sigma, X) \to 2^X$ maps a term t to the set of variables which occur in t, i.e. $vars(t) = \{x \in X \mid \text{there exists } p \in pos(t) \text{ with } t|_p = x\}$. Note that we may use *vars* with an arbitrary arity. We call $p \in pos(t)$ a *variable position* if $t|_p$ is a variable. A term t is called *ground* if $vars(t) = \emptyset$. The set of all ground terms over Σ is denoted by $\mathcal{T}(\Sigma)$.

A variable x which is a subterm of $t[x]_p$ at position p (the occurrence of a variable in $t[x]_p$ at position p) is called *linear* (in t) if x occurs only once in t, i.e. $t|_q \neq x$ for all $q \in pos(t)$ with $p \neq q$. A term t is called *linear* if every variable (occurrence) is linear in t. We define a *shallow term t* as a flat term of the form $f(x_1, \ldots, x_n)$ where all arguments are (shallow) variables, i.e. a term t is called *shallow* if (i) t is a variable or (ii) t is of the form $f(x_1, \ldots, x_n)$ where each x_i with $1 \leq i \leq n$ is a variable. In the literature, shallow terms have also been defined as terms in which variables may only occur as shallow variables, i.e. arbitrary ground terms are allowed in t. In our context, however, we demonstrate that this definition does not increase the expressiveness of the according (shallow) equational theories. Note that a shallow term may not be linear.

Semi-linearity generalizes the concept of non-linear shallow variables to an arbitrary depth of certain non-linear variable occurrences. Intuitively, viewing terms as trees, semi-linearity requires all occurrences of a non-linear variable in a term to occur at the same depth and the paths leading to the occurrences have to be labeled by the same function symbols.

Definition 2.2.1 (Semi-linear Term)
A term t is called *semi-linear* if (i) either t is a variable, or else (ii) t is of the form $f(t_1, \ldots, t_n)$ such that (ii.a) each t_i with $1 \leq i \leq n$ is semi-linear and (ii.b) $t_i = t_j$ for all i and j with $1 \leq i, j \leq n$ and $vars(t_i) \cap vars(t_j) \neq \emptyset$.

Any (non-linear) shallow term is semi-linear but not vice versa. A term t is called *pseudo-linear* if, for all variables $x \in vars(t)$, all occurrences of x in t occur at the same depth in t, i.e. $|p| = |q|$ for all $x \in vars(t)$ with $t[x]_p$ and $t[x]_q$. Any semi-linear term is pseudo-linear but not vice versa. Note that the intuitive generalization of pseudo-linearity over semi-linearity is that pseudo-linearity abstracts from the function symbols which occur on the path to the distinct non-linear occurrences of a variable. For instance, the term $f(g(x), g(x), h(y, y))$ is semi-linear and pseudo-linear but neither linear nor shallow whereas the term $f(g(x), g(x), h(x, y))$ is pseudo-linear but not semi-linear. The term $f(x, x, y)$ is shallow, semi-linear, and pseudo-linear but not linear.

A *substitution* is a function $\sigma{:}X \to \mathcal{T}(\Sigma, X)$ such that $\sigma(x) \neq x$ for only finitely many variables x. We denote substitutions by the lowercase greek letters σ and τ. The (finite) set of variables that σ does not map to themselves is called the *domain of σ*, denoted by $Dom(\sigma)$, i.e. $Dom(\sigma) = \{x \in X \mid \sigma(x) \neq x\}$. If $Dom(\sigma) = \{x_1, \ldots, x_n\}$ then we may write

σ as $\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$. The *codomain of* σ is given by $Cod(\sigma) = \{\sigma(x) \mid x \in Dom(\sigma)\}$. We say that σ *instantiates* x if $x \in Dom(\sigma)$. Given a substitution σ and a finite set of variables $Y \subseteq X$, $\sigma|_Y$ is defined as the substitution with $Dom(\sigma|_Y) = Dom(\sigma) \cap Y$ and, for all variables $x \in X$, $\sigma|_Y(x) = \sigma(x)$ if $x \in Y$ and $\sigma|_Y(x) = x$, otherwise. If Y is a singleton $\{x\}$ we may denote $\sigma|_{\{x\}}$ by $\sigma|_x$. Any substitution can be extended to a mapping $\sigma{:}\mathcal{T}(\Sigma, X) \to \mathcal{T}(\Sigma, X)$ by the usual homomorphic extension. The *composition* $\sigma\tau$ of two substitutions σ and τ is defined as $\sigma\tau(x) = \sigma(\tau(x))$. Given a term t, we may write $t\sigma$ instead of $\sigma(t)$. A term t is called an *instance* of a term s which is called a *generalization* of t if and only if there exists a substitution σ such that $s\sigma = t$.

An *equation*, denoted by $s \approx t$, is an unordered pair of terms s and t. We define $size(s \approx t)$ by $size(s) + size(t)$. For the straightforward extension of the other notions for terms to equations we assume that $\approx$ is a special binary symbol (with arity 2). In particular, an equation $s \approx t$ is called *linear* if both terms s and t are linear. Note that variables in a linear equation $s \approx t$ may occur in s and t. An equation $s \approx t$ is called *shallow* if both terms s and t are shallow. Similar to the linear case, shallow variables in a shallow equation $s \approx t$ may occur in s and t.

Definition 2.2.2 (Semi-linear Equation)
An equation $s \approx t$ is called *semi-linear* if (i) s and t are semi-linear, and (ii) if s is of the form $f(s_1, \ldots, s_n)$ then (ii.a) if t is a variable then for all i with $1 \leq i \leq n$ and $t \in vars(s_i)$ we have that $s_i = t$, or (ii.b) if t is of the form $g(t_1, \ldots, t_m)$ then $s_i = t_j$ for all i and j with $1 \leq i \leq n$, $1 \leq j \leq m$, and $vars(s_i) \cap vars(t_j) \neq \emptyset$.

Any (non-linear) shallow equation is semi-linear but not vice versa. An equation $s \approx t$ is called *pseudo-linear* if, for all variables $x \in vars(s, t)$, all occurrences of x in $s \approx t$ occur at the same depth in $s \approx t$, i.e. $|p| = |q|$ for all $x \in vars(s, t)$ with $(s \approx t)[x]_p$ and $(s \approx t)[x]_q$. Any semi-linear equation is pseudo-linear but not vice versa. For instance, the equations $f(g(x), g(x), y) \approx h(y, g(x), y)$ and $f(g(x), g(x), y) \approx y$ are semi-linear and pseudo-linear but neither linear nor shallow whereas $f(g(x), g(x)) \approx x$ is even not pseudo-linear.

An equation $s \approx t$ is called *collapsing* if either t is a variable, or else s is a variable. It is called *non-collapsing* if s and t are non-variable terms. It is called *universal* if s and t are variables. An equation $s \approx t$ is called *disjoint* if $vars(s) \cap vars(t) = \emptyset$. Otherwise, it is called *shared*.

2.3 First-Order Languages

Throughout this work, we use a *first-order language* $\mathcal{L}$ (with equality) where the alphabet of $\mathcal{L}$ consists of a countably infinite set of *predicate symbols*, a signature Σ, a countably infinite set of variables X, the *logical connectives* $\neg$, $\wedge$, and $\vee$, the *quantifiers* $\forall$ and $\exists$, and the symbol $\approx$. Predicate symbols are denoted by uppercase letters P, R, S, and T. Each predicate symbol P is associated with a non-negative integer n, the *arity* of P. We call a predicate symbol with arity 0 a *propositional variable* while a predicate symbol with arity 1 is called *monadic* or *unary*.

We define the *formulae* of $\mathcal{L}$ as usual, inductively, by (i) any *atom* $P(t_1, \dots, t_n)$ with P is a predicate symbol and $\{t_1, \dots, t_n\} \subseteq \mathcal{T}(\Sigma, X)$ is a formula, (ii) any equation $s \approx t$ with $s, t \in \mathcal{T}(\Sigma, X)$ is a formula, and (iii) if Φ and Ψ are formulae and $x \in X$ is a variable then $\neg\Phi$, $\Phi \wedge \Psi$, $\Phi \vee \Psi$, $\forall x \Phi$, and $\exists x \Phi$ are formulae. We denote atoms by the uppercase letters A and B. Note that, for simplicity, we may represent an atom A by an equation of the form $A \approx \top$ where $\top$ is a special constant symbol. We define the *size* of a formula Φ (Ψ), denoted by $size(\Phi)$, inductively, by $size(P(t_1, \dots, t_n)) = 1 + size(t_1) + \ldots + size(t_n)$, $size(s \approx t)$ as already defined above, $size(\neg\Phi) = size(\Phi)$, $size(\Phi \wedge \Psi) = size(\Phi) + size(\Psi)$, $size(\Phi \vee \Psi) = size(\Phi) + size(\Psi)$, $size(\forall x \Phi) = size(\Phi)$, and $size(\exists x \Phi) = size(\Phi)$. The universal and existential closure of a formula Φ which does not contain any quantifiers is the formula $\forall\, x_1, \dots, x_n\, (\Phi)$ and $\exists\, x_1, \dots, x_n\, (\Phi)$, respectively, where $\{x_1, \dots, x_n\} = vars(\Phi)$. We may also denote the closure by $\forall\Phi$ and $\exists\Phi$, respectively. A (*positive*) *literal* L is an atom A or an equation $s \approx t$ while a (*negative*) *literal* L is a negated atom $\neg A$ or a negated equation $\neg(s \approx t)$ (a *disequation* $s \not\approx t$). An atom (literal) $(\neg)A$ is called *monadic* if A is of the form $P(t)$ where P is a monadic predicate symbol.

A *clause* is a disjunction $L_1 \vee \ldots \vee L_n$ of literals with $n \geq 0$ where all variables in the clause are assumed to be universally quantified. A clause with $n = 1$ is called a *unit clause*. Clauses are denoted by the uppercase letters C and D. We may also use the multiset representation $\{L_1, \dots, L_n\}$ to denote a clause $L_1 \vee \ldots \vee L_n$. Another representation of a clause C is the sequent style notation $\Gamma \rightarrow \Delta$ where the *antecedent* Γ is the multiset of all negative literals of C and the *succedent* Δ is the multiset of all positive literals of C. We may also use the uppercase greek letters Λ and Π to denote the antecedent and succedent of a clause, respectively. Note that an antecedent Γ is (implicitly) assumed to be the conjunction of the literals in Γ whereas a succedent Δ can be seen as a disjunction of the literals in Δ. A clause of the form $\rightarrow \Delta$ is called *positive* and a clause of the form $\Gamma \rightarrow$ is called *negative*. A formula which is a conjunction of clauses is said to be in *clausal normal form*. A *set of clauses* is (implicitly) assumed to be the conjunction of the clauses in the set. Clause sets are denoted by the uppercase letters N and M.

2.4 First-Order Structures

A *model* for formulae of the first-order language $\mathcal{L}$ (with equality) consists of a non-empty set $\mathcal{D}$, called the *domain*, and a mapping $\mathcal{I}$, called an *interpretation*, that associates to each function symbol $f \in \Sigma$ with arity n a function $f^{\mathcal{I}}{:}\mathcal{D}^n \rightarrow \mathcal{D}$, to each predicate symbol P with arity n a relation $P^{\mathcal{I}} \subseteq \mathcal{D}^n$, and to $\approx$ a binary relation on $\mathcal{D}^2$. An interpretation $\mathcal{I}$ is said to *interpret* a function symbol, a predicate symbol, or $\approx$ by the associated function or relation, respectively. We call $\mathcal{I}$ an *equality interpretation* if $\approx$ is interpreted by $\mathcal{I}$ as a congruence relation. In this case, the model and $\mathcal{I}$ are called an *equality model* and an *equality interpretation*, respectively.

We define Herbrand interpretations, as usual, on the domain $\mathcal{T}(\Sigma)$ where the function symbols "are interpreted by themselves". Note that we explicitly define the notions of truth (validity, satisfiability) only for (sets of) clauses with respect to Herbrand interpretations. The extensions to formulae and (non-Herbrand) interpretations are assumed as usual. A *Herbrand interpretation* I is a set of ground atoms and ground equations. We may also denote Herbrand interpretations by the uppercase letter J. For any predicate symbol P, we define $I(P) = \{(t_1, \dots, t_n) \mid P(t_1, \dots, t_n) \in I\}$. Similarly as above, a Herbrand interpretation is called an *equality (Herbrand) interpretation* if $\approx$ is interpreted as a congruence relation.

Given a (equality) Herbrand interpretation I, a ground atom A (a ground equation $s \approx t$, a ground (positive) literal L of the form A $(s \approx t)$) is called *true* in I if A $(s \approx t)$ is in I, and is called *false* in I, otherwise. A ground (negative) literal L of the form $\neg A$ $(\neg(s \approx t))$ is true [false] in I if and only if A $(s \approx t)$ is false [true] in I. A ground clause is called *true* in I if one of its literals is true in I, and is called *false* in I, otherwise. If a ground expression (atom, equation, literal, clause) E is true in I we write $I \models E$ and also say that I *satisfies* E. A (equality) Herbrand interpretation I is said to satisfy a non-ground clause if it satisfies all its ground instances. A clause is called a *tautology* if it is true in any (Herbrand) interpretation.

Note that the definition of Herbrand interpretations is fixed for the domain $\mathcal{T}(\Sigma)$ which implies that the notion of a Herbrand model coincides with Herbrand interpretations. We may therefore reuse the notion of a model to call a Herbrand interpretation, though ambiguously, a *model* of a set N of clauses if it satisfies all clauses in N. In the sequel, by a model we mean a Herbrand interpretation in the above sense, if not stated otherwise. A set N of clauses (with equality) is called *consistent* or *satisfiable* if N has a (equality) model, and it is called *inconsistent* or *unsatisfiable*, otherwise. The *entailment problem* for a set N of clauses (with equality) is to decide for any ground clause C if C is true in every (equality) model of N, written $N \models C$. We say that N *entails* C or that C is *valid in* N whenever $N \models C$.

2.5 Equational Theories and Term Rewriting

A binary relation $\Rightarrow$ on $\mathcal{T}(\Sigma, X)$ is called *closed under substitutions* if and only if $s \Rightarrow t$ implies that $s\sigma \Rightarrow t\sigma$ for all terms s and t and all substitutions σ. It is called *closed under context application* if and only if $s \Rightarrow t$ implies that $u[s]_p \Rightarrow u[t]_p$ with $p \in pos(u)$ for all terms s, t, and u. A binary relation $\Rightarrow$ on $\mathcal{T}(\Sigma, X)$ is called a *rewrite relation* if it is closed under substitutions and context application. As usual, $\overset{+}{\Rightarrow}$ denotes the transitive closure and $\overset{*}{\Rightarrow}$ denotes the reflexive transitive closure of $\Rightarrow$. Moreover, $\Leftrightarrow$ denotes the symmetric closure and $\overset{*}{\Leftrightarrow}$ denotes the reflexive symmetric transitive closure of $\Rightarrow$.

An *equational theory* $\mathcal{E}$ is a finite set of equations which are implicitly assumed to be universally quantified. $\mathcal{E}$ is called *shallow* (*linear*, *semi-linear*) if the equations in $\mathcal{E}$ are shallow (linear, semi-linear). The *reduction relation* $\Rightarrow_{\mathcal{E}} \subseteq \mathcal{T}(\Sigma, X) \times \mathcal{T}(\Sigma, X)$ (with respect to $\mathcal{E}$) is defined as $u \Rightarrow_{\mathcal{E}} v$ if and only if there exists an equation $s \approx t \in \mathcal{E}$, a position $p \in pos(u)$, and a substitution σ such that $u|_p = s\sigma$ and $v = u[t\sigma]_p$. Two terms u and v are called *unifiable with respect to an equational theory* $\mathcal{E}$ if and only if there exists a substitution σ such that $u\sigma \overset{*}{\Leftrightarrow}_{\mathcal{E}} v\sigma$. We call the problem of deciding whether two terms are unifiable with respect to an equational theory $\mathcal{E}$ the *E-unifiability problem* (with respect to $\mathcal{E}$). The *word problem* (with respect to $\mathcal{E}$) is a particular instance of the E-unifiability problem where only ground terms u and v are considered. By Birkhoff's Theorem we have that $\mathcal{E} \models u \approx v$ if and only if $u \overset{*}{\Leftrightarrow}_{\mathcal{E}} v$. Note that each equation $s \approx t$ in $\mathcal{E}$ may be represented by a clause $\rightarrow s \approx t$ in a clause set N. Then u and v are unifiable with respect to $\mathcal{E}$ if and only if the set $N \cup \{u \approx v \rightarrow\}$ of clauses is unsatisfiable.

A *rewrite rule*, denoted by $s \Rightarrow t$, is an ordered pair of terms s and t. A *term rewriting system* R (or *rewrite system*) is a finite set of rewrite rules. The *reduction relation* $\Rightarrow_R \subseteq \mathcal{T}(\Sigma, X) \times \mathcal{T}(\Sigma, X)$ (with respect to R) is defined as $u \Rightarrow_R v$ if and only if there exists a rewrite rule $s \Rightarrow t \in R$, a position $p \in pos(u)$, and a substitution σ such that $u|_p = s\sigma$ and $v = u[t\sigma]_p$ where $s\sigma$ is called a *redex* and p is called a *redex position*. We say that u *rewrites to* v (with respect to R) if $u \Rightarrow_R v$. A term that cannot be rewritten (with respect to R) is called *irreducible* (with respect to R), and *reducible*, otherwise. A term t' is called a *normal form of* t (with respect to R) if $t \overset{*}{\Rightarrow}_R t'$ and t' is irreducible with respect to R. Note that $\Rightarrow_R$ is a rewrite relation.

A strict partial ordering $\succ$ on $\mathcal{T}(\Sigma, X)$ has the *subterm property* if $t \succ t|_p$ for all terms $t \in \mathcal{T}(\Sigma, X)$ and all positions $p \in pos(t) \setminus \{\epsilon\}$. A *reduction ordering* is a well-founded strict partial ordering that is also a rewrite relation. A rewrite system R is called *terminating* if $\Rightarrow_R \subseteq \succ$ where $\succ$ is a reduction ordering. We call R *confluent* whenever a term s can be rewritten to terms t and u, i.e. $s \overset{*}{\Rightarrow}_R t$ and $s \overset{*}{\Rightarrow}_R u$, then t and u are *joinable* (with respect to R), denoted by $t \Downarrow_R u$, for some term v, i.e. $t \overset{*}{\Rightarrow}_R v$ and $u \overset{*}{\Rightarrow}_R v$. The relation $t \Downarrow_R u$ indicates that the equation $t \approx u$ *converges* or has a *rewrite proof* (with respect to R). A terminating and confluent rewrite system is called *convergent*. Given an equational theory $\mathcal{E}$, we call a convergent rewrite system R *complete for* $\mathcal{E}$ if (i) the rewrite rules of R are contained in $\overset{*}{\Leftrightarrow}_{\mathcal{E}}$ (soundness) and (ii) the equations of $\mathcal{E}$ are contained in $\overset{*}{\Leftrightarrow}_R$ (adequacy). If R is complete for $\mathcal{E}$ then $\overset{*}{\Leftrightarrow}_{\mathcal{E}} = \overset{*}{\Leftrightarrow}_R$ which implies that R decides the E-unifiability problem with respect to $\mathcal{E}$. Note that convergent rewrite systems define unique normal forms and thus, for convergent rewrite system, two terms are equivalent if and only if they rewrite to the same normal form.

We assume that *critical pairs* are defined as usual. A rewrite rule $s \Rightarrow r$ is called *left-linear* (*right-linear*) if s is linear (t is linear). A rewrite system is called *left-linear* (*right-linear*) if all of its rewrite rules are left-linear (right-linear). A rewrite system is called *orthogonal* if it is left-linear and has no critical pairs. Note that any orthogonal rewrite system is confluent.

Chapter 3

Soft Typing

Throughout this work we consider *clausal inference systems* which are based on the particular data structure of clauses. Any first-order formula can be transformed to the so-called *clausal normal form* which is a conjunction of clauses. The clausal normal form transformation preserves satisfiability. Anything from naive to sophisticated clausal normal form transformation can be found in the literature. Clausal normal forms are represented by sets of clauses. We use the following notions and notations for clausal inference systems.

Definition 3.0.1 (Clausal Inference System)
A (*clausal*) *inference scheme* π of the form

$$\text{Infer}\frac{C' \quad C}{D}$$

with the *main premise* C, the *side premise* C', and the *conclusion* D is a relation $\mathcal{N}\times\mathcal{N}\times\mathcal{N}$ on sets $\mathcal{N}$ of clauses where C', C, and D are clause schemes. Ambiguously, a (*clausal*) *inference* π is a tuple (C', C, D) of clauses C', C, and D, which are instantiations of the main and side premises and the conclusion, respectively, of an inference scheme π. The conclusion of an inference π is also denoted by $\mathcal{C}(\pi)$. Unary and n-ary inferences (schemes) are defined analogously. A *clausal inference system* Π is a set of inference schemes. Let N be a set of clauses. The set $\Pi(N)$ denotes the subset of all inferences in Π with premises in N. The set of all conclusions from premises in N by inferences in Π is denoted by $\mathcal{C}(\Pi(N))$.

Note that we abbreviate the term (clausal) inference scheme to (clausal) inference. The operational behavior of an inference from premises in N is that the conclusion of the inference is added to N. A clausal inference system induces a relation on sets of ground clauses. In other words, an inference, which is derived in some clause set, will also be derived in any other clause set which contains the same premises. This implies that any system of this kind describes an inherently *local* form of theorem proving processes in the sense that inferences are drawn independently from the *global* context of the particular premises.

Soft typing is a general framework that extends clausal inference systems by additional parameters such that global aspects of a problem set may yield better control of the inference process. By soft typing for a particular inference system Π, we mean a *refinement* of Π with respect to a particular context (clause set) N such that the refinement contains less inferences from N than Π. The context N is evaluated by semantic considerations with respect to certain *model hypotheses* of N. We obtain soft typing for semantic and ordered resolution, c.f. Section 3.1.1 and 3.1.2, and for superposition as demonstrated in Section 3.1.3. Note that a refinement of Π is not a clausal inference system in general, as some restrictions of the refinement may involve the global context of the particular problem. In order to elaborate on the fundamental ideas of soft typing we recall the basic concepts of resolution and introduce soft typing for unrefined resolution.

In the sequel, we assume the ground level as default. The *lifting problem*, i.e. the problem of finding a suitable approximation of the ground level on the non-ground level, is of quite some importance to obtain an effective theorem proving procedure for first-order logic. We discuss the lifting problem in Section 3.5. In the other parts of this chapter, however, we are interested in properties of the calculi and, in particular, of soft typing which are independent from these issues.

3.1 Introduction to Soft Typing

The *resolution calculus* R or simply resolution was invented by Robinson (1965). The resolution calculus is sound and refutationally complete. A calculus (inference system) is *sound* if any inference of the calculus (inference system) is sound, i.e. any conclusion of a sound inference is a logical consequence of the premises. A calculus is *refutationally complete* if, in any (fair) *derivation* by the inferences of the calculus from an unsatisfiable formula, a contradiction is eventually derived. See Section 3.4 for a formalization of the notion of a (fair) derivation.

A calculus is called *complete* if any logical consequence of the problem can be derived by the inferences of the calculus. A complete calculus is refutationally complete whereas, in general, the opposite is not true. However, to demonstrate the unsatisfiability of a formula refutational completeness is sufficient. An example for a refutationally complete calculus which is not complete is the resolution calculus R. We define R on the ground level. It consists of the *Resolution* inference rule and the *Factoring* inference rule.

Definition 3.1.1 (Resolution)
The following inference is called *Resolution*:

$$\text{Infer}\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$

where (i) $C \vee A$ and $D \vee \neg A$ are ground clauses and A is a ground atom.

We call the side premise $C \vee A$ and the main premise $D \vee \neg A$ the *positive* and the *negative* premise, respectively, of the inference and the conclusion $C \vee D$ the *resolvent*.

Definition 3.1.2 (Factoring)
The following inferences are called *Factoring*:

$$\text{Infer}\frac{C \vee \neg A \vee \neg A}{C \vee \neg A}$$

where (i) $C \vee \neg A \vee \neg A$ is a ground clause and A is a ground atom; and

$$\text{Infer}\frac{C \vee A \vee A}{C \vee A}$$

where (i) $C \vee A \vee A$ is a ground clause and A is a ground atom.

The first rule is called *Negative Factoring* whereas the second rule is called *Positive Factoring*. In order to demonstrate the refutational completeness of a clausal inference system, we may abstract from the notion of a derivation. It is actually sufficient to look at the result of a derivation.

Definition 3.1.3 (Saturation)
Let Π be a clausal inference system. A set N of clauses is called *saturated* with respect to Π, if N is closed under Π, i.e. if $\mathcal{C}(\Pi(N)) \subseteq N$.

A clausal inference system Π is refutationally complete if, for any set N of clauses which is saturated with respect to Π, either N is satisfiable, or else N contains a contradiction. In clausal theorem proving, the *empty clause* is the witness for a contradiction. The empty clause represents the empty disjunction and is therefore always false. From an operational point of view, the process of exhaustively applying inference rules is called *saturation*. In this way, Π provides a *saturation-based theorem proving method*.

The following analysis of resolution is an easy exercise which already contains the main ideas of proofs of the refutational completeness of general clausal inference systems. The idea is based on a particular binary tree called the *semantic tree*. It goes back to a well-known proof by Robinson (1965).

Theorem 3.1.4 (Soundness and Completeness of Resolution)
The resolution calculus is sound and refutationally complete.

Proof Let N be an unsatisfiable set of ground clauses which is saturated with respect to R. The Herbrand base of N is any countable set $\{A_1, A_2, A_3, \dots\}$ of ground atoms. A semantic tree $\mathcal{T}$ is a labeled binary tree where the root node has no label and otherwise at level n all left children are labeled A_n and all right children are labeled $\neg A_n$. Depending on the order in which the atoms are listed, there are many semantic trees. To identify a unique tree, we may assume some fixed total ordering on the ground atoms with A_1 being the minimal object. Note that the ordering is well-founded. We observe that each branch θ of $\mathcal{T}$ completely determines a Herbrand interpretation I by assuming that L is true in I for all L on θ. Each ground atom in the Herbrand base or its negation appears on θ, but not both.

A *path* in a tree is a sequence of nodes starting at the root node and proceeding from parent to child. Maximal paths are called *branches*. A path θ in $\mathcal{T}$ *contradicts* a clause C if, for each literal L in C, the negation of L appears on θ. A path is N*-closed* if θ contradicts some clause in N. A node l is called a *failure node for* N if the path to l is N-closed.

The construction is intended to show that the root node of $\mathcal{T}$ is a failure node for N. This implies that the empty clause is in N since the trivial path from the root node to itself only contradicts the empty clause. Every branch of $\mathcal{T}$ is N-closed, for otherwise there would be a branch θ of $\mathcal{T}$ that induced a Herbrand model of N. Let θ be a branch of $\mathcal{T}$ and let C be a clause in N that θ contradicts. Since C is finite, there must be some finite initial segment of θ that contradicts C. Thus every branch in $\mathcal{T}$ has a finite initial N-closed segment that contains a failure node for N. Let $\mathcal{T}^*$ be the subtree of $\mathcal{T}$ in which every descendant of a failure node has been deleted. In order to show that $\mathcal{T}^*$ is the trivial tree, consisting of just the root node, we suppose otherwise.

Since, by König's Lemma, $\mathcal{T}^*$ is finite, there is a maximal branch θ in $\mathcal{T}^*$ of the form $\theta' \circ A$ where A is, without loss of generality, a left child and θ' is the path to the parent of A. We assume that A also denotes the according ground atom in the Herbrand base. Note that A is a failure node and θ is N-closed but θ' is not. Let $\neg A$ be the right sibling of A in $\mathcal{T}^*$ where $\neg A$ is also a failure node, for otherwise the shortest N-closed path beginning with $\theta' \circ \neg A$ would be longer than θ. Since θ' is not N-closed, $\theta' \circ \neg A$ is an N-closed branch of $\mathcal{T}^*$.

Let C' and C be some clauses in N contradicted by $\theta' \circ \neg A$ and $\theta' \circ A$, respectively. Then A must occur positively in C', for otherwise θ' would already contradict C'. Likewise $\neg A$ must occur in C. We call this situation a *local contradiction*. The Resolution rule is a method to *resolve* this local contradiction. It follows that θ' already contradicts the resolvent D. We may assume that Factoring has been exhaustively applied on the occurrences of A in both clauses. However, since N is saturated with respect to R, D must be in N which implies that θ' is N-closed. This is a contradiction to the assumption that $\mathcal{T}^*$ is non-trivial. ■

The proof shows which properties of the calculus are important to obtain refutational completeness, i.e. (i) if an unsatisfiable clause set does not yet contain the empty clause, then there must be some local contradiction such that Resolution is applicable, and (ii) the conclusion must be in some sense simpler in order to ensure that the empty clause is eventually derived. In general, a calculus that enjoys these two properties is refutationally complete. Note that the semantic tree approach achieves (ii) indirectly by an ordering on the Herbrand interpretations induced by the semantic tree (the maximal branch). However, the approach is limited as the incorporation of arbitrary interpretations is difficult. Not only semantic resolution but also ordered resolution, as they both deal with particular model constructions, require a method which is free of the particular choice of interpretations and whose ordering concept to show termination is independent from the construction of the interpretation. In the sequel, we assume that an arbitrary but fixed well-founded and total ordering on ground clauses is given. The smallest clause is the empty clause. The clause ordering is an alternative to the previous ordering on Herbrand interpretation to achieve a termination argument.

The notion of a local contradiction can be generalized to the notion of a *counterexample* for some model such that situations like a factor or, for equational reasoning, a redex in a positive literal are also covered.

Definition 3.1.5 (Counterexample)
Let N be a set of ground clauses and let C be a clause in N. If C is false in an interpretation I we call it a *counterexample* for I. A counterexample C is called *minimal in N* if C is the smallest counterexample in N with respect to some well-founded and total ordering on ground clauses.

We will later see that a so-called admissible clause ordering as given in Definition 3.1.15 is the intended well-founded and total ordering on ground clauses. A *candidate model* can be viewed as representing a model hypothesis for N that is based on the currently available knowledge about N. If a clause is false in the model, more inferences are required. Conversely, clauses which have the appropriate truth value need not be considered for inferences.

Definition 3.1.6 (Candidate Model)
Let N be a set of ground clauses which does not contain the empty clause. Let I be a mapping that assigns to N an (equality) Herbrand interpretation I^N. We call I a *model functor* and I^N the *candidate model for N*.

The *reduction property* of a clausal inference system Π generalizes the crucial properties (i) and (ii) such that, for Π to be refutationally complete, there must always be an inference in Π that reduces at least the minimal counterexample (local contradiction) to a smaller counterexample (smaller falsifying branch interpretation). An inference whose conclusion is always in some sense smaller than at least its main premise is called *monotone*. If N is a set of ground clauses and C a ground clause (not necessarily in N), N_C denotes the set of clauses D in N such that D is smaller than C with respect to a well-founded and total ordering on ground clauses.

Definition 3.1.7 (Reduction Property)
Let Π be a clausal inference system. We say that Π has the *reduction property for counterexamples* (with respect to a model functor I) if, for all sets N of ground clauses and for all minimal counterexamples C for I^N in N, there exists an inference in Π with main premise C, side premise C' which is true in I^N and a counterexample for $I^{N_{C'}}$, and a conclusion D which is a smaller counterexample for I^N than C with respect to a well-founded and total ordering on ground clauses. We call an inference from C to D a *reduction inference* and we say C is *reduced* to D. An inference from C' and C to D where C is a counterexample for I^N in N, but which may not be a minimal counterexample, is called a *weak reduction inference*.

The proposition below states that a clausal inference system which enjoys the reduction property is indeed refutationally complete. Note that the proof, which implicitly involves a Noetherian induction, does not exploit the requirement that a side premise C' which is involved in the reduction of a counterexample for I^N in N is a counterexample for $I^{N_{C'}}$. However, the inference systems which are shown to be compatible with soft typing match this requirement. We will later see that soft typing makes particular use of this aspect.

Proposition 3.1.8 (Bachmair & Ganzinger (1997, 1998*a*))
Let N be a set of ground clauses and let Π be a clausal inference system which has the reduction property. Suppose that N is saturated with respect to Π. Then N is either satisfiable, or else N contains the empty clause.

Proof Suppose that Π has the reduction property for counterexamples with respect to some model functor I. If the empty clause is not in N, either I^N is a model of N, or else N contains a minimal counterexample for I^N in N. The latter case is impossible. Otherwise, by the reduction property, C can be reduced to a smaller counterexample D, which, if $\mathcal{C}(\Pi(N)) \subseteq N$, must be contained in N, and hence would contradict the minimality of C. ■

The reduction property of a clausal inference system Π describes sufficient requirements of Π to be a refutationally complete theorem proving method. It guarantees that there are always enough inferences in Π to reduce at least the minimal counterexamples of a particular candidate model. Thus we also call a reduction inference *indispensable*. On the other hand, we may abstract from the minimality of counterexamples and call an inference, which is not a weak reduction inference, *dispensable*. Note that a weak reduction inference π from a clause C, which is not a reduction inference, is also not needed for refutational completeness. Weak reduction inferences can be avoided by a preference on small clauses with respect to the clause ordering.

The motivation of a refinement of Π, in the sense of soft typing, is to single out dispensable inferences in Π. In general, however, not all dispensable inferences can be avoided. In fact, the set of inferences in any effective inference system must be an approximation of the set of reduction inferences since the identification of counterexamples is undecidable. However, the permanent exclusion of dispensable inferences is not possible since, in general, the status (context) of clauses may change in a theorem proving derivation. In other words, the notion of being a counterexample is not stable in a derivation. However, dispensable inference may be *blocked* dynamically.

Definition 3.1.9 (Blocked Inference)
Let I be a model functor and let N be a set of ground clauses. We say that an inference with a ground clause C as main premise and a ground clause C' as side premise is *blocked in* N, if (i) C' is a counterexample for I^N, or (ii) C' is true in $I^{N_{C'}}$, or (iii) C is true in I^N.

We define soft typing for a clausal inference system Π operationally as an upper approximation of the reduction inferences of Π from premises in a clause set N. From a denotational point of view, soft typing for Π is not a relation on clause sets in the sense of clausal inference systems, since the problem set N and the model functor I are involved as additional parameters. Consequently, by soft typing for Π we mean a certain relation on sets of premises and conclusions (clauses), a set of contexts (clause sets), and a model functor I.

Definition 3.1.10 (Soft Typing)
Let N be a set of ground clauses and let Π be a clausal inference system. The refinement of Π with respect to N which contains only the non-blocked inferences in Π is called *soft typing for* Π.

The process of an exhaustive application of soft typing for Π to a clause set is called *weak saturation.*

Definition 3.1.11 (Weak Saturation)
Let N be a set of ground clauses and let Π be a clausal inference system. The set N is called *weakly saturated* with respect to Π, if N is closed under the soft typing for Π.

Note that N is weakly saturated with respect to Π if and only if N is saturated with respect to the soft typing for Π. The corollary below is an immediate consequence of Proposition 3.1.8.

Corollary 3.1.12
Let N be a set of ground clauses and let Π be a clausal inference system which has the reduction property. Suppose that N is weakly saturated with respect to Π. Then N is either satisfiable, or else N contains the empty clause.

In the following, we apply the concept of the reduction property for counterexamples to obtain an alternative proof of refutational completeness of resolution. The construction is based on a particular model functor which, as we will see later, essentially carries over to the proof of refutational completeness of ordered resolution. The parameters of the model functor are certain *admissible orderings* on atoms, literals, and clauses. We define an admissible atom ordering as a well-founded and total ordering on atoms and show in which way admissible atom orderings can be extended to admissible orderings on literals and clauses.

Definition 3.1.13 (Admissible Atom Ordering)
An *atom ordering* $\succ_a$ is a strict partial ordering on atoms. $\succ_a$ is called *admissible* if $\succ_a$ is well-founded, and total on ground atoms. A ground atom A is called [strictly] maximal with respect to a multiset of literals Γ, if for any L in Γ with $L = (\neg)B$ we have $A \succeq_a B$ $[A \succ_a B]$.

Definition 3.1.14 (Admissible Literal Ordering)
Let $\succ_a$ be an admissible atom ordering. An ordering $\succ_l$ on literals is called a *literal ordering*. $\succ_l$ is called *admissible* if (i) $\succ_l$ is well-founded, (ii) the restriction of $\succ_l$ to ground literals is a total ordering, (iii) if $L = (\neg)A$ and $L' = (\neg)A'$ are ground literals, then $L \succ_l L'$, whenever (iii.1) $A \succ_a A'$, or (iii.2) $A = A'$ and the literal L is negative, whereas L' is positive.

Note that for any positive literal A, $\neg A$ is greater than A, whereas $\neg A$ is smaller than B if B is a positive literal with $B \succ_a A$. The multiset extension of an admissible literal ordering induces an ordering on ground clauses.

Definition 3.1.15 (Admissible Clause Ordering)
Let $\succ_l$ be an admissible literal ordering. A *clause ordering* $\succ_c$ is the multiset extension of $\succ_l$. We call $\succ_c$ *admissible* if $\succ_l$ is admissible.

Admissible clause orderings are well-founded and total on ground clauses. Moreover, an admissible clause ordering is compatible with the underlying atom ordering. If the maximal atom in a ground clause C is greater than the maximal atom in a ground clause D then $C \succ_c D$. We shall consider only admissible orderings from now on. We say that a clause $C \vee A$ is *reductive for* the atom A, if A is a strictly maximal atom with respect to C. Given a set of ground clauses N, we use induction with respect to $\succ_c$ to define a model functor I for resolution on N as follows.

Definition 3.1.16 (Model Functor for Resolution)
Let $\succ_c$ be an admissible clause ordering. Let N be a set of ground clauses and let C be a clause in N. Suppose that ε_D and I_D^N have been defined for all clauses D for which $C \succ_c D$. Then

$$I_C^N = \bigcup_{C \succ_c D} \varepsilon_D$$

where

$$\varepsilon_C = \{A\}$$

if (i) $C = C' \vee A$ is reductive for A and (ii) C is false in I_C^N. Otherwise, $\varepsilon_C = \emptyset$. I_C^N is called the *partial interpretation up to C*. We say that a clause C *produces* the atom A if $\varepsilon_C = \{A\}$ and call C a *productive clause*. The *model functor for resolution* I assigns to N the Herbrand interpretation $I^N = \bigcup_{C \in N} \varepsilon_C$.

Note that for any clause C the interpretations I_C^N, I^{N_C} and $\bigcup_{D \in N_C} \varepsilon_D$ coincide. The construction is designed to render the formulas of N true in I^N. The partial interpretation I_C^N is intended to be a model of the set N_C of those clauses in N that are smaller than C. The interpretation ε_C is meant to be a minimal extension of I_C^N that makes C true.

Lemma 3.1.17 (Bachmair & Ganzinger (1994))
Let $\succ_c$ be an admissible clause ordering. Let I be the model functor for resolution and let N be a set of ground clauses. Let C be a clause $\Gamma, B \rightarrow \Delta$ in N and let D be a clause in N with $D \succeq_c C$. If B is false in I_C^N then it is also false in I_D^N and I^N.

Proof Suppose that B is false in I_C^N. Suppose that there is an atom A which is true in $I^N \setminus I_C^N$. Then $A \succ_a B$ since clauses which produce atoms smaller than B are also smaller than C. This implies that B is false in I_D^N and I^N. ■

Recall the proof of the refutational completeness of resolution. The crucial Resolution step involves the two clauses C' and C where A is the maximal literal in C' and C, respectively, with respect to the fixed atom ordering. Similarly, Factoring is also required only on A and $\neg A$, respectively. Hence, Resolution as well as Factoring may be restricted such that only maximal literals are resolved and factorized with respect to some well-founded total atom ordering.

There is also another obvious improvement of resolution. Only one of the two Factoring rules is actually required for refutational completeness. Suppose that we skip Negative Factoring. That is, only positive literals are factorized. However, since occurrences of literals have to be factorized only if they are also involved in local contradictions, Positive Factoring and Resolution serve for sufficient factorization of negative literals. We can actually prove that resolution has the reduction property for counterexamples of I^N even under both improvements. This implies that soft typing for resolution with these improvements is refutationally complete.

Lemma 3.1.18
Let I be the model functor for resolution. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then there exists an inference in R from C such that

1. the conclusion is a smaller counterexample for I^N than C; and
2. in case of a Resolution inference, C is the main premise and the side premise is a productive clause.

Proof Suppose that C contains a negative literal A which is maximal in C, i.e. C is of the form $C = A, \Lambda \rightarrow \Pi$. Since $I^N \not\models C$ we have that $A \in I^N$, $\Lambda \subseteq I^N$, and $\Pi \cap I^N = \emptyset$. Let C' be a reductive clause of the form $\Gamma \rightarrow \Delta, A$ that produces A. Note that C' is true in I^N. Thus there is a Resolution inference between C and C' where D is the conclusion $\Gamma, \Lambda \rightarrow \Delta, \Pi$. It follows that D is smaller than C with respect to $\succ_c$ since the maximal literal in C has been resolved. Since C' is false in $I^{N_{C'}}$ we have that $\Gamma \subseteq I^{N_{C'}} \subseteq I^N$ and $\Delta \cap I^{N_{C'}} = \emptyset$. By Lemma 3.1.17 the interpretation I^N and the partial interpretation $I^N_{C'}$ coincide for atoms which are smaller than A which implies that $\Delta \cap I^N = \emptyset$ and therefore that D is a counterexample for I^N.

Suppose that A is a maximal atom in C such that $C = \Lambda \rightarrow \Pi, A, A$. If C contains only one occurrence of A, then C would be a productive clause which contradicts the assumption that C is a counterexample for I^N. Thus there is a Positive Factoring inference on C where D is the conclusion $\Lambda \rightarrow \Pi, A$. Again, D is a counterexample for I^N and D is smaller than C with respect to $\succ_c$. ■

Throughout this work, we use the concept of the reduction property to analyze and to prove the refutational completeness of the refinements by soft typing. The most important parameter is the model functor which has to be defined in a suitable manner for each refinement. However, soft typing as an undecidable concept demands decidable approximations of blocking, possibly with additional properties, e.g., stability throughout derivations for the permanent exclusion of blocked inferences. Recall that the notion of being a counterexample is, in general, not stable in a derivation. The abstract notion of *redundancy* of clauses proposed by Bachmair & Ganzinger (1994) is a concept for the approximation of permanent non-counterexamples. More precisely, it is a sufficient criterion for a clause such that the clause cannot become a minimal counterexample in any derivation. Note that this point of view is different to the original motivation of redundancy. In (Bachmair & Ganzinger 1994), redundancy has been introduced as an approximation of the set of clauses which do not contribute to the construction of the candidate model.

Definition 3.1.19 (Redundant Clause)
Let N be a set of ground clauses. A ground clause C is called *redundant in* N if $N_C \models C$.

Note that the clause C is not necessarily a member of N. $N_C \models C$ means that C is entailed by the members of N which are smaller than C with respect to a well-founded and total ordering on ground clauses. Suppose that C is a clause which is redundant in N. Then C can never be a minimal counterexample for any candidate model I^N since, by the minimality of C, I^N would be a model of N_C and thus also a model of C. Suppose that N' is the set of all non-redundant clauses in N. Then we also have that $N'_C \models C$, i.e. there are always non-redundant clauses in N_C which already entail C. It follows that C can also not be a minimal counterexample for any model of N'_C. This fact implies that redundancy is stable in any derivation. Redundancy is independent from the particular model of N_C, i.e. a redundant clause can never be a minimal counterexample for any model functor.

So far, redundancy provides a filter of clauses in a set N of ground clauses which are involved as main premises in some inference. On the other hand, a clause C' which is involved as a side premise is true in some candidate model I^N. However, C' is a counterexample for the candidate model $I^{N_{C'}}$. Suppose that Π is a clausal inference system which has the reduction property. If we can show that any clause C', which is involved as the side premise in a reduction inference in Π, is actually a minimal counterexample for the candidate model $I^{N_{C'}}$, then redundancy applies also to side premises. In other words, in this case we can prove that $I^{N_{C'}}$ is a model of $N_{C'}$. Using the model functor for resolution we may strengthen Lemma 3.1.18 in this regard.

Lemma 3.1.20
Let I be the model functor for resolution and let $\succ_c$ be the according admissible clause ordering. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then $I^{N_{C'}} \models N_{C'}$ for all clauses C' in $N_C \cup \{C\}$.

Proof Let C' be a clause in $N_C \cup \{C\}$. In order to show that $I^{N_{C'}}$ is a model of $N_{C'}$ suppose otherwise that there is a clause D of the form $\Gamma \to \Delta$ in $N_{C'}$ such that $I^{N_{C'}} \not\models D$, i.e. $\Gamma \subseteq I^{N_{C'}}$ and $\Delta \cap I^{N_{C'}} = \emptyset$, and, in particular, $\Gamma \subseteq I^{N_D}$ and $\Delta \cap I^{N_D} = \emptyset$. Since C is the minimal counterexample in N for I^N and $C \succ_c D$ we have that D is true in I^N. Thus there is a clause D' in N where $D' \succ_c D$ that produces an atom A in Δ into I^N. It follows that D is also reductive for A for otherwise $D \succ_c D'$. However, then D would be productive and thus true in $I^{N_{C'}}$ which shows the contradiction. ■

We will later see that this property does not hold for the model functor I of ordered resolution with selection used in the original work on soft typing for ordered resolution (Ganzinger et al. 1997). Using I, ordered resolution with selection where redundancy is imposed on main and side premises is refutationally complete unless dispensable inferences are blocked. This implies that ordered resolution with selection and redundancy requires dispensable inferences from non-counterexamples, c.f. Example 3.1.39. In this way, we arrive at a trade-off of soft typing in combination with redundancy. Soft typing has been designed as a general framework for the optimization of proof search by the incorporation of the global aspects of a problem set. However, the incompatibility can be fixed in two different ways, either by an appropriate modification of the clause ordering, or else by a modification of the model functor. We will discuss both solutions in detail in Section 3.1.2.

The following definition introduces the *strong reduction property* for clausal inference systems which essentially is the reduction property with the additional minimality requirement on side premises. Using the above lemma, we obtain that unrefined resolution has the strong reduction property.

Definition 3.1.21 (Strong Reduction Property)
Let Π be a clausal inference system. We say that Π has the *strong reduction property for counterexamples* (with respect to a model functor I) if, for all sets N of ground clauses and all minimal counterexamples C for I^N in N, there exists an inference in Π with main premise C, side premise C' which is true in I^N and a minimal counterexample for $I^{N_{C'}}$, and a conclusion D which is a smaller counterexample for I^N than C with respect to a well-founded and total ordering on ground clauses.

Redundancy may be imposed on clauses which are involved as main or side premises in inferences of a clausal inference system which has the strong reduction property. It is also possible to apply redundancy to the conclusions of the inferences as the following definition suggests.

Definition 3.1.22 (Redundant Inference)
Let N be a set of ground clauses and let π be an inference with main premise C and conclusion D. The inference π is called *redundant in* N if $N_C \models D$.

Suppose that Π is a clausal inference system which has the strong reduction property. Recall that any clause C which is involved as the main premise in a reduction inference in Π to a conclusion D is a minimal counterexample for some candidate model I^N. It follows that I^N is a model of N_C. Thus any reduction inference in Π is non-redundant, for otherwise $N_C \models D$ and thus I^N would satisfy D which implied that D is not a counterexample for I^N. Hence, the notion of a redundant inference is a sufficient criterion for an inference to be permanently blocked. Finally, redundancy induces a concept of (*weak*) *saturation* *up to redundancy* where only (non-blocked) non-redundant inferences from non-redundant clauses are considered.

Definition 3.1.23 ((Weak) Saturation up to Redundancy)
Let N be a set of ground clauses and let Π be a clausal inference system. The set N is called (*weakly*) *saturated up to redundancy* with respect to Π, if any (non-blocked) inference in Π from non-redundant premises in N is redundant in N.

The following theorem summarizes the discussion on the strong reduction property and redundancy. Any clausal inference system with the strong reduction property is refutationally complete even when all inferences are restricted to non-blocked non-redundant inferences from non-redundant clauses. As a corollary, we obtain that weak saturation up to redundancy with respect to unrefined resolution is refutationally complete.

Theorem 3.1.24
Let N be a set of ground clauses and let Π be a clausal inference system which is sound and has the strong reduction property for counterexamples with respect to the model functor I. Suppose that N is weakly saturated up to redundancy with respect to Π. Then N is either satisfiable, or else N contains the empty clause.

Proof Suppose that N does not contain the empty clause. If I^N is not a model of N, then N contains a minimal counterexample C for I^N. Since Π has the strong reduction property there is a non-blocked inference from N with main premise C, side premise C', and a conclusion D, such that D is a smaller counterexample for I^N than C and the clause C' is true in I^N and a minimal counterexample in N for $I^{N_{C'}}$. We may assume that C and C' are non-redundant, for otherwise C would be entailed by N_C and thus, by the minimality of C, be true in I^N. Similarly, C' would be entailed by $N_{C'}$ and thus, by the minimality of C', be true already in $I^{N_{C'}}$. By weak saturation, the non-blocked inference is redundant, i.e. $N_C \models D$. But this implies, by the minimality of C, that D is true in I^N, which is a contradiction. In sum, I^N is a model of N. ■

If we just consider saturation up to redundancy, a stronger form of the above theorem is possible (Bachmair & Ganzinger 1997, Bachmair & Ganzinger 1998*a*). Let N be a set of ground clauses which is saturated up to redundancy. Then, in contrast to weak saturation, any subset of N is also saturated up to redundancy. In other words, the relational character of an inference system applied to N is not affected by redundancy.

Theorem 3.1.25 (Bachmair & Ganzinger (1997, 1998*a*))
Let N be a set of ground clauses and let Π be a clausal inference system which is sound and has the reduction property for counterexamples with respect to the model functor I. Suppose that N is saturated up to redundancy with respect to Π. Then N is either satisfiable, or else N contains the empty clause.

Note that, if N is satisfiable and saturated, then I^N is a model of N. In fact, if M is the set of non-redundant clauses in N, then I^M is already a model of M and thus also a model of N.

Finally, we motivate an abstract notion of *simplification* based on redundancy in the context of soft typing. Redundancy can be seen as a concept for the identification of permanent non-counterexamples. Simplification is then the reduction of a clause, counterexample or not, to a permanent non-counterexample, i.e. simplification proves clauses (premises) to be permanent non-counterexamples. The proof may contain new clauses (conclusions) which are, however, smaller than the original. In this way, simplification approximates the set of reduction inferences by a modification of the problem set itself.

Redundancy is a concept which is not effective. However, many simplification techniques in today's saturation-based theorem provers are actually effective approximations of redundancy, e.g., subsumption or, for equational reasoning, conditional rewriting, and thus can be shown to be completeness preserving for inference systems with the (strong) reduction property. In this sense, by simplification, we mean a reduction of clauses to redundant clauses where the reduction produces a proof of redundancy by clauses which are smaller than the originals.

Definition 3.1.26 (Simplification)
Let N be a set of clauses and let C be a clause in N. We call ρ an *admissible simplification* of the form

$$\text{Simplify}\frac{C}{D}$$

if C is redundant in $(N \setminus \{C\}) \cup \{D\}$. The clause C is called the *premise* of the simplification. The *conclusion* D of the simplification replaces the premise C in N.

3.1.1 Semantic Resolution

We recall the basic ideas of semantic resolution (Slagle 1967) which is a particular refinement of the resolution calculus. The semantic filter is based on the truth value of clauses with respect to an arbitrary model hypothesis. In order to show that semantic resolution has the reduction property we need, for technical reasons, another variant of Resolution called *Negative Resolution* which is equal to Resolution up to the polarity of the premises. This rule can be avoided if we assume that a certain renaming of the literals is applied according to the model hypothesis. However, we prefer to use the additional inference as follows.

Definition 3.1.27 (Negative Resolution)
The following inference is called *Negative Resolution*:

$$\text{Infer}\frac{C \vee \neg A \qquad D \vee A}{C \vee D}$$

Then the *semantic resolution calculus* R_S is defined as a clausal inference system which consists of the inference rules from the resolution calculus plus the Negative Resolution rule.

Definition 3.1.28 (Semantic Resolution)
Let J be a model. *Semantic resolution* R_S is a refinement of the resolution calculus R where the inferences by (Negative) Resolution are restricted to premises which have distinct truth values with respect to J.

A clause $C \vee \neg A$ is said to be *negatively reductive for* the atom A, if A is a strictly maximal atom with respect to C. The model functor for semantic resolution is designed to render the false (negatively) reductive clauses in J true. These clauses are supposed to be the side premises in the reduction inferences by (Negative) Resolution.

Definition 3.1.29 (Model Functor for Semantic Resolution)
Let J be a model and let $\succ_c$ be an admissible clause ordering. Let N be a set of ground clauses. The model J divides the set N of clauses into two disjoint partitions N' and $N \setminus N'$ such that J falsifies all clauses in N' and satisfies $N \setminus N'$. Let C be a clause in N'. Suppose that ε_D, δ_D, and I_D^N have been defined for all clauses $D \in N'$ for which $C \succ_c D$. Then

$$I_C^N = (J \cup \bigcup_{C \succ_c D \in N'} \varepsilon_D) \setminus \bigcup_{C \succ_c D \in N'} \delta_D$$

where

$$\varepsilon_C = \{A\} \text{ and } \delta_C = \emptyset$$

if (i) $C = C' \vee A$ is reductive for A and (ii) C is false in I_C^N, or else

$$\varepsilon_C = \emptyset \text{ and } \delta_C = \{A\}$$

if (i) $C = C' \vee \neg A$ is negatively reductive for A and (ii) C is false in I_C^N, or else

$$\varepsilon_C = \emptyset \text{ and } \delta_C = \emptyset.$$

I_C^N is called the *partial interpretation up to* C. We say that a clause C (*negatively*) *produces* the atom A if $\varepsilon_C = \{A\}$ ($\delta_C = \{A\}$) and call C a (*negatively*) *productive clause*. The *model functor for semantic resolution* I assigns to N the Herbrand interpretation $I^N = (J \cup \bigcup_{C \in N'} \varepsilon_C) \setminus \bigcup_{C \in N'} \delta_C$.

Since there are no local contradictions in N', the construction of the model functor implies that $I^N \models N''$ with N'' is the subset of N' such that N'' contains the (negatively) reductive clauses in N'. In general, however, I^N is not a model of $N \setminus N'$, e.g., if N is unsatisfiable. In Section 3.6.1, we will use this model functor to discuss *ordered semantic hyper-linking* (Plaisted 1994, Plaisted & Zhu 1997). Note that in the construction of the model functor we assume that an admissible clause ordering is given. However, due to the absence of local contradictions in N' admissibility could be relaxed such that a partial ordering on ground clauses is possible where clauses with complementary literals are incomparable, c.f. (Plaisted 1994).

The following lemma shows that semantic resolution enjoys the reduction property and, moreover, that any (Negative) Resolution inference from a minimal counterexample involves a main premise which is true in J and a side premise which is false in J. In case of a Factoring inference from a minimal counterexample we may assume that the involved main premise is a clause in $N' \setminus N''$ which is false in J.

Lemma 3.1.30
Let J be a model and let I be the model functor for semantic resolution with an admissible clause ordering $\succ_c$. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then there exists an inference in R_S from C such that

1. the conclusion is a smaller counterexample for I^N than C; and
2. in case of a (Negative) Resolution inference, C is true in J and the side premise C' is false in J and $I^{N_{C'}}$.

Proof The model J divides the set N of clauses into two disjoint partitions N' and $N \setminus N'$ such that J falsifies all clauses in N' and satisfies $N \setminus N'$. Suppose that C is a clause in $N \setminus N'$. Since C is true in J but a counterexample for I^N, there is at least one positive occurrence of an atom A in C such that $A \in J$ and $A \notin I^N$, or at least one negative occurrence of an atom A in C such that $A \notin J$ and $A \in I^N$. In any case, it follows that there is a clause C' in N' which (negatively) produces the atom A. In particular, we have that C' is false in $I^{N_{C'}}$. An inference by (Negative) Resolution from C and C' results then in a smaller counterexample for I^N than C. Note that C is true in J and that C' is false in J.

Suppose that C is a clause in N'. Then C contains several (positive, or else negative) occurrences of the maximal atom A, for otherwise C would not be a counterexample of I^N. Factoring on A produces a smaller counterexample for I^N than C. ■

The obvious improvements of unrefined resolution are not all possible for semantic resolution. Both versions of Factoring are required. However, Factoring is only required on counterexamples. In a (Negative) Resolution inference only the side premise may be restricted to (negatively) reductive clauses. However, the occurrence of the resolved literal in the main premise of a (Negative) Resolution inference may be non-maximal.

Example 3.1.31
The following table contains an inconsistent set of clauses. We assume that semantic resolution is used with the model hypothesis J where A is false and B is true in J. In addition, we assume the atom ordering $B \succ_a A$. Due to the semantic filter given by J we may restrict the (Negative) Resolution inferences to premises with different truth values with respect to J. The table lists the clauses in ascending order.

$\sharp$	Clause C	ε_C	δ_C	Remarks
1	$\rightarrow A, B$	$\emptyset$	$\emptyset$	true in J, false in I^N, positive clause
2	$A \rightarrow B$	$\emptyset$	$\emptyset$	true in J, true in I^N
3	$B \rightarrow A$	$\emptyset$	$\{B\}$	false in J, true in I^N, negatively productive
4	$A, B \rightarrow$	$\emptyset$	$\emptyset$	true in J, true in I^N, negative clause

Note that I^N is empty since B has been removed from J. There are two Resolution inferences from (1) and (3) as well as (3) and (4) yielding (after Factoring) the unit clauses $\rightarrow A$ and $B \rightarrow$, respectively. Note that the inference from (3) and (4) is actually dispensable. Constructing the candidate model for the new clause set proceeds as follows:

$\sharp$	Clause C	ε_C	δ_C	Remarks
1	$\rightarrow A$	$\{A\}$	$\emptyset$	false in J, true in I^N, factorized resolvent of (1) and (3), productive
2	$\rightarrow A, B$	$\emptyset$	$\emptyset$	true in J, true in I^N, positive clause
3	$A \rightarrow B$	$\emptyset$	$\emptyset$	true in J, false in I^N
4	$B \rightarrow$	$\emptyset$	$\{B\}$	false in J, true in I^N, factorized resolvent of (3) and (4), negatively productive
5	$B \rightarrow A$	$\emptyset$	$\emptyset$	false in J, true in I^N
6	$A, B \rightarrow$	$\emptyset$	$\emptyset$	true in J, true in I^N, negative clause

Note that I^N now consists of the atom A. There are two Resolution inferences from (1) and (3) as well as (3) and (4) yielding the unit clauses $\rightarrow B$ and $A \rightarrow$, respectively. Both clauses are true in J which implies that the model I^N does not change. We simply list the two conclusions.

$\sharp$	Clause C	ε_C	δ_C	Remarks
7	$\rightarrow B$	$\emptyset$	$\emptyset$	true in J, false in I^N, resolvent of (1) and (3)
8	$A \rightarrow$	$\emptyset$	$\emptyset$	true in J, false in I^N, resolvent of (3) and (4)

Note that tautologies are not listed in the table. Compared to unrefined resolution the search space becomes narrower but deeper. There are non-deterministic choices but not as many as in unrefined resolution. The contradiction can be derived in the third level.

In general, semantic resolution does not enjoy the strong reduction property. However, using the model functor for semantic resolution we may strengthen Lemma 3.1.30 to show that semantic resolution enjoys a restricted form of the strong reduction property.

Lemma 3.1.32
Let J be a model. Let I be the model functor for semantic resolution and let $\succ_c$ be an admissible clause ordering. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Let N'' be the set of (negatively) reductive clauses from N which are false in J. Then $I^{N_{C'}} \models N''_{C'}$ for all clauses C' in N_C.

Proof By construction of the model functor, $N''_{C'}$ is exactly the set of clauses in N which is rendered true in the candidate model $I^{N_{C'}}$. ■

The lemma implies that any clause which is involved as a side premise in a (Negative) Resolution inference is a minimal counterexample in N'' for $I^{N_{C'}}$, but not in N. This implies that soft typing for semantic resolution is compatible with a relaxed form of redundancy $N''_C \models C$ of a ground clause C. In other words, simplification may only involve (negatively) reductive clauses which are false in J. In Section 3.6.1, we argue that ordered semantic hyper-linking is compatible with simplification of this form.

3.1.2 Ordered Resolution

Ordered resolution is a refinement of resolution with two additional parameters, an admissible atom ordering and a selection function (Bachmair & Ganzinger 1994). We have already seen that Resolution as well as Factoring may be restricted to maximal occurrences of literals. However, the preference of maximal negative literals may be overruled by a *selection function* that indicates for each clause which occurrences of negative literals are *selected* in the clause. Any inference from premises with selected literals is forced to take place on the selected literals. That is, Positive Factoring on a clause with selected literals is not possible as well as side premises with selected literals are excluded from Resolution inferences.

Definition 3.1.33 (Selection Function)
A *selection function* S assigns to each ground clause a possibly empty set of occurrences of negative literals. If C is a clause, the literal occurrences in $S(C)$ are called *selected*. $S(C) = \emptyset$ indicates that no literal is selected.

The *ordered resolution calculus* R_O consists of the inference rules Ordered Resolution and Ordered Factoring. Ordered Factoring corresponds to Positive Factoring with additional ordering constraints and selection.

Definition 3.1.34 (Ordered Resolution)
The following inference is called *Ordered Resolution*:

$$\text{Infer}\,\frac{C \vee A \qquad D \vee \neg A}{C \vee D}$$

where (i) $C \vee A$ is reductive for A, (ii) no literal is selected in C, and (iii) $\neg A$ either is selected, or else is maximal with respect to D.

In an Ordered Resolution inference the negative premise is the main premise whereas the positive premise is the side premise. An Ordered Resolution inference is *redundant in* N if the conclusion is entailed by clauses in N smaller than the negative premise.

Definition 3.1.35 (Ordered Factoring)
The following inference is called *Ordered Factoring*:

$$\text{Infer}\frac{C \vee A \vee A}{C \vee A}$$

where (i) A is maximal with respect to C and (ii) no literal is selected in the premise.

An Ordered Factoring inference is *redundant in N* if the conclusion follows from clauses in N which are smaller than its premise. In order to show the reduction property of ordered resolution we include the selection function into the definition of the model functor, i.e. a clause can only contribute to the candidate model if it does not contain any selected literal. However, in the limit N of a derivation any clause C which is false in I_C^N does not contain any selected literal anyway. In other words, condition (iii) below could be omitted if we would not conceptually separate saturation and the reduction of counterexamples.

Definition 3.1.36 ((Prototypical) Model Functor for Ordered Resolution)
Let $\succ_c$ be an admissible clause ordering and let S be a selection function. Let N be a set of ground clauses and let C be a clause in N. Suppose that ε_D and I_D^N have been defined for all clauses D for which $C \succ_c D$. Then

$$I_C^N = \bigcup_{C \succ_c D} \varepsilon_D$$

where

$$\varepsilon_C = \{A\}$$

if (i) $C = C' \vee A$ is reductive for A, (ii) C is false in I_C^N, and (iii) C contains no selected atom. Otherwise, $\varepsilon_C = \emptyset$. The *(prototypical) model functor for ordered resolution I* assigns to N the Herbrand interpretation $I^N = \bigcup_{C \in N} \varepsilon_C$.

The following lemma shows that ordered resolution enjoys the reduction property. The proof essentially follows the proof of Lemma 3.1.18. In addition, selected literals has to be considered. However, any side premise that is involved in the reduction of a counterexample is a productive clause and, therefore, does not contain any selected literals. In other words, negative literals can be viewed as side conditions which have to be proved. Selection is a mechanism to control this process. Only clauses without any selected literals are considered to be proved and may be involved in the reduction of counterexamples.

Lemma 3.1.37
Let I be the prototypical model functor for ordered resolution. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then there exists an inference in R_O from C such that

1. the conclusion is a smaller counterexample for I^N than C; and
2. in case of an Ordered Resolution inference, C is the main premise and the side premise is a productive clause.

Proof All cases except for selection have been shown in the proof of Lemma 3.1.18. Suppose that C contains a selected negative literal A, i.e. $C = \Lambda, A \rightarrow \Pi$. We proceed similarly to the case where A is maximal. Let C' be a reductive clause of the form $\Gamma \rightarrow \Delta, A$ that produces A. Note that Γ does not contain any selected literal by condition (iii) of the model functor and C' is true in I^N. Thus there is an Ordered Resolution inference between C and C' where D is the conclusion $\Gamma, \Lambda \rightarrow \Delta, \Pi$. It follows that D is a smaller counterexample than C with respect to $\succ_c$. ■

Example 3.1.38
The table contains the same inconsistent set of clauses as in the Example 3.1.31. We assume that ordered resolution and the prototypical model functor I is used whereas redundancy is not considered. In addition, we assume the atom ordering $B \succ_a A$ and an empty selection function. The example describes the construction of the partial interpretations I_C^N. The table lists the clauses in ascending order.

$\sharp$	Clause C	I_C^N	ε_C	Remarks
1	$\rightarrow A, B$	$\emptyset$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
2	$A \rightarrow B$	$\{B\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
3	$B \rightarrow A$	$\{B\}$	$\emptyset$	false in I_C^N, false in I^N, B is maximal
4	$A, B \rightarrow$	$\{B\}$	$\emptyset$	true in $I_C^N = I^N$, true in I^N, B is maximal

There are two Ordered Resolution inferences from (1) and (3) as well as (2) and (4) yielding (after Ordered Factoring) the unit clauses $\rightarrow A$ and $A, A \rightarrow$, respectively. Note that the inference from (2) and (4) is actually dispensable. Constructing the partial interpretations for the new clause set proceeds as follows:

$\sharp$	Clause C	I_C^N	ε_C	Remarks
1	$\rightarrow A$	$\emptyset$	$\{A\}$	false in I_C^N, true in I^N, productive, A is maximal
2	$\rightarrow A, B$	$\{A\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
3	$A, A \rightarrow$	$\{A\}$	$\emptyset$	false in I_C^N, false in I^N
4	$A \rightarrow B$	$\{A\}$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
5	$B \rightarrow A$	$\{A, B\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
6	$A, B \rightarrow$	$\{A, B\}$	$\emptyset$	now false in $I_C^N = I^N$, false in I^N, B is maximal

There is only one Ordered Resolution inference from (1) and (3), respectively, which leads to the contradiction.

Using the prototypical model functor for ordered resolution combined with an admissible clause ordering, Lemma 3.1.37 does not carry over to show the strong reduction property of ordered resolution. That is, for any side premise C' which is involved in a reduction inference by Ordered Resolution C' is some counterexample in N for $I^{N_{C'}}$. In other words, $I^{N_{C'}}$ is, in general, not a model of $N_{C'}$.

Example 3.1.39
The example demonstrates the problem which is actually due to the combination of an admissible clause ordering and selection. We assume the atom ordering $B \succ_a A' \succ_a A$. Moreover, suppose that the atom A is selected in the clause $A \to B$. The table describes the construction of I^N for an inconsistent set N of clauses which the table lists in ascending order.

$\sharp$	Clause C	I_C^N	ε_C	Remarks
1	$\to A$	$\emptyset$	$\{A\}$	false in I_C^N, true in I^N, productive, A is maximal
2	$\to A'$	$\{A\}$	$\{A'\}$	false in I_C^N, true in I^N, productive, A' is maximal
3	$A \to B$	$\{A', A\}$	$\emptyset$	false in I_C^N, true in I^N, A is selected, B is maximal
4	$A', A \to B$	$\{A', A\}$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
5	$B \to$	$\{B, A', A\}$	$\emptyset$	false in $I_C^N = I^N$, false in I^N

The clauses (1) and (2) produce the atoms A and A', respectively. As a consequence, the clause (3) is false in I_C^N but does not produce B into I^N since A is selected. The atom B is then produced by the clause (4). However, (5) is the minimal counterexample in N for I^N but $I^{N_{C'}}$ is not a model for $N_{C'}$ if C' is the clause (4).

To overcome this problem we consider below a different construction of the model functor. Another solution is to use the prototypical model functor in combination with either a restriction of the selection function, or else a modification of the definition of admissible clause orderings. For the latter choice the selection status of negative literals has to be part of the ordering. Suppose that the clause (4) is smaller than the clause (3). Constructing the partial interpretations for the modified clause ordering proceeds as follows:

♯	Clause C	I_C^N	ε_C	Remarks
1	$\rightarrow A$	$\emptyset$	$\{A\}$	false in I_C^N, true in I^N, productive, A is maximal
2	$\rightarrow A'$	$\{A\}$	$\{A'\}$	false in I_C^N, true in I^N, productive, A' is maximal
4	$A', A \rightarrow B$	$\{A', A\}$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
3	$A \rightarrow B$	$\{B, A', A\}$	$\emptyset$	true in I_C^N, true in I^N, A is selected, B is maximal
5	$B \rightarrow$	$\{B, A', A\}$	$\emptyset$	false in $I_C^N = I^N$, false in I^N

Note that $I^{N_{C'}}$ is a model for $N_{C'}$ for any clause which is smaller than (5).

Definition 3.1.40 ((Prototypical) Admissible Clause Ordering)
Let $\succ_l$ be an admissible literal ordering and let S be a selection function. Let M be the multiset of the literal occurrences in a clause C. If C is reductive for A and $S(C)$ is non-empty, then $M' = M \cup \{A\}$, or else $M' = M$. The multiset M' is called the *S-representation* of C. A *clause ordering* $\succ_c$ is the multiset extension of $\succ_l$ on S-representations. We call $\succ_c$ *admissible* if $\succ_l$ is admissible.

This version of an admissible clause ordering is still compatible with the underlying atom ordering. If the maximal atom in a ground clause C is greater than the maximal atom in a ground clause D then $C \succ_c D$. Using the prototypical model functor for ordered resolution and the prototypical admissible clause ordering we may strengthen Lemma 3.1.37 to show that ordered resolution actually has the strong reduction property.

Lemma 3.1.41
Let I be the prototypical model functor for ordered resolution and let $\succ_c$ be a prototypical admissible clause ordering. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then $I^{N_{C'}} \models N_{C'}$ for all clauses C' in $N_C \cup \{C\}$.

Proof Let C' be a clause in $N_C \cup \{C\}$. In order to show that $I^{N_{C'}}$ is a model of $N_{C'}$ suppose otherwise that there is a clause D of the form $\Gamma \rightarrow \Delta$ in $N_{C'}$ such that $I^{N_{C'}} \not\models D$, i.e. $\Gamma \subseteq I^{N_{C'}}$ and $\Delta \cap I^{N_{C'}} = \emptyset$, and, in particular, $\Gamma \subseteq I^{N_D}$ and $\Delta \cap I^{N_D} = \emptyset$. Since C is the minimal counterexample in N for I^N and $C \succ_c D$ we have that D is true in I^N. Thus there is a clause D' in N where $D' \succ_c D$ that produces an atom A in Δ into I^N. It follows that D is also reductive for A for otherwise $D \succ_c D'$. Since D is not productive there are selected literal occurrences in D. However, in this case D would again be greater than D' with respect to $\succ_c$ which shows the contradiction. ■

Example 3.1.42
For an example of soft typing for ordered resolution, we use again the same inconsistent set of clauses as in the Example 3.1.31 and 3.1.38. We assume the atom ordering $B \succ_a A$

and an empty selection function. The following table describes the construction of the partial interpretations I_C^N. Clauses are listed in ascending order.

$\sharp$	Clause C	I_C^N	ε_C	Remarks
1	$\rightarrow A, B$	$\emptyset$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
2	$A \rightarrow B$	$\{B\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
3	$B \rightarrow A$	$\{B\}$	$\emptyset$	false in I_C^N, false in I^N, B is maximal
4	$A, B \rightarrow$	$\{B\}$	$\emptyset$	true in $I_C^N = I^N$, true in I^N, B is maximal

According to our definition, the only non-blocked inference is by Ordered Resolution from (1) and (3), yielding (after Ordered Factoring) the unit clause $\rightarrow A$. In this way, soft typing singles out the dispensable inference from (2) and (4). Constructing the partial interpretations for the new clause set proceeds as follows:

$\sharp$	Clause C	I_C^N	ε_C	Remarks
1	$\rightarrow A$	$\emptyset$	$\{A\}$	false in I_C^N, true in I^N, productive, A is maximal
2	$\rightarrow A, B$	$\{A\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
3	$A \rightarrow B$	$\{A\}$	$\{B\}$	false in I_C^N, true in I^N, productive, B is maximal
4	$B \rightarrow A$	$\{A, B\}$	$\emptyset$	true in I_C^N, true in I^N, B is maximal
5	$A, B \rightarrow$	$\{A, B\}$	$\emptyset$	now false in $I_C^N = I^N$, false in I^N, B is maximal

Again, only one inference is non-blocked, the Ordered Resolution inference from (3) and (5), respectively. From this we derive $A \rightarrow$, and then, by a Resolution step with (1), the contradiction. We observe that by choosing counterexamples, computing non-blocked inferences can be made a completely deterministic refutation process for this example, whereas ordering restrictions alone or semantic resolution is not deterministic.

In order to demonstrate that ordered resolution with selection has the strong reduction property, there is, beside the modification of the clause ordering, another more general solution which also carries over to solve similar problems which arise in the context of superposition. We may incorporate the minimality requirement for counterexamples of partial interpretations into the construction of the model functor I. That is, a clause C may only be productive, if C does not change the truth value of the clauses which are smaller than C. This gives an immediate proof for the strong reduction property. However, in general, there are less productive clauses in a set N of ground clauses and thus more counterexamples with respect to I^N. In other words, we may turn a productive clause C into a (minimal) counterexample, in order to enable the system to accomplish some reduction steps on C, e.g., triggered by selection, before C can be used as a side premise for the reduction of other counterexamples. We propose a modification of the model functor for ordered resolution in this regard.

Definition 3.1.43 (Model Functor for Ordered Resolution)
Let $\succ_c$ be an admissible clause ordering of Definition 3.1.15 and let S be a selection function. Let N be a set of ground clauses and let C be a clause in N. Suppose that ε_D and I_D^N have been defined for all clauses D for which $C \succ_c D$. Then

$$I_C^N = \bigcup_{C \succ_c D} \varepsilon_D$$

where

$$\varepsilon_C = \{A\}$$

if (i) $C = C' \vee A$ is reductive for A, (ii) C is false in I_C^N, (iii) C contains no selected atom, and (iv) $I_C^N \cup \{A\} \not\models D$ for all clauses $D \in N$ with $I_C^N \not\models D$ and $C \succ_c D$. Otherwise, $\varepsilon_C = \emptyset$. The *model functor for ordered resolution* I assigns to N the Herbrand interpretation $I^N = \bigcup_{C \in N} \varepsilon_C$.

The following lemma repeats that ordered resolution enjoys the reduction property for a combination of the modified model functor and the original admissible clause ordering. Intuitively, the productive clauses of the model functor have the same properties as the productive clauses of the former prototypical construction.

Lemma 3.1.44
Let I be the model functor for ordered resolution of Definition 3.1.43. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then there exists an inference in R_O from C such that

1. the conclusion is a smaller counterexample for I^N than C; and
2. in case of an Ordered Resolution inference, C is the main premise and the side premise is a productive clause.

Proof The proof follows exactly the proof of Lemma 3.1.37. ■

More interestingly, using the model functor for ordered resolution we may strengthen Lemma 3.1.44 to show that ordered resolution actually has the strong reduction property even with the original admissible clause ordering.

Lemma 3.1.45
Let I be the model functor for ordered resolution of Definition 3.1.43 and let $\succ_c$ be an admissible clause ordering. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then $I^{N_{C'}} \models N_{C'}$ for all clauses C' in $N_C \cup \{C\}$.

Proof Let C' be a clause in $N_C \cup \{C\}$. In order to show that $I^{N_{C'}}$ is a model of $N_{C'}$ suppose otherwise that there is a clause D of the form $\Gamma \rightarrow \Delta$ in $N_{C'}$ such that $I^{N_{C'}} \not\models D$, i.e. $\Gamma \subseteq I^{N_{C'}}$ and $\Delta \cap I^{N_{C'}} = \emptyset$, and, in particular, $\Gamma \subseteq I^{N_D}$ and $\Delta \cap I^{N_D} = \emptyset$. Since C is the minimal counterexample in N for I^N and $C \succ_c D$ we have that D is true in I^N. However, this already shows the contradiction, since, by condition (iv) of the definition of the model functor, a clause D which is false in I^N_D cannot become true by a greater productive clause. That is, a clause D' in N where $D' \succ_c D$ cannot produce an atom A in Δ into I^N, for otherwise D would become true in $I^N_{D'} \cup \{A\}$. ■

Example 3.1.46
The example demonstrates the behavior of soft typing for ordered resolution. The problem corresponds to the problem of the Example 3.1.39. We assume the atom ordering $B \succ_a A' \succ_a A$. Moreover, suppose that the atom A is selected in the clause $A \rightarrow B$. The table describes the construction of I^N for an inconsistent set N of clauses which the table lists in ascending order.

♯	Clause C	I^N_C	ε_C	Remarks
1	$\rightarrow A$	$\emptyset$	$\{A\}$	false in I^N_C, true in I^N, productive, A is maximal
2	$\rightarrow A'$	$\{A\}$	$\{A'\}$	false in I^N_C, true in I^N, productive, A' is maximal
3	$A \rightarrow B$	$\{A', A\}$	$\emptyset$	false in I^N_C, false in I^N, A is selected, B is maximal
4	$A', A \rightarrow B$	$\{A', A\}$	$\emptyset$	false in I^N_C, false in I^N, non-productive, B is maximal
5	$B \rightarrow$	$\{A', A\}$	$\emptyset$	false in $I^N_C = I^N$, true in I^N

The clauses (1) and (2) produce the atoms A and A', respectively. As a consequence, the clause (3) is false in I^N_C but does not produce B into I^N since A is selected. However, neither clause (4) produces the atom B, for otherwise (3) would become true. Thus clause (3) is the minimal counterexample in N for I^N. The non-blocked inference by Ordered Resolution from (1) and (3) derives the clause $\rightarrow B$ that triggers the derivation of the empty clause in the next step.

3.1.3 Superposition

Soft typing is a general framework for the incorporation of semantic control in clausal inference systems. In the previous sections, we have seen that soft typing for semantic and ordered resolution are refutationally complete. The motivation of this section is to demonstrate that soft typing is also compatible with theorem proving technology for first-order logic with equality. The *superposition calculus* (with *selection*) Bachmair & Ganzinger (1994), is a well-known example in this category. Soft typing for superposition is in fact refutationally complete. The proofs are similar to the proofs for ordered resolution but technically more involved. We need generalized versions of admissible orderings. In particular, since equality induces a congruence on the Herbrand universe, our conventional admissible atom ordering has to be refined to an admissible ordering on terms.

Definition 3.1.47 (Admissible Term Ordering)
An ordering $\succ$ on terms is called a *term ordering*. $\succ$ is called *admissible* if $\succ$ is a reduction ordering and the restriction of $\succ$ to ground terms is a total ordering.

Note that an admissible term ordering induces an admissible ordering on atoms. However, this atom ordering is more restrictive, since the underlying term ordering has to be a reduction ordering, at least when equality is present.

Definition 3.1.48 (Admissible Literal Ordering)
Let $\succ$ be an admissible term ordering. An ordering $\succ_l$ on literals is called a *literal ordering*. $\succ_l$ is called *admissible* if (i) $\succ_l$ is well-founded, (ii) the restriction of $\succ_l$ to ground literals is a total ordering, (iii) if L and L' are ground literals, then $L \succ_l L'$, whenever (iii.1) $\max(L) \succ \max(L')$, or (iii.2) $\max(L) = \max(L')$ and the literal L is negative, whereas L' is positive. By $\max(L)$, we denote the maximal term in L with respect to $\succ$. A ground literal L is called [strictly] maximal with respect to a multiset of literals Γ, if for any L' in Γ we have $L \succeq_l L'$ [$L \succ_l L'$].

Note that for any positive literal $s \approx t$, $s \not\approx t$ is greater than $s \approx t$, whereas $s \approx t$ is smaller than $s \not\approx u$ with $s \succ t \succ u$. The multiset extension of an admissible literal ordering induces an ordering on ground clauses. We use the admissible clause ordering for ordered resolution of Definition 3.1.15. We say that a clause $\Gamma \rightarrow \Delta, s \approx t$ is *reductive for* $s \approx t$, if $t \not\succeq s$ and $s \approx t$ is strictly maximal with respect to $\Gamma \rightarrow \Delta$.

Admissible clause orderings are well-founded and total on ground clauses. Moreover, an admissible clause ordering is compatible with the underlying term ordering. If the maximal term in a ground clause C is greater than the maximal term in a ground clause D then $C \succ_c D$. This implies that the clause ordering is also compatible with the induced atom ordering. If the maximal atom in a ground clause C is greater than the maximal atom in a ground clause D then $C \succ_c D$. We shall consider only admissible orderings from now on. Based on the admissible orderings superposition enjoys the strong reduction property, see below.

The *superposition calculus* S is a saturation-based theorem proving method for full first-order clauses with equality. Superposition is a refinement of *paramodulation* by additional ordering restrictions which have been motivated by the *Knuth-Bendix completion* procedure. The completion procedure is a well-known concept for the computation of *convergent* rewrite systems. Recall that convergent rewrite systems define unique normal forms and can, therefore, be used as convenient representations of equality interpretations (candidate models). Superposition generalizes the idea of Knuth-Bendix completion for term rewriting systems to first-order clauses with equality. The calculus consists of the following inference rules.

Definition 3.1.49 (Superposition Left)
The following inference is called *Superposition Left*:

$$\text{Infer}\frac{\Gamma \to \Delta, s \approx t \qquad u[s'] \approx v, \Lambda \to \Pi}{u[t]\sigma \approx v\sigma, \Gamma\sigma, \Lambda\sigma \to \Delta\sigma, \Pi\sigma}$$

where (i) σ is the most general unifier of s and s', (ii) the clause $\Gamma\sigma \to \Delta\sigma, s\sigma \approx t\sigma$ is reductive for $s\sigma \approx t\sigma$ and no literal is selected in $\Gamma\sigma$, (iii) $v\sigma \not\succeq u\sigma$ and $u\sigma \approx v\sigma$ is selected, or else no literal is selected in $u\sigma \approx v\sigma, \Lambda\sigma$ and $u\sigma \approx v\sigma$ is maximal with respect to $\Lambda\sigma \to \Pi\sigma$, and (iv) s' is not a variable.

Definition 3.1.50 (Superposition Right)
The following inference is called *Superposition Right*:

$$\text{Infer}\frac{\Gamma \to \Delta, s \approx t \qquad \Lambda \to \Pi, u[s'] \approx v}{\Gamma\sigma, \Lambda\sigma \to \Delta\sigma, \Pi\sigma, u[t]\sigma \approx v\sigma}$$

where (i) σ is the most general unifier of s and s', (ii) the clause $\Gamma\sigma \to \Delta\sigma, s\sigma \approx t\sigma$ is reductive for $s\sigma \approx t\sigma$ and no literal is selected in $\Gamma\sigma$, (iii) the clause $\Lambda\sigma \to \Pi\sigma, u\sigma \approx v\sigma$ is reductive for $u\sigma \approx v\sigma$ and no literal is selected in $\Lambda\sigma$, (iv) s' is not a variable, and (v) $s\sigma \approx t\sigma \not\succeq_l u\sigma \approx v\sigma$.

Definition 3.1.51 (Equality Resolution)
The following inference is called *Equality Resolution*:

$$\text{Infer}\frac{u \approx v, \Lambda \to \Pi}{\Lambda\sigma \to \Pi\sigma}$$

where (i) σ is the most general unifier of u and v, (ii) $u\sigma \approx v\sigma$ is selected, or else no literal is selected in $u\sigma \approx v\sigma, \Lambda\sigma$ and $u\sigma \approx v\sigma$ is maximal with respect to $\Lambda\sigma \to \Pi\sigma$.

Definition 3.1.52 (Equality Factoring)
The following inference is called *Equality Factoring*:

$$\text{Infer}\frac{\Lambda \to \Pi, u \approx v, s \approx t}{v\sigma \approx t\sigma, \Lambda\sigma \to \Pi\sigma, s\sigma \approx t\sigma}$$

where (i) σ is the most general unifier of s and u, (ii) $s\sigma \approx t\sigma$ is maximal with respect to $\Lambda\sigma \to \Pi\sigma$, and (iii) no literal is selected in $\Lambda\sigma$.

The following construction of the model functor is a generalization of the model functor for ordered resolution. The partial interpretation I_C^N is inductively constructed as the congruence closure of the *partial rewrite system* R_C. Note that any R_C is a left-reduced ground rewrite system, i.e. for every rewrite rule $s \approx t$ in R_C the term s is irreducible by $R_C \setminus \{s \approx t\}$. It follows from the well-known critical pair lemma for term rewrite systems that any left-reduced, well-founded ground rewrite system is convergent. Thus the truth value of an equation $s \approx t$ can be determined by rewriting, i.e. $s \approx t \in I$ if and only if $s \Downarrow_R t$.

Definition 3.1.53 (Model Functor for Superposition)
Let $\succ_c$ be an admissible clause ordering and let S be a selection function. Let N be a set of ground clauses and let C be a clause in N. Suppose that R_D, ε_D, and I_D^N have been defined for all clauses D for which $C \succ_c D$. Then

$$R_C = \bigcup_{C \succ_c D} \varepsilon_D \text{ and } I_C^N = R_C^*.$$

where

$$\varepsilon_C = \{s \approx t\}$$

if C is a clause $C' \vee s \approx t$ such that (i) C is reductive for $s \approx t$, (ii) C contains no selected equation, (iii) C is a counterexample for I_C^N, (iv) s is irreducible by R_C, and (v) $I_C^N \cup \{s \approx t\} \not\models D$ for all clauses $D \in N_C \cup \{C'\}$ with $I_C^N \not\models D$. Otherwise, $\varepsilon_C = \emptyset$. We say that a clause C *produces* the equation $s \approx t$ if $\varepsilon_C = \{s \approx t\}$ and call C a *productive clause.* The *model functor for superposition* I assigns to N the equality Herbrand interpretation $I^N = \bigcup_{C \in N} \varepsilon_C$.

This version of the model functor employs, by condition (v), a generalization of condition (v'), C' is a counterexample for $I_C^N \cup \{s \approx t\}$, which has already been used in the construction of the model functor in (Bachmair & Ganzinger 1998*a*). Actually, the simpler form is required to obtain the reduction property when a reductive clause C of the form $\Lambda \to \Pi, u \approx v', u \approx v$, where $u \approx v$ denotes the strictly maximal literal, is involved as a side premise in some reduction inference. Suppose that $I_C^N \models v \approx v'$ holds. In this case, if C would be productive, then the conclusion of the inference would not be a counterexample for I^N.

In contrast to this *local* situation, a more complicated *global* scenario may arise when several clauses of the form of C, which contain the same maximal term u, occur in N. In this case when using condition (v'), such clauses may render smaller clauses of this form, which are false in the candidate model, true subsequently. This implies that, in contrast to ordered resolution with selection, it is not possible to obtain the strong reduction property for superposition by an adaption of the clause ordering. However, in our construction of the model functor, condition (v') is strictly included and, moreover, similar to the model functor for ordered resolution, productive clauses are restricted such that smaller clauses cannot become true subsequently. Thus we obtain the strong reduction property for superposition even with selection as an immediate consequence, see below.

Similar to Lemma 3.1.17, using admissible clause orderings, we can show that the construction of the candidate model for superposition is monotone in the sense that a clause D does not affect the truth value of equations which occur negatively in clauses in N_D.

Lemma 3.1.54 (Bachmair & Ganzinger (1994))
Let $\succ_c$ be an admissible clause ordering. Let I be the model functor for superposition and let N be a set of ground clauses. Let C be a clause $\Gamma, s \approx t \to \Delta$ in N and let D be a clause in N with $D \succeq_c C$. If $s \approx t$ is false in I_C^N then it is also false in I_D^N and I^N.

Proof Suppose that $s \approx t$ is false in I_C^N. If s' and t' are the normal forms of s and t, respectively, with respect to R_C, then $s' \neq t'$. Furthermore, if $u \approx v$ is a rule in $R \setminus R_C$, then $u \succ s \succeq s'$ and $v \succ t \succeq t'$. Therefore, s' and t' are in normal form with respect to R, which implies that $s \approx t$ is false in I_D^N and I^N. ■

The following proof to show that superposition has the reduction property is similar to the proof for ordered resolution. However, the presence of equality involves a more complicated case analysis.

Lemma 3.1.55
Let I be the model functor for superposition. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then there exists an inference in S from C such that

1. the conclusion is a smaller counterexample for I^N than C; and
2. in case of a Superposition Left or Superposition Right inference, C is the main premise and the side premise is a productive clause.

Proof Suppose that C contains a selected or maximal negative literal $u \approx v$, i.e. $C = u \approx v, \Lambda \to \Pi$. Since $I^N \not\models C$ we have that $\Lambda \subseteq I^N$, $u \approx v \in I^N$, and $\Pi \cap I^N = \emptyset$. If $u = v$ then Equality Resolution derives the conclusion $\Lambda \to \Pi$ which is smaller than C and also a counterexample for I^N.

Suppose that $u \succ v$. Since $u \approx v$ is true in I^N there is a reductive clause C' of the form $\Gamma \to \Delta, s \approx t$ that produces $s \approx t$ where s is a subterm of $u[s]$. Note that Γ does not contain any selected literal and C' is true in I^N. Thus there is a Superposition Left inference between C and C' where D is the conclusion $u[t] \approx v, \Gamma, \Lambda \to \Delta, \Pi$. It follows that D is smaller than C with respect to $\succ_c$. Since C' is false in $I_{C'}^N$ we have that $\Gamma \subseteq I_{C'}^N \subseteq I^N$ and $\Delta \cap I_{C'}^N = \emptyset$. By Lemma 3.1.54 all $w \approx w' \in \Delta$ are also false in I^N which implies that $\Delta \cap I^N = \emptyset$ and therefore that D is a counterexample for I^N.

Suppose that C contains no selected literals and $u \approx v$ is a maximal positive equation in C where $u \succ v$, i.e. $C = \Lambda \to \Pi, u \approx v$. Note that u is the maximal term in C. Since $I^N \not\models C$ we have that $\Lambda \subseteq I^N$, $\Pi \cap I^N = \emptyset$, and $u \approx v \notin I^N$. If Π is of the form $\Pi', u \approx v$ then by Equality Factoring we obtain the clause D of the form $v \approx v, \Lambda \to \Pi', u \approx v$. Clearly, D is a smaller counterexample than C for I^N.

Suppose that Π does not contain any equation $u \approx v$. Note that in this case C is reductive for $u \approx v$. If u is reducible by R_C, then there is a reductive clause C' of the form $\Gamma \to \Delta, s \approx t$ that produces $s \approx t$ where s is a subterm of $u[s]$. Note that Γ does not contain any selected literal, C' is true in I^N, and $(u \approx v) \succ_l (s \approx t)$ since C' is smaller than C. Thus there is a Superposition Right inference between C and C' where D is the conclusion $\Gamma, \Lambda \to \Delta, \Pi, u[t] \approx v$. It follows that D is smaller than C with respect to $\succ_c$. Since C' is false in $I_{C'}^N$ we have that $\Gamma \subseteq I_{C'}^N \subseteq I^N$ and $\Delta \cap I_{C'}^N = \emptyset$. By Lemma 3.1.54 all $w \approx w' \in \Delta$ are also false in I^N which implies that $\Delta \cap I^N = \emptyset$ and therefore that D is a counterexample for I^N.

Suppose that u is irreducible by R_C. In this case C fulfills all conditions of a productive clause except that $\Lambda \to \Pi$ might not be a counterexample for $I_C^N \cup \{u \approx v\}$. Consequently, this must be the reason for C not being productive. Note that a clause D in N which is smaller than C must be true in I_C^N, for otherwise C would not be the minimal counterexample for I^N, c.f. Lemma 3.1.56. Suppose that there is an equation $s \approx t$ in Π which is true in $I_C^N \cup \{u \approx v\}$ where $s \succ t$. Thus s is reducible by $u \approx v$ which implies that $s = u$ for otherwise u would not be the maximal term in C. Moreover, $v \approx t$ must be true in I_C^N and therefore also in I^N. Note that $s \approx t$ is maximal with respect to $\Lambda \to \Pi'$ where $\Pi = \Pi', s \approx t$. By Equality Factoring we obtain the clause D of the form $v \approx t, \Lambda \to \Pi', s \approx t$. It follows that D is a smaller counterexample than C for I^N. ■

Using the model functor for superposition we may strengthen Lemma 3.1.55 to show that superposition has the strong reduction property with respect to an admissible clause ordering based on an admissible term ordering.

Lemma 3.1.56
Let I be the model functor for superposition and let $\succ_c$ be the according admissible clause ordering. Let N be a set of ground clauses which does not contain the empty clause and let C be the minimal counterexample in N for I^N. Then $I^{N_{C'}} \models N_{C'}$ for all clauses C' in $N_C \cup \{C\}$.

Proof Let C' be a clause in $N_C \cup \{C\}$. In order to show that $I^{N_{C'}}$ is a model of $N_{C'}$ suppose otherwise that there is a clause D of the form $\Gamma \to \Delta$ in $N_{C'}$ such that $I^{N_{C'}} \not\models D$, i.e. $\Gamma \subseteq I^{N_{C'}}$ and $\Delta \cap I^{N_{C'}} = \emptyset$, and, in particular, $\Gamma \subseteq I^{N_D}$ and $\Delta \cap I^{N_D} = \emptyset$. Since C is the minimal counterexample in N for I^N and $C \succ_c D$ we have that D is true in I^N. However, this already shows the contradiction, since, by condition (v) of the definition of the model functor, a clause D which is false in I_D^N cannot become true by greater productive clauses. That is, a clause D' in N where $D' \succ_c D$ cannot produce an equation $s \approx t$ into I^N such that some equation $u \approx v$ in Δ becomes true in $I_{D'}^N \cup \{s \approx t\}$, for otherwise D would become true in $I_{D'}^N \cup \{s \approx t\}$. ■

Superposition as a theorem proving method for first-order logic with equality has been further optimized in various directions by, however, mostly syntactic concepts. In general, a refinement is compatible with soft typing (and redundancy) if it enjoys the (strong) reduction property. However, in some cases the notion of counterexamples and/or redundancy have to be adapted in order to achieve refutational completeness for the refinement. In particular, a weakening of the notion of counterexamples may lead to impractical versions of soft typing since an effective approximation may be difficult.

An example is the *strict superposition calculus* (Bachmair & Ganzinger 1997, Bachmair & Ganzinger 1998*b*) which is simply the superposition calculus without Equality Factoring. The refinement is not compatible with arbitrary tautology elimination and, therefore, does not have the reduction property. Based on so-called weak counterexamples, which are defined by a notion of *direct provability* rather than semantic validity, an according weak reduction property can be established. Direct provability of an equation $u \approx v$ is obtained by a (direct) rewrite proof of $u \approx v$ with respect to a rewrite system R that uses only equation $s \approx t$ from R which are smaller than or equal to $u \approx v$. Similar to standard superposition, the rewrite system R represents the candidate model I^N.

A clause $\Gamma \to \Delta$ is a weak counterexample for R if and only if (i) for each negative literal $u \approx v$ in Γ we have $u \Downarrow_R v$ and (ii) no positive equation in Δ has a direct rewrite proof in R. Note that R is constructed as a left-reduced, well-founded ground rewrite system which implies, by the critical pair lemma, that $u \Downarrow_R v$ is equivalent to $u \approx t \in I$ for I being the congruence closure of R. Hence, *weak blocking* of $\Gamma \to \Delta$ may become effective by approximating (i) the truth value of Γ using the same approximation techniques as for standard blocking, in particular, the approximations discussed in this work. By weak blocking we mean a straightforward adaption of blocking to weak counterexamples. The situation for weak blocking based on (ii), the direct provability of Δ, is more involved since the term ordering has to be considered as an additional parameter by any approximation.

Ordered chaining calculi, which have been developed by Bachmair & Ganzinger (1995) as a generalization of superposition for possibly non-symmetric transitive relations R_t, are also compatible with soft typing. The candidate models are constructed as models in which R_t is a transitive relation. For an effective approximation of blocking, decidable fragments of first-order logic with transitive relations (Ganzinger, Hustadt, Meyer & Schmidt 1999) could be used. Ordered chaining may also be optimized to a strict version in a similar way as superposition (Bachmair & Ganzinger 1997). An inference rule called transitivity resolution, that generalizes Equality Factoring to the symmetric case, can be avoided. As a consequence, using a similar construction of weak counterexamples, the effective approximation of weak blocking is possible by approximating (i) from above using the same techniques as for the non-strict case.

Strict superposition is compatible with a notion of *weak redundancy* which is a particular form of redundancy based on weak counterexamples. However, not all superposition inferences at or below variable positions, i.e. positions of variable occurrences, are weakly redundant. This problem is actually an instance of the general *lifting problem* which arises in the context of non-ground clauses, see Section 3.5. However, certain techniques of another refinement of superposition, the so-called *basic superposition* (Bachmair, Ganzinger, Lynch & Snyder 1992, Nieuwenhuis & Rubio 1992), motivated Bachmair & Ganzinger (1997) to generalize strict superposition to *strict basic superposition* such that superposition inferences at or below variable positions can be avoided.

The idea of basic superposition is to block superposition inferences at so-called *substitution positions*. A substitution position is a generalization of a variable position where the variable positions of a clause C are "remembered" in all successors of C, e.g., by some marking mechanism. That is, a substitution position in a clause C refers to a position at or below a variable position in a predecessor of C. For the basic variant of paramodulation, a further optimization is possible by the so-called *variable abstraction* (Bachmair, Ganzinger, Lynch & Snyder 1995). A substitution position may also include the redex position of a previous basic paramodulation inference. Superposition is a generalization of paramodulation with additional ordering restrictions. Note that we use specialized basic superposition/paramodulation calculi in Sections 6.3, 6.7, and 7.1 as decision procedures for non-trivial equational theories. In particular, variable abstraction provides for an elegant result for the decidability of unification with respect to certain equational theories.

Syntactic concepts like basic restrictions or even more involved constraints like the equality and ordering constraints in (Nieuwenhuis & Rubio 1995) usually yield refinements which are compatible with soft typing. The general motivation of these methods is to find a better approximation of the ground level by restricting the set of possible ground instances of non-ground clauses and the inferences on the non-ground level. Thus compatibility with soft typing is achieved whenever the lifting problem for the refinements can be solved, see Section 3.5.

3.2 Minimal Model Semantics

The blocking concept of soft typing induces the need for (i) suitable representations of candidate models and (ii) decision procedures for the validity and/or satisfiability problem in these representations in order to make effective use of this method. Recall that an inference is blocked depending on the truth value of its premises with respect to the candidate model defined by some model functor. If we take a set N of (Horn) clauses as the representation of (the approximation of) a candidate model the *minimal model semantics* of N is a suitable concept for (i) to relate the whole class of (Herbrand) models of N to the candidate model. Our discussion on minimal model semantics is based on the introduction to the semantics of logic programs by Apt (1990). Note that (ii) is addressed in Chapter 4 through 7.

The following discussion is based on first-order logic with equality. Note that a non-equality atom A may be encoded as an equation $A \approx \top$ where $\top$ is a special constant symbol. We assume, in the sequel, that an admissible atom ordering is induced by an admissible term ordering. If equality is not involved we may relax the atom ordering to a general admissible atom ordering, i.e. it does not have to be compatible with a reduction ordering on terms.

The *least model* I of a clause set N is defined as the unique Herbrand model of N which is subset included in any other Herbrand model of N. However, I does not always exist for arbitrary N. The *minimal model* I of N is defined as a Herbrand model of N such that any Herbrand model which is properly subset included in I is not a model of N.

Definition 3.2.1 (Least and Minimal Model)
Let I be an (equality) Herbrand model of a set N of clauses. We call I the *least model of* N if $I \subseteq J$ for all (equality) Herbrand models J of N. We call I a *minimal model of* N if $J \not\models N$ for all (equality) Herbrand models J with $J \subset I$.

Note that the least model is minimal but not vice versa, e.g., take $N = \{A \vee B\}$ where A and B are ground atoms. Then $\{A\}$ and $\{B\}$ are minimal but not least models of N. In general, there are several minimal model for an arbitrary set N of clauses. However, for certain classes of satisfiable clause sets there is always a unique least Herbrand model, see below.

Suppose that N has the least Herbrand model I. Let J be a superset of I such that J is still a Herbrand model of N. Suppose that C is a clause in N of the form $P(t_1, \dots, t_n), \Gamma \rightarrow \Delta$. If the extension of P is empty in J, then it is also empty in I. Thus C is not only true in J but also in I. If the emptiness of P in J is decidable, then we already have a decidable approximation of the problem $N \models C$, but not of $N \not\models C$. Moreover, if I is a superset of some candidate model I^M of some set M of clauses, then the problem $I^M \models C$ is also decidable. In this case N is a suitable representation for the approximation of blocking. The dual problem $I^M \not\models C$ could be approximated by some subset J of I where I must be a subset of I^M. For a general concept see Section 3.3.

The following definition introduces a particular form of clauses called *Horn clauses* which contain at most one positive occurrence of a literal. It is well-known that any satisfiable set H of Horn clauses has a unique minimal Herbrand model which corresponds to the least Herbrand model of H.

Definition 3.2.2 (Horn Clause)
A clause with at most one positive literal is called a *Horn clause*.

For an arbitrary set N of clauses, using a different ordering concept for interpretations, it is also possible to identify a unique *perfect model* of N (Bachmair & Ganzinger 1991) which is the minimal model with respect to an ordering $\succ^i$ on interpretations. The ordering $\succ^i$ is the multiset extension of the *inverse* $\prec$ of the term ordering $\succ$. The existence of minimal models (with respect to $\succ^i$) for consistent, infinite sets of clauses is not trivial, as $\succ^i$ is not well-founded in general. Moreover, perfect models are minimal (the least models) with respect to subset inclusion in the sense of Definition 3.2.1. Thus representations with perfect model semantics can also be used for an approximation of blocking. However, the problem (ii), i.e. the identification of decision procedures, is difficult for non-Horn clause sets. Another problem is to verify that the perfect model of the representation is always the appropriate approximation of the candidate model.

Throughout this work, we only employ Horn clause sets for the effective approximation of blocking. We use a generalization of the well-known *immediate consequence operator* for Horn clauses without equality to Horn clauses with equality.

Definition 3.2.3 (Immediate Consequence Operator)
Let H be a set of ground Horn clauses. The *immediate consequence operator* T_H maps interpretations I to interpretations $T_H(I)$ by

$$T_H(I) = (T'_H(I))^*$$

where

$$T'_H(I) = \{u \approx v \mid \exists\, \Gamma \rightarrow u \approx v \in H \text{ with } \Gamma \subseteq I\}.$$

The lemma relates the Herbrand models of a set H of Horn clauses with the operator T_H. As a consequence, any such Herbrand model turns out to be a so-called *pre-fixpoint* of T_H, see below. Thus we may concentrate on pre-fixpoints of the consequence operator in order to investigate the Herbrand models of H.

Lemma 3.2.4
Let H be a set of (non-negative) ground Horn clauses and let I be an (equality) Herbrand interpretation. Then $I \models H$ if and only if $T_H(I) \subseteq I$.

Proof Suppose that $I \models H$ holds and that $u \approx v \in T_H(I)$. Thus $u \approx v$ is in the congruence closure of $T'_H(I)$. Suppose that the equational proof of $u \approx v$ consists of the equations $s_1 \approx t_1, \ldots, s_n \approx t_n$ in $T'_H(I)$. Hence, there are clauses $\Gamma_1 \rightarrow s_1 \approx t_1, \ldots, \Gamma_n \rightarrow s_n \approx t_n$ in H such that $\Gamma_1, \ldots, \Gamma_n \subseteq I$. Since I is a model of H it follows that the equations $s_1 \approx t_1, \ldots, s_n \approx t_n$ are true in I and therefore $u \approx v$ is also true in I if I is an equality interpretation.

Suppose that $T_H(I) \subseteq I$ holds and let C be a ground Horn clause $\Gamma \rightarrow u \approx v$ in H with $\Gamma \subseteq I$. By definition we have that $u \approx v \in T'_H(I)$ and thus $u \approx v \in T_H(I)$. We conclude that $u \approx v$ is true in I. ■

The definition introduces not only pre-fixpoints but also *fixpoints* of mappings from interpretations to interpretations. The idea is to show that T_H has, among all pre-fixpoints, also fixpoints and, moreover, by the Fixpoint Theorem (Tarski 1955) below, that there is a *least fixpoint* for T_H. The least fixpoint of T_H, as it is the least Herbrand model of H, is a suitable characterization of all Herbrand models of H.

Definition 3.2.5 ((Pre-)Fixpoint)
Let I be an (equality) Herbrand interpretation and let T be a mapping from interpretations to interpretations. If $T(I) \subseteq I$ holds, I is called a *pre-fixpoint of* T. If $T(I) = I$ holds, I is called a *fixpoint of* T.

According to the Fixpoint Theorem, any *monotone* operator has a least fixpoint. If the operator is also *continuous* then it reaches the least fixpoint after n iterations with $n < \omega$.

Definition 3.2.6 (Monotonicity, Continuity)
Let T be a mapping from interpretations to interpretations. T is called *monotone* if, for all (equality) Herbrand interpretations I and J, $I \subseteq J$ implies $T(I) \subseteq T(J)$. We call T *continuous* if, for every infinite sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$ of (equality) Herbrand interpretations

$$T(\bigcup_{n=0}^{\infty} I_n) = \bigcup_{n=0}^{\infty} T(I_n)$$

holds.

Note that a continuous mapping is monotone but not vice versa. The following lemma shows that the immediate consequence operator generalized to Horn clauses with equality is indeed continuous.

Lemma 3.2.7
Let H be a satisfiable set of Horn clauses. Then T_H is continuous.

Proof Let $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$ be an infinite sequence of (equality) Herbrand interpretations. Let I be the interpretation $\bigcup_{n=0}^{\infty} I_n$. Suppose that $u \approx v \in T_H(I)$ holds. Thus $u \approx v$ is in the congruence closure of $T'_H(I)$. Suppose that the equational proof of $u \approx v$ consists of the equations $s_1 \approx t_1, \ldots, s_m \approx t_m$ in $T'_H(I)$. Hence, there are clauses $\Gamma_1 \to s_1 \approx t_1, \ldots, \Gamma_m \to s_m \approx t_m$ in H such that $\Gamma_1, \ldots, \Gamma_m \subseteq I$. Since the sequence of interpretations is ascending with respect to subset inclusion, there is an interpretation I_k such that $\Gamma_1, \ldots, \Gamma_m \subseteq I_k$. It follows that the equations $s_1 \approx t_1, \ldots, s_m \approx t_m$ are true in $T'_H(I_k)$ which implies that $u \approx v$ is true in $T_H(I_k)$ and thus $u \approx v$ is also true in $\bigcup_{n=0}^{\infty} T_H(I_n)$.

In order to show the other direction we prove first that T_H is monotone. Suppose that I and J are (equality) Herbrand interpretations with $I \subseteq J$ and that $u \approx v$ is true in $T_H(I)$. Thus $u \approx v$ is in the congruence closure of $T'_H(I)$. Suppose that the equational proof of $u \approx v$ consists of the equations $s_1 \approx t_1, \ldots, s_m \approx t_m$ in $T'_H(I)$. Hence, there are clauses $\Gamma_1 \to s_1 \approx t_1, \ldots, \Gamma_m \to s_m \approx t_m$ in H such that $\Gamma_1, \ldots, \Gamma_m \subseteq I$ and thus $\Gamma_1, \ldots, \Gamma_m \subseteq J$. It follows that $u \approx v$ is true in $T_H(J)$ which shows that T_H is monotone.

Suppose that $u \approx v$ is true in $\bigcup_{n=0}^{\infty} T_H(I_n)$. By monotonicity of T_H and since the sequence of interpretations is ascending with respect to subset inclusion, there is an interpretation I_k such that $u \approx v$ is true in $T_H(I_k)$. We conclude that $u \approx v$ is true in $T_H(\bigcup_{n=0}^{\infty} I_n)$. ■

The *consequence closure* of the immediate consequence operator T_H constructively defines the minimal model of a satisfiable set H of Horn clauses.

Definition 3.2.8 (Consequence Closure)
Let I be an (equality) Herbrand interpretation and let T be a mapping from interpretations to interpretations. We define T^n by $T^0(I) := I$ and $T^{n+1}(I) := T(T^n(I))$. By T^ω we denote the interpretation $\bigcup_{n<\omega} T^n(\emptyset)$.

Theorem 3.2.9 (Fixpoint Theorem (Tarski 1955))
Let T be a mapping from interpretations to interpretations. If T is monotone then T has a least fixpoint which is also its least pre-fixpoint. If T is continuous then T^ω is its least pre-fixpoint and its least fixpoint.

As a corollary, we obtain that T_H^ω computes the minimal model of any satisfiable set H of Horn clauses (with equality).

Corollary 3.2.10
Let H be a satisfiable set of ground Horn clauses. Then T_H^ω is the least and thus also the minimal model of H.

In the sequel, we denote, by T^H, the minimal model of a satisfiable set H of Horn clauses.

Definition 3.2.11 (Minimal Model)
Let H be a set of (non-ground) Horn clauses. T^H denotes the interpretation $T_{H'}^\omega$, where H' is the set of all ground instances of H.

Suppose that H has actually been derived, e.g., by an *abstraction* of some other set N of clauses. Given a candidate model I^N constructed from the clauses in N, the minimal model semantics of H allow us to relate the class of Herbrand models of H to I^N using the minimal model T^H of H. Further questions like the relation of satisfiability of clauses in T^H and I^N may be studied in this regard. In the next section, we introduce a formal concept of abstractions to address these questions.

3.3 Abstraction

Soft typing is based on the observation that the derivation of reduction inferences is sufficient for refutational completeness. Blocking is a method to distinguish reduction inferences from non-reduction inferences. In general, however, blocking is undecidable. More precisely, the problem whether a clause C in a set N of clauses is true or false in some candidate model I^N is undecidable. However, there might be *approximations* of I^N for which this problem becomes decidable. Informally, the approximation of some object is a representation of the object which is in some sense simpler than the original. The motivation of an approximation is to achieve a representation of the original for which more properties are decidable while some properties of the original are preserved in the approximation. For instance, it might be desirable that the satisfiability of C in the approximation implies the satisfiability of C in the original. An *abstraction* is a description of how to obtain the approximation of the original. We define an abstraction operationally as a transformation rule which copies its premise from a set of clauses, transforms the premise into one or more conclusions, and finally adds the conclusions to another set of clauses.

Definition 3.3.1 (Abstraction)
An *abstraction* is written as a rule of the form

$$\text{Abstract}\,\frac{C}{\begin{array}{c}D_1\\ \vdots\\ D_n\end{array}}$$

where C is the *premise* and the $D_1, \ldots, D_n$ are the *conclusions* of the abstraction. Let N and M be sets of clauses and let C be a clause in N. The abstraction transforms C in N into the conclusions $D_1, \ldots, D_n$ and adds the conclusions to M.

If M is the result of an exhaustive application of some abstraction to N and if, in addition, the class of models of M satisfies the conditions of the following definition, then we call M an approximation. More generally, we define approximations, independently of a particular representation, as a class of interpretations with the following properties.

Definition 3.3.2 (Approximation)
Let I be an (equality) interpretation and let $\mathcal{F}$ be a class of formulae. We call a class of (equality) interpretations $\mathcal{J}$ an *approximation of I for $\mathcal{F}$* if for the class $\mathcal{F}$ of formu-

lae either validity (satisfiability) in $\mathcal{J}$ implies validity (satisfiability) in I, or, conversely, validity (satisfiability) in I implies validity (satisfiability) in $\mathcal{J}$. An approximation $\mathcal{J}$ is called *decidable for* $\mathcal{F}$ if validity (satisfiability) in $\mathcal{J}$ is decidable for the formulas in $\mathcal{F}$.

Suppose that H is a satisfiable set of Horn clauses which has been obtained by some abstraction from another set N of clauses. Since the minimal model T^H of H is a subset of any other model from the class $\mathcal{J}$ of Herbrand models of H, it is interesting to see if some candidate model I^N of N is included in T^H, or, in other words, if T^H is an *upper approximation* of I^N.

Definition 3.3.3 (Upper Approximation)
Let $\mathcal{J}$ be an approximation of an (equality) interpretation I for some class of formulae. An (equality) interpretation $J \in \mathcal{J}$ is called an *upper approximation* of I, if $I \subseteq J$.

Example 3.3.4
Suppose that $\mathcal{J}$ is an approximation in which satisfiability of conjunctions of atoms is decidable. Let N be a set of ground clauses and let I be a model functor. Consider a clause $C = A_1, \ldots , A_n \rightarrow \Delta$. We assume that $J \in \mathcal{J}$ is an upper approximation of any $I^{N_{C\sigma}}$, for the ground instances $C\sigma$ of C. If $J \not\models \exists\,(A_1 \wedge \ldots \wedge A_n)$ then $I^{N_{C\sigma}} \not\models A_1\sigma \wedge \ldots \wedge A_n\sigma$, for any ground instance $C\sigma$ of C. Thus any ground instance $C\sigma$ of C is true in $I^{N_{C\sigma}}$ which implies that any clausal inference with premise C is blocked in N.

Definition 3.3.5 (Lower Approximation)
An (equality) interpretation J is called a *lower approximation* of another (equality) interpretation I, if $J \subseteq I$.

Example 3.3.6
Suppose that $\mathcal{J}$ is an approximation in which validity of disjunctions of atoms is decidable. Let N be a set of ground clauses and let I be a model functor. Consider a clause $C = \Gamma \rightarrow A, A_1, \ldots , A_n$. We assume that $J \in \mathcal{J}$ is a lower approximation of any I^{N_D}, for the ground instances D of C. If $J \models \forall\,(A_1 \vee \ldots \vee A_n)$ then $I^{N_D} \models \forall\,(A_1 \vee \ldots \vee A_n)$, for any ground instance D of C. Thus any ground instance D of C is true in I^{N_D} which implies that C cannot be the side premise of a non-blocked inference with A.

A satisfiable set H of Horn clauses is of particular interest since H enjoys the existence of a unique minimal model which also implies that decidability results with respect to the class of Herbrand models of H are, in general, easier to obtain than for sets of non-Horn clauses. For decidability purposes we consider *Horn theories* which are finite satisfiable sets of Horn clauses.

Definition 3.3.7 (Horn Theory)
A *Horn theory* is a finite satisfiable set of Horn clauses.

In the sequel, by a set of clauses, we always mean a finite set of clauses. However, note that in this section we are not yet concerned with decidability aspects. They are treated in Chapter 4 through 7. Here, we are interested in possible abstractions from arbitrary sets of clauses, i.e., in particular, the model properties of the Horn theories with respect to their originals. The interesting abstraction parameters are therefore on the level of the logical structure of clauses, i.e. strictly above the term structure of the clauses. For example, an obvious abstraction to obtain a Horn theory from a set of general clauses is to transform disjunctions into conjunctions as the *Distribution* rule suggests.

Definition 3.3.8
The following abstraction is called *Distribution*:

$$\text{Abstract}\frac{\Gamma \to L_1, \ldots, L_n}{\begin{array}{c}\Gamma \to L_1\\ \vdots \\ \Gamma \to L_n\end{array}}$$

where (i) L_i is either an atom or an equation for all i with $1 \leq i \leq n$ and (ii) $n > 0$.

Only clauses with a non-empty succedent are subject to Distribution such that the result is indeed a satisfiable set of Horn clauses. We call a Horn theory H obtained by Distribution from a set N of clauses a *static Horn theory* since H is in some sense sufficiently conservative to approximate even some candidate model of another set M of clauses which has been obtained by inferring inferences from N and/or by simplifying N. We will clarify this point in Section 3.4.

Definition 3.3.9 (Static Horn Theory)
Let N be a set of clauses. Let H be the result of an exhaustive application of the Distribution rule to N. We call H the *static Horn theory of* N.

The following easy proposition shows that the minimal model of the static Horn theory of some clause set N is an upper approximation of the candidate model I^N at least with respect to the model functor for ordered resolution. The generalization to the model functor for superposition is also not difficult.

Proposition 3.3.10
Let I be the model functor for ordered resolution. Let N be a set of clauses which does not contain the empty clause and let M be the set of all ground instances of clauses in N. Let H be the static Horn theory of N. The minimal model T^H is an upper approximation of I^M.

Proof The proof is by induction on an admissible atom ordering $\succ_a$. Let C be a ground instance $\Gamma\sigma \to \Delta\sigma, A\sigma$ in M of a clause in N which produces $A\sigma$ into I^M. Note that $A\sigma$ must be strictly maximal in C. We have to show that $A\sigma \in T^H$. As the induction hypothesis we assume that if $B \in I^M$ then $B \in T^H$ for all ground atoms B with $A\sigma \succ_a B$. By definition, H contains the clause $\Gamma \to A$. Since C is productive we have that $\Gamma\sigma \subseteq I^M$ and thus, by the induction hypothesis, $\Gamma\sigma \subseteq T^H$. We conclude that $A\sigma$ is also in the minimal model T^H. ■

Only productive clauses which are, in particular, reductive for the produced atom, participate in the construction of the candidate model for ordered resolution as well as for superposition. This suggests the following ordered refinement of the Distribution rule to strictly maximal positive literals, which still transforms sufficiently many clauses for an appropriate upper approximation, see below.

Definition 3.3.11
The following abstraction is called *Ordered Distribution*:

$$\text{Abstract}\frac{\Gamma \to \Delta, L}{\Gamma \to L}$$

where (i) L is either an atom or an equation and (ii) L is strictly maximal in the premise.

In contrast to a static Horn theory, a *dynamic Horn theory* which has been obtained by Ordered Distribution does not provide for an upper approximation for arbitrary descendants of the original set, since different clauses may become productive. However, the repeated revision of a dynamic Horn theory in a theorem proving process may result in an improved upper approximation which is closer to the current original than the static Horn theory.

Definition 3.3.12 (Dynamic Horn Theory)
Let N be a set of clauses. Let H be the result of an exhaustive application of the Ordered Distribution rule to N. We call H the *dynamic Horn theory of* N.

Similar to Proposition 3.3.10 for static Horn theories, it is easy to show that the minimal model of the dynamic Horn theory of N is in fact an upper approximation of the candidate model I^N with respect to the model functor for ordered resolution. The generalization to the model functor for superposition is also not difficult.

Proposition 3.3.13
Let I be the model functor for ordered resolution. Let N be a set of clauses which does not contain the empty clause and let M be the set of all ground instances of clauses in N. Let H be the dynamic Horn theory of N. The minimal model T^H is an upper approximation of I^M.

Proof Since the candidate model I^M contains only the atoms which are strictly maximal in the corresponding productive clauses the minimal model T^H is an upper approximation of I^M, c.f. Proposition 3.3.10. ■

The previous discussion has been dedicated to certain abstractions of arbitrary sets of clauses and, in particular, to the properties of the according approximations. The rest of this section is devoted to a general concept for the connection of approximations with the blocking concept of soft typing, i.e., in essence, the effective identification of clauses which can be subject to blocking with respect to some approximation. We propose the concept of a *typing function* similar to the selection function for ordered resolution and superposition. In general, only certain parts, e.g., the antecedent, of a clause C may be evaluated with respect to some approximation. The typing function indicates which literal occurrences in C are to be checked for the control of the blocking state of C.

Definition 3.3.14 (Typing Function)
A *typing function* T assigns to each ground clause a possibly empty set of occurrences of literals. If C is a clause, the literal occurrences in $T(C)$ are called *typed*. $T(C) = \emptyset$ indicates that no literal is typed.

In Chapter 4 through 7, we study decidable approximations which allow for an effective computation of, e.g., the emptiness of the extension of certain predicates or the unifiability of first-order terms with respect to non-trivial equational theories. This motivates the definition of a *type* which can be seen as the negative (conditional) part of a clause.

Definition 3.3.15 (Type)
The existential closure of a conjunction of atoms (equations) is called an (*equational*) *type*. Let C be a clause $\Gamma \rightarrow \Delta$ and let T be a typing function. If $T(C) \subseteq \Gamma$ then $T(C)$ is called the (*equational*) *type of* C. We call C a *typed clause* whenever $T(C)$ is not empty.

A *blocked type* is a type of a clause C which is false in some interpretation of a class of interpretations. In this way, a blocked type serves as a sufficient constraint to block inferences from or to C, see below.

Definition 3.3.16 (Blocked Type)
Let $\mathcal{I}$ be a class of (equality) interpretations. The type Γ of a clause $\Gamma, \Lambda \rightarrow \Delta$ is called *blocked with respect to* $\mathcal{I}$, if $\mathcal{I} \not\models \exists\, x_1, \ldots, x_n\, (\Gamma)$ where $\{x_1, \ldots, x_n\} = \mathit{vars}(\Gamma)$.

The following lemma shows that any inference from or to a clause with a blocked type is blocked with respect to any clausal inference system which has the (strong) reduction property.

Lemma 3.3.17
Let Π be a clausal inference system which has the (strong) reduction property for counterexamples with respect to a model functor I. Let N be a set of clauses and let M be the set of all ground instances of clauses in N. Let C be a clause $\Gamma, \Lambda \rightarrow \Delta$ where Γ is the type of C. If Γ is blocked with respect to $\{I^M\}$ then any inference in Π either from C or with C as conclusion is blocked.

Proof Suppose that Γ is blocked with respect to I^M, i.e. $I^M \not\models \exists\, x_1, \ldots, x_n\, (\Gamma)$ where $\{x_1, \ldots, x_n\} = \mathit{vars}(\Gamma)$. Thus we may also assume that $I^{M_{C\sigma}} \not\models \exists\, x_1, \ldots, x_n\, (\Gamma)$, for all ground instances $C\sigma$ of C. Suppose that there is a ground instance $C\sigma$ of C where $C\sigma$ is false in $I^{M_{C\sigma}}$. However, in this case we have that $I^{M_{C\sigma}} \models \Gamma\sigma$ which is a contradiction to the assumption that Γ is blocked. Thus we may assume that any $C\sigma$ is true in $I^{M_{C\sigma}}$

and I^M. In particular, any $C\sigma$ is a non-productive clause. It follows that any inference in Π from C or with C as conclusion is blocked. ■

In the following proposition we summarize the discussion on types and remark that static and dynamic Horn theories are appropriate approximations for the blocking of inferences which involve typed clauses. Any Horn theory whose minimal model is an upper approximation of the static or dynamic Horn theory may also be used in this way.

Proposition 3.3.18
Let I be the model functor for ordered resolution or superposition. Let N be a set of clauses which does not contain the empty clause and let M be the set of all ground instances of clauses in N. Let H be the static or dynamic Horn theory of N. Let H_u be another Horn theory such that the minimal model T^{H_u} is an upper approximation of the minimal model T^H. Let C be a clause $\Gamma, \Lambda \rightarrow \Delta$ where Γ is the type of C. If Γ is blocked with respect to H_u, then Γ is blocked with respect to H which implies that Γ is blocked with respect to $\{I^M\}$.

Since arbitrary Horn theories are, in general, undecidable it is important that further (decidable) approximations can be incorporated for effective blocking. More precisely, by a decidable Horn theory H, we mean that there is some theory of formulae (types) ϕ over H for which $T^H \models \phi$ is decidable. We devote Chapter 4 through 7 to the identification of decidable theories of certain types.

Definition 3.3.19 (Theory of Types)
Let H be a Horn theory and let $\mathcal{F}$ be a class of formulae. Let $\mathcal{F}_H$ be the set of formulae in $\mathcal{F}$ which contain only predicate and function symbols from H. We call $\mathcal{F}_H$ the *theory of $\mathcal{F}$ over H* and say that $\mathcal{F}_H$ is *decidable*, if $T^H \models \phi$ is decidable for any formula ϕ in $\mathcal{F}_H$. If $\mathcal{F}$ is the class of all first-order formulae, $\mathcal{F}_H$ is called the *first-order theory over H*. We call $\mathcal{F}_H$ the *theory of types over H*, if $\mathcal{F}$ is the class of types.

3.4 Effectiveness

Soft typing for a clausal inference system Π has been introduced as a refinement of Π where, in particular, dispensable inferences are blocked dynamically by some model hypothesis. Soft typing is refutationally complete if Π enjoys the reduction property for counterexamples. This property has been shown for various calculi by an analysis of the *local* behavior of the inference systems. In contrast, the compatibility of soft typing with redundancy can be proved by an analysis of the result of a *theorem proving derivation*, i.e. the final (weakly) saturated set of clauses. Systems which apply to the strong reduction property are possible candidates for soft typing with redundancy.

In order to speak precisely about theorem proving processes we need a formalization of the notion of a theorem proving derivation for soft typing. The following definition of theorem proving derivations has been originally proposed by Bachmair & Ganzinger

(1994). Note that, as a prerequisite to the discussion below, that redundancy of clauses and inferences is stable under the addition of new clauses as well as under the deletion of redundant clauses.

Lemma 3.4.1 (Bachmair & Ganzinger (1994))
Let N and N' be two sets of clauses and let Π be a clausal inference system.

1. If $N \subseteq N'$ then any inference in Π, or any clause, which is redundant in N is also redundant in N'.
2. If $N \subseteq N'$ and all clauses in $N' \setminus N$ are redundant in N' then any inference in Π, or any clause, which is redundant in N' is also redundant in N.

Proof Part (1) is obvious. For Part (2) recall that for a redundant clause C in N there are always non-redundant clauses in N_C which already entail C. Thus we may assume that the conclusion of any inference π in Π, or any clause C, which is redundant in N' is entailed by non-redundant clauses. Since all clauses in $N' \setminus N$ are assumed to be redundant in N' we conclude that π and C are redundant in N. ■

As a consequence, we define a theorem proving derivation by some clausal inference system Π as the process of adding conclusions from inferences in Π and deleting redundant clauses triggered by simplification.

Definition 3.4.2 (Theorem Proving Derivation)
Let Π be a clausal inference system. A (finite or countably infinite) sequence N_0, N_1, N_2, ... of sets of ground clauses is called a *theorem proving derivation* by Π if each set N_{i+1} can be obtained from its predecessor N_i either (i) by adding a set of clauses that can be deduced by Π from N_i, or else (ii) by deleting a subset of clauses which are all redundant in N_i.

The set $N_\infty = \bigcup_j \bigcap_{k \geq j} N_k$ is called the *limit* of the derivation. Clauses in N_∞ are called *persisting*. A theorem proving derivation is called *fair* with respect to Π if every inference in Π from premises in N_∞ is redundant in $\bigcup_j N_j$. It is called *weakly fair* with respect to Π if every inference π in Π from premises in N_∞, where π is not blocked in N_∞, is redundant in $\bigcup_j N_j$.

From an operational point of view, fairness requires an inference system not to delay any non-redundant inference infinitely. A suitable approximation of (weak) fairness is to consider periodically inferences from the currently smallest clause C with respect to its size, i.e. the number of symbols in C. Since the ordering by size is well-founded each clause will eventually be considered. This concept also solves the problem of soft typing with respect to non-monotone sequences of candidate models in theorem proving derivations. In general, it is an open problem which properties of candidate models are monotone in any derivation. In particular, it might be possible that a clause C is blocked and non-blocked periodically but is never considered in inferences during the non-blocked phase. However, there are again only finitely many non-blocked clauses smaller than C with respect to size which implies that C will eventually be considered unless C is blocked forever. See Section 3.5 for a discussion of (weak) fairness of theorem proving derivations on the non-ground level.

The following lemma does not require the inference system to be sound. Below, we exploit this freedom for a generalization of theorem proving derivations to derivation trees where unsound addition of clauses is possible under certain circumstances.

Lemma 3.4.3
Let Π be a clausal inference system. Let N_0, N_1, N_2, $\ldots$ be a (weakly) fair theorem proving derivation with respect to Π. Then (i) the clauses in $(\bigcup_j N_j) \setminus N_\infty$ are redundant in the limit N_∞ which (ii) is (weakly) saturated up to redundancy.

Proof If the sequence of N_i is a theorem proving derivation, any clause C in $(\bigcup_j N_j) \setminus N_\infty$ is redundant in some N_j, hence by part (1) of Lemma 3.4.1 redundant in $\bigcup_j N_j$ which shows (i). If the theorem proving derivation by Π is (weakly) fair, every (non-blocked) inference in Π from N_∞ is redundant in $\bigcup_j N_j$ and, therefore, using part (2) of Lemma 3.4.1, redundant in N_∞. Thus the limit N_∞ is (weakly) saturated up to redundancy which implies (ii). ■

We arrive at the main theorem for (soft typing for) clausal inference systems on the ground level. Any (weakly) fair theorem proving derivation with respect to a sound system which has the (strong) reduction property will eventually derive the empty clause from any unsatisfiable set N_0 of clauses. Conversely, if N_0 is satisfiable, then I^{N_∞} is a model of N_0 ($I^{\bigcup_j N_j}$ is a model of N_0).

Theorem 3.4.4
Let Π be a clausal inference system which is sound and has the (strong) reduction property. Let N_0, N_1, N_2, $\ldots$ be a (weakly) fair theorem proving derivation with respect to Π. If $\bigcup_j N_j$ does not contain the empty clause, then N_∞ is (weakly) saturated up to redundancy and N_0 is consistent.

Proof Since the N_i constitute a (weakly) fair theorem proving derivation with respect to Π, by part (ii) of Lemma 3.4.3, the set N_∞ is (weakly) saturated up to redundancy. In case Π enjoys the reduction property and N_∞ is saturated up to redundancy, by Theorem 3.1.25, we have that I^{N_∞} is a model of N_∞. Since, by part (i) of Lemma 3.4.3, the clauses in $(\bigcup_j N_j) \setminus N_\infty$ are redundant in N_∞, we also have that I^{N_∞} is a model of $\bigcup_j N_j$. If Π has the strong reduction property and N_∞ is weakly saturated up to redundancy, by Theorem 3.1.24, we have that $I^{\bigcup_j N_j}$ is a model of $\bigcup_j N_j$. ■

The main theorem provides a general framework for deferring inferences due to blocking. The effective use of blocking requires approximations of candidate models. Another component, when we consider theorem proving derivations, is to control the computation of approximations relative to the progress of the derivation. Recall that, in the previous section, we have already introduced static and dynamic Horn theories. In general, we have to expect that approximations which have to be revised dynamically to be appropriate provide for a better approximation than static approximations. However, the dynamic re-computation can be expensive. A *static approximation* corresponds to the border where a one-time computation at the beginning of the derivation is sufficient for an appropriate approximation of all upcoming candidate models.

Definition 3.4.5 (Static Approximation)
Let Π be a clausal inference system and let I be a model functor. Let N be a set of clauses and let N_0 be the set of all ground instances of clauses in N. Let $N_0, N_1, N_2, \ldots$ be any (weakly) fair theorem proving derivation with respect to Π. Let $\mathcal{S}$ be a class of (equality) interpretations and let $\mathcal{F}$ be a class of formulae. We call $\mathcal{S}$ a (*partial*) *static approximation for* $\mathcal{F}$ if the class $\mathcal{S}$ is an approximation of each I^{N_i} (I^{N_C} where $N = N_i$, for any clause C) for $\mathcal{F}$ where N_i does not contain the empty clause. A (partial) static approximation is called *decidable* if $\mathcal{S}$ is a decidable approximation.

Most importantly, clauses which have a type that is blocked with respect to a static approximation can be permanently excluded from the search process. Static Horn theories are an example of (partial) static approximations for types with respect to any theorem proving derivation by soft typing for ordered resolution. The generalization to soft typing for superposition is straightforward as Proposition 3.3.10 can easily be extended in this regard.

Lemma 3.4.6
Let I be the model functor for ordered resolution. Let N be a set of clauses and let N_0 be the set of all ground instances of clauses in N. Let H be the static Horn theory of N. Let $N_0, N_1, N_2, \ldots$ be a (weakly) fair theorem proving derivation with respect to ordered resolution. Then H is a static approximation for types.

Proof The proof is by induction on the length of a fair theorem proving derivation. By Proposition 3.3.10 the minimal model T^H is an upper approximation of I^{N_0}. Thus the unsatisfiability of types in H implies the unsatisfiability in I^{N_0}. It follows that H is an approximation of I^{N_0} for types. As the induction hypothesis we assume that H is an approximation of I^{N_i} for types. In particular, we may assume that the hypothesis holds for any admissible atom ordering $\succ_a$. That is, T^H contains all atoms which occur in the succedent of any clause in N_i. Consequently, no matter if N_{i+1} has been obtained by adding logical consequences or by deleting redundant clauses, H is also an approximation of $I^{N_{i+1}}$ for types. ■

In contrast to static approximations, a *dynamic approximation* involves the dynamic character of the approximation in the sense that the approximation may at most be revised at each step in the derivation. However, the frequency of updates is left open and may vary from approximation to approximation.

Definition 3.4.7 (Dynamic Approximation)
Let Π be a clausal inference system and let I be a model functor. Let N be a set of clauses and let N_0 be the set of all ground instances of clauses in N. Let N_0, N_1, N_2, … be any (weakly) fair theorem proving derivation with respect to I. Let $\mathcal{D}$ be a (finite or countably infinite) sequence $\mathcal{J}_0$, $\mathcal{J}_1$, $\mathcal{J}_2$, … of classes of (equality) interpretations. Let $\mathcal{F}$ be a class of formulae. We call $\mathcal{D}$ a *(partial) dynamic approximation for* $\mathcal{F}$ if, for all i, the class $\mathcal{J}_i$ is an approximation of I^{N_i} (I^{N_C} where $N = N_i$, for any clause C) for $\mathcal{F}$ where N_i does not contain the empty clause. A (partial) dynamic approximation is called *decidable* if each $\mathcal{J}_i$ is a decidable approximation.

Clauses which have a type that is blocked with respect to a dynamic approximation may only be temporarily excluded from the inference process. Note that any static approximation is also a dynamic approximation but not vice versa. In the context of derivations by ordered resolution (superposition), a dynamic Horn theory represents a particular dynamic approximation for types which, in general, has to be revised at each step in the derivation. The addition of logical consequences as well as the deletion of redundant information may affect the set of reductive clauses from which the candidate model is constructed.

Lemma 3.4.8
Let I be the model functor for ordered resolution. Let N_0 be a set of clauses and let M_0 be the set of all ground instances of clauses in N_0. Let M_0, M_1, M_2, … be a (weakly) fair theorem proving derivation with respect to ordered resolution where the N_i denote the corresponding sets on the non-ground level. The sequence of dynamic Horn theories H_i of N_i is a dynamic approximation for types.

Proof Due to Proposition 3.3.13 each minimal model T^{H_i} is an upper approximation of the according interpretation I^{M_i}. Following the proof of Lemma 3.4.6 each H_i is an approximation of the according (equality) interpretation I^{M_i} for types. ■

The previous discussion on theorem proving derivations and their approximations essentially involves *linear* theorem proving processes. At each point in time either deduction or simplification steps are possible. However, following the general principle of divide and conquer, branching of derivations can be useful. We introduce, as a generalization of theorem proving derivations, so-called *theorem proving derivation trees* by admitting deduction steps to split a set of clauses N into $k \geq 1$ alternatives $N \cup M_1, \ldots, N \cup M_k$ of clause sets $N \cup M_i$ such that N is consistent if and only if $N \cup M_i$ is consistent for some $1 \leq i \leq k$. For example, we might split $N \cup \{C \vee D\}$ on a clause $C \vee D$ with variable-disjoint subclauses C and D into two branches $N \cup \{C \vee D, C\}$ and $N \cup \{C \vee D, D\}$. Note that in both subtrees of $N \cup \{C \vee D\}$ the clause $C \vee D$ becomes redundant and can be deleted. In this way, resolution-based theorem proving methods may be generalized to Davis-Putnam-like procedures, which significantly improves the performance of such methods at least on propositional problems. If clauses are split into subclauses which share some variables, then any derivation in the according subtrees has to accomplish the dependencies caused by the instantiation of common variables. This approach is related to tableaux-based theorem proving methods. Other applications of derivation trees can be found in Section 3.6.2.

Definition 3.4.9 (Theorem Proving Derivation Tree)
Let Π be a clausal inference system. Let N^Θ be a tree such that each node in N^Θ is labeled by a set of ground clauses. We call N^Θ a *theorem proving derivation tree* if the labels on each branch N^θ in N^Θ induce a (finite or countably infinite) sequence N_0^θ, N_1^θ, N_2^θ, ... of sets of ground clauses and, for each non-leaf node N in N^Θ, its successors $N_1, \ldots, N_k$ are either obtained from N (i.a) by adding a set of clauses that can be deduced by Π from N or (i.b) by deleting a subset of clauses which are all redundant in N, or else (ii) each N_i is a set $N \cup M_i$ of ground clauses where N is consistent if and only if N_i is consistent for some $1 \leq i \leq k$.

Note that a branch in a derivation tree is not exactly a theorem proving derivation since (ii) may introduce arbitrary new clauses. However, limit and fairness of derivations naturally carry over to branches in a derivation tree. The set $N_\infty^\theta = \bigcup_j \bigcap_{k \geq j} N_k^\theta$ is called the *limit* of the branch θ. A theorem proving derivation tree is called *fair* with respect to Π if, for every branch N^θ in N^Θ, every inference in Π from premises in N_∞^θ is redundant in $\bigcup_j N_j^\theta$. It is called *weakly fair* with respect to Π if, for every branch N^θ in N^Θ, every non-blocked inference in Π from premises in N_∞^θ is redundant in $\bigcup_j N_j^\theta$. The following theorem is a generalization of the main Theorem 3.4.4 to derivation trees.

Theorem 3.4.10
Let Π be a clausal inference system which is sound and has the (strong) reduction property. Let N^Θ be a (weakly) fair theorem proving derivation tree with respect to Π. Then the limit N_∞^θ of each branch θ in N^Θ is (weakly) saturated, and N_0 is inconsistent if and only if on every branch θ the empty clause is in $\bigcup_j N_j^\theta$.

Proof We may use part (ii) of Lemma 3.4.3 to derive that for each branch θ in N^Θ, the limit N_∞^θ is (weakly) saturated. Note that Lemma 3.4.3 is true regardless of the soundness of deduction steps and can, therefore, be applied to any branch in the deduction tree. Suppose that there is a branch θ such that $\bigcup_j N_j^\theta$ does not contain the empty clause. In case Π enjoys the reduction property and N_∞^θ is saturated up to redundancy, by Theorem 3.1.25, we have that $I^{N_\infty^\theta}$ is a model of N_∞^θ. Since, by part (i) of Lemma 3.4.3, the clauses in $(\bigcup_j N_j^\theta) \setminus N_\infty^\theta$ are redundant in N_∞^θ, we also have that $I^{N_\infty^\theta}$ is a model of $\bigcup_j N_j^\theta$ and, therefore, a model of N_0. If Π has the strong reduction property and N_∞^θ is weakly saturated up to redundancy, by Theorem 3.1.24, we have that $I^{\bigcup_j N_j^\theta}$ is a model of $\bigcup_j N_j^\theta$ and, therefore, a model of N_0.

On the other hand, suppose that N_0 is consistent. By the definition of theorem proving derivation trees splitting preserves consistency on at least one branch θ. Since each deduction step in a theorem proving derivation is sound, $\bigcup_j N_j^\theta$ does not contain the empty clause. ∎

A generalization of static and dynamic approximations to derivation trees is straightforward. From a practical point of view, there are encouraging results using the theorem prover Spass (Weidenbach, Meyer, Cohrs, Engel & Keen 1998) which computes tree-like derivations with splitting on variable-disjoint clause parts.

3.5 Lifting

Soft typing for clausal inference systems has been introduced as a general framework for semantically guided theorem proving. The definition of soft typing on ground clauses and the appropriate notions of redundancy, saturation, and derivations allow for a clear investigation of the various properties of this concept. The *lifting problem*, on the other hand, involves questions which arise when approximating an inference system defined on the ground level by an appropriately *lifted* version of the inference system on the non-ground level. For an effective use of soft typing for non-ground clauses, we may lift soft typing by the following approximations of blocking, redundancy, and theorem proving derivations. Note that by a ground instance $\pi\sigma$ of some inference π from a clausal inference system Π we mean the ground inference in Π obtained from π by instantiating the premises and conclusion of π with the same substitution σ.

Definition 3.5.1 (Blocked Inference (Lifted))
Let I be a model functor and let N be a set of clauses. Let Π be a clausal inference system. Suppose that M is the set of ground instances of N. We say that an inference π from Π with a clause C as main premise and a clause C' as side premise is *blocked in* N, if for all ground instances $\pi\sigma$ of π from Π we have that (i) $C'\sigma$ is a counterexample for I^M, or (ii) $C'\sigma$ is true in $I^{M_{C'\sigma}}$, or (iii) $C\sigma$ is true in I^M.

Then soft typing for a clausal inference system Π on the non-ground level is the refinement of Π with respect to a set N of clauses which contains only the non-blocked inferences in Π. In order to combine soft typing with redundancy we need a suitable approximation of redundant clauses and inferences on the non-ground level.

Definition 3.5.2 (Redundant Clause (Lifted))
Let N be a set of clauses. A clause C is called *redundant in* N if all ground instances of C are redundant in the set of ground instances of N.

Definition 3.5.3 (Redundant Inference (Lifted))
Let N be a set of clauses. The inference π of a clausal inference system Π is called *redundant in* N if all ground instances of π from Π are redundant in the set of ground instances of N.

The generalization of (weak) saturation up to redundancy to the non-ground level is immediate. A *general theorem proving derivation* N_0, N_1, N_2, $\ldots$ is a derivation on the non-ground level in the sense that the sequence of sets of ground instances of each N_i forms a theorem proving derivation.

Definition 3.5.4 (General Theorem Proving Derivation)
Let Π be a clausal inference system. A (finite or countably infinite) sequence N_0, N_1, N_2, … of sets of clauses is called a (*general*) *theorem proving derivation* by Π if the corresponding sequence M_0, M_1, M_2, …, where each M_i is the set of ground instances of N_i, is a theorem proving derivation.

A (general) theorem proving derivation N_0, N_1, N_2, … is called (*weakly*) *fair* with respect to Π if the corresponding sequence M_0, M_1, M_2, …, where each M_i is the set of ground instances of N_i, is a (weakly) fair theorem proving derivation. Note that a definition of (weak) fairness as in the ground case does not work for the non-ground case since the set of ground instances of N_∞ is, in general, not equal to M_∞ but may be a strict subset of M_∞. For instance, the set of ground instances of N_∞ may be a strict subset of M_∞ if there are non-ground clauses, say C and a variant D of C, which do not persist but which are periodically added and removed by, e.g., non-strict subsumption. However, the ground instances of C (D) can be persisting clauses in the ground derivation whenever either C or D is present in each N_i. We observe that a general theorem proving derivation is (weakly) fair if there is some clause set $N \subseteq \bigcup_j N_j$ where M_∞ is a subset of the set of ground instances of N such that every inference π in Π from premises in N (where π is not blocked in N) is redundant in $\bigcup_j N_j$.

Similar to the ground case, a suitable approximation of (weak) fairness is to consider periodically inferences from the currently smallest clause C with respect to its size, i.e. the number of symbols in C. Since the ordering by size is also well-founded for non-ground clauses each (ground) clause will eventually be considered. Beside this particular selection scheme, we may also associate a counter to each clause that is increased each time the clause changes its blocking status or is involved in a non-strict subsumption step. A clause with a counter beyond a certain threshold is then excluded from non-strict subsumption or blocking.

An important property to achieve refutational completeness on the non-ground level is that any non-redundant ground inference π in Π must be *liftable* in the sense that π is the ground instance of a (non-ground) inference in Π. In the literature, this particular problem is usually referred to as the *lifting problem*. For resolution on the non-ground level, it is unification which provides for a suitable approximation such that any inference by (Negative) Resolution or Factoring is liftable, even the redundant inferences. For superposition, only the non-redundant inferences are liftable which is, however, sufficient. See also (Bachmair & Ganzinger 1997) for the lifting problem of (strict, basic) superposition. We obtain as a corollary of the main theorem 3.4.4 that (soft typing for) semantic resolution, ordered resolution, and superposition are refutationally complete for sets of non-ground clauses.

Corollary 3.5.5
Let N_0, N_1, N_2, … be a (weakly) fair general theorem proving derivation with respect to semantic resolution R_S, ordered resolution R_O, or superposition S. Then $\bigcup_j N_j$ is (weakly) saturated up to redundancy and N_0 is consistent if $\bigcup_j N_j$ does not contain the empty clause.

The generalization of theorem proving derivation trees to the non-ground level is straightforward. The notion of static and dynamic approximations has been defined for any (weakly) fair theorem proving derivation on the ground level. The generalization to the non-ground level is obvious. In the sequel, we will speak of (non-ground) derivations in the context of static and dynamic approximations, as the approximations are usually obtained from non-ground sets of clauses.

3.6 Related Work

Soft typing for clausal inference system has been introduced as a general framework for the semantically guided inference process with respect to the knowledge about certain model hypotheses. Gelernter (1960) has shown by his geometry theorem prover that already some limited understanding of the problem set may have some impact on syntactic proof procedures. However, the prover was restricted to a particular domain within plane geometry. The work by Slagle (1967) on semantic resolution marks the beginning of semantically guided resolution-based inference systems with respect to arbitrary first-order model domains. Basically, inferences may be restricted to counterexamples of the particular (candidate) model. Reiter (1976) proposes a similar generalization for natural deduction systems, like the geometry theorem prover from (Gelernter 1960), to arbitrary first-order models. Following these ideas, another theorem proving concept called *hierarchical deduction*, which is related to model elimination (Loveland 1969), is enhanced by general models in (Wang 1985, Wang & Bledsoe 1987).

However, effectiveness of semantics requires for a more involved analysis of the (candidate) models, since the identification of counterexamples is, of course, undecidable, in general. In Chapter 4 through 7, we propose methods to infer automatically *abstractions* of candidate models for approximating validity and/or satisfiability of clauses in these models. The concept is related to methods which have been called *soft typing* (Cartwright & Fagan 1991, Cartwright & Felleisen 1996) in the programming language area. As motivated by Frühwirth et al. (1991), we use certain (decidable) sets of Horn clauses (with equality) to represent effective approximations of candidate models.

Our concept of soft typing for clausal inference systems is, however, not restricted to these particular model representations. For example, a Herbrand model can be approximated by a finite set of ground (or constrained) unit clauses where truth as well as falsity of atoms may explicitly be represented. Models of this kind are generated by a resolution-based inference system for (dis-)equality constrained clauses (Caferra & Zabel 1990, Caferra & Zabel 1992). Similar to semantic resolution, the resolution inferences can be filtered with respect to satisfiability in the model (Caferra & Peltier 1995*b*, Caferra & Peltier 1995*a*) where effectiveness is achieved by the decidable constraint language. As a consequence, the approach can be seen as being essentially semantic resolution for the enriched language of (dis-)equality constrained clauses.

Leitsch (1997) discusses automated model building based on hyper-resolution decision procedures. The model building process results for certain classes of clause sets in ground atom representations of Herbrand models. We also refer to the comprehensive discussion of this approach in (Fermüller & Leitsch 1996).

3.6.1 Soft Typing for Semantic Resolution

A standard perspective of refutational theorem proving is to view refutational proof procedures as reduction inference systems with respect to a well-founded ordering. The construction of a particular candidate model is only implicitly assumed as some vehicle to show the refutational completeness. In contrast, the model generation paradigm motivates refutational theorem proving by an explicit construction of the candidate models. A model generation procedure identifies counterexamples and extends the model hypothesis accordingly in order to satisfy the counterexamples.

A proof procedure which has been developed in the spirit of model generation is the semantic hyper-linking method (Plaisted, Alexander, Chu & Lee 1992, Chu & Plaisted 1994*a*, Chu & Plaisted 1994*b*). There is also a generalization to ordered semantic hyper-linking (Plaisted 1994, Plaisted & Zhu 1997) which incorporates orderings on ground atoms. We argue below that this concept is an instance of soft typing for semantic resolution since (i) ordered semantic hyper-linking constructs exactly the candidate model obtained from the model functor for semantic resolution and (ii) the saturation criterion of ordered semantic hyper-linking is an instance of saturation with respect to soft typing for semantic resolution.

Suppose that N is a set of clauses and that J is a model such that satisfiability of (ground) literals with respect to J is decidable. Essentially, ordered semantic hyper-linking computes ground instances of clauses in N, which are counterexamples for J, and extends J to J' by the maximal literals of the ground instances. Note that, in this way, J' remains an effective representation. In Section 3.1.1, we observed that soft typing for semantic resolution is compatible with the abstract notion of redundancy restricted to the counterexamples for J. Hence, ordered semantic hyper-linking is compatible with simplification which involves, however, only the (negatively) reductive counterexamples for J.

Clause-linking (hyper-linking) of clauses in N with literals obtained from J (J') enumerates the appropriate ground instances from N. In contrast to resolution, clause-linking resolves complementary literals just to compute the most general unifier. More precisely, a clause-linking step computes the simultaneous most general unifier σ by hyper-resolving all literals in a clause C and then derives $C\sigma$ as the conclusion. Thus clause-linking uses unification in a similar way as resolution for the identification of suitable ground instances of an already unsatisfiable subset of clauses. The approach is then refutationally complete in combination with further enumeration of ground instances and a propositional satisfiability check. In this way, clause-linking distinguishes what resolution combines in a single inference rule, i.e. the identification of suitable ground instances and the unsatisfiability test. However, even in the context of ordered semantic hyper-linking, since $C\sigma$ may still not be ground, a further enumeration of ground instances cannot be avoided, if the input semantics J is only decidable for ground literals.

The following simple proposition implies (i) by observing that the candidate model I^N for semantic resolution corresponds exactly to what has been called the "least model" by Plaisted & Zhu (1997).

Proposition 3.6.1
Let J be a Herbrand model and let N be set of ground clauses. Let I be the model functor for semantic resolution. Suppose that J falsifies the clauses in $N' \subseteq N$ and satisfies the clauses in $N \setminus N'$. Let N'' be the set of (negatively) reductive clauses in N'. Then I^N is a model of N''.

Note that the set E of strictly maximal literals in (negatively) productive clauses from N'' are called "eligible literals" in (Plaisted & Zhu 1997). These literals describe the difference between J and I^N, i.e. E corresponds up to negation to the set $(I^N \setminus J) \cup (J \setminus I^N)$. We arrive at an immediate corollary of the above proposition.

Corollary 3.6.2
Let J be a Herbrand model and let N be set of ground clauses. Let J' be the model constructed from J and N by ordered semantic hyper-linking. If J' is a model of N then N is saturated with respect to soft typing for semantic resolution.

The corollary demonstrates that (ii) the saturation criterion of ordered semantic hyper-linking is an instance of saturation with respect to soft typing for semantic resolution.

3.6.2 Soft Typing for Ordered Resolution

The model generation paradigm has also been studied by Manthey & Bry (1988) who developed the Satchmo theorem prover. In contrast to ordered semantic hyper-linking (Plaisted 1994, Plaisted & Zhu 1997), Satchmo constructs models based on sets of Horn clauses and maintains theorem proving derivation trees. Suppose that N is a set of clauses. Satchmo computes consistent subsets H of Horn clauses of N and then selects counterexamples C for the minimal model T^H of H in order to compute hyper-resolution inferences from C and H. The conclusions are positive clauses which are considered as potential extensions for H not only to satisfy C but also to satisfy N in the limit. More specifically, each positive literal in a conclusion may individually extent H such that we arrive at a tree-like derivation scheme. Our construction of theorem proving derivation trees provides for a suitable formalization of this effect. Conversely, if all clauses in N are true in T^H then the set is considered saturated, as it is, obviously, consistent.

For an effective model construction, clauses are required to be range-restricted. A clause is range-restricted whenever all variables in the succedent also occur in the antecedent. Any set of clauses can be transformed into an essentially equivalent set of range-restricted clauses. By hyper-resolving on range-restricted clauses, new positive ground clauses are obtained and the process branches with regard to the disjuncts by extending H individually with each ground atom of the conclusions. Satchmo theorem proving processes are, therefore, theorem proving derivation trees. We arrive at a major drawback by observing that, in general, an enumeration of all ground instances is required. Unification, as in saturation-based methods, may help as demonstrated by ordered semantic hyper-linking.

For SATCHMO and MGTP, the problem has been addressed in two different lines of research. Loveland, Reed & Wilson (1995) propose certain approximation techniques for the identification of counterexamples in order to limit the combinatorial explosion and to improve the construction of H. The ideas have been implemented in the SATCHMORE system. The MGTP theorem prover, on the other hand, has been developed within the fifth generation project in Japan as an efficient implementation of SATCHMO's theorem proving concept. Note that SATCHMO has been implemented as a prototypical system using Prolog. Similar to the SATCHMORE system, MGTP improves upon the original work in several ways as reported in the more recent paper by Hasegawa, Fujita & Koshimura (1997). SATCHMORE essentially computes, as the improvement upon SATCHMO, an upper approximation T^U of the finite failure set T_F^H of H up to the antecedent literals in N. The finite failure set T_F^H is the set of ground atoms which cannot be derived from H in any finite derivation. The set T_F^H corresponds to the Herbrand base up to the maximal model of H when we restrict our attention to Herbrand models. Note that T^H as well as T_F^H are recursively enumerable for any Horn clause set H whereas the maximal model of H is not. The minimal model T^H can be seen as the set of ground atoms which can be finitely derived from H whereas the maximal model is the minimal model plus the ground atoms which can be derived from H in infinite (cyclic) derivations. SATCHMORE checks the positive ground literals Δ for $\Delta \subseteq T^U$ where Δ is the conclusion of an hyper-resolution inference from a potential counterexample $\Gamma \to \Delta$ and H. In this case, the literals in Δ may serve as suitable extensions of H. Conversely, if there is a ground atom A in Δ such that $A \notin T^U$, then A is already derivable from H which implies that C is not a counterexample.

In contrast, MGTP uses a generalization of magic sets (Bancilhon, Maier, Sagiv & Ullman 1986) and (Beeri & Ramakrishnan 1991) to non-Horn magic sets (Hasegawa, Inoue, Ohta & Koshimura 1997) which approximate directly the minimal model T^H of H by a lower approximation T_L. Then, potential counterexamples are checked in the dual way with respect to T_L. Note that the check with respect to the smaller T_L may also be more efficient compared to the computation with respect to T^H. Recently, Ohta, Inoue & Hasegawa (1998) demonstrated that this concept actually coincides with SATCHMORE with respect to derivation trees assuming the construction of T^U in SATCHMORE is slightly weakened.

In the following, we demonstrate that the semantic methods in SATCHMO and, in this way, also in MGTP are essentially an instance of soft typing for ordered resolution. If we abstract from the particular structure of theorem proving derivations, we can show that (i) for any set H of Horn clauses constructed by SATCHMO there is a candidate model obtained from the model functor for ordered resolution which is exactly the minimal model of H and (ii) the saturation criterion of SATCHMO is in fact a special case of our notion of saturation with respect to soft typing for ordered resolution. More precisely, for satisfiable sets H of Horn clauses it turns out that one may always find an admissible atom ordering for which (i) the construction of the model functor I for ordered resolution yields the minimal model of H and thus (ii) H is saturated with respect to soft typing for ordered resolution which implies, if T^H is a model of N, that all inferences from non-reductive clauses are blocked and thus that N is weakly saturated, see below. The ordering has to be constructed from the immediate consequence operator T_H.

Definition 3.6.3 (Compatible Atom Ordering)
Let H be a set of Horn clauses without equality. Let ι_A, for any ground atom A in the minimal model T^H of H, denote the minimal index n for which A is in $T_H^n(\emptyset)$. For atoms A not in T^H we set $\iota_A := \infty$. If $\succ_a$ is an ordering on ground atoms such that $A \succ_a B$ whenever $\iota_A > \iota_B$, then we call $\succ_a$ *compatible* with H.

For compatible admissible atom orderings the model functor I for ordered resolution of Definition 3.1.43 is simply another method for generating the minimal model T^H of H.

Proposition 3.6.4
Let I be the model functor for ordered resolution. Let H be a satisfiable subset of Horn clauses of some clause set N, and let $\succ_a$ be an admissible atom ordering that is compatible with H. Let C be a ground clause where $(\neg)A$ is a maximal atom in C.

1. Let B be some ground atom. If $A \succ_a B$ then B is in T^H if and only if B is in I^H_C.
2. If $\neg A$ in C or $\{A, A\} \subseteq C$ then A is in T^H if and only if A is in I^H_C.
3. H is weakly saturated with respect to R_O and $\succ_a$.
4. $I^H_C \subseteq I^N_C$.

Proof The proof for (1) is by induction on the atom ordering. We assume as the induction hypothesis that (1) holds for all clauses D in which the maximal atom is smaller than A. Suppose that B is true in T^H. By definition of T^H there is a Horn clause $D = B_1, \ldots, B_n \rightarrow B$ in H such that $\{B_1, \ldots, B_n\} \subseteq T^H$. Since $B \succ_a B_1, \ldots, B_n$ we may apply the induction hypothesis and infer that $\{B_1, \ldots, B_n\} \subseteq I^H_D$. Thus the clause D produces B if B has not already been produced. In both cases B is true in I^H_C. For the other direction suppose B is true in I^H_C. Then there is a productive Horn clause $D = B_1, \ldots, B_n \rightarrow B$ in H. By the induction hypothesis we derive that $\{B_1, \ldots, B_n\} \subseteq T^H$. Thus B is also true in T^H.

For (2) suppose that A is true in T^H. By definition of T^H there is a Horn clause $D = A_1, \ldots, A_n \rightarrow A$ in H such that $\{A_1, \ldots, A_n\} \subseteq T^H$. Note that D is strictly smaller than C. Since $A \succ_a A_1, \ldots, A_n$ we may use (1) to infer that $\{A_1, \ldots, A_n\} \subseteq I^H_C$ and thus $\{A_1, \ldots, A_n\} \subseteq I^H_D$. Hence, either A is true in I^H_D, or else D is a productive clause. In both cases we conclude that A is true in I^H_C. For the other direction suppose A is true in I^H_C. Then there is a productive Horn clause $D = A_1, \ldots, A_n \rightarrow A$ in H. By (1) we derive that $\{A_1, \ldots, A_n\} \subseteq T^H$. Thus A is also true in T^H.

(3) holds since, by (1) and (2), it follows that I^H is a model of H and therefore any inference from clauses in H is blocked.

To show (4) we use induction on C. Suppose that A is in I_C^H. Then there exists a clause $D = \Gamma \rightarrow A$ in H_C which produces A into I^H. In particular, any B in Γ is smaller than A and $\Gamma \subseteq I_D^H$. Using the induction hypothesis for D we may infer that $\Gamma \subseteq I_D^N$. Therefore, either A is in I_D^N, or else D produces A into I^N. In both cases we conclude that A is in I_C^N. ■

Observe that, part (4) of the above Proposition, motivates a proof for the other direction $I_C^N \subseteq I_C^H$ which is, however, only true for non-reductive C. We can show this property by induction using a specific construction K_C^H which contains I_C^H.

Definition 3.6.5
Let I be the model functor for ordered resolution. Let H be a set of ground Horn clauses and let C be a possibly non-Horn ground clause with maximal atom A. By K_C^H we denote the interpretation

$$K_C^H = \begin{cases} I_C^H \cup \{A\} & \text{if } A \in T^H \text{ and } C \text{ is reductive for } A, \\ I_C^H & \text{otherwise.} \end{cases}$$

K_C^H is different from I_C^H only if C is reductive for A and if A gets eventually produced into T^H by a clause D in H such that $D \succeq_c C$.

Theorem 3.6.6
Let H be a satisfiable subset of Horn clauses of some clause set N, and let $\succ_a$ be an admissible atom ordering that is compatible with H. Suppose that T^H satisfies N.

1. For any ground clause C, we have $I_C^N \subseteq K_C^H$.
2. Any non-reductive clause C in N is true in I_C^N.

Proof We prove (1) by induction on C. Suppose that A is in I_C^N. Then there exists a clause $D = \Gamma \rightarrow \Delta, A$ in N_C which produces A into I^N. In particular, any B in Γ, Δ is smaller than A with respect to $\succ_a$, and $\Gamma \subseteq I_D^N$, as well as $\Delta \cap I_D^N = \emptyset$. Using the induction hypothesis for D we may infer that $\Gamma \subseteq K_D^H$ and thus $\Gamma \subseteq I_D^H$. Hence, by part (1) of Proposition 3.6.4 we get $\Gamma \subseteq T^H$. By part (4) of Proposition 3.6.4, we get $\Delta \cap I_D^H = \emptyset$. Again by part (1) of this proposition we conclude that $\Delta \cap T^H = \emptyset$. For T^H to satisfy D, the atom A must therefore be true in T^H. If A is strictly smaller than the maximal atom in C then, by part (1) of Proposition 3.6.4, A is in I_C^H. Similarly, if A is the maximal atom in C but C is not reductive for A, then, by part (2) of the same proposition, A is in I_C^H. If C is reductive for A then by definition of K_C^H we have that A is in K_C^H.

Part (2) is an immediate consequence of (1) by observing that, according to (1) and part (4) of Proposition 3.6.4, the interpretations I_C^H and I_C^N coincide for non-reductive clauses C. By part (1) and (2) of Proposition 3.6.4 I_C^H assigns the same truth values as T^H to the atoms in C. Thus I_C^N is also a model of C. ■

If T^H is not only a model of a Horn subset H of N but also a model of N itself then Theorem 3.6.6 demonstrates that any non-reductive and possibly non-Horn clause C from N is true in I_C^N. It follows that N is weakly saturated since any inference from clauses in N is blocked. Note that any main premise of an Ordered Resolution inference as well as any main premise of an Ordered Factoring step is a non-reductive clause.

Corollary 3.6.7
Let N be a set of ground clauses and let H be a consistent Horn subset of N. If T^H satisfies N then N is weakly saturated with respect to any admissible atom ordering that is compatible with H and any selection function.

This corollary indicates that (ii) the Satchmo saturation criterion is a special case of saturation with respect to soft typing for ordered resolution.

3.6.3 Abstraction in Derivations

Abstraction in the context of automated proof search has also been received quite some attention from the perspective of theorem proving derivations. In (Plaisted 1980, Plaisted 1981) proof guidance based on abstractions is obtained by successful resolution-based proof derivations in some "abstract space" which has been derived from the original problem set. The idea is to map the original clause set into an abstract form, e.g., several clauses are identified in an equivalence class, and to search for a proof first in this abstract form. A successful *abstract derivation* is then used as a template for the proof on the original level which is supposed to fill in the "gaps" of the abstract proof. An obvious generalization of this idea is to build up a hierarchy of abstractions where the abstract derivation on level $n + 1$ guides the proof search of level n. Plaisted (1986) motivates that not only the input set itself may be subject to (input) abstraction but also the (conclusions of) resolution operations which arise in the abstract derivation. However, the application of simplification methods in abstract derivations seems to be difficult while preserving completeness. For instance, subsumption in abstract derivations may delete significant information which is actually non-redundant in the original level. The problem is actually due to the somewhat tight combination of abstractions and derivations.

The concept of proof guidance by abstract derivations may also be seen independently from a particular inference system (Giunchiglia & Walsh 1989, Giunchiglia & Walsh 1992). Moreover, Giunchiglia & Walsh (1993) state that "false proofs" on the abstract level, as they have already been mentioned by Plaisted (1981), cannot be avoided, in general. A false proof corresponds to a successful abstract derivation from an inconsistent abstract space which has been obtained from a consistent original. In particular, a consistent subset (theory) of some inconsistent problem set may be mapped to an inconsistent abstraction. Proof attempts on the original level according to false proofs may therefore lead to substantial overhead. Note that false proofs are an instance of the general problem of trivial abstract spaces. Hence, in combination with the incompatibility of simplification techniques, we consider the incorporation of abstract derivations as problematic.

Chapter 4

Sort Theories

Soft typing for clausal inference systems is a general framework for semantically guided theorem proving. A derivation by an inference system can be controlled using the dynamic blocking concept of soft typing for clauses and inferences. An effective use of blocking, however, requires decidable approximations of candidate models. In Section 3.3, we introduced (static, dynamic) Horn theories as a formalism for the representation of approximations of candidate models. A Horn theory H enjoys the existence of a unique minimal model T^H of H which allows for a convenient relation of all models of H and some candidate model. However, validity and satisfiability of clauses with respect to a general Horn theory are undecidable. Thus further abstraction steps are needed for decidability. Throughout this chapter, by a decidable Horn theory H, we mean that the theory of monadic types over H is decidable.

We demonstrate that a specialized form of ordered resolution with selection, the so-called *sort resolution calculus* (Weidenbach 1996*a*), can be used as a general decision procedure for certain variants of so-called *sort theories*. We present effective transformations to infer automatically abstractions of Horn theories which result in the class of decidable sort theories. In this way, we obtain decidable approximations for a particular form of types. In Chapter 5 through 7, we follow exactly this concept to obtain similar results for other forms of Horn theories and for non-trivial equational theories which can be seen as Horn theories with equality.

Frühwirth et al. (1991) have already proposed to use certain classes of logic programs to infer type information for logic programs. Logic programs and Horn theories are essentially equivalent. The first step is to consider only *monadic Horn clauses* in which only monadic predicates occur. Note that Horn theories which contain only monadic Horn clauses are still undecidable, in general. Further restrictions on the structure of terms which occur in the clauses of the theory are needed. We include equality in the definitions below to obtain a continuous definition which can also be used in the subsequent chapters.

Definition 4.0.1 (Monadic Horn Clause)
A Horn clause $\Gamma \rightarrow A$ where Γ contains only monadic atoms and equations and A is a monadic atom or an equation is called a *monadic Horn clause*. A monadic Horn clause

$\Gamma \to S(t)$ is called *shallow* (*linear*, *semi-linear*) if t is shallow (linear, semi-linear). A monadic Horn clause $\Gamma \to s \approx t$ is called *shallow* (*linear*, *semi-linear*) if $s \approx t$ is shallow (linear, semi-linear).

A *monadic Horn theory* consists of monadic Horn clauses only. Note that, in general, a shallow and/or linear monadic Horn theory is a semi-linear monadic Horn theory but not vice versa.

Definition 4.0.2 (Monadic Horn Theory)
A *monadic Horn theory* $\mathcal{H}$ is a finite set of monadic Horn clauses. $\mathcal{H}$ is called *shallow* (*linear*, *semi-linear*) if all monadic Horn clauses in $\mathcal{H}$ are shallow (linear, semi-linear).

In the sequel, we will show that certain (non-linear) shallow monadic Horn theories are decidable and that (semi-linear) linear monadic Horn theories can be transformed into essentially equivalent (non-linear) shallow monadic Horn theories. We may obtain a monadic Horn theory from an arbitrary Horn theory by *Monadic Abstraction* where all non-monadic atoms are simply left out. We may improve the result using, for example, a particular form of *term encoding* which transforms a non-monadic atom $P(t_1, \ldots, t_n)$ into a monadic atom $S(p(t_1, \ldots, t_n))$ where S is a common monadic predicate symbol for all non-monadic atoms and p is a new function symbol with the same arity as P. In (Frühwirth et al. 1991), the symbol S is denoted by the monadic predicate symbol *type*. In order to use this form of term encoding for blocking, the original clauses have to be extended with additional S constraints. However, from a conceptual point of view, this does not affect the analysis of decidable monadic Horn theories, below.

Definition 4.0.3
The following abstraction is called *Monadic Abstraction*:

$$\text{Abstract}\frac{\Lambda \to L}{\Gamma \to L}$$

where (i) L is either a monadic atom or an equation and (ii) Γ is the maximal subset of Λ such that Γ contains monadic atoms and equations only.

Recall that static and dynamic Horn theories are approximations of arbitrary clause sets where disjunctions of positive literals have been transformed into conjunctions. A dynamic Horn theory is restricted to the reductive clauses of the according static Horn theory. An exhaustive application of Monadic Abstraction yields the *static* and *dynamic monadic Horn theory*, respectively.

Definition 4.0.4 (Static/Dynamic Monadic Horn Theory)
Let N be a set of clauses. Let H_S and H_D be the static and dynamic Horn theories of N, respectively. Let $\mathcal{H}_S$ and $\mathcal{H}_D$ be the result of an exhaustive application of Monadic Abstraction to H_S and H_D, respectively. We call $\mathcal{H}_S$ the *static monadic Horn theory of* N and $\mathcal{H}_D$ the *dynamic monadic Horn theory of* N.

The following statement is an easy corollary of Proposition 3.3.10 and 3.3.13. Using Monadic Abstraction, we simply restrict the approximation to monadic atoms and equations. The generalization to the model functor for superposition is also not difficult.

Corollary 4.0.5
Let I be the model functor for ordered resolution. Let N be a set of clauses which does not contain the empty clause and let M be the set of all ground instances of clauses in N. Let $\mathcal{H}_S$ and $\mathcal{H}_D$ be the static and dynamic monadic Horn theories of N, respectively. Then the minimal models $T^{\mathcal{H}_S}$ and $T^{\mathcal{H}_D}$ are upper approximations of I^M with respect to monadic atoms and equations.

Consequently, we define *monadic (equational) types* as the types which may still be subject to an effective evaluation of satisfiability with respect to the monadic abstractions of static and dynamic Horn theories.

Definition 4.0.6 (Monadic (Equational) Type)
A type which contains only monadic atoms is called a *monadic type*. It is called a *monadic equational type* if all predicate symbols in the type are monadic. Let C be a clause $\Gamma \rightarrow \Delta$ and let T be a typing function. If $T(C) \subseteq \Gamma$ and $T(C)$ contains only monadic predicate symbols (and equations) then $T(C)$ is called the *monadic (equational) type of C*. If each term which occurs in a monadic (equational) type is a variable then we call it a *monadic (equational) variable type*.

The following corollary manifests the goal of this chapter (and also of Chapter 5 through 7). We are interested in automatic abstractions of static/dynamic monadic Horn theories which lead to decidable approximations for monadic (equational) types. The unsatisfiability of these types with respect to the approximations allows for an effective blocking of clauses which contain monadic (equational) type information.

Corollary 4.0.7
Let N_0 be a set of clauses. Then the static monadic Horn theory of N_0 is a static approximation for monadic (equational) types. Let N_0, N_1, N_2, ... be a (weakly) fair theorem proving derivation. The sequence of dynamic monadic Horn theories $\mathcal{H}_i$ of N_i is a dynamic approximation for monadic (equational) types.

We may refer to the static or dynamic monadic Horn theory of a set N of clauses simply by the *monadic Horn theory of N*.

4.1 Introduction to Sort Theories

We study so-called *sort theories* which are a particular form of monadic Horn theories without equality. By a sort theory we mean what has been called the "relativization of a sort theory" in (Weidenbach 1998), see the example below. A particular refinement of ordered resolution with selection called the *sort resolution calculus* (Weidenbach 1996*a*) serves as a general decision procedure for the satisfiability of monadic types with respect to a sort theory. The sort resolution calculus is based on a particular data structure of the clauses where certain negative monadic literals are represented separately as *sort constraints* of the clauses. This notion allows for a technically simpler presentation of a particular combination of the atom ordering and the selection strategy in the sort resolution calculus. Thus sort constraints are simply convenient but not essential.

Definition 4.1.1 (Sort Constraint)
A conjunction Ψ of monadic atoms is called a *sort constraint*.

Note that the similarity of sort constraints and monadic types is coincidental as the concept of sort constraints does not naturally carry over to monadic equational types which can be decided by (*basic*) *sorted superposition* (Jacquemard et al. 1998*a*), c.f. Section 6.3, 6.7, and 7.1. In contrast to sort constraints, the type of a clause C denotes which part of C may be responsible for an effective blocking with respect to a decidable theory. We distinguish sort constraints and monadic types as particular data structures to obtain, on one hand, decision procedures and, on the other hand, to effectively control soft typing.

Definition 4.1.2 (Sorted Clause)
Let $\Lambda \to \Delta$ be a clause. Let $\Psi \subseteq \Lambda$ be a multiset of monadic atoms in Λ. We write $\Lambda \to \Delta$ as

$$\Psi \,||\, \Gamma \to \Delta$$

where $\Gamma = \Lambda \setminus \Psi$. We call $\Psi \,||\, \Gamma \to \Delta$ a *sorted clause* where Ψ is the sort constraint.

A sorted clause $\Psi \,||\, \Gamma \to \Delta$ is logically equivalent to the clause $\Psi, \Gamma \to \Delta$.

Example 4.1.3
If all atoms in a sort constraint are of the form $S(x)$ with S a monadic predicate and x a variable then clauses are relativizations of clauses containing sorted variables (Weidenbach 1998). For example, the sorted clause $R(x_S, y_S) \to P(x_S)$ where x_S is a variable of sort S corresponds via relativization to the sorted clause $S(x), S(y) \,||\, R(x,y) \to P(x)$ that is logically equivalent to the clause $S(x), S(y), R(x,y) \to P(x)$.

Resolution inferences, as defined in the sort resolution calculus, can be restricted to side premises with *solved sort constraints*. This guarantees a particular form of the involved side premises which later provides for a simple termination argument of sort resolution. Conversely, unsolved sort constraints are selected, while resolution is applied to transform the sort constraint into solved form.

Definition 4.1.4 (Solved Sort Constraint)
A sort constraint Ψ in a clause $\Psi \,||\, \Gamma \to \Delta$ is called *solved*, if each occurrence of a term in Ψ is a variable x and there is at least one occurrence of x in Γ or Δ such that x occurs as the argument of a function. A sort constraint Ψ in which each occurrence of a term in Ψ is a variable is called *non-alternating* whenever, for all $T(x), T'(y) \in \Psi$, $T \neq T'$ implies that $x \neq y$. Otherwise, we may call Ψ *alternating.*

The notion of "sort declarations" has been originally motivated within the area of *sorted unification*, c.f. (Weidenbach 1996*a*). Here, by a *sort declaration*, we mean the relativized version of a "sort declaration" in the traditional sense, i.e. a sort declaration is a certain form of a sorted Horn clause.

Definition 4.1.5 (Sort Declaration)
A sorted clause $S_1(x_1), \ldots, S_n(x_n) \,\|\, \rightarrow S(t)$ is called a *sort declaration* if either the sort constraint $S_1(x_1), \ldots, S_n(x_n)$ is solved, or else t is a variable with $t = x_1 = \ldots = x_n$. A sort declaration is called a *term declaration*, if t is not a variable and a *subsort declaration*, otherwise. A subsort declaration is called *trivial* if $n = 0$. A term declaration of the form $\Psi \,\|\, \rightarrow S(t)$ where Ψ is non-alternating is called *non-alternating*. Otherwise, we may call a term declaration *alternating*. A sort declaration is called *shallow* (*linear*, *semi-linear*) if t is shallow (linear, semi-linear). Sort declarations as well as subsort declarations are also called *declarations*.

Note that non-trivial subsort declarations have an unsolved sort constraint. We arrive at the formal definition of a *sort theory*, which is again the relativization of a "sort theory" in the traditional sense, i.e. a sort theory is a finite set of sort declarations.

Definition 4.1.6 (Sort Theory)
A *sort theory* $\mathcal{S}$ is a finite set of declarations. $\mathcal{S}$ is called *shallow* (*linear*, *semi-linear*) if all sort declarations in $\mathcal{S}$ are shallow (linear, semi-linear). $\mathcal{S}$ is called *non-alternating* if $\mathcal{S}$ does not contain any alternating term declarations and non-trivial subsort declarations. Otherwise, we may call $\mathcal{S}$ *alternating.*

Note that, in general, a sort theory is a monadic Horn theory but not vice versa while a shallow and/or linear sort theory is a semi-linear sort theory but not vice versa. The motivation of this classification is to distinguish shallow and semi-linear sort theories. We demonstrate that (non-linear) shallow sort theories are decidable and that semi-linear sort theories can be effectively transformed into essentially equivalent (non-linear) shallow sort theories. Using this transformation, we can extend the decidability result to semi-linear sort theories. The notion of alternation is inspired by alternating tree automata which are considered in Section 4.2. Note that a non-alternating sort theory contains only declarations with solved sort constraint. Monadic Horn theories, in general, do not comply with the restriction of sort theories. *Sort Abstraction* removes the improper information and transforms a monadic Horn theory $\mathcal{H}$ (without equality) into a sort theory $\mathcal{S}$ in such a way that the minimal model $T^{\mathcal{S}}$ is an upper approximation of $T^{\mathcal{H}}$.

Definition 4.1.7
The following abstraction is called *Sort Abstraction*:

$$\text{Abstract}\,\frac{S_1(x_1), \ldots, S_n(x_n), \Gamma \rightarrow S(t)}{S_1(x_1), \ldots, S_n(x_n) \,\|\, \rightarrow S(t)}$$

where (i) $S_1(x_1), \ldots, S_n(x_n)$ is maximal such that it contains only monadic atoms of the form $T(x)$ with $x \in \mathit{vars}(S(t))$.

Sort Abstraction removes certain monadic atoms which occur negatively in the antecedent of a clause. We present a better approximation in this regard in Chapter 5. However, there is a trade-off for decidability, since the approximation requires then all clauses to be linear declarations.

Proposition 4.1.8
Let $\mathcal{H}$ be a monadic Horn theory without equality. The result of an exhaustive application of Sort Abstraction to $\mathcal{H}$ is a sort theory $\mathcal{S}$ while the minimal model $T^{\mathcal{S}}$ of $\mathcal{S}$ is an upper approximation of the minimal model $T^{\mathcal{H}}$ of $\mathcal{H}$.

Proof By the construction of Sort Abstraction it follows immediately that any exhaustive application of the rule is finite and results in a sort theory. The minimal model $T^{\mathcal{S}}$ is in fact an upper approximation of $T^{\mathcal{H}}$ since all clauses with positive literals are abstracted into $\mathcal{S}$ while only negative literals are removed. ■

In the sequel, we call a sort theory $\mathcal{S}$ which has been obtained by an exhaustive application of Sort Abstraction to the static or dynamic monadic Horn theory $\mathcal{H}$ of a set N of clauses the *sort theory of* N. In order to emphasize that $\mathcal{S}$ has been obtained from $\mathcal{H}$ we may also say that $\mathcal{S}$ is the *static* or *dynamic sort theory of* N, respectively. The following corollary is an immediate consequence of Proposition 4.1.8 and Corollary 4.0.7.

Corollary 4.1.9
Let N_0 be a set of clauses without equality. The static sort theory of N_0 is a static approximation for monadic types. Let $N_0, N_1, N_2, \ldots$ be a fair theorem proving derivation. The sequence of dynamic sort theories of each N_i is a dynamic approximation for monadic types.

4.2 Tree Automata

We shall relate (non-linear) shallow sort theories to finite (bottom-up) tree automata as demonstrated by Weidenbach (1998) in a comprehensive discussion of the systematic correspondence between various sort theories and distinct classes of tree automata. The automata approach can be seen as an equally powerful concept as the representation of sort theories by monadic Horn clauses. In Section 6.2, however, we show that the automata approach does not capture the satisfiability problem with respect to so-called sorted shallow equational theories whereas the generalization of monadic Horn clauses with equality in combination with sorted superposition does. There is also a relation to set constraints which we, however, do not consider here and instead refer to (Charatonik & Podelski 1998, Weidenbach 1996*a*).

We adopt a definition of tree automata by means of Horn clauses. This definition, though non-standard, is equivalent to the original definition by Thatcher & Wright (1968) and Doner (1970). The language accepted by a tree automaton is a set of ground terms. Typical questions related to tree automata are closure properties under boolean operations like union, intersection, or complementation, and non-emptiness or finiteness tests.

Definition 4.2.1 ((Alternating) Tree Automaton)
An *alternating tree automaton* $\mathcal{A}$ is a finite set of linear shallow term declarations of the form $S_1(x_1), \ldots, S_n(x_n) \parallel \rightarrow S(f(x_1, \ldots, x_n))$ with $n \geq 0$ and subsort declarations of the form $S_1(x), \ldots, S_m(x) \parallel \rightarrow S(x)$ with $m > 1$. We call $\mathcal{A}$ a (*non-alternating*) *tree automaton* if $\mathcal{A}$ does not contain any subsort declarations.

We denote the class of (alternating) tree automata by *Rec* and call a tree automaton a *Rec* automaton. Following tree automata terminology, the monadic predicates are called *states* and the term declarations of $\mathcal{A}$ are *transition rules* or simply *transitions*. Subsort declarations of the form $S_1(x), \ldots, S_m(x) \parallel \rightarrow S(x)$ with $m > 1$ correspond to so-called *alternation rules* whereas subsort declarations of the form $S(x) \parallel \rightarrow T(x)$ correspond to so-called *ϵ-transitions*. Note that ϵ-transitions are not included in our definition of tree automata. However, any (non-alternating) tree automaton with additional ϵ-transitions can be transformed in polynomial time into an equivalent automaton which contains only non-alternating linear shallow term declarations in compliance with Definition 4.2.1.

A ground term t is *recognized* by an (alternating) tree automaton $\mathcal{A}$ in a state S if $\mathcal{A} \models S(t)$. Let $\mathcal{S}$ be a subset of all states of $\mathcal{A}$. We call the states in $\mathcal{S}_\mathrm{f}$ the *final states* (final predicates) of $\mathcal{A}$. A ground term t is recognized by $\mathcal{A}$ with respect to $\mathcal{S}_\mathrm{f}$ if t is recognized by $\mathcal{A}$ in a final state in $\mathcal{S}_\mathrm{f}$. The set $L(\mathcal{A})$ of ground terms which are recognized by a *Rec* automaton $\mathcal{A}$ with respect to the set of final states of $\mathcal{A}$ is called the *language recognized by* $\mathcal{A}$. A set L of ground terms is called a *recognizable language* with respect to *Rec* if there is a *Rec* automaton such that $L = L(\mathcal{A})$. By *Rec*, we ambiguously denote the class of languages which are recognizable with respect to *Rec*. The *Rec* class is closed under boolean operations. A non-alternating tree automaton $\mathcal{A}$ is called (*bottom-up*) *deterministic* whenever $\mathcal{A}$ does not contain any two rules of the form $\Psi \parallel \rightarrow S(f(x_1, \ldots, x_n))$ and $\Psi \parallel \rightarrow T(f(x_1, \ldots, x_n))$. Every recognizable language in *Rec* is recognized by some (bottom-up) deterministic tree automaton $\mathcal{A}$ such that a ground term cannot be recognized by $\mathcal{A}$ in more than one state. A non-alternating tree automaton $\mathcal{A}$ is called (*top-down*) *deterministic* whenever $\Psi = \Phi$ for any two rules in $\mathcal{A}$ of the form $\Psi \parallel \rightarrow S(f(x_1, \ldots, x_n))$ and $\Phi \parallel \rightarrow S(f(x_1, \ldots, x_n))$. Every recognizable language in *Rec* is recognized by some *completely specified* tree automaton $\mathcal{A}$ such that every ground term is recognized by $\mathcal{A}$ at least in one state. The following problems have extensively been studied in the area of tree automata.

Definition 4.2.2 (Non-emptiness Problem)
The *non-emptiness problem of Rec* is the problem whether the language $L(\mathcal{A})$ recognized by a tree automaton $\mathcal{A}$ is non-empty, i.e. whether $L(\mathcal{A}) \neq \emptyset$.

Definition 4.2.3 (Membership Problem)
The *membership problem of Rec* is the problem whether a given ground term t is recognized by a tree automaton $\mathcal{A}$, i.e. whether $t \in L(\mathcal{A})$.

Definition 4.2.4 (Intersection Non-emptiness Problem)
The *intersection non-emptiness problem of Rec* is the problem whether the intersection of the languages recognized by a finite sequence of tree automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ is non-empty,

i.e. whether $L(\mathcal{A}_1) \cap \ldots \cap L(\mathcal{A}_n) \neq \emptyset$.

In the sequel, we may use straightforward adaptions of the above problems for other types of automata or theories as well. The non-emptiness problem of *Rec* is decidable in linear time and the membership problem of *Rec* is decidable in polynomial time. The intersection non-emptiness problem of *Rec* is EXPTIME-complete (Frühwirth et al. 1991, Seidl 1994, Veanes 1997). In particular, Frühwirth et al. (1991) merely sketch the proof idea which is based on the fact that exponential-time algorithms can be simulated by alternating polynomial-space algorithm (Chandra, Kozen & Stockmeyer 1981). Seidl (1994) shows the EXPTIME-completeness for deterministic (top-down) tree automata. Veanes (1997) gives a detailed proof of EXPTIME-completeness for deterministic (bottom-up) tree automata which implies the EXPTIME-hardness already for deterministic tree automata. For a comprehensive discussion of the complexity of (non-alternating) tree automata see (Veanes 1997). The following proposition states that (alternating) linear shallow sort theories are essentially (alternating) tree automata up to a polynomial increase in size.

Proposition 4.2.5
Any (alternating) tree automaton is an (alternating) linear shallow sort theory while any (alternating) linear shallow sort theory $\mathcal{S}$ can be transformed into an (alternating) tree automaton $\mathcal{A}$ where the set of final states of $\mathcal{A}$ is the set of predicate symbols which occur in $\mathcal{S}$ such that $L(\mathcal{A}) = \{t \mid \mathcal{S} \models S(t)\}$ and $\mathit{size}(\mathcal{A})$ is polynomially bounded with respect to $\mathit{size}(\mathcal{S})$.

Proof Given an (alternating) linear shallow sort theory $\mathcal{S}$, we have to transform all clauses in $\mathcal{S}$ which are trivial subsort declarations and term declarations that do not conform with transition rules. A trivial subsort declaration $\| \rightarrow S'(x)$ can be represented by a so-called *universal* tree automaton $\mathcal{A}_u$ that simply recognizes all ground terms in the state S'. Then $\mathcal{A}_u$ has only one state S' while $\mathit{size}(\mathcal{A}_u)$ is linear in the number of function symbols which occur in the clauses of $\mathcal{S}$. A linear shallow term declaration of the form $\Psi \| \rightarrow S(f(\ldots, x, \ldots)$ with $x \notin \mathit{vars}(\Psi)$ can then be replaced by the term declaration $S'(x), \Psi \| \rightarrow S(f(\ldots, x, \ldots)$. An alternating linear shallow term declaration of the form $\Psi, \Phi \| \rightarrow S(t)$ where Ψ is maximal such that Ψ is of the form $S_1(x), \ldots, S_m(x)$ for some $x \in \mathit{vars}(t)$ can be represented by a modified term declaration $T(x), \Phi \| \rightarrow S(t)$ and an alternation rule $S_1(x), \ldots, S_m(x) \| \rightarrow T(x)$. ■

The representation of tree automata by monadic Horn clauses immediately shows the close relationship of the automata-theoretic approach and the sort-theoretic approach though there are stronger limits for the automata-theoretic approach when incorporating semantic equality, c.f. Section 6.2. However, we shall also relate non-linear shallow sort theories to a particular generalization of tree automata. Bogaert & Tison (1992) introduce a generalized form of $\mathit{Rec}_{\neq}$ automata with (dis-)equality constraints in the transition rules. A (dis-)equality constraint is a boolean combination of syntactic equalities. The class $\mathit{Rec}_{\neq}$ of recognizable languages with respect to $\mathit{Rec}_{\neq}$ is a strict superclass of *Rec*. The constraints impose (dis-)equality tests in transition rules between brother subterms. A strict subclass of $Rec_{\neq}$ automata is the class of $Rec_{=}$ automata without negation in the constraints. Non-linear shallow term declarations may represent $Rec_{=}$ automata in the following way. Suppose that $\mathcal{A}$ is a $Rec_{=}$ automaton. We may transform the constraint in each transition rule of $\mathcal{A}$ into a disjunction $E_1 \vee \ldots \vee E_n$ of conjunctions E_i of equations. Then we generate n copies of the transition rule containing the respective conjunction E_i of equations. Note that the size of the resulting automaton is exponential in the size of the constraints of the original automaton. However, in particular, the hardness results mentioned below have been obtained already for the class of automata which contain only conjunctions of equations.

Definition 4.2.6 ((Alternating) $Rec_{=}$ Automaton)
A finite set of (non-linear) shallow term declarations of the form $S_1(x_1), \ldots, S_n(x_n) \parallel \rightarrow S(f(x_1, \ldots, x_n))$ with $n \geq 0$ and subsort declarations of the form $S_1(x), \ldots, S_m(x) \parallel \rightarrow S(x)$ with $m > 1$ is called an *alternating tree automaton $\mathcal{A}$ with equality constraints between brother subterms*. We call $\mathcal{A}$ a (*non-alternating*) *tree automaton with equality constraints between brother subterms* if $\mathcal{A}$ does not contain any subsort declarations.

The variables x_i, which are not necessarily distinct, sufficiently represent the conjunction of syntactic equalities. The notion of recognized ground terms, languages, determinism, and completeness is the same for (alternating) tree automata with equality constraints between brother subterms as for (alternating) tree automata. The class $Rec_{\neq}$ is strictly larger than $Rec_{=}$, e.g., the set of non-well-balanced trees (terms) is recognizable by a $Rec_{\neq}$ automaton but not by a $Rec_{=}$ automaton (Bogaert & Tison 1992).

Note that $Rec_{\neq}$ is closed under boolean operations, under determinism and complete specification, whereas $Rec_{=}$ is closed under union and intersection but not under complementation. The non-emptiness problem of $Rec_{\neq}$ is decidable though EXPTIME-complete (Comon et al. 1997) already for automata which contain at most a conjunction of equations in each transition rule and no disequality constraints. Bogaert & Tison (1992) show that the same problem is NP-hard for non-deterministic $Rec_{\neq}$ automata and that it can be solved in polynomial time for deterministic $Rec_{\neq}$ automata. We obtain, as an immediate corollary of Proposition 4.2.5, that (alternating) non-linear shallow sort theories are essentially (alternating) $Rec_{=}$ automata up to a polynomial increase in size.

Corollary 4.2.7
Any (alternating) $Rec_{=}$ automaton is an (alternating, non-linear) shallow sort theory while any (alternating) linear shallow sort theory $\mathcal{S}$ can be transformed into an (alternating) $Rec_{=}$ automaton $\mathcal{A}$ where the set of final states of $\mathcal{A}$ is the set of predicate symbols which occur in $\mathcal{S}$ such that $L(\mathcal{A}) = \{t \mid \mathcal{S} \models S(t)\}$ and $size(\mathcal{A})$ is polynomially bounded with respect to $size(\mathcal{S})$.

In Section 6.2, we show that the tree automata approach even with (dis-)equality constraints between brother terms, however, does not capture the satisfiability problem with respect to so-called sorted shallow equational theories whereas the generalization of monadic Horn clauses with equality in combination with sorted superposition does.

4.3 Decidability

In order to show that satisfiability of monadic types with respect to (non-linear) shallow sort theories is decidable, we employ *sort resolution* which is a particular instance of ordered resolution with selection. We prefer a specialized version of ordered resolution in order to abstract from the combination of a particular admissible atom ordering and a selection strategy for sort constraints. Under the proviso that the admissible atom ordering is compatible with the subterm property, sort resolution is refutationally complete for monadic Horn theories without equality where the antecedent literals of every clause have been transformed into the sort constraint. An atom ordering which is induced by an admissible term ordering is compatible with the subterm property. We assume admissible literal orderings of Definition 3.1.14 and admissible clause orderings of Definition 3.1.15. The *sort selection* strategy selects the unsolved part of a sort constraint with a particular priority on non-variable terms. More precisely, literals containing non-variable terms are selected first, followed by so-called *empty sorts*, i.e. literals containing variables that do not occur in the non-constraint part of the clause. Literals of subsort declarations are selected last.

Definition 4.3.1 (Sort Selection)
Let C be a sorted clause of the form $\Psi \,\|\, \rightarrow \Pi$ where Π is either a monadic atom $T(u)$ or empty. A literal occurrence $S(t) \in \Psi$ is selected whenever (i) t is a non-variable term, or else (ii) Ψ contains only variables and $t \notin vars(\Pi)$, or else (iii) Ψ contains only variables (in particular, t is a variable) with $t = u$. No other literal occurrence in C is selected. We call this selection strategy *sort selection*. A literal occurrence which is selected by sort selection is called *sort selected*.

The sort resolution calculus only consists of the *Sort Constraint Resolution* rule. Factoring is not required for Horn clause sets.

Definition 4.3.2 (Sort Constraint Resolution)
The following inference is called *Sort Constraint Resolution*:

$$\text{Infer}\frac{\Phi \,\|\, \rightarrow S(s) \qquad S(u), \Psi \,\|\, \Lambda \rightarrow \Pi}{\Phi\sigma, \Psi\sigma \,\|\, \Lambda\sigma \rightarrow \Pi\sigma}$$

where (i) σ is the most general unifier of s and u, (ii) Φ is solved, (iii) $S(u)$ is sort selected, and (iv) Π is either empty or contains exactly one atom.

The refutational completeness of sort resolution on sets of sorted monadic Horn clauses with empty antecedents is a consequence of the results in (Bachmair & Ganzinger 1994). In Chapter 5, we employ Sort Constraint Resolution among other rules also for sets of sorted clauses with non-empty antecedents. The model functor for ordered resolution provides for an appropriate model construction also for sort resolution. The adaption to sorted clauses is straightforward.

Corollary 4.3.3 (Bachmair & Ganzinger (1994))
Let I be the model functor for ordered resolution. Let H be a set of sorted monadic Horn clauses with empty antecedents. We assume that equality is not present in H. Suppose that H is saturated by sort resolution. Then either H contains the empty clause, or else I^H is a model of H.

Note that if I^H is a model of a saturated set H, then I^H corresponds to the minimal model $T^{H'}$ of H' (Weidenbach 1999) where H' contains the term declarations and trivial subsort declarations of H. The clauses in $H \setminus H'$ have unsolved sort constraints which implies that these clauses cannot be productive.

In order to show that the saturation process by sort resolution always terminates for shallow sort theories, we need *sort condensing* which is a particular instance of condensing for sort constraints. Condensing is essentially factorization where the conclusion subsumes the premise.

Definition 4.3.4 (Sort Condensing)
The following simplification is called *Sort Condensing*:

$$\text{Simplify}\frac{S(u), S(u), \Psi \,\|\, \Lambda \rightarrow \Pi}{S(u), \Psi \,\|\, \Lambda \rightarrow \Pi}$$

where (i) u is a possibly non-variable term.

Sort Condensing is a quite restricted form of general condensing. Sort Condensing can be done in quadratic time in the size (number of symbols) of the sort constraint. It is an admissible simplification rule since any premise of Sort Condensing is redundant in the presence of the conclusion. This is also the case for the more restrictive variant of redundancy which is needed for (basic) sorted superposition/paramodulation, c.f. Section 6.3, 6.7, and 7.1. There is a certain class of shallow monadic Horn clauses which includes shallow sort theories and which is closed under sort resolution. However, this class is, in general, not finite. Assuming a finite signature Sort Condensing identifies sufficiently many clauses to demonstrate effectively that this class is finite up to Sort Condensing and variant clauses.

Similarly, Joyner Jr. (1976) characterizes a bound term depth and a bound cardinality of variable components as sufficient conditions for a class of clause sets to be finite up to condensing and variant clauses. Any set of clauses with respect to some finite number of predicate and function symbols, where the depth of terms as well as the cardinality of variable components of these clauses is bound by some constant, is finite up to condensing and variant clauses. For our purpose, Sort Condensing already provides for a sufficient contraction of clauses.

Lemma 4.3.5
Let $\mathcal{S}$ be a (non-linear) shallow sort theory. Then $\mathcal{S}$ can be finitely saturated by sort resolution and Sort Condensing. The productive clauses in the saturated set form a (non-linear) shallow sort theory which consists of term declarations and trivial subsort declarations.

Proof The only clauses in a shallow sort theory which have an unsolved sort constraint are non-trivial subsort declarations. Only descendants of non-trivial subsort declarations are involved as main premises in Sort Constraint Resolution inferences. Conversely, the side premise C' of a Sort Constraint Resolution inference must have a solved sort constraint, i.e. C' is either a (non-linear) shallow term declaration or a trivial subsort declaration. The idea of the proof is then to show that (i) there is a certain subclass of shallow monadic Horn theories which includes $\mathcal{S}$ and which is closed under Sort Constraint Resolution and (ii) this class is finite with respect to Sort Condensing and variant clauses. For (i) we shall argue that the following class of clauses is closed under Sort Constraint Resolution.

$$T_1(t), \ldots, T_n(t), S_1(x_1), \ldots, S_m(x_m) \parallel \rightarrow S(t)$$

where n and m are possibly zero and t is a shallow term. Note that the sort constraint $S_1(x_1), \ldots, S_m(x_m)$ does not have to be solved and $\{x_1, \ldots, x_m\} \subseteq \mathit{vars}(t)$. We call a clause of the above form the *invariant*.

Obviously, the class includes any (non-linear) shallow sort theory. Any inference that involves an invariant as the side premise is only possible if $n = 0$ and the sort constraint of the invariant is solved, i.e. if the invariant is a (non-linear) shallow term declaration or a trivial subsort declaration. Thus an invariant that is involved as the main premise may only be reduced to another invariant by a (non-linear) shallow term declaration or by a trivial subsort declaration which implies (i). Since any invariant contains at most one shallow term t, there only finitely many invariants over a finite signature up to Sort Condensing and variant clauses, which implies (ii). ■

Non-alternating (non-linear) shallow sort theories as well as (alternating, non-linear) shallow sort theories without non-trivial subsort declarations are already saturated by sort resolution. In other words, saturation by sort resolution is the transformation of an alternating sort theory (tree automata) into an essentially equivalent sort theory without non-trivial subsort declarations (alternation rules). However, the alternation in the saturated theory is then encoded in alternating term declarations.

Proposition 4.3.6
Let $\mathcal{S}$ be an alternating (non-linear) shallow sort theory. Then $\mathcal{S}$ can be finitely saturated by sort resolution and Sort Condensing in simply exponential time with respect to $\mathit{size}(\mathcal{S})$ resulting in a sort theory $\mathcal{S}'$ such that $\mathit{size}(\mathcal{S}')$ is at most simply exponentially larger than $\mathit{size}(\mathcal{S})$.

Proof We show that the number of clauses may increase at most simply exponential where each new clause is polynomially bound in size. Given an alternating (non-linear) shallow sort theory $\mathcal{S}$ over a signature Σ, we may assume that Σ contains only the function symbols which occur in clauses in $\mathcal{S}$. Let n be the number of non-trivial subsort declarations in $\mathcal{S}$ and let n_p be the number of distinct monadic atoms in the sort constraints of the n declarations. Suppose that we compute the "reachability" closure among the n non-trivial subsort declarations. More precisely, the number of non-trivial subsort declarations which can be derived from the n non-trivial subsort declarations by an exhaustive application of sort constraint resolution (when selecting all negative sort constraint literals) is at most $n * 2^{n_p}$ denoted by n_Ψ. We consider n_Ψ as a conservative approximation of the number of clauses which are actually derived in a recursive process from non-trivial subsort declarations and new term declarations.

Let m be the number of function symbols in Σ and let k be the maximal arity among all arities of the function symbols in Σ. The instantiation of the non-trivial subsort declaration $S_1(x), \dots, S_l(x) \parallel \rightarrow S(x)$ with the largest sort constraint by a term $f(y_1, \dots, y_k)$ results in a clause C of the form $S_1(f(y_1, \dots, y_k)), \dots, S_l(f(y_1, \dots, y_k)) \parallel \rightarrow S(f(y_1, \dots, y_k))$. Clearly, there are at most $n_\Psi * m$ distinct (linear) instances of this form of the n_Ψ non-trivial subsort declarations and even $n_\Psi * m * 2^{k-1}$ distinct non-linear instances. Let r be the number of term declarations in $\mathcal{S}$ of the form $\Psi \parallel \rightarrow S_1(f(y_1, \dots, y_k))$. Thus sort constraint resolution may derive at most r conclusions from C and these declarations. Suppose that r is the maximal number of term declarations of the above form for all monadic atoms which occur in the sort constraint of a non-trivial subsort declaration. Then there are at most l^r conclusions derivable from C and the term declarations in $\mathcal{S}$. We conclude that there are at most $l^r * n_\Psi * m$ new term declarations derivable by sort constraint resolution in a linear theory. Note that l^r reduces to a polynomial whenever the set of term declarations in $\mathcal{S}$ corresponds to a top-down deterministic automaton.

In a non-linear theory at most $l^r * n_\Psi * m * 2^{k-1}$ new term declarations are derivable. Note that the exponential increase caused by non-linear term declarations can polynomially be represented by syntactic equality constraints which allow, in addition to the conjunction, also the disjunction of equations. ■

The previous proposition states that saturation by sort resolution transforms an alternating (non-linear) shallow sort theory $\mathcal{S}$ into an equivalent at most simply exponentially larger theory $\mathcal{S}'$ which does not contain non-trivial subsort declarations. We show that a subsequent transformation of the remaining alternating term declarations in $\mathcal{S}'$ results in an essentially equivalent non-alternating theory $\mathcal{S}''$ which is still simply exponentially larger than the original theory $\mathcal{S}$. Intuitively, the complexity of the second transformation depends only on the number of "states" in the theory and not on the number of clauses. The idea is inspired by the work of Devienne, Talbot & Tison (1997) and is based on a subset construction on the powerset of all monadic predicate symbols which occur in $\mathcal{S}$. Consider the following inference rule.

Definition 4.3.7 (State Union)
The following inference is called *State Union*:

$$\text{Infer}\frac{\Psi' \parallel \rightarrow S_\Psi(s) \qquad \Phi' \parallel \rightarrow S_\Phi(u)}{\Psi'\sigma, \Phi'\sigma \parallel \rightarrow S_{\Psi\cup\Phi}(s)\sigma}$$

where (i) σ is the most general unifier of s and u.

From an automata-theoretic point of view, if the same input symbol can be read in two states S_Ψ and S_Φ then we may assume that this symbol can be read in a state $S_{\Psi\cup\Phi}$

which is supposed to recognize the intersection of all ground terms recognized in S_Ψ and S_Φ. The exhaustive application of State Union is continued by an exhaustive application of the following simplification rule which actually replaces the premise by the conclusion.

Definition 4.3.8 (De-Alternation)
The following simplification is called *De-Alternation*:

$$\text{Simplify}\frac{S_\Phi(x), S_{\Phi'}(x), \Psi' \parallel \rightarrow \Pi}{S_{\Phi\cup\Phi'}(x), \Psi' \parallel \rightarrow \Pi}$$

Obviously, De-Alternation encodes the alternation over one variable into a single state. In this way, an alternating (non-linear) shallow sort theory $\mathcal{S}'$ without non-trivial subsort declarations can be transformed into a non-alternating theory $\mathcal{S}''$ such that the minimal model of $\mathcal{S}'$ is essentially equivalent to the minimal model of $\mathcal{S}''$.

Proposition 4.3.9
Let $\mathcal{S}$ be an alternating (non-linear) shallow sort theory without non-trivial subsort declarations and let $\mathcal{P}$ be the set of predicate symbols which occur in $\mathcal{S}$. Then $\mathcal{S}$ can be transformed into a non-alternating (non-linear) shallow sort theory $\mathcal{S}'$ such that (i) $T^{\mathcal{S}}(P_1)\cap\ldots\cap T^{\mathcal{S}}(P_n) = T^{\mathcal{S}'}(S_{\{P_1,\ldots,P_n\}})$ for any set of predicate symbols $P_1,\ldots,P_n \in \mathcal{P}$ and (ii) $\mathit{size}(\mathcal{S}')$ is at most simply exponentially larger than $\mathit{size}(\mathcal{S})$ with respect to the cardinality of $\mathcal{P}$.

Proof Given an alternating (non-linear) shallow sort theory $\mathcal{S}$ without non-trivial subsort declarations, we may assume that any predicate symbol $P \in \mathcal{P}$ is equivalent to the predicate symbol $S_{\{P\}}$. Suppose that $\mathcal{S}'$ has been obtained by an exhaustive application of State Union to the clauses in $\mathcal{S}$ followed by an exhaustive application of De-Alternation. The proof of (i) can be done by a simple induction on an admissible atom ordering which is compatible with $\mathcal{S}$ and $\mathcal{S}'$, respectively. For (ii) observe that only State Union introduces new clauses which implies that $\mathcal{S}'$ is larger than $\mathcal{S}$ by at most $2^{|\mathcal{P}|}$ more clauses. ■

The idea of State Union and De-Alternation marks only the beginning of possible improvements to represent efficiently (alternating) theories. For instance, so-called *binary decision diagrams* (BDD) (Bryant 1986, Bryant 1992) may be used, in practice, as an appropriate data structure for an efficient computation of the satisfiability status of (monadic) types.

The next theorem summarizes the discussion and states that the theory of monadic types over shallow sort theories is decidable. Any shallow sort theory may be finitely saturated by sort resolution such that the candidate model of the saturation is the minimal model of the theory. By a subsequent saturation with sort resolution between the saturated theory and a negated monadic type we can effectively compute the satisfiability of the type with respect to the theory.

Theorem 4.3.10 (Weidenbach (1998))
Let $\mathcal{S}$ be a (non-linear) shallow sort theory. The theory $\mathcal{F}_{\mathcal{S}}$ of monadic types over $\mathcal{S}$ is decidable.

Proof Let $\exists\, x_1, \dots, x_n\, (\Psi)$ be a monadic type in $\mathcal{F}_\mathcal{S}$ where $\{x_1, \dots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{S} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds if and only if $T^\mathcal{S} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds if and only if $T^\mathcal{S} \not\models \forall\, x_1, \dots, x_n\, (\neg\Psi)$ holds. In order to check whether $\mathcal{S} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds we add as a goal the clause $\Psi \parallel \rightarrow$ to the saturated set $\mathcal{S}'$ and saturate the result by Sort Constraint Resolution. By Lemma 4.3.5 $\mathcal{S}$ can be finitely saturated into $\mathcal{S}'$. Subsequently, only steps between clauses in $\mathcal{S}'$ and the goal are required. Moreover, since the goal is purely negative we can delete all clauses with unsolved sort constraints from $\mathcal{S}'$. The saturation process terminates since any application of Sort Constraint Resolution either reduces the multiset of all term depths or the overall number of variables. ■

Example 4.3.11
We demonstrate the inference process of soft typing for ordered resolution triggered by static and dynamic sort theories. We assume that a typing function identifies the sort constraint of each clause C as the monadic type of C. Hence, the notion of blocked types induces the notion of *blocked sort constraints*, i.e. a sort constraint is blocked if it is false with respect to the static or dynamic sort theory. For simplicity, we use the clause representation by sorted clauses in order to highlight the blocking part of a clause. For Ordered Resolution and Ordered Factoring inferences, the clauses are assumed to be in the standard format. Suppose that $\succ_a$ is an admissible atom ordering induced by the ordering $R \succ S \succ T \succ Q$ on the predicate symbols where $\succ_a$ is compatible with the subterm property. Let N be a set of clauses as listed in the following table. Maximal literals are marked by *.

$$\begin{array}{lrl}
(1) & \parallel & \rightarrow Q(a)^* \\
(2) & \parallel & \rightarrow R(a,a)^* \\
(3) & \parallel & \rightarrow S(a)^*, T(a) \\
(4) & Q(x), T(x) \parallel R(x,x)^* & \rightarrow S(x) \\
(5) & S(x) \parallel R(x,y)^* & \rightarrow T(y) \\
(6) & T(y) \parallel & \rightarrow R(x, f(f(y)))^* \\
(7) & T(x) \parallel R(x, f^4(x))^* \rightarrow &
\end{array}$$

The dynamic sort theory $\mathcal{S}_D$ with respect to N consists of the two facts $Q(a)$ and $S(a)$ generated from the clauses (1) and (3), respectively. The sort constraints of clauses (4), (6), and (7) are blocked, because T is empty with respect to $\mathcal{S}_D$. There is only one Ordered Resolution step from (2) and (5) yielding (after resolution with (3) on $S(a)$ and factorization of $T(a)$)

$$(8) \qquad \parallel \qquad \rightarrow T(a)^*$$

Now clause (3) becomes redundant and $\mathcal{S}_D$ changes to $\{Q(a), T(a)\}$. The clauses (4), (6), and (7) are no longer blocked. Nevertheless, the Ordered Resolution inferences from (4) and (6), and (6) and (7) result in clauses with a blocked sort constraint. Thus both inferences are blocked since any ground instance of the inferences is blocked, c.f. the lifted version of blocked inferences in Section 3.5. An appropriate approximation of this notion using sorted clauses is to consider the sort constraint of the conclusion. We may assume that an inference is blocked if the sort constraint of the conclusion is blocked. The only possible inference is from (2) and (4) eventually generating

$$(9) \qquad \| \qquad \rightarrow S(a)^*$$

The atom $S(a)$ is added to $\mathcal{S}_D$ and now the inference from (5) and (6) is no longer blocked, resulting in the clause

$$(10) \qquad T(x) \,\| \qquad \rightarrow T(f(f(x)))^*$$

The clause (10) is added to $\mathcal{S}_D$ and an Ordered Resolution step from (6) and (7) results in the clause $T(x), T(f(f(x))) \,\| \rightarrow$ from which the empty clause in derivable in three Ordered Resolution steps.

With respect to our above example, the static sort theory $\mathcal{S}_S$ contains the ground facts $Q(a)$, $S(a)$, $T(a)$ and the two clauses $Q(x), T(x) \,\| \rightarrow S(x)$, $\| \rightarrow T(y)$. The sort T collapses to include arbitrary elements. Nevertheless, the possible inference from (4) and (6) results in a sort constraint that is unsolvable with respect to $\mathcal{S}_S$ allowing to delete the conclusion of this inference.

By relating (alternating) linear shallow sort theories to tree automata, we obtain the following complexity result.

Theorem 4.3.12
The satisfiability problem of the theory of monadic variable types over (alternating) linear shallow sort theories is EXPTIME-complete.

Proof We show that the EXPTIME-hardness holds already for non-alternating linear shallow sort theories by a reduction of the intersection non-emptiness problem of tree automata (Veanes 1997). Given n tree automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ we may assume without loss of generality that the state sets of the automata are mutually disjoint and that each automaton $\mathcal{A}_i$ with $1 \leq i \leq n$ has exactly one final state S_{f_i}. We collect the transition rules of all automata in a non-alternating linear shallow sort theory $\mathcal{S}$, c.f. Proposition 4.2.5. Then $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_n) \neq \emptyset$ if and only if $\mathcal{S} \nvDash S_{\mathrm{f}_1}(x), \dots, S_{\mathrm{f}_n}(x) \,\| \rightarrow$. Note that the intersection non-emptiness problem does not remain EXPTIME-hard if the number of automata is bounded which implies, in particular, that the satisfiability problem is EXPTIME-hard if the number of monadic predicates is not bounded.

We show that the problem is in EXPTIME. Given an alternating linear shallow sort theory $\mathcal{S}$ and a monadic variable type $\exists\, x\,(\Psi)$ from the theory over $\mathcal{S}$, we add the non-trivial subsort declaration $\Psi \,\| \rightarrow S_{\mathrm{f}}(x)$ to $\mathcal{S}$ and transform the result in simply exponential time into a non-alternating linear shallow sort theory $\mathcal{S}'$ due to (Devienne et al. 1997) or, alternatively, by Proposition 4.3.6 and 4.3.9. We may assume that S_{f} does not occur in $\mathcal{S}$. Devienne et al. (1997) demonstrate that a so-called *quasi-automaton* can be transformed in simply exponential time into an equivalent (bottom-up deterministic) tree automaton.

The class of quasi-automata strictly embeds the class of alternating linear sort theories. By Proposition 4.2.5, $\mathcal{S}'$ can be transformed in polynomial time into an equivalent tree automaton $\mathcal{A}$ where we assume that S_{f} is the only final state of $\mathcal{A}$. Then $\mathcal{S} \models \exists\, x\,(\Psi)$ if and only if $L(\mathcal{A}) \neq \emptyset$. Note that the non-emptiness test of tree automata is decidable in linear time. ■

The above theorem implies the EXPTIME-hardness of the satisfiability problem not only of the theory of monadic variable types but also of the theory of (general) monadic types over (alternating) non-linear shallow sort theories.

Corollary 4.3.13
The satisfiability problem of the theory of monadic (variable) types over (alternating, non-linear) shallow sort theories is EXPTIME-hard.

In the proof of Proposition 5.2.8, we will show the EXPTIME-completeness of the satisfiability problem of the theory of (general) monadic types over linear shallow type theories. We obtain as an immediate corollary that the satisfiability problem of the theory of monadic types over (alternating) linear shallow sort theories is EXPTIME-complete. The EXPTIME-completeness of the same problem for non-linear theories remains open.

Corollary 4.3.14
The satisfiability problem of the theory of monadic (variable) types over (alternating) linear shallow sort theories is EXPTIME-complete.

From the perspective of sorted unification, sort resolution implements the test of *well-sortedness* and *emptiness of sorts*, as well as *sorted unification* (Weidenbach 1996*a*) on the sort constraint literals. More precisely, the saturation by sort resolution of a shallow sort theory with a negated ground monadic type corresponds to the test of well-sortedness whereas a negated monadic variable type is an emptiness test of sorts. Recall that a monadic variable type is a monadic type with each term in the type being a variable. The same situation using a negated monadic type with arbitrary terms corresponds to sorted unification with respect to *non-empty* "well-sorted" unifiers. In contrast, sorted unification may also produce *empty* "well-sorted" unifiers which is reflected by sort resolution in the derivation of a monadic type that corresponds to an emptiness test for sorts, as demonstrated by the following theorem.

Theorem 4.3.15 (Weidenbach (1998))
Let $\mathcal{S}$ be a (non-linear) shallow sort theory and let C be a negative clause of the form $S_1(t_1), \ldots, S_n(t_n) \parallel \rightarrow$. We can derive a clause $T_1(y_1), \ldots, T_k(y_k) \parallel \rightarrow$ from C and $\mathcal{S}$ by Sort Constraint Resolution if and only if the sorted unification problem $x_1 = t_1, \ldots, x_n = t_n$ has a "well-sorted" most general unifier with respect to the sort theory $\mathcal{S}$ where each x_i is new and has the sort S_i.

It is important to see that sorted unification does not involve an emptiness test on the sorts $T_1(y_1), \ldots, T_k(y_k)$ which explains the difference in complexity compared to the satisfiability problem of the theory of monadic types. This emptiness test could express an intersection non-emptiness test from an automata-theoretic point of view. Using the following theorem of Weidenbach (1998) in combination with Theorem 4.3.15, we may infer that the satisfiability problem of the theory of monadic types over (non-linear) shallow sort theories is NP-hard.

Theorem 4.3.16 (Weidenbach (1998))
Sorted unification in (non-linear) shallow sort theories is NP-complete, finitary and the number of "well-sorted" most general unifiers is simply exponential in the size of the sort theory and the unification problem.

If we assume that the number of distinct monadic predicate symbols is fixed, an emptiness test for sorts can be done in polynomial time following another theorem of Weidenbach (1998).

Theorem 4.3.17 (Weidenbach (1998))
Suppose that the number of sorts is constant. Then emptiness of sorts with respect to (non-linear) shallow sort theories can be decided in polynomial time.

4.4 Semi-linear Approximations

The theory of monadic types over shallow sort theories is decidable. This result has been improved by Weidenbach (1996*a*) using a so-called (*Semi-linear*) *Flattening* transformation of certain non-shallow sort theories into essentially equivalent shallow sort theories. The transformation introduces a new monadic predicate for each occurrence of a proper non-variable subterm in a term declaration where the number of new symbols is linear in the number of function symbols which occur in the sort theory. The minimal model of the resulting shallow sort theory is at least an upper approximation of the minimal model of the original sort theory. Flattening replaces in a top-down manner each occurrence of a non-variable proper subterm t_i by a variable x where an additional sort constraint on x restricts the ground instances of x to the "well-sorted" ground instances of t. Note that Flattening abstracts from non-linear non-shallow occurrences of variables.

Definition 4.4.1
The following abstraction is called *Flattening*:

$$\text{Abstract}\,\frac{\Psi, \Psi' \parallel \rightarrow P(f(t_1, \ldots, t_i, \ldots, t_n))}{\begin{array}{c}\Psi' \parallel \rightarrow S_{t_i}(t_i)\\ S_{t_i}(x), \Psi, \Psi'' \parallel \rightarrow P(f(t_1, \ldots, x, \ldots, t_n))\end{array}}$$

where (i) Ψ, Ψ' is solved, (ii) t_i is a non-variable term, (iii) Ψ' is maximal such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t_i)$, (iv) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (v) S_{t_i} is a new monadic predicate, and (vi) x is a new variable.

Example 4.4.2
Let $\mathcal{S}$ be a sort theory which contains a term declaration C of the form $S(x), S(y) \,\|\, \rightarrow T(f(g(x), g(x), y, y))$. An exhaustive application of Flattening to $\mathcal{S}$ results in a sort theory $\mathcal{S}'$ which contains $S_{g(x)}(z), S'_{g(x)}(z'), S(y) \,\|\, \rightarrow T(f(z, z', y, y))$ as the "flattened" version of C and new declarations of the form $S(x) \,\|\, \rightarrow S_{g(x)}(g(x))$ and $S(x) \,\|\, \rightarrow S'_{g(x)}(g(x))$. The non-linear shallow occurrences of the variable y are not renamed. $\mathcal{S}'$ is a non-linear shallow sort theory.

In the sequel, a monadic predicate S_t refers to the new predicate which has been introduced by Flattening for a non-variable term t. The newly introduced declarations represent, from an automata-theoretic point of view, tree automata which recognize in a state S_t exactly the "well-sorted" ground instances of the original term t. The following proposition suggests that Flattening may be applied to arbitrary sort theories. However, minimal models are preserved only for sort theories in which each term declaration may contain, beside linear occurrences of variables, also shallow non-linear occurrences of variables. In other words, all occurrences of variables in term declarations have to be linear except for shallow occurrences. For arbitrary sort theories the transformation yields a sort theory whose minimal model is an upper approximation of the minimal model of the original theory.

Proposition 4.4.3
Let $\mathcal{S}$ be a sort theory. Exhaustive application of Flattening to $\mathcal{S}$ terminates and results in a (non-linear) shallow sort theory.

Proof Termination follows from the fact that the Flattening replaces a clause by two clauses which both have less function symbols than the premise. Flattening is not applicable to a clause that is a shallow declaration, since all terms at depth two of the positive atom are always variables (if they exist). On the other hand, if the direct subterm of a positive atom is not shallow, it has a subterm at depth two which is not a variable and therefore Flattening applies. Hence, Flattening terminates in a shallow sort theory. Observe that Flattening introduces a new monadic predicate symbol for each occurrence of a non-variable proper subterm in a term declaration. Thus the number of new symbols is linear in the number of function symbols which occur in the sort theory. Note that the result may contain non-linear shallow sort declarations if and only if $\mathcal{S}$ contains sort declarations with non-linear shallow variable occurrences in positive atoms since these occurrences are not renamed. ■

In the sequel we call a sort theory which has been obtained by an exhaustive application of Flattening to a sort theory $\mathcal{S}$ the *flat approximation of* $\mathcal{S}$. We already mentioned that a flat approximation preserves the minimal model of certain sort theories up to the new predicates while the minimal model of the flat approximation of arbitrary sort theories yields an upper approximation of the original minimal model. As a preliminary lemma, we show that (i) the extension of a new predicate S_t in the minimal model of the flat approximation consists only of ground instances of t and that (ii) these ground instances are "well-sorted" with respect to the sort constraints of the original occurrence of t.

Lemma 4.4.4
Let $\mathcal{S}$ be a sort theory and let $\mathcal{S}_F$ be the flat approximation of $\mathcal{S}$. Let I be the minimal model of $\mathcal{S}_F$. Let t be a non-variable term for which a new monadic predicate symbol S_t has been introduced by Flattening. Let C be the immediate conclusion $\Psi \parallel \to S_t(t)$ of Flattening on t. Then for all ground atoms $S_t(s)$ which are true in I there is a ground substitution σ such that (i) $s = t\sigma$ and (ii) $\Psi\sigma$ is true in I.

Proof The proof is by structural induction on terms. Let t be a non-variable term for which a new monadic predicate symbol S_t has been introduced by Flattening. Let C_F be the final sort declaration $\Phi, \Psi' \parallel \to S_t(t')$ in $\mathcal{S}_F$ where Ψ' is the maximal subset of Ψ such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t')$ and Φ contains only new atoms introduced by Flattening. For simplicity we may assume that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(\Psi)$. Suppose that there is a ground atom $S_t(s)$ which is true in I. The statements (i) and (ii) obviously hold if t is a constant since Ψ is empty in this case. Suppose that t is a non-constant term $f(t_1, \ldots, t_n)$. Thus t' is a term of the form $f(x_1, \ldots, x_n)$. We assume as the hypothesis that (i) and (ii) hold for all proper non-variable subterms in t. It follows that there is a ground substitution τ such that $S_t(t')\tau = S_t(s)$. Note that S_t occurs positively only once in $\mathcal{S}_F$, namely in C_F. Since I is a model of C_F and Φ, Ψ' is solved we have that $\Phi\tau \subseteq I$ and $\Psi'\tau \subseteq I$. Note that Φ contains the new atoms $S_{t_i}(x_i)$ for all non-variable terms t_i in t and thus each $S_{t_i}(x_i)\tau$ is true in I. By the induction hypothesis we may assume that there is a ground substitution σ such that $x_i\tau = t_i\sigma$ and $\Psi_i\sigma$ is true in I for each non-variable term t_i where each Ψ_i is the maximal subset of Ψ such that $\mathit{vars}(\Psi_i) \subseteq \mathit{vars}(t_i)$. Note that $\Psi = \Psi' \cup \bigcup_i \Psi_i$ up to factoring. If t_i is a variable then we may simply let $t_i\sigma := x_i\tau$ which implies that $\Psi'\sigma$ is true in I. We conclude that $s = t\sigma$ holds and $\Psi\sigma$ is true in I. ■

Flattening may be improved by an identification of exact copies of non-variable subterm occurrences. Consider the above example where C is a term declaration of the form $S(x), S(y) \parallel \to T(f(g(x), g(x), y, y))$. We may identify the occurrences of the subterm $g(x)$. Thus the result of an improved flattening of C is a shallow term declaration $S_{g(x)}(z), S(y) \parallel \to T(f(z, z, y, y))$ and only one shallow term declaration for $g(x)$ of the form $S(x) \parallel \to S_{g(x)}(g(x))$. More generally, an according adaption of Flattening to the so-called *Semi-linear Flattening* improves the abstraction for the semi-linear non-shallow occurrences of variables.

Definition 4.4.5
The following abstraction is called *Semi-linear Flattening*:

$$\text{Abstract}\frac{\Psi, \Psi' \parallel \to A[t]_{p_1}}{\begin{array}{c}\Psi' \parallel \to S_t(t)\\ S_t(x), \Psi, \Psi'' \parallel \to A[p_1, \ldots, p_n/x]\end{array}}$$

where (i) A is a monadic atom, (ii) Ψ, Ψ' is solved, (iii) t is a non-variable subterm at position p_1 with $|p_1| = 2$, (iv) the positions $p_1, \ldots, p_n$ refer to all positions q of t in A with

$|q| = 2$, (v) Ψ' is maximal such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t)$, (vi) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (vii) S_t is a new monadic predicate, and (viii) x is a new variable.

The proof of the following proposition on the termination of Semi-linear Flattening is similar to the proof of Proposition 4.4.3. The complexity for the number of new predicate symbols is still linear in the number of function symbols which occur in the original sort theory.

Proposition 4.4.6
Let $\mathcal{S}$ be a sort theory. Exhaustive application of Semi-linear Flattening to $\mathcal{S}$ terminates and results in a (non-linear) shallow sort theory.

In the sequel we call a sort theory which has been obtained by an exhaustive application of Semi-linear Flattening to a sort theory $\mathcal{S}$ the *semi-linear flat approximation of $\mathcal{S}$*. The following lemma on the minimal model properties of the new predicates introduced by Semi-linear Flattening corresponds to the technical Lemma 4.4.4 for Flattening.

Lemma 4.4.7
Let $\mathcal{S}$ be a sort theory and let $\mathcal{S}_{SF}$ be the semi-linear flat approximation of $\mathcal{S}$. Let I be the minimal model of $\mathcal{S}_{SF}$. Let t be a non-variable term for which a new monadic predicate symbol S_t has been introduced by Semi-linear Flattening. Let C be the immediate conclusion $\Psi \parallel \rightarrow S_t(t)$ of Semi-linear Flattening on t. Then for all ground atoms $S_t(s)$ which are true in I there is a ground substitution σ such that (i) $s = t\sigma$ and (ii) $\Psi\sigma$ is true in I.

Proof The proof is similar to the proof of Lemma 4.4.4. The important observation for the extension of Flattening to Semi-linear Flattening is that for a term t only exact copies of t, which occur at the same depth where t occurs, are abstracted into a unique sort. Even variants of t are abstracted into different sorts, for otherwise the result would already be too restrictive. ■

The (Semi-linear) Flattening abstracts sort theories such that the minimal models of the approximations are upper approximations of minimal models of the original sort theories.

Proposition 4.4.8
Let $\mathcal{S}$ be a sort theory and let $\mathcal{S}_F$ be the (semi-linear) flat approximation of $\mathcal{S}$. The minimal model J of $\mathcal{S}_F$ is an upper approximation of the minimal model I of $\mathcal{S}$.

Proof Let $\succ_a$ be an admissible atom ordering which is compatible with $\mathcal{S}$. The proof is by induction on $\succ_a$. Let $P(t)\tau$ be a ground atom which is true in I. Thus there is a clause C of the form $\Psi \parallel \rightarrow P(t)$ in $\mathcal{S}$. Since Ψ is solved we have that $\Psi\tau$ is ground and true in I. Let C_F be the corresponding flattened clause $\Phi, \Psi' \parallel \rightarrow P(t')$ in $\mathcal{S}_F$ where Φ contains the new atoms introduced by (Semi-linear) Flattening and Ψ' is a subset of Ψ, c.f. Definition 4.4.1 (4.4.5). For simplicity we may assume that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(\Psi)$. For the base case suppose that Ψ is empty which implies that Ψ' is empty. We have to show that there is a ground substitution σ such that $\Phi\sigma$ is true in J and $t\tau = t'\tau\sigma$. This is indeed the case using statement (ii) below which can be shown for the base case without using the hypothesis (i). We assume as the hypothesis that (i) for all A where $P(t)\tau \succ_a A$ if $A \in I$ then $A \in J$. Note that $\Psi\tau$ is true in I and by the hypothesis (i) $\Psi\tau$ is also true in J which implies that $\Psi'\tau$ is true in J. Similar to the base case it remains to show that there is a ground substitution σ such that $\Phi\sigma$ is true in J and $t\tau = t'\tau\sigma$. If t is a constant then Ψ is empty. This case belongs to the base case. If t is a variable then $P(t')\tau$ is obviously true in J since Φ is empty and $\Psi'\tau$ is true in J. Suppose that t is a complex term $f(t_1, \dots, t_n)$.

We show that (ii) for each proper non-variable subterm s of t there is a ground atom $S_s(s')$ which is true in J. The proof is by structural induction on terms. Suppose that $f(t_1, \dots, t_n)$ is a term such that each t_i is either a variable or a constant. By definition each constant t_i is abstracted into a term declaration $\| \rightarrow S_{t_i}(t_i)$ in $\mathcal{S}_F$ which implies that the base case of (ii) holds. Suppose that $f(t_1, \dots, t_n)$ is a term such that at least one t_i is a complex term. We assume as the hypothesis that the statement (ii) holds for all proper subterms of the complex terms t_i. Let D be a term declaration $\Phi', \Psi'' \| \rightarrow S_{t_i}(t_i')$ in $\mathcal{S}_F$ for a complex term t_i where Φ' contains the new atoms and Ψ'' is a subset of Ψ, c.f. Definition 4.4.1 (4.4.5). For simplicity we may assume that $\mathit{vars}(\Psi'') \subseteq \mathit{vars}(\Psi)$. Since $\Psi\tau$ is true in I we have that $\Psi''\tau$ is true in I. The hypothesis (i) implies that $\Psi''\tau$ is true in J. By the hypothesis (ii) there is a ground substitution σ such that $\Phi'\sigma$ is true in J. Thus $S_{t_i}(t_i')\tau\sigma$ is true in J which implies (ii).

By Lemma 4.4.4 (Lemma 4.4.7), part (i), we may assume that there is a ground substitution σ such that (ii) holds with $s' = s\sigma$. We conclude that there is a ground substitution σ such that $\Phi\sigma$ is true in J and $t\tau = t'\tau\sigma$. ■

Semi-linear Flattening yields exact approximations for semi-linear sort theories. The proof of the following proposition actually shows that Flattening yields not only an exact approximation for linear sort theories but also for sort theories in which each term declaration may contain, beside linear occurrences of variables, also shallow non-linear occurrences of variables. However, semi-linearity already subsumes this case.

Proposition 4.4.9
Let $\mathcal{S}$ be a linear (semi-linear) sort theory and let $\mathcal{S}_F$ be the (semi-linear) flat approximation of $\mathcal{S}$. The minimal model J of $\mathcal{S}_F$ is equivalent to the minimal model I of $\mathcal{S}$ up to the new atoms introduced by (Semi-linear) Flattening.

Proof The direction $I \subseteq J$ has been shown in the proof of Proposition 4.4.8. In order to show that $J \subseteq I$ up to the new atoms suppose that $\succ_a$ is an admissible atom ordering which is compatible with $\mathcal{S}_F$. The proof is by induction on $\succ_a$. Let $P(t')\tau\lambda$ be a ground atom which is true in J where P is not a new symbol. Thus there is a clause C_F of the form $\Phi, \Psi' \| \rightarrow P(t')$ in $\mathcal{S}_F$ where t' is a shallow term and Φ contains only new atoms introduced by (Semi-linear) Flattening. Since C_F is a sort declaration we may assume that $\Phi\lambda \subseteq J$ and $\Psi'\tau \subseteq J$. Let C be the according original clause $\Psi \| \rightarrow P(t)$ in $\mathcal{S}$ where

Ψ' is a subset of Ψ. For simplicity we may assume that $vars(\Psi') \subseteq vars(\Psi)$. For the base case both Φ and Ψ' are empty. Thus t is shallow and $\Psi' = \Psi$ for otherwise Φ would not be empty. Consequently, Ψ is empty which implies that $P(t)\tau\lambda$ is true in I.

We assume as the hypothesis that for all A where A is not a new atom and $P(t')\tau\lambda \succ_a A$ if $A \in J$ then $A \in I$. If t' is a constant then Φ and Ψ' are both empty. This case belongs to the base case. If t' is a variable then Φ is empty which implies that $\Psi' = \Psi$. By the hypothesis we have that $\Psi'\tau$ is true in I. Thus $P(t)\tau\lambda$ is true in I. Suppose that t' is a shallow term $f(x_1, \dots, x_n)$. Thus t is a term of the form $f(t_1, \dots, t_n)$. Again, by the hypothesis we have that $\Psi'\tau$ is true in I. It remains to show that there is a substitution σ such that $t\tau\sigma = t'\tau\lambda$ and $\Psi\tau\sigma$ is true in I. By definition Φ contains the new atoms $S_{t_i}(x_i)$ for each non-variable unique term t_i. Since $\Phi\lambda$ is true in J we may apply Lemma 4.4.7 to derive that there is a ground substitution σ such that $x_i\lambda = t_i\sigma$ and $\Psi_i\sigma$ is true in J for each non-variable term t_i where each Ψ_i is the maximal subset of Ψ such that $vars(\Psi_i) \subseteq vars(t_i)$. By the hypothesis $\Psi_i\sigma$ is true in I for each non-variable term t_i. Note that $\Psi = \Psi' \cup \bigcup_i \Psi_i$ up to factoring. Since only exact copies of a non-variable term t_i at the same depth are abstracted into a unique sort $S_{t_i}(x_i)$ the ground substitution σ is already of the form such that $t\tau\sigma = t'\tau\lambda$. We conclude that $P(t)\tau\sigma$ is true in I since $\Psi\tau\sigma$ is true in I. ■

It turns out that the theory of monadic types over semi-linear sort theories is decidable (Weidenbach 1996*a*). Semi-linear Flattening is an effective abstraction from semi-linear sort theories to non-linear shallow sort theories for which satisfiability of monadic types has been shown to be decidable.

Theorem 4.4.10
Let $\mathcal{S}$ be a semi-linear sort theory. Then the theory $\mathcal{F}_\mathcal{S}$ of monadic types over $\mathcal{S}$ is decidable.

Proof Let $\mathcal{S}_F$ be the semi-linear flat approximation of $\mathcal{S}$. By Proposition 4.4.6 $\mathcal{S}_F$ can be effectively computed where $\mathcal{S}_F$ is a shallow sort theory. Due to Theorem 4.3.10 the theory of monadic types over shallow sort theories is decidable. Let $\exists\, x_1, \dots, x_n\, (\Psi)$ be a monadic type in $\mathcal{F}_\mathcal{S}$ where $\{x_1, \dots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{S} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds if and only if $T^\mathcal{S} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds and, by Proposition 4.4.9, if and only if $T^{\mathcal{S}_F} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds under the proviso that Ψ does not contain any new atoms introduced by Semi-linear Flattening. It follows that the theory of monadic types over semi-linear sort theories is decidable. ■

The following corollary is a consequence of Corollary 4.3.13.

Corollary 4.4.11
The satisfiability problem of the theory of monadic (variable) types over (alternating) semi-linear sort theories is EXPTIME-hard.

For non-alternating semi-linear sort theories, we can derive the EXPTIME-completeness of the non-emptiness problem from the EXPTIME-completeness of the same problem with respect to $Rec_=$ automata (Comon et al. 1997). Flattening transforms any non-alternating semi-linear sort theory into a linearly larger non-alternating non-linear shallow sort theory.

Corollary 4.4.12
The non-emptiness problem of non-alternating semi-linear sort theories is EXPTIME-complete.

The following corollary is a consequence of Corollary 4.3.14 and Proposition 4.4.3.

Corollary 4.4.13
The satisfiability problem of the theory of monadic (variable) types over (alternating) linear sort theories is EXPTIME-complete.

Weidenbach (1999) argues that even the full first-order theory over semi-linear sort theories is decidable by, however, automata-theoretic means. More precisely, the first-order theory over $Rec_{\neq}$ has been shown to be decidable (Comon & Delor 1994) while a non-linear shallow sort theory can be seen as a $Rec_{\neq}$ automaton. In contrast to the automata-theoretic approach, for an effective saturation-based computation in the first-order theory over semi-linear sort theories, Skolemization has to be carried out with respect to the minimal models of the sort theories.

Theorem 4.4.14 (Weidenbach (1999))
Let $\mathcal{S}$ be a semi-linear sort theory. Then the first-order theory over $\mathcal{S}$ is decidable.

We restrict our attention to saturation-based decision procedures and conclude that (Semi-linear) Flattening is a suitable concept for effective soft typing with respect to arbitrary static/dynamic sort theories and monadic types.

Corollary 4.4.15
Let N_0 be a set of clauses without equality. The (semi-linear) flat approximation of the static sort theory of N_0 is a decidable static approximation for monadic types. Let N_0, N_1, N_2, ... be a fair theorem proving derivation. The sequence of (semi-linear) flat approximations of the dynamic sort theories of each N_i is a decidable dynamic approximation for monadic types.

Chapter 5

Type Theories

We study so-called *type theories* which are certain monadic Horn theories in order to identify decidable fragments for which at least the theory over monadic types is decidable. Recall that sort theories are a special case of monadic Horn theories. In particular, the antecedent of clauses in a sort theory is restricted to a form in which each occurrence of a term is a variable. Sort Abstraction transforms an arbitrary monadic Horn theory into a sort theory by excluding the improper literals from the antecedent of each clause. *Type Abstraction* carries out a similar transformation where, however, all monadic atoms are kept in the approximation. In this way, arbitrary monadic atoms may occur in the antecedent (sort constraint) of the clauses in the approximation. We call these clauses *type declarations* which strictly generalize sort declarations such that the non-constraint part of the antecedent may contain monadic literals. An appropriate representation of type declarations are sorted clauses of the form $\Psi \parallel \Theta \rightarrow S(t)$ where Ψ is the sort constraint and Θ contains arbitrary monadic atoms.

Definition 5.0.1 (Type Declaration)
A sorted clause $\Psi \parallel \Theta \rightarrow S(t)$ is called a *type declaration* if $\Psi \parallel \;\rightarrow S(t)$ is a sort declaration and Θ contains monadic atoms only. It is called *proper* whenever Θ is non-empty. A type declaration is called *shallow* (*linear*, *semi-linear*) if t is shallow (linear, semi-linear).

As a generalization of sort theories, we define *type theories* as finite sets of type declarations.

Definition 5.0.2 (Type Theory)
A *type theory* $\mathcal{T}$ is a finite set of type declarations. $\mathcal{T}$ is called *shallow* (*linear*, *semi-linear*) if all type declarations in $\mathcal{T}$ are shallow (linear, semi-linear).

Note that, in general, a type theory is a monadic Horn theory but not vice versa. We demonstrate that linear shallow type theories are decidable and that arbitrary monadic Horn theories can be effectively transformed into (linear shallow) type theories. Monadic Horn theories, in general, do not comply with the restriction of type theories. More precisely, Type Abstraction removes the improper information and transforms a monadic Horn theory $\mathcal{H}$ (without equality) into a type theory $\mathcal{T}$ in such a way that the minimal model $T^{\mathcal{T}}$ is an upper approximation of $T^{\mathcal{H}}$. There is, however, a trade-off for decidability as linearity has to be imposed on the positive literals in each clause.

Definition 5.0.3
The following abstraction is called *Type Abstraction*:

$$\text{Abstract}\,\frac{S_1(x_1),\ldots,S_n(x_n),\Theta,\Gamma\to S(t)}{S_1(x_1),\ldots,S_n(x_n)\parallel\Theta\to S(t)}$$

where (i) $S_1(x_1),\ldots,S_n(x_n)$ is maximal such that it contains only monadic atoms of the form $T(x)$ with $x\in \mathit{vars}(t)$ and (ii) Γ is maximal such that Γ contains non-monadic atoms and equations only.

Sort Abstraction may remove certain monadic atoms which occur negatively in the antecedent of a clause. Type Abstraction improves the approximation obtained by Sort Abstraction such that all monadic atoms in the antecedent are kept in the abstracted clause. However, there is a trade-off for decidability, since the approximation has to be further abstracted such that all clauses are linear type declarations.

Proposition 5.0.4
Let $\mathcal{H}$ be a monadic Horn theory without equality. The result of an exhaustive application of Type Abstraction to $\mathcal{H}$ is a type theory $\mathcal{T}$ while the minimal model $T^{\mathcal{T}}$ of $\mathcal{T}$ corresponds to the minimal model $T^{\mathcal{H}}$ of $\mathcal{H}$.

Proof By the construction of Type Abstraction it follows immediately that any exhaustive application of the rule is finite and results in a type theory. The minimal model $T^{\mathcal{T}}$ corresponds to $T^{\mathcal{H}}$ since all clauses with positive literals are abstracted into $\mathcal{T}$. ■

We call a type theory $\mathcal{T}$ which has been obtained by an exhaustive application of Type Abstraction to the static or dynamic monadic Horn theory $\mathcal{H}$ of a set N of clauses the *type theory of* N. In order to emphasize that $\mathcal{T}$ has been obtained from $\mathcal{H}$ we may also say that $\mathcal{T}$ is the *static* or *dynamic type theory of* N, respectively. The following corollary is an immediate consequence of Proposition 5.0.4 and Corollary 4.0.7 for sets of clauses without equality.

Corollary 5.0.5
Let N_0 be a set of clauses without equality. The static type theory of N_0 is a static approximation for monadic types. Let $N_0, N_1, N_2, \ldots$ be a fair theorem proving derivation. The sequence of dynamic type theories of each N_i is a dynamic approximation for monadic types.

Type theories are represented by sorted monadic Horn clauses where the antecedent of the clauses is divided into a sort constraint part and a clause part. The Sort Constraint Resolution inference rule operates on the sort constraint part whereas Type Resolution has been designed to resolve the literals in the clause part of the antecedent. However, from a technical point of view, saturation by type resolution may generate clauses of the form $\Psi \parallel S_1(x_1), \ldots, S_n(x_n) \to \Pi$ which may be transformed into logically equivalent clauses of the form $\Psi, S_1(x_1), \ldots, S_n(x_n) \parallel \to \Pi$. The *Type Simplification* does so and guarantees in this way not only that we obtain the same form of productive clauses as in the saturation of sort theories but also provides for technical simplicity for the extension of type theories to equality, c.f. Chapter 7.

Definition 5.0.6 (Type Simplification)
The following inference is called *Type Simplification*:

$$\text{Simplify}\frac{\Psi \parallel \Lambda \to \Pi}{\Psi, \Lambda \parallel \to \Pi}$$

where (i) Λ contains only monadic atoms of the form $S(x)$.

Type Simplification can be justified as a simplification rule in the sense that the premise becomes redundant in the presence of the conclusion whenever the admissible clause ordering distinguishes the occurrences of negative literals in the constraint and non-constraint part. In the sequel, we assume that Type Simplification is applied eagerly with the highest priority in any saturation by type resolution.

5.1 Logic Programs

We shall relate type theories to logic programs with monadic predicates. In fact, type theories, which are finite sets of monadic Horn clauses, correspond exactly to logic programs with monadic predicates (and goals). Hence, in the sequel, by a *monadic logic program* (with monadic predicates only) or simply by a *logic program*, we mean a type theory which does not contain any negative Horn clauses. Note that (monadic) logic programs are already powerful enough to encode any recursively enumerable language. This motivates a classification of decidable fragments of logic programs. An example of a class of logic programs, for which, e.g., the membership problem is EXPTIME-complete (Frühwirth et al. 1991), is the class of *type programs*. Intuitively, a type program corresponds to a "decoupled" version of its original. The so-called *Projection* abstraction transforms logic programs into type programs. Projection "decouples" the relation between the variables which occur in positive atoms in a logic program.

Example 5.1.1
Suppose that a logic program $\mathcal{T}$ contains the clauses $\parallel \to Q(f(a,b))$; $\parallel \to Q(f(b,a))$, and $\parallel Q(f(x,y) \to P(f(x,y))$. The minimal model $T^{\mathcal{T}}$ of $\mathcal{T}$ consists of the ground atoms $Q(f(a,b))$, $Q(f(b,a))$, $P(f(a,b))$, and $P(f(b,a))$. The type program obtained by Projection contains the clauses $\parallel \to Q(f(a,b))$; $\parallel \to Q(f(b,a))$; $\parallel Q(f(x,y)) \to S_1(x)$; $\parallel Q(f(x,y)) \to S_2(y)$, and $S_1(x), S_2(y) \parallel \to P(f(x,y))$. The minimal model of the type program is an upper approximation of $T^{\mathcal{T}}$ with the additional atoms $P(f(a,a))$ and $P(f(b,b))$, up to the new atoms.

We define type programs in terms of particular linear type theories. Observe that the class of type programs is a strict superclass of the class of linear sort theories.

Definition 5.1.2 (Type Program)
Let $\mathcal{T}$ be a type theory. We call $\mathcal{T}$ a *type program* if for each clause C in $\mathcal{T}$ either (i) C is a linear term (sort) declaration, or else (ii) C is a type declaration of the form $\Psi \parallel \Theta \rightarrow S(x)$ where the variable x may occur (non-linearly) in Ψ and Θ.

We call a clause of the form of case (ii) an *expansion clause*. Note that for any subsort declaration there is a logically equivalent expansion clause but not vice versa. Projection introduces for each variable x in the linearized version of a positive atom in a clause C an expansion clause which carries the "type" information of x with respect to the antecedent of C.

Definition 5.1.3
The following abstraction is called *Projection*:

$$\text{Abstract}\,\frac{\Psi \parallel \Theta \rightarrow S(t)}{\begin{array}{r}S_{k_1}(y_1), \ldots, S_{k_m}(y_m) \parallel \quad \rightarrow S(t') \\ \Psi \parallel \Theta \rightarrow S_1(x_1) \\ \vdots \\ \Psi \parallel \Theta \rightarrow S_n(x_n)\end{array}}$$

where (i) $\Psi \parallel \Theta \rightarrow S(t)$ is not a linear term declaration, (ii) t is a non-variable term, (iii) t' is a linear term with $t'\sigma = t$ and σ is a renaming such that the variables in $\mathit{vars}(t') \setminus \mathit{vars}(t)$ are new, (iv) $\{x_1, \ldots, x_n\} = \mathit{vars}(t) \cap \mathit{vars}(\Psi, \Theta)$, (v) $\{y_1, \ldots, y_m\} \subseteq \mathit{vars}(t')$ is maximal such that for all i with $1 \leq i \leq m$ there is a j with $1 \leq j \leq n$ such that $y_i\sigma = x_j$ where $k_i = j$, and (vi) for all j with $1 \leq j \leq n$ the monadic predicate symbol S_j is new.

The following proposition is an immediate consequence of the construction of Projection. In particular, Projection terminates on any logic program, since each application of the abstraction rule transforms a clause into a linear term declaration.

Proposition 5.1.4
Let $\mathcal{T}$ be a logic program. Exhaustive application of Projection to $\mathcal{T}$ terminates and results in a type program.

Proof Termination follows from the fact that Projection transforms the premise into a linear term declaration and several expansion clauses which are not subject to a subsequent abstraction step. Projection generates from clauses, which are not yet linear term declarations, only linear term declarations and expansion clauses which implies that the result of an exhaustive application of Projection is in fact a type program. ■

In the sequel, we call a logic program which has been obtained by an exhaustive application of Projection to a logic program $\mathcal{T}$ the *type program of* $\mathcal{T}$. The following proposition follows by an easy induction on an admissible atom ordering.

Proposition 5.1.5
Let $\mathcal{T}$ be a logic program. The minimal model of the type program of $\mathcal{T}$ is an upper approximation of the minimal model $T^{\mathcal{T}}$.

Recall the concept of (Semi-linear) Flattening for an effective abstraction of arbitrary sort theories into (semi-linear) shallow approximations. We present a related *Typed Flattening* method combined with a certain *Linearization* rule for the transformation of type theories into linear shallow type theories. The minimal model of the approximation is shown to be an upper approximation of the minimal model of the original. The theory of monadic types over linear shallow type theories is decidable (Weidenbach 1999). In the context of sort theories, (semi-)linear sort theories have been identified as classes of sort theories for which (Semi-linear) Flattening yields an exact approximation. However, in contrast to the (semi-)linear sort theories, an exact syntactic characterization of a class of theories from which Typed Flattening and Linearization yield an exact approximation of the original is difficult due to the complex variable occurrences in the monadic Horn clauses. A sufficient condition is, for example, that any two distinct variables which occur in the positive literal of a clause C do not occur both in any variable disjoint partition of the antecedent of C.

The motivation to present type programs and Projection is to show that (Typed) Flattening of type programs yields an exact approximation in the sense that the minimal model of the original logic program is essentially preserved. Moreover, Typed Flattening of arbitrary logic programs (combined with Linearization) improves upon Projection in the sense that the minimal models of the type programs obtained by Projection are upper approximations of the minimal models of the *linear flat approximations*.

Given a type program, (Typed) Flattening transforms each linear term declaration in the program into a linear shallow term declaration. In this way, we obtain logic programs which are called *uniform programs* by, e.g., Charatonik, McAllester, Niwinski, Podelski & Walukiewicz (1998) who show that the non-emptiness problem for uniform programs is EXPTIME-complete (even for sets of infinite trees). Frühwirth et al. (1991) give a definition of uniform programs which is syntactically different but, in essence, describes the same class. The only difference of uniform programs to type programs is that the term declarations in uniform programs are shallow.

Definition 5.1.6 (Uniform Program)
Let $\mathcal{T}$ be a type program. We call $\mathcal{T}$ a *uniform program* if each term declaration in $\mathcal{T}$ is shallow.

Frühwirth et al. (1991) show that the minimal models of uniform programs express exactly the class of *regular sets*, i.e. sets of terms (trees) which are recognizable by a tree automaton. This observation also follows from the fact that uniform programs can be finitely saturated by sort resolution such that the set of productive clauses in the saturated set form a linear shallow sort theory which in turn corresponds to a tree automaton, c.f. Lemma 5.2.5, as demonstrated in the next section. Another motivation of this lemma is to show that the theory of monadic types over linear shallow type theories is decidable.

5.2 Decidability

In order to show that satisfiability of monadic types with respect to linear shallow type theories is decidable, we employ *type resolution* which is an adaption of sort resolution for type theories. Type resolution is also a particular instance of ordered resolution with selection. Under the proviso that the admissible atom ordering is compatible with the subterm property, type resolution is refutationally complete for monadic Horn theories without equality. An atom ordering which is induced by an admissible term ordering is compatible with the subterm property. We assume admissible literal orderings of Definition 3.1.14 and admissible clause orderings of Definition 3.1.15. Note that the satisfiability problem has already been shown to be decidable by Weidenbach (1999). We present a similar proof which is later shown to carry over to more general results with respect to certain equational theories, c.f. Chapter 7.

Given a sorted clause of the form $\Psi \,\|\, \Theta \rightarrow \Pi$ where Θ contains monadic atoms and Π is either a monadic atom or empty, the *type selection* strategy selects the occurrences of negative monadic literals in Θ at first with a particular priority on non-variable terms. In fact, the type selection strategy is equivalent for literal occurrences in Θ and in the unsolved part of Ψ where the occurrences in Θ have a higher priority. Recall that the sort selection strategy selects literals containing non-variable terms first, followed by so-called *empty sorts*, i.e. literals containing variables that do not occur in the non-constraint part of the clause. Literals of subsort declarations are selected last. The motivation of this strategy is to prohibit an increasing term depth during a saturation process by type resolution. The potential increase of the term depth by one may only be tolerated in non-trivial subsort declarations in favor of a technically simpler presentation.

Definition 5.2.1 (Type Selection)
Let C be a sorted clause of the form $\Psi \,\|\, \Theta \rightarrow \Pi$ where Θ contains monadic atoms and Π is either a monadic atom $T(u)$ or empty. A literal $S(t) \in \Psi$ is selected whenever Θ is empty and (i) t is a non-variable term, or else (ii) Ψ contains only variables and $t \notin \mathit{vars}(\Pi)$, or else (iii) Ψ contains only variables (in particular, t is a variable) with $t = u$. A literal $S(t) \in \Theta$ is selected whenever (iv) t is a non-variable term, or else (v) Θ contains only variables and $t \notin \mathit{vars}(\Pi)$, or else (vi) Θ contains only variables (in particular, t is a variable) with $t = u$. No other literal in C is selected. We call this selection strategy *type selection*. A literal which is selected by type selection is called *type selected*.

Type selection is a generalization of sort selection for clauses with non-empty non-constraint antecedents. Note that we could employ case (i) through (iii) of type selection for a termination proof of the saturation of (non-linear) shallow sort theories by sort resolution since the involved clauses have an empty non-constraint antecedent part. For the decidability result on linear shallow type theories, the additional cases (iv) through (vi) are required.

The type resolution calculus consists of the Sort Constraint Resolution rule and the *Type Resolution* rule. Type Resolution corresponds to Sort Constraint Resolution on the non-constraint part of the involved clauses. Factoring is not required for Horn clause sets.

Definition 5.2.2 (Type Resolution)
The following inference is called *Type Resolution*:

$$\text{Infer}\frac{\Phi \parallel \;\to S(s) \qquad \Psi \parallel S(u), \Lambda \to \Pi}{\Phi\sigma, \Psi\sigma \parallel \Lambda\sigma \to \Pi\sigma}$$

where (i) σ is the most general unifier of s and u, (ii) Φ is solved, (iii) $S(u)$ is type selected, and (iv) Π is either empty or contains exactly one atom.

The refutational completeness of type resolution and Type Simplification is a consequence of the results in (Bachmair & Ganzinger 1994). The model functor for ordered resolution provides for an appropriate model construction also for type resolution. The adaption to sorted clauses is straightforward.

Corollary 5.2.3 (Bachmair & Ganzinger (1994))
Let I be the model functor for ordered resolution. Let H be a set of sorted monadic Horn clauses without equality. Suppose that H is saturated by type resolution and Type Simplification. Then either H contains the empty clause, or else I^H is a model of H.

In order to show that the saturation process by type resolution always terminates for linear shallow type theories, we need *intersection condensing* which is, similar to Sort Condensing, a particular instance of condensing for sort constraints. Condensing is essentially factorization where the conclusion subsumes the premise. Intersection Condensing is needed to contract equal conjunctions of atoms in sort constraints which may express intersection non-emptiness problems.

Definition 5.2.4 (Intersection Condensing)
The following simplification is called *Intersection Condensing*:

$$\text{Simplify}\frac{S_1(x), \ldots, S_n(x), S_1(y), \ldots, S_n(y), \Psi \parallel \;\to \Pi}{S_1(x), \ldots, S_n(x), \Psi \parallel \;\to \Pi}$$

where (i) $x, y \notin \mathit{vars}(\Psi, \Pi)$.

Intersection Condensing is a quite restricted form of general condensing. Intersection Condensing can be done in polynomial time in the size (number of symbols) of the sort constraint. It is an admissible simplification rule since any premise of Intersection Condensing is redundant in the presence of the conclusion. This is also the case for the more restrictive variant of redundancy which is needed for (basic) sorted superposition/paramodulation, c.f. Section 6.3, 6.7, and 7.1.

Lemma 5.2.5 (Weidenbach (1999))
Let $\mathcal{T}$ be a linear shallow type theory. Then $\mathcal{T}$ can be finitely saturated by type resolution, Type Simplification, Sort Condensing, and Intersection Condensing. The productive clauses in the saturated set form an (alternating) linear shallow sort theory which consists of (alternating) linear shallow term declarations and trivial subsort declarations.

Proof The idea of the proof is to show that (i) there is a certain subclass of linear shallow monadic Horn theories which includes $\mathcal{T}$ and which is closed under type resolution and (ii) the conclusion of each inference by Sort Constraint Resolution or Type Resolution is either (ii.a) strictly smaller than the main premise with respect to the lexicographic combination of the multiset of all term depths and the number of monadic atoms in the sort constraint and the antecedent, or else (ii.b) a clause of the form $T_1(t), \ldots, T_n(t), S_1(x_1), \ldots, S_m(x_m) \,\|\, \rightarrow S(u)$ where u and t are non-variable shallow terms. Note that there are only finitely many clauses of the form of case (ii.b) over a finite signature up to Sort Condensing, Intersection Condensing, and variant clauses. For (i) we shall argue that the following class of clauses is closed under type resolution and Type Simplification.

$$\Psi_1, \ldots, \Psi_n, S_1(x_1), \ldots, S_m(x_m) \,\|\, \Theta \rightarrow S(t)$$

where n and m are possibly zero, each Ψ_i is of the form $T_1(t_i), \ldots, T_{k_i}(t_i)$ where each t_i is a non-variable term, Θ contains monadic atoms of the form $S(u)$ with u is a non-variable term, and t is a linear shallow term. Note that the constraint $S_1(x_1), \ldots, S_m(x_m)$ does not have to be solved. We call a clause of the above form the *invariant*.

The class of invariants includes any linear shallow type theory. Any inference that involves an invariant as the side premise is only possible if $n = 0$, Θ is empty, and the sort constraint of the invariant is solved, i.e. if the invariant is a linear shallow term declaration or a trivial subsort declaration. Note that the eager application of Type Simplification guarantees that a clause of the form $\Psi \,\|\, S_1(x_1), \ldots, S_m(x_m) \rightarrow \Pi$ is simplified to a clause of the form $\Psi, S_1(x_1), \ldots, S_m(x_m) \,\|\, \rightarrow \Pi$. Thus an invariant that is involved as the main premise may only be reduced to another invariant by a linear shallow term declaration or by a trivial subsort declaration which implies (i).

Consider a Sort Constraint Resolution (Type Resolution) inference from an invariant and a linear shallow term declaration. If the invariant is a non-trivial subsort declaration, then the conclusion is a clause of the form of case (ii.b) where $t = u$. If the invariant is a clause of the form $S_1(x_1), \ldots, S_m(x_m) \,\|\, \rightarrow \Pi$ where there is at least one variable x_i with $1 \leq i \leq m$ and $x_i \notin \mathit{vars}(\Pi)$, then the conclusion is a clause of the form of case (ii.b). In all other cases, the conclusion is a clause with a strictly smaller multiset of all term depths which implies (ii.a). On the other hand, an inference from any invariant and a trivial subsort declaration decreases the number of monadic atoms in the sort constraint (antecedent) by one which shows (ii).

The saturated set may still contain clauses with an unsolved sort constraint and/or non-empty antecedents. However, these clauses cannot be productive which implies that the set of productive clauses are term declarations and trivial subsort declarations which form a linear shallow sort theory. ■

The process of saturation by sort resolution of some linear shallow type theory is strongly related to what has been called *type inference* by Frühwirth et al. (1991). In particular, uniform programs are transformed into so-called *regular programs* similar to the saturation process by sort resolution which transforms a linear shallow type theory into a linear shallow sort theory. In fact, from a model-theoretic point of view, the distinct syntactic classes are essentially equivalent whereas the classes of linear shallow type theories and linear shallow sort theories are syntactically distinguished from uniform programs and regular programs, respectively, but have a strong similarity. More precisely, the above lemma (and (Frühwirth et al. 1991)) shows that linear shallow type theories do not improve upon the expressiveness of linear shallow sort theories. Saturation always yields a candidate model which is completely specified by a linear shallow sort theory. Note that the class of linear shallow sort theories corresponds to the class of tree automata which confirms the observation of Frühwirth et al. (1991) that the minimal models of uniform programs express exactly the class of *regular sets*, i.e. sets of terms (trees) which are recognizable by a tree automaton.

In the context of soft typing, linear shallow type theories may improve the extraction of "regular" information encoded by some clause set compared to linear shallow sort theories. In Section 5.3, we will show that Typed Flattening, which is similar to the Flattening of arbitrary sort theories, computes linear shallow approximations of type theories. In particular, we improve upon the Projection of arbitrary logic programs into type programs.

The following proposition states that, similar to (non-linear) shallow sort theories, saturation by type resolution transforms a linear shallow type theory in simply exponential time into an equivalent linear shallow sort theory. However, in practice, we expect potentially larger saturated theories compared to the saturation of sort theories due to the additional inferences on non-constraint antecedent atoms. In the following proof the size of these atoms is involved as an exponent in the overall number of derivable clauses. Moreover, these atoms may encode arbitrary intersection non-emptiness problems which are checked during saturation.

Proposition 5.2.6
Let $\mathcal{T}$ be a linear shallow type theory. Then $\mathcal{T}$ can be finitely saturated by type resolution, Type Simplification, Sort Condensing, and Intersection Condensing in simply exponential time with respect to $\mathit{size}(\mathcal{T})$ resulting in a type theory $\mathcal{T}'$ such that $\mathit{size}(\mathcal{T}')$ is at most simply exponentially larger than $\mathit{size}(\mathcal{T})$.

Proof We follow the proof of Proposition 4.3.6. We show that the number of clauses may increase at most simply exponential where each new clause is polynomially bound in size. Given a linear shallow type theory $\mathcal{T}$ over a signature Σ, we may assume that Σ contains only the function symbols which occur in clauses in $\mathcal{T}$. Let n be the number of proper type declarations in $\mathcal{T}$ and let q be the number of distinct monadic predicate symbols which occur in the clauses in $\mathcal{T}$. In contrast to the proof of Proposition 4.3.6, it is difficult to compute the "reachability" closure among the n proper type declarations. Instead, we estimate the overall number n_T of derivable new type declarations with respect to the powerset 2^q independently from the number of term declarations in $\mathcal{T}$ to avoid recursive dependencies.

Let m be the maximal size $\mathit{size}(\Theta)$ of the proper type declarations of the form $\Psi \parallel \Theta \rightarrow S(t)$ in $\mathcal{T}$, let k be the maximal number of distinct variables which occur in a proper type declaration in $\mathcal{T}$, and let k_a be the maximal arity among all arities of the function symbols in Σ. Then we may assign to k variables $k * 2^q$ distinct combinations of monadic atoms of the form $T(x)$ which implies that $n * k * 2^q$ linear shallow type declarations of the form $\Psi \,\|\, \rightarrow S(t)$ where Ψ contains only monadic atoms of the form $T(x)$ can be obtained from the n proper type declarations. By i, we refer ambiguously to the quantity $n * k * 2^q$ and the form of the derivable clauses. More generally, there are at most $n * k * k_a * 2^{(m+1)*q}$ linear shallow type declarations derivable from the n proper type declarations.

However, further computations are only possible on the clauses of the form (i). Suppose that C is a clause $\Psi \,\|\, \rightarrow S(t)$ of the form (i). Whenever Ψ contains a monadic atom $T(x)$ with $x \notin \mathit{vars}(t)$ saturation by type resolution actually computes intersection non-emptiness tests on Ψ. Note that there are at most k (q, if $k > q$) intersection non-emptiness problems encoded in C (with respect to Intersection Condensing), since variables which do not occur in the positive atom originate from the non-constraint part of some proper type declaration. Let m_f be the number of function symbols in Σ. Then, from an intersection non-emptiness problem, we may derive $m_f * k_a * 2^q$ subproblems (with respect to Sort Condensing) using (alternating) linear shallow term declarations of the form $\Phi \,\|\, \rightarrow T(f(y_1, \dots, y_{k_a}))$. We conclude that there are at most $i * q * m_f * k_a * 2^q$ clauses derivable from clauses of the form (i). ■

Similar to the saturated alternating shallow sort theories which are obtained from saturation by sort resolution, a saturated type theory can be further transformed by State Union and De-Alternation into an equivalent non-alternating linear shallow sort theory due to Proposition 4.3.9. The next theorem summarizes the discussion and states that the theory of monadic types over linear shallow type theories is decidable. Any linear shallow type theory may be finitely saturated by sort resolution such that the candidate model of the saturation is the minimal model of the theory. By a subsequent saturation with sort resolution between the saturated theory and a negated monadic type we can effectively compute the satisfiability of the type with respect to the theory. A negated monadic type is a negative clause and thus any clause with an unsolved sort constraint may safely be removed from the saturated theory. For the satisfiability test of monadic types with respect to a linear shallow type theory, we arrive at the same test with respect to linear shallow sort theories.

Theorem 5.2.7
Let $\mathcal{T}$ be a linear shallow type theory. The theory $\mathcal{F}_\mathcal{T}$ of monadic types over $\mathcal{T}$ is decidable.

Proof Let $\exists\, x_1, \dots, x_n\, (\Psi)$ be a monadic type in $\mathcal{F}_\mathcal{T}$ where $\{x_1, \dots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{T} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds if and only if $T^\mathcal{T} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds if and only if $T^\mathcal{T} \not\models \forall\, x_1, \dots, x_n\, (\neg\Psi)$ holds. In order to check whether $\mathcal{T} \models \exists\, x_1, \dots, x_n\, (\Psi)$ holds we add as a goal the clause $\Psi \,\|\, \rightarrow$ to the saturated set $\mathcal{T}'$ and saturate the result by type resolution and Type Simplification. By Lemma 5.2.5 $\mathcal{T}$ can be finitely saturated into $\mathcal{T}'$. Subsequently, only steps between clauses in $\mathcal{T}'$ and the goal are required. Moreover, since the goal is purely negative we can delete all clauses with unsolved sort constraints from $\mathcal{T}'$ and obtain $\mathcal{T}''$ which is a linear shallow sort theory. The Theorem 4.3.10 states that the theory of monadic types over (non-linear) shallow sort theories is decidable. ■

Devienne et al. (1997) demonstrate that uniform programs can be transformed into essentially equivalent (non-alternating) tree automata in simply exponential time which implies that the non-emptiness test of uniform programs is in EXPTIME. Following these ideas, we show that the satisfiability problem of the theory of monadic types over linear shallow type theories is EXPTIME-complete. In the next section, we can show that this result carries over to type programs.

Proposition 5.2.8
The satisfiability problem of the theory of monadic (variable) types over linear shallow type theories is EXPTIME-complete.

Proof The EXPTIME-hardness follows from the EXPTIME-completeness of the satisfiability problem of the theory of monadic variable types over (alternating) linear shallow sort theories. We show that the problem is in EXPTIME. Given an alternating linear shallow type theory $\mathcal{T}$ and a monadic type $\exists\, x_1, \ldots, x_n\, (\Psi)$ from the theory over $\mathcal{T}$, we add the expansion clause $\Psi \,\|\, \rightarrow S_{\mathrm{f}}(x)$ to $\mathcal{T}$ where x is a new variable and transform the result in simply exponential time into a non-alternating linear shallow sort theory $\mathcal{T}'$ due to Proposition 5.2.6 and 4.3.9. We may assume that S_{f} does not occur in $\mathcal{T}$. By Proposition 4.2.5, $\mathcal{T}'$ can be transformed in polynomial time into an equivalent tree automaton $\mathcal{A}$ where we assume that S_{f} is the only final state of $\mathcal{A}$. Then $\mathcal{T} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ if and only if $L(\mathcal{A}) \neq \emptyset$. Note that the non-emptiness test of tree automata is decidable in linear time. ■

An immediate corollary of the previous proposition is that the satisfiability problem of the theory of monadic (variable) types over uniform programs is EXPTIME-complete.

Corollary 5.2.9
The satisfiability problem of the theory of monadic (variable) types over uniform programs is EXPTIME-complete.

We obtain as a corollary of Theorem 4.4.14 and Lemma 5.2.5 that the full first-order theory over linear shallow type theories is decidable by, however, automata-theoretic means, c.f. the discussion of Theorem 4.4.14.

Corollary 5.2.10 (Weidenbach (1999))
Let $\mathcal{T}$ be a linear shallow type theory. Then the first-order theory over $\mathcal{T}$ is decidable.

5.3 Linear Approximations

The theory of monadic types over linear shallow type theories is decidable. A transformation of arbitrary type theories into linear shallow type theories is possible similar to the

(Semi-linear) Flattening of sort theories. We demonstrate that Typed Flattening combined with Linearization transforms arbitrary type theories (logic programs) into linear shallow type theories such that the minimal model of the approximation is an upper approximation of the minimal model of the original. For type programs, (Typed) Flattening yields an even equivalent linear shallow type theory (uniform program) with respect to minimal models up to the new predicates. We also show that Typed Flattening with Linearization improves upon the Projection abstraction with respect to minimal models. The distinction of Typed Flattening and Linearization is not essential but considerably simplifies the proof presentation.

Typed Flattening introduces a new monadic predicate for each occurrence of a proper non-variable subterm in the positive atom of a clause where the number of new symbols is linear in the number of function symbols which occur in the theory. The minimal model of the resulting shallow type theory is at least an upper approximation of the minimal model of the original theory. Typed Flattening replaces in a top-down manner each occurrence of a non-variable proper subterm t_i by a new variable x where an additional sort constraint on x restricts the ground instances of x to the "well-sorted" ground instances of t. Typed Flattening abstracts from non-linear non-shallow occurrences of variables.

Definition 5.3.1
The following abstraction is called *Typed Flattening*:

$$\text{Abstract}\,\frac{\Phi, \Psi, \Psi' \parallel \Theta \rightarrow P(f(t_1, \ldots, t_i, \ldots, t_n))}{\begin{array}{c}\Psi' \parallel \Theta \rightarrow S_{t_i}(t_i) \\ S_{t_i}(x), \Phi, \Psi, \Psi'' \parallel \Theta \rightarrow P(f(t_1, \ldots, x, \ldots, t_n))\end{array}}$$

where (i) Φ contains all new atoms of the form $S_t(y)$, (ii) Ψ, Ψ' is maximal such that Ψ, Ψ' is solved in the premise, (iii) t_i is a non-variable term, (iv) Ψ' is maximal such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t_i)$, (v) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (vi) S_{t_i} is a new monadic predicate, and (vii) x is a new variable.

Example 5.3.2
Let $\mathcal{T}$ be a type theory which contains a clause C of the form $S(x), S(y) \parallel S(f(x, y)) \rightarrow P(f(g(x), y, y))$. An exhaustive application of Typed Flattening to $\mathcal{T}$ results in a non-linear shallow type theory $\mathcal{T}'$ which contains $S_{g(x)}(z), S(y) \parallel S(f(x, y)) \rightarrow P(f(z, y, y))$ as the "flattened" version of C and a new declaration of the form $S(x) \parallel S(f(x, y)) \rightarrow S_{g(x)}(g(x))$. The non-linear shallow occurrences of the variable y are not renamed.

In the sequel, a monadic predicate S_t refers to the new predicate which has been introduced by Typed Flattening for a non-variable term t. The newly introduced declarations represent, from an automata-theoretic point of view, tree automata which recognize in a state S_t exactly the "well-sorted" ground instances of the original term t. The following proposition suggests that Typed Flattening may be applied to arbitrary type theories. However, minimal models are preserved only for, e.g., type programs. For arbitrary type theories the transformation yields a theory whose minimal model is an upper approximation of the minimal model of the original theory.

Proposition 5.3.3
Let $\mathcal{T}$ be a type theory. Exhaustive application of Typed Flattening to $\mathcal{T}$ terminates and results in a (non-linear) shallow type theory.

Proof Termination follows from the fact that Typed Flattening replaces a clause by two clauses which both have less function symbols than the premise. Typed Flattening is not applicable to a shallow monadic Horn clause, since all terms at depth two of the positive atom are always variables (if they exist). On the other hand, if the direct subterm of a positive atom is not shallow, it has a subterm at depth two which is not a variable and therefore Typed Flattening applies. Hence, Typed Flattening terminates in a shallow type theory. Note that the result may contain non-linear shallow monadic Horn clauses if and only if $\mathcal{T}$ contains monadic Horn clauses with non-linear shallow variable occurrences since these occurrences are not transformed. ■

In the sequel, we call a type theory which has been obtained by an exhaustive application of Typed Flattening to a type theory $\mathcal{T}$ the *flat approximation of* $\mathcal{T}$. The following proposition states that the flat approximation of type programs are uniform programs where the minimal model of the type program is essentially preserved.

Proposition 5.3.4
Let $\mathcal{T}$ be a type program. The flat approximation $\mathcal{T}_F$ of $\mathcal{T}$ (achieved by (Typed) Flattening) is a uniform program and the minimal model of $\mathcal{T}_F$ is equivalent to the minimal model of $\mathcal{T}$ up to the new atoms introduced by (Typed) Flattening.

Proof The only clauses in $\mathcal{T}$ which are subject to (Typed) Flattening are linear term declarations that form a linear sort theory. In fact, the flat approximation of a linear sort theory is a linear shallow sort theory, c.f. Proposition 4.4.3. Note that Typed Flattening of linear sort theories corresponds exactly to Flattening of linear sort theories. By Proposition 4.4.9, the flat approximation (by Flattening) of a linear sort theory has the same minimal model as the original theory up to the new atoms. Thus the result is a uniform program with a minimal model that corresponds to the minimal model of $\mathcal{T}$ up to the new atoms. ■

We may conclude that the theory of monadic types over type programs is decidable. (Typed) Flattening is an effective abstraction from type programs to uniform programs. The satisfiability of monadic types over linear shallow type theories has been shown to be decidable, c.f. Theorem 5.2.7. The class of linear shallow type theories includes the class of uniform programs.

Theorem 5.3.5
Let $\mathcal{T}$ be a type program. Then the theory $\mathcal{F}_{\mathcal{T}}$ of monadic types over $\mathcal{T}$ is decidable.

Proof Let $\mathcal{T}_F$ be the flat approximation of $\mathcal{T}$. By Proposition 5.3.4, we have that $\mathcal{T}_F$ is a uniform program. Due to Theorem 5.2.7, the theory of monadic types over linear shallow type theories is decidable. Let $\exists\, x_1, \ldots, x_n\,(\Psi)$ be a monadic type in $\mathcal{F}_\mathcal{T}$ where $\{x_1, \ldots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{S} \models \exists\, x_1, \ldots, x_n\,(\Psi)$ holds if and only if $T^\mathcal{S} \models \exists\, x_1, \ldots, x_n\,(\Psi)$ holds and, by Proposition 5.3.4, if and only if $T^{\mathcal{S}_F} \models \exists\, x_1, \ldots, x_n\,(\Psi)$ holds under the proviso that Ψ does not contain any new atoms introduced by Typed Flattening. It follows that the theory of monadic types over type programs is decidable. ■

Since the transformation of a type program into an equivalent uniform program can be computed in polynomial time, we obtain as a corollary of Proposition 5.2.8 that the satisfiability problem of the theory of monadic types over type programs is EXPTIME-complete.

Corollary 5.3.6
The satisfiability problem of the theory of monadic (variable) types over type programs is EXPTIME-complete.

The Corollary 5.2.10 states that the first-order theory over linear shallow type theories is decidable (Weidenbach 1999). The Proposition 5.3.4 implies that type programs can be transformed into essentially equivalent uniform programs which are, in fact, linear shallow type theories. Thus the full first-order theory over type programs is decidable by, however, automata-theoretic means. Recall that the first-order theory over $Rec_{\neq}$ has been shown to be decidable (Comon & Delor 1994) while a (non-linear) shallow sort theory can be seen as a $Rec_{=}$ automaton. Uniform programs can be finitely saturated by sort resolution such that the candidate model of the saturated set is defined by productive clauses which form a shallow sort theory. The candidate model corresponds to the minimal model of the saturated set.

Corollary 5.3.7
The first-order theory over type programs is decidable.

As we have already shown in the Example 5.3.2, Typed Flattening does not *linearize* the shallow occurrences of non-linear variables in the positive atoms in each clause of a theory. However, for the decidability of shallow type theories, it seems to be difficult to avoid the *Linearization* of the positive atoms. For example, Weidenbach (1999) argues that the sorted unification problem with respect to arbitrary sort theories, which is undecidable, can be reduced to the same problem with respect to non-linear shallow type theories. Observe that Linearization improves upon a naive renaming by additional "sort" information for each renamed variable.

Definition 5.3.8
The following abstraction is called *Linearization*:

$$\text{Abstract}\frac{\Psi, \Psi' \parallel \Theta \to L[x]_p}{\begin{array}{c}S_{x_1}(x_1), \ldots, S_{x_n}(x_n), \Psi, \Psi' \parallel \Theta \to L[p_1/x_1, \ldots, p_n/x_n] \\ \Psi' \parallel \Theta \to S_{x_1}(x) \\ \vdots \\ \Psi' \parallel \Theta \to S_{x_n}(x)\end{array}}$$

where (i) L is a monadic atom, (ii) x is a non-linear variable in L at position p, (iii) the positions $p_1, \ldots, p_n$ refer to all other positions of x in L, (iv) Ψ is maximal such that $x \notin vars(\Psi)$, (v) each S_{x_i} is a new monadic predicate, and (vi) the $x_1, \ldots, x_n$ do not occur in the premise.

The following proposition suggests that Linearization may be applied to arbitrary type theories. However, for a *linear* flat approximation, we prefer to apply Typed Flattening first, for, in general, a better approximation may be obtained.

Proposition 5.3.9
Let $\mathcal{T}$ be a type theory. Exhaustive application of Linearization to $\mathcal{T}$ terminates and results in a linear type theory.

Proof Termination follows from the fact that Linearization replaces a clause by several clauses which all have less non-linear variable occurrences than the premise. Linearization is not applicable to a linear monadic Horn clause. On the other hand, if there is a non-linear variable occurrence in a positive atom Linearization applies. Hence, Linearization terminates in a linear type theory. Note that if $\mathcal{T}$ is a shallow type theory Linearization only renames the variables in shallow occurrences. The result is a linear shallow type theory. ■

In the sequel, we call a type theory $\mathcal{T}'$ which has been obtained by an exhaustive application of Typed Flattening to a type theory $\mathcal{T}$ followed by an exhaustive application of Linearization to $\mathcal{T}'$ the *linear flat approximation of* $\mathcal{T}$. In order to show that the minimal model of a linear flat approximation is an upper approximation of the minimal model of the original theory, we need a technical lemma similar to Lemma 4.4.4 and 4.4.7, respectively, for the (semi-linear) flat approximation of sort theories. We show that (i) the extension of each predicate P in the minimal model of the linear flat approximation consists only of ground instances of the linear renaming of a term t which occurs in a positive atom in the original theory and that (ii) these ground instances are "well-sorted" with respect to the sort constraints of the original occurrence of t at least on one variable in the linear renaming of t. In particular, by case (i), we mean that the linear flat approximation preserves the term structure of the original terms which occur in positive atoms.

Lemma 5.3.10
Let $\mathcal{T}$ be a type theory and let $\mathcal{T}_F$ be the linear flat approximation of $\mathcal{T}$. Let I be the minimal model of $\mathcal{T}_F$. Suppose that there is a ground atom $P(t')\tau$ which is true in I where t' is a linear shallow term. Let C_F be the clause $\Phi', \Phi, \Psi' \parallel \Theta' \to P(t')$ in $\mathcal{T}_F$ that

produces $P(t')\tau$ into I. Let C be the (non-linear, non-shallow) clause $\Psi \parallel \Theta \rightarrow P(t)$ which corresponds to C_F where Ψ is maximal such that $vars(\Psi) \subseteq vars(t)$ and each term which occurs in Ψ is a variable. We may assume that Φ' contains all new atoms introduced by Linearization, Φ contains all new atoms introduced by Typed Flattening, Ψ' is the maximal subset of Ψ such that $vars(\Psi') \subseteq vars(t')$, and $\Theta' = \Theta$. Then there is a ground substitution σ such that (i) $t'\tau = t''\sigma$ where t'' is the linear renaming of t and (ii) for all variables $x \in vars(t'')$ there is a ground substitution λ such that $(\Psi, \Theta)\sigma|_x\lambda$ is true in I.

Proof The proof is by structural induction on terms. For simplicity we may assume that $vars(\Psi') \subseteq vars(\Psi)$ and $vars(\Theta') = vars(\Theta)$. Suppose that there is a ground atom $P(t')\tau$ which is true in I. If t is a constant then (i) and (ii) obviously hold. However, note also that in this case Ψ is empty and thus Ψ' is empty. Φ' and Φ are also empty. Since $t = t'$ and C_F is true in I there is a ground substitution λ such that $\Theta'\lambda$ is true in I.

Suppose that t is a non-constant term $f(t_1, \ldots, t_n)$. Thus t' is a term of the form $f(x_1, \ldots, x_n)$ and t'' is a term of the form $f(t''_1, \ldots, t''_n)$ where each t''_i is the linear renaming of t_i. We assume as the hypothesis that both statements hold for all proper non-variable subterms in t. Let τ_F and τ_L be two ground substitutions such that $P(t')\tau_F\tau_L = P(t')\tau$ where $Dom(\tau_F) = vars(\Phi)$ and $Dom(\tau_L) = vars(t') \backslash Dom(\tau_F)$. Note that the substitution τ_F instantiates the variables in t' which represent the flattened non-variable arguments t_i of t. The substitution τ_L instantiates the variables in t' which represent the linearized shallow variables t_i of t.

Since I is a model of C_F and $vars(\Phi', \Phi, \Psi') \subseteq vars(t')$ we have that $(\Phi'\tau_L, \Phi\tau_F, \Psi'\tau_L) \subseteq I$. Moreover, there is a minimal substitution λ such that $\Theta'\tau_L\lambda \subseteq I$ where $Dom(\lambda) \subseteq vars(\Theta')$. Note that Φ' contains the new atoms $S_{x_i}(x_i)$ introduced by Linearization for the non-linear shallow variables in t and thus each $S_{x_i}(x_i)\tau_L$ is true in I. Note that Φ contains the new atoms $S_{t_i}(x_i)$ introduced by Typed Flattening for all non-variable terms t_i in t and thus each $S_{t_i}(x_i)\tau_F$ is true in I. By the induction hypothesis for (i) we may assume that there is a ground substitution σ such that $x_i\tau_F = t''_i\sigma$ for each non-variable term t_i. If t_i is a variable then we may simply let $t''_i\sigma\colon= x_i\tau_L$. Note that this also implies that $\Psi'\sigma$ is true in I. It follows that $t'\tau = t''\sigma$ holds which implies (i).

From the induction hypothesis for (ii) it follows that for each non-variable term t_i and for all variables $x \in vars(t''_i)$ there is a ground substitution λ' such that $(\Psi_i, \Theta')\sigma|_x\lambda' \subseteq I$ where each Ψ_i is the maximal subset of Ψ such that $vars(\Psi_i) \subseteq vars(t_i)$. Note that $\Psi = \Psi' \cup \bigcup_i \Psi_i$ up to factoring. We may assume that the hypothesis applies to Θ' since Typed Flattening always inherits this part of the antecedent completely. What remains are the terms t_i which are variables. $\Theta'\sigma|_{t''_i} = \Theta'\tau_L|_{x_i}$ holds for each t_i which is a variable since $t''_i\sigma = x_i\tau_L$. It follows that there is a ground substitution λ' such that $\Theta'\sigma|_{t''_i}\lambda' = \Theta'\tau_L$ and thus $\Theta'\sigma|_{t''_i}\lambda'\lambda = \Theta'\tau_L\lambda$. Therefore, we have that $\Theta'\sigma|_{t''_i}\lambda'\lambda \subseteq I$.

Let t_i be a variable which corresponds to a non-renamed variable x_i in t'. Since $\Psi'\sigma$ is true in I there is a ground substitution λ' such that $\Psi'\sigma|_{t''_i}\lambda'$ is true in I. Suppose that t_i is a variable which corresponds to a renamed variable x_i in t'. Thus there is a new atom $S_{x_i}(x_i)$ in Φ' introduced by Linearization. Since the clause containing $S_{x_i}(x_i)$ as the positive atom is true in I and by the definition of Linearization, there is a ground substitution λ' such that $\Psi''\sigma|_{x_i}\lambda'$ is true in I where Ψ'' is the maximal subset of Ψ such that $vars(\Psi'') = \{x_i\}$. Since all Ψ_i and Ψ'' are pairwise variable-disjoint we conclude that there is a substitution λ' such that $(\Psi, \Theta)\sigma|_x\lambda'$ is true in I for all variables $x \in vars(t'')$.

In fact, we could also prove a stronger statement for (ii) for the shallow variables of a flattened term. Let T be the set of all shallow variables t''_i in t''. Then $\sigma|_T = \tau_L|_T$. Informally, the relation with respect to Ψ, Θ among the variables which occur as arguments of a linear shallow term is not lost by the linear flat approximation since such variables are not approximated. ■

Typed Flattening combined with Linearization abstract type theories such that the minimal models of the approximations are upper approximations of minimal models of the original theories.

Proposition 5.3.11
Let $\mathcal{T}$ be a type theory and let $\mathcal{T}_F$ be the linear flat approximation of $\mathcal{T}$. The minimal model $T^{\mathcal{T}_F}$ is an upper approximation of the minimal model $T^{\mathcal{T}}$.

Proof The proof is similar to the induction proof of Proposition 4.4.8 since the hypothesis extends to include also the non-constraint antecedent part of the clauses in $\mathcal{T}$. This part is completely inherited by Typed Flattening. The prerequisites for an appropriate proof are contained in the proof of part (i) of Lemma 5.3.10. ■

We restrict our attention to saturation-based decision procedures and conclude that Typed Flattening/Linearization is a suitable concept for effective soft typing with respect to arbitrary static/dynamic type theories and monadic types.

Corollary 5.3.12
Let N_0 be a set of clauses without equality. The linear flat approximation of the static type theory of N_0 is a (decidable) static approximation for monadic types. Let N_0, N_1, $N_2, \ldots$ be a fair theorem proving derivation. The sequence of linear flat approximations of the dynamic type theories of each N_i is a (decidable) dynamic approximation for monadic types.

The following proposition states that the minimal model of the type program of a logic program $\mathcal{T}$ (type theory) obtained by Projection is a (proper) upper approximation of the minimal model of the linear flat approximation of $\mathcal{T}$. The proof shows the potential strictness by a counterexample for the other direction.

Proposition 5.3.13
Let $\mathcal{T}$ be a type theory and let $\mathcal{T}_F$ be the linear flat approximation of $\mathcal{T}$. Let $\mathcal{T}_P$ be the type theory which has been obtained by an exhaustive application of Projection to $\mathcal{T}$. Then $T^{\mathcal{T}_P}$ is an upper approximation of $T^{\mathcal{T}_F}$ up to the new atoms introduced by the abstractions but not vice versa.

Proof Let $\succ_a$ be an admissible atom ordering which is compatible with $\mathcal{T}_F$. The proof is by induction on $\succ_a$. Let $P(t')\tau$ be true in $T^{\mathcal{T}_F}$ where P is not a new symbol. Thus there is a clause C_F of the form $\Phi', \Phi, \Psi' \,\|\, \Theta' \to P(t')$ in $\mathcal{T}_F$ and the original clause C of the form $\Psi \,\|\, \Theta \to P(t)$ in $\mathcal{T}$ where Ψ is maximal such that $vars(\Psi) \subseteq vars(t)$ and each term which occurs in Ψ is a variable, t' is a linear shallow term, Φ' contains all new atoms introduced by Linearization, Φ contains all new atoms introduced by Typed Flattening, Ψ' is the maximal subset of Ψ such that $vars(\Psi') \subseteq vars(t')$, and $\Theta' = \Theta$. For simplicity we may assume that $vars(\Psi') \subseteq vars(\Psi)$ and $vars(\Theta') = vars(\Theta)$. Let C_P be the according clause $S_1(y_1), \ldots, S_m(y_m) \,\|\, \to P(t'')$ in $\mathcal{T}_P$ where t'' is the linear renaming of t. Furthermore, there are m clauses $C_i = \Psi \,\|\, \Theta \to S_i(y_i)$ in $\mathcal{T}_P$. For simplicity we may assume that $vars(C_i) \subseteq vars(C)$ for each C_i. For the base case Ψ and Θ are empty. Thus Φ', Ψ', and Θ' are empty and $m = 0$. It follows that $P(t'')\lambda$ is true in $T^{\mathcal{T}_P}$ for all ground substitutions λ. By Lemma 5.3.10, part (i), there is a ground substitution σ such that $t'\tau = t''\sigma$ which implies that $P(t')\tau$ is true in $T^{\mathcal{T}_P}$.

We assume as the hypothesis that for all B where $P(t')\tau \succ_a B$ and B is not a new atom if $B \in T^{\mathcal{T}_F}$ then $B \in T^{\mathcal{T}_P}$. If t' is a variable or a constant then $P(t')\tau$ is obviously in $T^{\mathcal{T}_P}$. Note that a clause containing a positive atom of this form is not part of the abstractions. Consequently, we may apply the hypothesis directly to all antecedent literals in C_F. Finally, we may assume that t' is a term of the form $f(x_1, \ldots, x_n)$, i.e. t is a term of the form $f(t_1, \ldots, t_n)$ and t'' is a term of the form $f(t''_1, \ldots, t''_n)$ where each t''_i is the linear renaming of t_i. By Lemma 5.3.10, part (i), there is a ground substitution σ such that $t'\tau = t''\sigma$. By part (ii) we have that for all variables $x \in vars(t'')$ there is a ground substitution λ such that $(\Psi, \Theta)\sigma|_x\lambda \subseteq T^{\mathcal{T}_F}$ which implies, by the induction hypothesis, that $(\Psi, \Theta)\sigma|_x\lambda \subseteq T^{\mathcal{T}_P}$. As a consequence, for all variables y_i there is a ground substitution λ such that each $S_i(y_i)\sigma|_{y_i}$ is true in $T^{\mathcal{T}_P}$ since $T^{\mathcal{T}_P}$ is a model of each C_i. Since $T^{\mathcal{T}_P}$ is also a model of C_P we have that $P(t'')\sigma$ is true in $T^{\mathcal{T}_P}$ and we conclude that $P(t')\tau$ is true in $T^{\mathcal{T}_P}$.

The following example shows that, in general, the minimal model $T^{\mathcal{T}_F}$ is not an upper approximation of $T^{\mathcal{T}_P}$. Suppose that $\mathcal{T}$ contains the clauses $\| \to Q(f(a,b))$; $\| \to Q(f(b,a))$, and $\| Q(f(x,y) \to P(f(x,y))$. The linear flat approximation of N contains the clauses $\| \to S_1(a)$; $\| \to S_2(b)$; $S_1(x), S_2(y) \,\|\, \to Q(f(x,y))$; $\| \to S_3(b)$; $\| \to S_4(a)$; $S_3(x), S_4(y) \,\|\, \to Q(f(x,y))$, and $\| Q(f(x,y) \to P(f(x,y))$. In this case, the minimal model $T^{\mathcal{T}_F}$ of the approximation is exactly the minimal model $T^{\mathcal{T}}$ up to the new atoms, i.e. $T^{\mathcal{T}_F}$ contains $Q(f(a,b))$, $Q(f(b,a))$, $P(f(a,b))$, and $P(f(b,a))$. On the other hand, the projection contains the clauses $\| \to Q(f(a,b))$; $\| \to Q(f(b,a))$; $\| Q(f(x,y)) \to S_1(x)$; $\| Q(f(x,y)) \to S_2(y)$, and $S_1(x), S_2(y) \,\|\, \to P(f(x,y))$. The minimal model $T^{\mathcal{T}_P}$ of the projection is a superset of $T^{\mathcal{T}_F}$ up to the new atoms. It contains up to the new atoms the additional atoms $P(f(a,a))$ and $P(f(b,b))$. ■

Chapter 6

Sorted Equational Theories

(Semi-linear) sort theories and (linear) type theories have been characterized as decidable fragments of the general class of monadic Horn theories. In particular, the theory of monadic types over these theories is decidable by saturation-based methods. The approximation of an arbitrary theory by a decidable fragment may allow to decide the satisfiability of a formula with respect to the original theory. We have shown that soft typing for clausal inference systems, as a general framework for semantically guided theorem proving, can effectively be used with respect to approximations by these sort and type theories.

Sort Abstraction and Monadic Horn Abstraction have been designed to filter out improper information from the clauses, in particular, any occurrences of equality. However, a generalization of monadic Horn theories which include equality may provide for a better approximation of the satisfiability of monadic types and may also allow to decide the unifiability of first-order terms with respect to these equational theories. We call the unifiability problem with respect to a (sorted) equational theory the *E-unifiability problem*, c.f. Chapter 2. The *word problem* is a particular instance of the E-unifiability problem on ground terms. The process of unifying terms with respect to an equational theory is called *E-unification*. For a formal definition of (sorted) equational theories see Definition 6.1.2 below. A decidable E-unifiability problem may allow the effective blocking with respect to (monadic) equational types.

A major aspect of the decidable sort and type theories is that these fragments can be naturally extended to decidable *sorted equational theories* and *typed equational theories*, respectively. We demonstrate that the decidability of E-unifiability with respect to these theories can be obtained by a generalization of sort resolution to *sorted superposition* (Weidenbach 1996*a*) which is a specialized form of (basic) superposition with selection (Bachmair et al. 1995, Nieuwenhuis & Rubio 1995). The decidable sort theories are strongly related to certain tree automata whereas the decidable sorted equational theories do not have an immediate counterpart in this area. In fact, we can show that even $Rec_{\neq}$ automata (Bogaert & Tison 1992) do not appropriately capture these fragments. The so-called *standard theories* (Nieuwenhuis 1996) are, however, closely related to the decidable sorted equational theories. Our results strictly improve upon the standard theories.

This chapter is also devoted to the analysis of abstractions that effectively transform general Horn theories with equality into the decidable equational theories such that the minimal model of the abstracted theory is an upper approximation of the minimal model of the original theory. We present effective transformations to infer automatically abstractions of Horn theories which lie in the class of decidable approximations.

6.1 Introduction to Sorted Equational Theories

We study so-called *sorted equational theories* which are a particular form of monadic Horn theories with equality. A sorted equational theory can be seen as a sort theory where positive occurrences of equations are possible. That is, a sorted equational theory consists of sort declarations and so-called *sorted equations* which correspond to equations with an additional sort information on the variables. An appropriate representation of a sorted equation is a sorted clause of the form $\Psi \,\|\, \rightarrow s \approx t$ where Ψ is the sort constraint. It turns out that the similarity of sort constraints and monadic types is a coincidence. For example, an extension of sort constraints by (dis-)equations leads to inference processes by sorted superposition which are difficult to control, c.f. Chapter 7. We distinguish sort constraints and monadic equational types as particular data structures to obtain, on one hand, decision procedures and, on the other hand, to effectively control soft typing.

Definition 6.1.1 (Sorted Equation)
A sorted clause of the form $S_1(x_1), \ldots, S_n(x_n) \,\|\, \rightarrow s \approx t$ is called a *sorted equation* if either $S_1(x_1), \ldots, S_n(x_n)$ is solved, or else $s \approx t$ is disjoint collapsing or disjoint universal with $\{x_1, \ldots, x_n\} \subseteq \mathit{vars}(s \approx t)$. A sorted equation $\Psi \,\|\, \rightarrow s \approx t$ is called *collapsing* (*non-collapsing*, *universal*, *disjoint*, *shared*) if $s \approx t$ is *collapsing* (*non-collapsing*, *universal*, *disjoint*, *shared*). A disjoint collapsing or disjoint universal sorted equation $\Psi \,\|\, \rightarrow s \approx t$ is called *trivial* if $\mathit{vars}(\Psi) \cap \mathit{vars}(s) = \emptyset$ or $\mathit{vars}(\Psi) \cap \mathit{vars}(t) = \emptyset$.

A sorted clause $\Psi \,\|\, s \approx t \rightarrow$ is called a *sorted disequation* if $\Psi \,\|\, \rightarrow s \approx t$ is a sorted equation. A sorted equation $\Psi \,\|\, \rightarrow s \approx t$ (disequation $\Psi \,\|\, s \approx t \rightarrow$) is called *shallow* (*linear*, *semi-linear*) if the equation $s \approx t$ is shallow (linear, semi-linear). A non-collapsing sorted equation of the form $\Psi \,\|\, \rightarrow s \approx t$ where Ψ is empty is called *non-alternating*. Otherwise, we may call a sorted equation *alternating*.

The notion of sort constraints allows for a technically simpler presentation of a particular combination of the term ordering and the selection strategy in the sorted superposition calculus. Moreover, we may impose so-called *basic restrictions* on the sort constraints to block further superposition inferences (Weidenbach 1996*a*). Thus sort constraints are convenient but not essential. We arrive at the formal definition of *sorted equational theories.*

Definition 6.1.2 (Sorted Equational Theory)
A *sorted equational theory* $\mathcal{E}$ is a finite set of sorted equations and sort declarations. $\mathcal{E}$ is called *shallow* (*linear*, *semi-linear*) if the sorted equations and sort declarations in $\mathcal{E}$ are shallow (linear, semi-linear). $\mathcal{E}$ is called *non-alternating* if $\mathcal{E}$ does not contain

any alternating term declarations, alternating sorted equations, and non-trivial subsort declarations. Otherwise, we may call $\mathcal{E}$ *alternating.*

Note that, in general, a sort theory is a sorted equational theory but not vice versa while a shallow and/or linear sorted equational theory is a semi-linear sorted equational theory but not vice versa. The motivation of this classification is to distinguish shallow and semi-linear theories. We demonstrate that (non-linear) shallow sorted equational theories are decidable and that semi-linear sorted equational theories can be effectively transformed into essentially equivalent (non-linear) shallow sorted equational theories. Using this transformation, we can extend the decidability result to semi-linear sorted equational theories. Monadic Horn theories, in general, do not obey the restriction of sorted equational theories. *Equational Sort Abstraction* removes the improper information and transforms a monadic Horn theory $\mathcal{H}$ into a sorted equational theory $\mathcal{E}$ in such a way that the minimal model $T^{\mathcal{E}}$ is an upper approximation of $T^{\mathcal{H}}$.

Definition 6.1.3
The following abstraction is called *Equational Sort Abstraction*:

$$\text{Abstract}\frac{S_1(x_1),\ldots,S_n(x_n),\Gamma\rightarrow L}{S_1(x_1),\ldots,S_n(x_n)\,\|\,\rightarrow L}$$

where (i) L is either a monadic atom or an equation and (ii) $S_1(x_1),\ldots,S_n(x_n)$ is maximal such that it contains only monadic atoms of the form $T(x)$ with $x\in \mathit{vars}(L)$.

Equational Sort Abstraction removes certain monadic atoms which occur negatively in the antecedent of a clause. We present a better approximation in this regard in Chapter 7. However, there is a trade-off for decidability, since the approximation requires all clauses to be linear declarations and equations.

Proposition 6.1.4
Let $\mathcal{H}$ be a monadic Horn theory. The result of an exhaustive application of Equational Sort Abstraction to $\mathcal{H}$ is a sorted equational theory $\mathcal{E}$ while the minimal model $T^{\mathcal{E}}$ of $\mathcal{E}$ is an upper approximation of the minimal model $T^{\mathcal{H}}$ of $\mathcal{H}$.

Proof By the construction of Equational Sort Abstraction it follows immediately that any exhaustive application of the rule is finite and results in a sorted equational theory. The minimal model $T^{\mathcal{E}}$ is in fact an upper approximation of $T^{\mathcal{H}}$ since all clauses with positive literals are abstracted into $\mathcal{E}$ while only negative literals are removed. ■

In the sequel, we call a sorted equational theory $\mathcal{E}$ which has been obtained by an exhaustive application of Equational Sort Abstraction to the static or dynamic monadic Horn theory $\mathcal{H}$ of a set N of clauses the *sorted equational theory of* N. In order to emphasize that $\mathcal{E}$ has been obtained from $\mathcal{H}$ we may also say that $\mathcal{E}$ is the *static* or *dynamic sorted equational theory of* N, respectively. The following corollary is an immediate consequence of Proposition 6.1.4 and Corollary 4.0.7.

Corollary 6.1.5
Let N_0 be a set of clauses. The static sorted equational theory of N_0 is a static approximation for monadic equational types. Let N_0, N_1, N_2, … be a fair theorem proving derivation. The sequence of dynamic sorted equational theories of each N_i is a dynamic approximation for monadic equational types.

6.2 Tree Automata and Standard Equational Theories

We shall relate sorted equational theories, on one hand, to certain combinations of tree automata with equational theories inspired by Comon (1995), and, on the other hand, to so-called *standard theories* (Nieuwenhuis 1996). On the way to an effective computation of the satisfiability of (ground) monadic equational types with respect to a sorted equational theory, the E-unifiability (word) problem with respect to the theory has to be solved. Recall that a sorted equational theory, in general, contains a sort theory and a finite set of sorted equations. In Section 4.2, we have already demonstrated that certain classes of sort theories correspond to classes of tree automata.

Given a signature Σ, we denote the closure of a set L of ground terms from $\mathcal{T}(\Sigma)$ with respect to an equational theory $\mathcal{E}$ by $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L){:}= \{s \in \mathcal{T}(\Sigma) \mid \exists t \in L \text{ with } t \overset{*}{\Leftrightarrow}_{\mathcal{E}} s\}$. Suppose that $\mathcal{E}$ is a linear shallow sorted equational theory which contains a sort theory $\mathcal{S}$ and a set $\mathcal{E}$ of equations with empty sort constraints. The sort theory $\mathcal{S}$ corresponds to a tree automaton $\mathcal{A}$ and $\mathcal{E}$ is an equational theory. Let L be the set of ground terms recognized by $\mathcal{A}$. Then the word problem $s \overset{*}{\Leftrightarrow}_{\mathcal{E}} t$ for ground terms s and t can be reduced to the membership problem for $s \in (\overset{*}{\Leftrightarrow}_{\mathcal{E}})(\{t\})$ if the closure set is recognizable. More generally, for a goal $s \approx t$ where s and t are both linear terms with $\mathit{vars}(s) \cap \mathit{vars}(t) = \emptyset$, E-unifiability with respect to $\mathcal{E}$ is equivalent to the non-emptiness problem of $\{s\sigma \mid \sigma \text{ is a ground substitution}\} \cap (\overset{*}{\Leftrightarrow}_{\mathcal{E}})(\{t\sigma \mid \sigma \text{ is a ground substitution}\})$. Note that the set of ground instances of s and t are both recognizable by tree automata.

Kaji, Toru & Kasami (1997) as well as Comon (1995) and Jacquemard (1996) investigate the recognizability of the closure of recognizable sets with respect to certain forms of term rewriting systems. Kaji et al. (1997) show the recognizability of the (right-)closure of a certain class of (right-)linear, confluent term rewriting systems applied to linear terms. The variables occurring in the left and right hand side of a rule $s \Rightarrow t$ are assumed to be linear in s and, moreover, s and the subterms of t are related by additional restrictions which can be effectively computed. Comon (1995), on the other hand, shows with the following theorem that the E-unifiability problem over linear (disjoint) terms with respect to certain linear equational theories is decidable. In fact, the construction of Comon (1995) can be used to decide the E-unifiability problem over linear terms with respect to the systems of Kaji et al. (1997). Jacquemard (1996) generalized the results of Comon (1995) to so-called linear *growing term rewriting systems*.

Theorem 6.2.1 (Comon (1995))
Let $\mathcal{E}$ be a linear equational theory where all variable occurrences in the equations in $\mathcal{E}$ are shallow. Let L be a recognizable language. Then $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ is a recognizable language.

Arbitrary ground terms may occur in equations in $\mathcal{E}$. However, these occurrences do not increase the expressiveness upon linear shallow equational theories as ground terms may always be represented by, e.g., a finite number of linear shallow sort declarations. We briefly sketch the idea of the proof of the above theorem. The construction is based on a tree automaton $\mathcal{A}_0$ which recognizes L and which contains a unique state S_{s_i} for each ground subterm s_i in each equation $f(s_1,\dots,s_n) \approx t$ in $\mathcal{E}$ such that s_i (and only s_i) is recognized by $\mathcal{A}_0$ in S_{s_i}. If s_i is a variable then S_{s_i} is the final state of a universal tree automaton. In some sense the subterms s_i are abstracted by $\mathcal{A}_0$. Note that $\mathcal{A}_0$ can be constructed such that the difference of the size of the automaton which recognizes L, i.e. the number of transition rules, and the size of $\mathcal{A}_0$ is linear in the size of $\mathcal{E}$, i.e. the number of function and predicate symbols which occur in the equations of $\mathcal{E}$. Then $\mathcal{A}_0$ is completed with respect to the inference rule *Automata Completion.*

Definition 6.2.2 (Automata Completion)
The following inference is called *Automata Completion*:

$$\text{Infer}\frac{f(s_1,\dots,s_n) \approx g(t_1,\dots,t_m) \qquad S_1(x_1),\dots,S_n(x_n) \parallel \to S(f(x_1,\dots,x_n))}{T_1(x_1),\dots,T_m(x_m) \parallel \to S(g(x_1,\dots,x_m))}$$

where (i) $f(s_1,\dots,s_n) \approx g(t_1,\dots,t_m)$ is a linear equation in which variables may only occur at shallow positions, (ii) if t_i with $1 \leq i \leq m$ is a ground term or a variable x with $x \notin \mathit{vars}(f(s_1,\dots,s_n))$, then $T_i{:}= S_{t_i}$, or else (iii) if t_i with $1 \leq i \leq m$ is a variable such that $t_i = s_j$ with $1 \leq j \leq n$, then $T_i{:}= S_j$.

The completion does not introduce new states which implies that only polynomially many new transition rules are added to $\mathcal{A}_0$ with respect to the number of rules of $\mathcal{A}_0$ and the number of equations in $\mathcal{E}$. The conditions (i) through (iii) can be checked in linear time in the size of $\mathcal{E}$. An inference by paramodulation from the premises of Automata Completion yields a clause of the form $S_1(s_1),\dots,S_n(s_n) \parallel \to S(g(t_1,\dots,t_m))$. Due to the construction of the Automata Completion, the conclusion is equivalent to the clause $T_1(x_1),\dots,T_m(x_m) \parallel \to S(g(x_1,\dots,x_m))$. In this way, we arrive at a close relation between the automata-theoretic approach and our saturation-based method using superposition/paramodulation. Note, however, that the idea of Automata Completion actually inspired our approach. Another justification to use concepts with a stronger expressiveness than automata-theoretic methods is that the recognizability result of Theorem 6.2.1 cannot be extended to non-linear shallow equational theories as stated in the following lemma. The proof is based on an easy pumping argument.

Lemma 6.2.3
There is a recognizable set L and a non-linear shallow equational theory $\mathcal{E}$ such that $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ is not recognizable.

Proof Suppose that the signature Σ consists of a binary function symbol f, a unary function symbol s, and a constant symbol a. Let L be the singleton $\{a\}$ and let $\mathcal{E}$ be the non-linear shallow equational theory $\{f(x,x) \approx a\}$. Suppose, for the purpose of a contradiction, that $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ is recognized by a tree automaton $\mathcal{A}$ where $\mathcal{S}$ denotes the distinguished set of final states of $\mathcal{A}$. We assume, without loss of generality, that $\mathcal{A}$ is deterministic. Since $\mathcal{A}$ has only finitely many states, there are two distinct terms $s^{n_1}(a)$ and $s^{n_2}(a)$ which are recognized by $\mathcal{A}$ in the same state S. Note that $s^n(a)$ denotes the term $s(s^{n-1}(a))$ for any $n \geq 0$ where $s^0(a) := a$. Since we have that $f(s^{n_1}(a), s^{n_1}(a)) \in (\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ and $\mathcal{A}$ is deterministic, there is a sorted clause (transition rule) of the form $S(x_1), S(x_2) \parallel \rightarrow S_{\mathrm{f}}(f(x_1, x_2))$ where $S_{\mathrm{f}} \in \mathcal{S}$ is a final state of $\mathcal{A}$. Thus it follows that $\mathcal{A} \models S_{\mathrm{f}}(f(s^{n_1}(a), s^{n_1}(a)))$. However, we may also conclude that $\mathcal{A} \models S_{\mathrm{f}}(f(s^{n_1}(a), s^{n_2}(a)))$ which shows the contradiction because the term $f(s^{n_1}(a), s^{n_2}(a))$ is not in $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$. ■

There is a simple $Rec_=$ automaton $\mathcal{A}$, though, which recognizes the language $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ from the above proof. Take $\parallel \rightarrow T(a)$; $T(x) \parallel \rightarrow T(s(x))$; $T(x), T(y) \parallel \rightarrow T(f(x,y))$, and $T(x) \parallel \rightarrow S_{\mathrm{f}}(f(x,x))$ as the only transition rules of $\mathcal{A}$ where S_{f} is the final state of $\mathcal{A}$. Then $L(\mathcal{A})$ is obviously the set $\{f(x,x) \mid x \in \mathcal{T}(\Sigma)\}$ of ground terms which corresponds to $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$. However, the next lemma states that there are also non-linear shallow equational theories such that even $Rec_{\neq}$ automata are too weak to recognize the according closure set. In the proof we use again a pumping argument which is based on (equivalence) classes of terms that correspond to well-balanced trees. Intuitively, we show that the syntactic (dis-)equations of the transition rules are too weak to distinguish the equivalence classes induced by the "semantic" equality of the equational theory.

Lemma 6.2.4
There is a recognizable set L and a non-linear shallow equational theory $\mathcal{E}$ such that $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ is not in $Rec_{\neq}$.

Proof Suppose that the signature Σ consists of the binary function symbols f and g and a constant symbol a. Consider the non-linear shallow equational theory $\mathcal{E}$, which consists of the equation $f(x,x) \approx g(x,x)$, and the set L of ground terms $\{g(s_1, s_2) \mid s_1, s_2 \in \mathcal{T}(\Sigma)\}$. Let L' be the closure $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ of ground terms with respect to $\mathcal{E}$. Suppose that L' is recognized by a $Rec_{\neq}$ automaton $\mathcal{A}$ where $\mathcal{S}$ denotes the distinguished set of final states of $\mathcal{A}$ and n is the number of states of $\mathcal{A}$. We assume, without loss of generality, that $\mathcal{A}$ is deterministic and completely specified.

We define a sequence of ground terms which correspond to well-balanced trees. Let t_1 be the ground term $f(a,a)$ and let, for all $i \geq 1$, t_{i+1} be the ground term $f(t_i, t_i)$. For all $i \geq 1$, the cardinality of the equivalence class of t_i with respect to $\overset{*}{\Leftrightarrow}_{\mathcal{E}}$ is $2^{2^i - 1}$. Let i_0 be the integer $\lceil \log(\log(2n+2)) \rceil$. Then, for each $i \geq i_0$, there are two syntactically distinct ground terms u_i and v_i which are equivalent to t_i with respect to $\overset{*}{\Leftrightarrow}_{\mathcal{E}}$ such that u_i and v_i are recognized by $\mathcal{A}$ in the same state S_i. Note that $f(u_i, v_i) \in L'$ which implies, by assumption, that $f(u_i, v_i)$ is recognized by $\mathcal{A}$ in a final state $T_i \in \mathcal{S}$. Since $\mathcal{A}$ is deterministic, it follows that there is a sorted clause (transition rule) C_i of the form $S_i(x_1), S_i(x_2) \parallel \rightarrow T_i(f(x_1, x_2))$ in $\mathcal{A}$ such that $\mathcal{A} \models C_i\sigma$ where σ is a substitution that maps x_1 to u_i and x_2 to v_i. We arrive at a "hierarchy" of transition rules C_i starting from $i \geq i_0$. In fact, for all $i \geq i_0$, the variables x_1 and x_2 in the respective clause C_i are distinct since there are always pairs of terms similar to u_i and v_i, for all $i \geq i_0$, which are syntactically distinct. In other words, a syntactic equality constraint cannot occur in any of these transition rules. Note also that the following argumentation is independent from the potential presence of syntactic disequations in the transition rules.

Since $\mathcal{A}$ has only finitely many states and transition rules, there are two distinct integers j and k with $j > k \geq i_0$ such that the respective clauses C_j and C_k in $\mathcal{A}$ are of the form $S_j(x_1), S_j(x_2) \parallel \; \rightarrow T_j(f(x_1, x_2))$ and $S_k(x_1), S_k(x_2) \parallel \; \rightarrow T_k(f(x_1, x_2))$, respectively, with $S_j = S_k$. Since x_1 is distinct from x_2 in C_j we have that $\mathcal{A} \models C_j\sigma$ where σ is a substitution that maps x_1 to a term u_j and x_2 to a term u_k. Note that u_j is equivalent to the term t_j with respect to $\mathcal{E}$ whereas u_k is equivalent to the term t_k with respect to $\mathcal{E}$. It follows that the term $f(u_j, u_k)$ is recognized by $\mathcal{A}$ in the final state T_j which shows the contradiction because this term is not in L'. ■

The above lemma implies that the syntactic equality constraints of $Rec_=$ automata are not suitable to solve the E-unifiability problem with respect to non-linear shallow equational theories. The class of non-linear shallow sorted equational theories is a strict generalization of $Rec_=$ automata. We demonstrate below that the combination of this class and saturation-based methods is an appropriate concept to address the E-unifiability problem.

Fassbender & Maneth (1996) also investigate the decidability of E-unification with respect to theories induced by certain term rewriting systems called top-down tree transducers. Syntactic restrictions based on distinct function and constructor alphabets apply here. They show that E-unification in top-down tree transducers with only one function symbol in the alphabet is decidable. Due to the constructor-based restrictions the results are difficult to compare to our characterization by semi-linearity. Otto, Narendran & Dougherty (1995) show that E-unification is decidable in equational theories axiomatized by confluent string-rewriting systems. Limet & Réty (1997) show the decidability of E-unification with respect to theories represented by a particular class of confluent, constructor-based term rewriting systems. The set of possibly infinite solutions is represented by so-called *tree tuple synchronized grammars*. A term rewriting system is transformed into such a grammar which then simulates narrowing. The additional restrictions on the term rewriting system are syntactic. However, semi-linear theories are difficult to compare to the constructor-based systems in this approach. Comon et al. (1994) investigate the properties of non-linear shallow theories which are an instance of semi-linear equational theories. Shallow presentations can be transformed into equivalent so-called cycle-syntactic presentations for which decidability of unification has been shown. The first-order theory of the quotient algebra $T(F)/_{=_E}$ is also shown to be decidable where F is finite and E is (non-linear) shallow. However, the proof techniques are entirely different to our approach.

Nieuwenhuis (1996) generalizes the result of Comon et al. (1994) to so-called *standard theories* that extend non-linear shallow theories in such a way that non-ground terms containing linear variable occurrences are allowed in certain restricted positions in both sides of the equations. An equation $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$ in a standard theory may contain linear terms s_i, respectively t_i, where all other equations with top symbol f, respectively g, must have linear term occurrences only at the position i. Non-linear variable occurrences are limited to shallow positions. Nieuwenhuis (1996) employs saturation-based methods which are closely related to our work. In fact, the decidability results are obtained by termination analyses of saturation under the *basic superposition calculus* from (Nieuwenhuis & Rubio 1995). In the sequel, we refer to standard theories by *standard equational theories* which are defined as follows.

Definition 6.2.5 (Standard Equational Theory)
A *standard signature* is a signature where every function symbol f with arity n has an associated set of *shallow positions* $sh(f)$ and a set of *linear positions* $lin(f)$ such that $sh(f) \cap lin(f) \approx \emptyset$ and $sh(f) \cup lin(f) = \{1, \ldots, n\}$. The argument s_i of a term t of the form $f(s_1, \ldots, s_n)$ is called a *shallow position argument of* t if $i \in sh(f)$ and is called a *linear position argument of* t if $i \in lin(f)$. A term t is called *standard* whenever t is a variable or a term of the form $f(s_1, \ldots, s_n)$ where for all i with $1 \leq i \leq n$ if $i \in sh(f)$ then s_i is a variable or a ground term or if $i \in lin(f)$ then s_i is a linear term in t. An equation $s \approx t$ is called *standard* whenever (i) s is linear and t is ground, or else (ii) s is a standard term of the form $f(\ldots, g(t), \ldots)$ and t is a variable, or else (iii) s and t are standard terms that share only shallow variables and no variable x with $x \in vars(s) \cup vars(t)$ is a shallow and a linear position argument of s or t. An equational theory which consists of standard equations only is called *standard*.

The class of semi-linear sorted equational theories is significantly larger than the class of (non-linear) shallow equational theories but, however, does not strictly include the standard equational theories because of case (ii) in the above definition. We present a strict generalization of semi-linear and standard equational theories in Section 6.8 by the so-called *semi-standard* sorted equational theories.

Theorem 6.2.6 (Nieuwenhuis (1996))
The E-unifiability problem with respect to standard equational theories is decidable.

We consider the following example which has been inspired by Nieuwenhuis (1996). The example shows that the limits of the approach are due to a weak representation of "schematic" information that appears during saturation. We therefore propose to use "sort" information as an appropriate concept for the finite representation of a larger class of theories. The particular data structure of "sorts" allow "sorted" refinements of (basic) superposition/paramodulation that make particular use of this information.

Example 6.2.7
Consider the equational theory given by the equations $f(g(x), y) \approx h(y)$ and $f(x, x) \approx g(x)$. The class of standard equational theories does not include this theory. The closure of the non-standard theory under basic superposition leads to an infinite set of equations $g(h^n(g(x))) \approx h^{n+1}(g(x))$ if the admissible term ordering is induced by a precedence $g \succ h$. The infinite expansion can be avoided by abstracting the linear (semi-linear) term $g(x)$ into a sort declaration $\| \rightarrow S(g(x))$. We transform the original theory into a sorted shallow

equational theory consisting of the sorted Horn clauses $\| \to S(g(x))$, $S(x) \| \to f(x,y) \approx h(y)$, and $\| \to f(x,x) \approx g(x)$. We can effectively saturate the sorted shallow equational theory under sorted superposition to $H \cup \{S(x) \to g(x) \approx h(x); S(x) \to S(h(x))\}$ where we assume that the admissible term ordering is induced by the precedence $f \succ g \succ h \succ S$.

The finite representation of infinite schematic terms has received quite some attention, see (Hermann & Salzer 1996) for an overview. We conclude that our methods are related, on one hand, to the combinations of tree automata with equational theories inspired by Comon (1995), and, on the other hand, to the standard equational theories in (Nieuwenhuis 1996). Unlike these two approaches, we use a particular refinement of (basic) superposition/paramodulation combined with a certain finite representation of schematic information which is generated during the saturation process.

6.3 Decidability

In order to show that satisfiability of monadic equational types with respect to (non-linear) shallow sorted equational theories is decidable, we employ *sorted superposition* which is a particular instance of (basic) superposition with selection. Similar to sort resolution, we prefer a specialized version of superposition based on the combination of an admissible term ordering and a particular selection strategy for sort constraints and (dis-)equations. For refutational completeness of sorted superposition, the atom ordering must be compatible with the subterm property. An atom ordering which is induced by an admissible term ordering is compatible with the subterm property. We assume admissible literal orderings of Definition 3.1.48 and admissible clause orderings of Definition 3.1.15.

Given a sorted clause of the form $\Psi \| \Theta, \Gamma \to \Pi$ where Θ contains monadic atoms, Γ contains equations, and Π is either an equation or a monadic atom, the *equality selection* strategy selects any occurrences of equations in Γ first. Whenever Γ is empty, occurrences of negative monadic literals in Θ are selected with a particular priority on non-variable terms. In fact, the equality selection strategy is equivalent for literal occurrences in Θ and in the unsolved part of Ψ where the occurrences in Θ have a higher priority. Recall that the sort selection strategy select literals containing non-variable terms first, followed by so-called *empty sorts*, i.e. literals containing variables that do not occur in the non-constraint part of the clause. Literals of subsort declarations are selected last whereas, for sorted equations, we generalize the strategy to select a negative occurrence of a literal $S(x)$ whenever the equation is of the form $x \approx t$ or $t \approx x$ with $x \notin \mathit{vars}(t)$. The motivation of the equality selection strategy is to prohibit an increasing term depth during a saturation process by sorted superposition. The potential increase of the term depth by one may only be tolerated in non-trivial subsort declarations and sorted equations of the above form in favor of a technically simpler presentation.

Definition 6.3.1 (Equality Selection)
Let C be a sorted clause of the form $\Psi \| \Theta, \Gamma \to \Pi$ where Θ contains monadic atoms, Γ contains equations, and Π is either an equation or a monadic atom or empty. A literal $S(t) \in \Psi$ is selected whenever Θ and Γ are empty and (i) t is a non-variable term, or

else (ii) Ψ contains only variables and $t \notin vars(\Pi)$, or else (iii) Ψ contains only variables (in particular, t is a variable) with $vars(\Psi) = \{t\}$ and Π is either a disjoint collapsing equation $s \approx t$ with $t \notin vars(s)$, or else a monadic atom $T(t)$. A literal $S(t) \in \Theta$ is selected whenever Γ is empty and (iv) t is a non-variable term, or else (v) Θ contains only variables and $t \notin vars(\Pi)$, or else (vi) Θ contains only variables (in particular, t is a variable) with $vars(\Theta) = \{t\}$ and Π is either a disjoint collapsing equation $s \approx t$ with $t \notin vars(s)$, or else a monadic atom $T(t)$. (vii) Each literal $s \approx t \in \Gamma$ is selected. No other literal in C is selected. We call this selection strategy *equality selection*. A literal which is selected by equality selection is called *equality selected*.

Equality selection is a generalization of sort selection and type selection for clauses with equality. Note that we only need case (i) through (iii) for a termination proof of the saturation of (non-linear) shallow sorted equational theories by sorted superposition since the involved clauses have an empty non-constraint antecedent part. For a similar result on linear shallow typed equational theories, the additional cases (iv) through (vi) are required. Finally, the decidability of full monadic equational types with respect to these theories requires all cases of the equality selection strategy.

The *sorted superposition calculus* consists of the inference rules *Sort Constraint Resolution*, *Sorted Superposition Left*, *Sorted Superposition Right*, and *Equality Resolution* whereas the *sorted paramodulation calculus* consists of the inference rules *Sort Constraint Resolution*, *Sorted Paramodulation Left*, *Sorted Paramodulation Right*, and *Equality Resolution*. By Sort Constraint Resolution, we mean an adaption of the inference rule of Definition 4.3.2 to sorted clauses with non-empty antecedents as given below. Both calculi are adaptions of superposition with selection (Bachmair & Ganzinger 1994) and paramodulation with selection, respectively, to sorted clauses in combination with the equality selection strategy with additional *basic restrictions* on the sort constraint inspired by the basic versions of the calculi (Bachmair et al. 1995, Nieuwenhuis & Rubio 1995). Superposition is a generalization of paramodulation with additional ordering restrictions.

We briefly recall the idea of basic restrictions in the context of superposition and paramodulation calculi. Recall that a position p in an expression is called *variable position* if a variable occurs at p. A superposition inference may be prohibited at so-called *substitution positions*. A substitution position is a generalization of a variable position where the variable positions of a clause C are "remembered" in all successors of C, e.g., by some marking mechanism. That is, a substitution position in a clause C refers to a position at or below a variable position in a predecessor of C. Note that we avoid a formal definition of substitution positions and basic restrictions in favor of a simpler presentation of the decidability proofs. However, we shall emphasize substitution positions by frames around subterms which occur at substitution positions, e.g., the occurrence of a term $g(x)$ at a substitution position in a literal $S(f(g(x)))$ is depicted by $S(f(\boxed{g(x)}))$. For the basic variant of paramodulation, a further optimization is possible by the so-called *variable abstraction* (Bachmair et al. 1995). A substitution position may also include the redex position of a previous basic paramodulation inference.

Sorted superposition exploits the fact that all subterm positions in a sort constraint are substitution positions. Thus superposition/paramodulation inferences on literals in the sort constraint are not needed. The inference rules of sorted superposition and sorted paramodulation are defined accordingly. Note that we only consider sets of sorted clauses which have been obtained by a transformation of arbitrary sets of clauses using Equational Sort Abstraction. Any term in the sort constraint of a sorted clause obtained this way is a variable. Bachmair & Ganzinger (1998*a*) speak of *schematic* sets of clauses which are characterized as a class of clause sets which are, as input sets, compatible with basic inference systems. More precisely, any substitution position of a clause C in a schematic set is a variable position of C. Any clause set obtained by Equational Sort Abstraction is a schematic set.

The saturation of (non-linear) shallow sorted equational theories by the sorted paramodulation calculus is a terminating process. For an effective computation of the satisfiability of monadic equational types, there is an elegant method based on a further refinement of sorted paramodulation to the *basic sorted paramodulation calculus* and variable abstraction. The idea has been proposed by Nieuwenhuis (1996). The refinement imposes basic restrictions on all substitution positions in the sort constraint as well as the actual clause part. The basic sorted paramodulation calculus consists of the inference rules *Sort Constraint Resolution*, *Basic Sorted Paramodulation Left*, *Basic Sorted Paramodulation Right*, and *Equality Resolution* whereas the *basic sorted superposition calculus* consists of the inference rules *Sort Constraint Resolution*, *Basic Sorted Superposition Left*, *Basic Sorted Superposition Right*, and *Equality Resolution*. From a technical point of view, sorted paramodulation uses a constraint inheritance strategy by which equality constraints obtained by unification are eagerly propagated to the non-constraint part of a sorted clause whereas the strategy for basic sorted paramodulation keeps the full equality constraint. Substitution positions can be represented by syntactic equality constraints. In Chapter 7 we demonstrate that basic sorted paramodulation also serves as a decision procedure for typed equational theories and the respective monadic equational types. We assume throughout this chapter that Sort Condensing and Intersection Condensing are eagerly applied. Both simplification rules are compatible with the particular redundancy concept which is required for the basic/sorted versions of the calculi since Sort Condensing and Intersection Condensing have been restricted to the factorization of variant sort constraint literals.

Definition 6.3.2 (Sort Constraint Resolution)
The following inference is called *Sort Constraint Resolution*:

$$\text{Infer}\frac{\Psi \parallel\ \to S(s) \qquad S(u), \Phi \parallel \Lambda \to \Pi}{\Psi\sigma, \Phi\sigma \parallel \Lambda\sigma \to \Pi\sigma}$$

where (i) σ is the most general unifier of s and u, (ii) Ψ is solved, (iii) $S(u)$ is equality selected, and (iv) no literal is equality selected in $\Lambda\sigma$.

Definition 6.3.3 ((Basic) Superposition/Paramodulation Left)
The following inference is called *Superposition Left*:

$$\text{Infer}\frac{\Psi \,\|\, \to s \approx t \qquad \Phi \,\|\, u[s'] \approx v, \Lambda \to \Pi}{\Psi\sigma, \Phi\sigma \,\|\, u[t]\sigma \approx v\sigma, \Lambda\sigma \to \Pi\sigma}$$

where (i) σ is the most general unifier of s and s', (ii) $t \not\succ s$ and Ψ is solved, (iii) $v \not\succ u$, (iv) $u\sigma \approx v\sigma$ is equality selected, or else no literal is equality selected in $u\sigma \approx v\sigma, \Lambda\sigma$, and (v) s' does not occur at a variable position. If condition (v) is replaced by (v') s' does not occur at a substitution position, we speak of a *Basic Superposition Left*. If we drop the requirement (iii), i.e. $v \not\succ u$, the inference is called a (*Basic*) *Paramodulation Left* inference. The conclusion of a Basic Paramodulation Left inference is refined to

$$\Psi\sigma, \Phi\sigma \,\|\, u\sigma[\![\boxed{t\sigma}]\!] \approx v\sigma, \Lambda\sigma \to \Pi\sigma$$

The refinement is called *variable abstraction* (Bachmair et al. 1995, Nieuwenhuis 1998). Note that Φ does not have to be solved.

Definition 6.3.4 ((Basic) Superposition/Paramodulation Right)
The following inference is called *Superposition Right*:

$$\text{Infer}\frac{\Psi \,\|\, \to s \approx t \qquad \Phi \,\|\, \to u[s'] \approx v}{\Psi\sigma, \Phi\sigma \,\|\, \to u[t]\sigma \approx v\sigma}$$

where (i) σ is the most general unifier of s and s', (ii) $t \not\succ s$ and Ψ is solved, (iii) $v \not\succ u$ and Φ is solved, s' does not occur at a variable position. If condition (v) is replaced by (v') s' does not occur at a substitution position, we speak of a *Basic Superposition Right*. If we drop the requirement $v \not\succ u$ of case (iii) the inference is called a (*Basic*) *Paramodulation Right* inference.

Definition 6.3.5 (Equality Resolution)
The following inference is called *Equality Resolution*:

$$\text{Infer}\frac{\Psi \,\|\, u \approx v, \Lambda \to \Pi}{\Psi\sigma \,\|\, \Lambda\sigma \to \Pi\sigma}$$

where (i) σ is the most general unifier of u and v and (ii) $u\sigma \approx t\sigma$ is equality selected, or else no literal is equality selected in $u\sigma \approx v\sigma, \Lambda\sigma$. Note that Ψ does not have to be solved.

We do not explicitly define the computation and maintainance of substitution positions in the basic variants of the inference rules. For simplicity, we assume that the substitution positions of clauses, which are involved as premises in some inferences, are inherited to the conclusions. The following corollary states that (basic) sorted superposition/paramodulation is refutationally complete for schematic sets of sorted monadic Horn clauses with equality.

Corollary 6.3.6 (Weidenbach (1996*a*))
Let N be a schematic set of sorted monadic Horn clauses (with equality). Suppose that N is saturated by (basic) sorted superposition/paramodulation. Then either N contains the empty clause, or else N is satisfiable.

The following lemma states that (non-linear) shallow sorted equational theories can be finitely saturated by sorted superposition. We show that there is a certain class of shallow monadic Horn clauses which includes shallow sorted equational theories and which is closed under sorted superposition. Assuming a finite signature, Sort and Intersection Condensing identify sufficiently many clauses to demonstrate effectively that this class is finite up to Sort and Intersection Condensing and variant clauses. The set of productive clauses in a set of clauses which has been finitely saturated by sorted superposition contains shared and trivial sorted equations and includes a (non-linear) shallow sort theory which consists of term declarations and trivial subsort declarations. The definition of productive clauses is straightforward by an adaption of the model functor for superposition to sorted superposition.

Lemma 6.3.7
Let $\mathcal{E}$ be a (non-linear) shallow sorted equational theory. Then $\mathcal{E}$ can be finitely saturated by sorted superposition, Sort Condensing, and Intersection Condensing. The productive clauses in the saturated set form a (non-linear) shallow sorted equational theory which consists of (non-linear) shallow term declarations, trivial subsort declarations, and (non-linear) shallow sorted equations with a solved sort constraint.

Proof The only clauses in a shallow sorted equational theory which have an unsolved sort constraint are non-trivial subsort declarations and disjoint collapsing/universal shallow sorted equations. The idea of the proof is to show that (i) there is a certain subclass of shallow monadic Horn theories with equality which includes $\mathcal{E}$ and which is closed under sorted superposition and (ii) this class is finite with respect to Sort and Intersection Condensing and variant clauses. For (i) we shall argue that the following class of clauses is closed under sorted superposition.

$$T_1(t), \dots, T_n(t), S_1(x_1), \dots, S_m(x_m) \parallel \rightarrow L$$

where n and m are possibly zero, L is either a monadic atom $S(u)$ or an equation $u \approx v$, t is a non-variable shallow term, and u and v are shallow terms. Note that the constraint $S_1(x_1), \dots, S_m(x_m)$ does not have to be solved. We call a clause of the above form the *invariant.*

Obviously, the class includes any (non-linear) shallow sorted equational theory. Any inference that involves an invariant as the side premise is only possible if $n = 0$ and the sort constraint of the invariant is solved, i.e. if the invariant is a (non-linear) shallow term declaration, a trivial subsort declaration, or a (non-linear) shallow sorted equation with a solved sort constraint. Thus an invariant that is involved as the main premise may only be reduced to another invariant by a (non-linear) shallow term declaration, a trivial subsort declaration, or a (non-linear) shallow sorted equation with a solved sort constraint. See the case analysis below. Since any invariant contains at most three shallow terms, there are only finitely many invariants over a finite signature up to Sort and Intersection Condensing and variant clauses, which implies (ii).

Let C be a clause of the form of the invariant and suppose $n = 0$. We consider all inferences in which C is involved as the main premise. Suppose that the sort constraint

$S_1(x_1), \ldots, S_m(x_m)$ is solved in C. In this case, C may only be involved in a Superposition Right inference. The involved side premise must also have a solved sort constraint. Both premises must be sorted equations. Suppose that the equation $s \approx s'$ in the side premise is disjoint collapsing, i.e. s' is a variable x and $x \notin \mathit{vars}(s)$. Note that the side premise does not contain a literal $S(x)$ in the sort constraint, for otherwise this literal would be selected. Since we do not superpose into variables, a Superposition Right inference results in a clause of the form of the invariant. In all other cases, the most general unifier is simply a renaming. It follows that any conclusion D is of the form of the invariant where the sort constraint of D may be unsolved but does not contain any non-variable term.

Now suppose that the sort constraint $S_1(x_1), \ldots, S_m(x_m)$ is unsolved in C. First, we consider the case in which C is either a non-trivial subsort declaration or a sorted equation where L is a disjoint collapsing equation of the form $x_i \approx t$ and $x_i \notin \mathit{vars}(t)$. Suppose that C is of one of these forms, i.e. there is at least one occurrence of a variable x_i such that the according literal $S_i(x_i)$ is selected. By Sort Constraint Resolution we may derive a clause D of the form of the invariant where n' is the number of occurrences of the non-variable term t in D. If the other involved premise is a term declaration then $n' = m - 1$ where m is the number of occurrences of x_i in the sort constraint of C. If it is a trivial subsort declaration then $n' = 0$.

For the second case, suppose that there is at least one occurrence of a variable x_i in the sort constraint of C with $x_i \notin \mathit{vars}(L)$. The according literal $S_i(x_i)$ is selected. By Sort Constraint Resolution we may derive a clause D of the form of the invariant where n' is the number of occurrences of the non-variable term t in D. If the other involved premise is a term declaration then $n' = m - 1$ where m is the number of occurrences of x_i in the sort constraint of C. If it is a trivial subsort declaration then $n' = 0$. However, the sort constraint in D may still be unsolved.

Suppose that $n > 0$ in C. The equality selection strategy prefers a literal $T_i(t)$ with a non-variable term t, i.e. potential occurrences of a literal $S_j(x_j)$ with $x_j \notin \mathit{vars}(L)$ are not selected in the presence of t. All literals $T_i(t)$ are selected. By Sort Constraint Resolution on some $T_i(t)$ we may derive a clause D of the form of the invariant where n' is the number of occurrences of t in D. In any case, we have that $n' = n - 1$ which implies (i).

The saturated set may still contain clauses with an unsolved sort constraint. However, these clauses cannot be productive since the unsolved part is equality selected which implies that the set of productive clauses are (non-linear) shallow term declarations, trivial subsort declarations, and (non-linear) shallow sorted equations with a solved sort constraint which form a (non-linear) shallow sorted equational theory. ■

Example 6.3.8
Consider the saturation process of the shallow sorted equational theory from the Example 6.2.7 by sorted superposition. We assume that the admissible term ordering $\succ$ is induced by the precedence $f \succ g \succ h \succ S$. In particular, we have that $f(x,y) \succ g(x) \succ h(x)$ and $f(x,y) \succ h(y)$ in any non-ground approximation of $\succ$.

$$\begin{array}{ll}
(1) & S(x) \,\|\, \rightarrow f(x,y) \approx h(y) \\
(2) & \,\|\, \rightarrow f(x,x) \approx g(x) \\
(3) & \,\|\, \rightarrow S(g(x))
\end{array}$$

The saturation process by sorted superposition generates the clauses (4) and (5) by Superposition Right inferences.

$$\begin{array}{ll}
(4) & S(x) \,\|\, \rightarrow g(x) \approx h(x) \\
(5) & S(x) \,\|\, \rightarrow S(h(x))
\end{array}$$

The clauses (1)–(5) are saturated by sorted superposition.

Corollary 6.3.9
Shallow sorted equational theories can be finitely saturated by sorted paramodulation.

Proof The only difference between sorted superposition and sorted paramodulation is that the sorted paramodulation calculus paramodulates into both sides of equations, as opposed to the ordering restrictions of superposition. However, we only paramodulate with right hand sides of equations that are not smaller than the respective left hand side. Therefore, the proof of Lemma 6.3.7 carries over to sorted paramodulation. ■

Example 6.3.10
With respect to the Example 6.3.8 sorted paramodulation produces the additional clause (6) by a Paramodulation Right inference from the clauses (2) and (4).

$$(6) \quad S(x) \,\|\, \rightarrow f(x,x) \approx h(x)$$

Note that the clause (1) subsumes the clause (6). However, the clauses (1)–(6) are saturated by sorted paramodulation.

Nieuwenhuis (1996) shows that every finite set of shallow equations can be closed under paramodulation in polynomial time. Starting from a non-alternating linear shallow sorted equational theory, sorted superposition/paramodulation essentially computes superposition/paramodulation steps among the non-collapsing equations and the term declarations which implies the following corollary.

Corollary 6.3.11
Let $\mathcal{E}$ be an non-alternating linear shallow sorted equational theory. Then $\mathcal{E}$ can be finitely saturated by sorted superposition/paramodulation, Sort Condensing, and Intersection Condensing in polynomial time with respect to $size(\mathcal{E})$ resulting in a sorted equational theory $\mathcal{E}'$ such that $size(\mathcal{E}')$ is polynomially larger than $size(\mathcal{E})$.

Collapsing equations could be used to derive non-trivial subsort declarations whereas non-linear equations may introduce alternation in the term declarations and thus possible intersection non-emptiness tests. It follows that the linearity requirement ensures that only non-emptiness tests are computed during saturation which are solvable in linear time. The above corollary indicates that equality adds only a polynomial degree in complexity. This observation and Proposition 4.3.6 imply that saturation by sorted superposition/paramodulation of an (alternating, non-linear) shallow sorted equational theories can be done in simply exponential time.

Proposition 6.3.12
Let $\mathcal{E}$ be an (alternating, non-linear) shallow sorted equational theory. Then $\mathcal{E}$ can be finitely saturated by sorted superposition/paramodulation, Sort Condensing, and Intersection Condensing in simply exponential time with respect to $size(\mathcal{E})$ resulting in a sorted equational theory $\mathcal{E}'$ such that $size(\mathcal{E}')$ is at most simply exponentially larger than $size(\mathcal{E})$.

Recall the subset construction in Section 4.3 which involves the inference rule State Union and the simplification rule De-Alternation. The Proposition 4.3.9 states that any alternating (non-linear) shallow sort theory $\mathcal{S}$ without non-trivial subsort declarations can be transformed into a non-alternating theory $\mathcal{S}'$ such that the minimal model of $\mathcal{S}$ is essentially equivalent to the minimal model of $\mathcal{S}'$. In particular, De-Alternation encodes the alternation over one variable into a single state. We may use the same technique to obtain the following corollary.

Corollary 6.3.13
Let $\mathcal{E}$ be an (alternating, non-linear) shallow sorted equational theory without non-trivial subsort declarations and let $\mathcal{P}$ be the set of predicate symbols which occur in $\mathcal{E}$. Then $\mathcal{E}$ can be transformed into a (non-linear) shallow sorted equational theory $\mathcal{E}'$ in which all term declarations are non-alternating and all sorted equations have non-alternating sort constraints such that (i) $T^{\mathcal{E}}(P_1) \cap \ldots \cap T^{\mathcal{E}}(P_n) = T^{\mathcal{E}'}(S_{\{P_1,\ldots,P_n\}})$ for any set of predicate symbols $P_1, \ldots, P_n \in \mathcal{P}$ and (ii) $size(\mathcal{E}')$ is at most simply exponentially larger than $size(\mathcal{E})$ with respect to the cardinality of $\mathcal{P}$.

Since Sort and Intersection Condensing are compatible with basic restrictions, the above results on finite saturation hold also for the basic variants of the sorted superposition/paramodulation calculus.

Corollary 6.3.14
Shallow sorted equational theories can be finitely saturated by basic sorted superposition and basic sorted paramodulation.

In particular, this corollary allows for a simple decidability proof of E-unification with respect to shallow sorted equational theories. There are only finitely many applications of Basic Paramodulation Left (and Equality Resolution) on some given (dis-)equation with respect to a clause set that is finitely saturated by basic sorted paramodulation. The subsequent applications of Sort Constraint Resolution have already been shown to be terminating.

Theorem 6.3.15
Unifiability with respect to finitely saturated shallow sorted equational theories is decidable.

Proof Two arbitrary terms u and v are unifiable if and only if we can derive the empty clause from the saturated theory and the goal clause $\parallel u \approx v \rightarrow$. Since the shallow sorted equational theory is saturated, no inferences inside the theory need to be considered. Furthermore, the goal is purely negative, so we can delete all clauses with an unsolved sort constraint from the saturated theory as well as all non-trivial subsort declarations. Let us in particular assume that our theory is saturated by (basic) sorted paramodulation. Following the equality selection strategy the disequation $u \approx v$ in the goal is selected. Thus only (Basic) Paramodulation Left or Equality Resolution inferences can be performed until the disequation is resolved. For a theory saturated under (basic) sorted paramodulation, every position in the disequation has only to be considered once for a Basic Paramodulation Left inference. Hence, the application of Basic Paramodulation Left and Equality Resolution terminates on the goal clause, resulting in a clause $S_1(t_1), \ldots, S_n(t_n) \parallel \rightarrow$. Now exhaustive application of Sort Constraint Resolution terminates on this clause, because the lexicographic combination of the overall number of variables in the clause and the multiset of all term depths decreases with any Sort Constraint Resolution application. ■

Recall Lemmata 6.2.3 and 6.2.4 which state that tree automata as well as $Rec_=$ are in some sense too weak to recognize the closure $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ of ground terms with respect to a recognizable set L and a non-linear shallow equational theory $\mathcal{E}$. In contrast, by Theorem 6.2.1, tree automata can recognize the closure $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ with respect to a recognizable set L and linear equational theories in which all variables are shallow (Comon 1995). We have shown that this result can be used to decide E-unifiability of linear "goal" terms s and t with $\mathit{vars}(s) \cap \mathit{vars}(t) = \emptyset$ with respect to $\mathcal{E}$. We conclude that, in contrast to the automata-theoretic approach, the particular data structure of (non-linear) shallow sorted equational theories in combination with sorted superposition are a suitable concept to relax (i) the linearity requirement on the equations of $\mathcal{E}$ and the goal and (ii) the variable disjointness of the goal terms. Our approach addresses (i) by the explicit representation of the equational information in the saturated set such that equality is not only involved in the (automata) completion but also in the recognization process of a goal and (ii) by an explicit control of the instantiation of the goal terms through unification, similar to so-called narrowing techniques.

Example 6.3.16
Consider the set N of clauses (1)–(6) of the Example 6.3.10 which has been saturated by (basic) sorted paramodulation. We compute the E-unifiability of two pairs of terms with respect to N. Given the pair of terms $f(x,y)$ and $h(y)$, we add the following (sorted) disequation to the theory.

$$(7) \qquad \parallel f(x,y) \approx h(y) \rightarrow$$

A Basic Paramodulation Left inference from the clause (1) and (7) results in the clause (8) of the form $S(x) \parallel \boxed{h(y)} \approx h(y) \rightarrow$. A subsequent Sort Constraint Resolution step from (3) and (8) yields the clause (9) of the form $\parallel \boxed{h(y)} \approx h(y) \rightarrow$. A final application of Equality

Resolution from (9) yields the empty clause. Therefore, $f(x,y)$ and $h(y)$ are unifiable with respect to N. Another example demonstrates that the two terms $f(a,x)$ and $h(x)$ where a is some constant are not unifiable with respect to N.

$$\begin{array}{lrl}
(7) & \| & f(a,x) \approx h(x) \rightarrow \\
(8) & S(a)\,\| & \qquad\qquad\quad \rightarrow \\
(9) & \| & g(a) \approx h(a) \quad \rightarrow
\end{array}$$

The saturation under sorted paramodulation does not produce the empty clause. However, the clauses (8) and (9) are produced before the procedure eventually stops. Thus the theory and the negated monadic equational type have a model which shows that $f(a,x)$ and $h(x)$ are not unifiable with respect to N.

The next theorem summarizes the discussion and states that the theory of monadic equational types over (non-linear) shallow sorted equational theories is decidable. Any shallow sorted equational theory may be finitely saturated by basic sorted paramodulation such that the candidate model of the saturation is the minimal model of the theory. By a subsequent saturation with basic sorted paramodulation between the saturated theory and a negated monadic equational type we can effectively compute the satisfiability of the type with respect to the theory. A negated monadic equational type is a negative clause and thus any clause with an unsolved sort constraint may safely be removed from the saturated theory.

Theorem 6.3.17
Let $\mathcal{E}$ be a shallow sorted equational theory. The theory $\mathcal{F}_\mathcal{E}$ of monadic equational types over $\mathcal{E}$ is decidable.

Proof Let $\exists\, x_1, \ldots, x_n\, (\Psi, \Theta, \Gamma)$ be a monadic equational type in $\mathcal{F}_\mathcal{E}$ where $\{x_1, \ldots, x_n\}$ is the set of free variables in Ψ, Θ, Γ. Moreover, Ψ is maximal such that Ψ contains only monadic atoms of the form $S(x)$ and $x \in \mathit{vars}(\Theta, \Gamma)$, Θ contains other monadic atoms, and Γ contains equations. Note that $\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi, \Theta, \Gamma)$ holds if and only if $T^\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi, \Theta, \Gamma)$ holds if and only if $T^\mathcal{E} \not\models \forall\, x_1, \ldots, x_n\, (\neg(\Psi, \Theta, \Gamma))$ holds. In order to check whether $\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi, \Theta, \Gamma)$ holds we add the sorted clause $\Psi \,\|\, \Theta, \Gamma \rightarrow$ to the saturated set $\mathcal{E}'$ and saturate the result by basic sorted paramodulation. By Corollary 6.3.14 $\mathcal{E}$ can be finitely saturated by basic sorted paramodulation into $\mathcal{E}'$. In the following we call a negative clause a *goal.* Only steps between clauses in $\mathcal{E}'$ and goals are required. Moreover, since goals are purely negative we can delete all clauses with unsolved sort constraint from $\mathcal{E}'$ as well as all non-trivial subsort declarations.

The equality selection strategy also applies to goals. If a goal contains equations then all equations are selected whereas the monadic atoms are not selected. In case a goal does not contain any equations all monadic atoms from the non-constraint part are selected whereas the literals in the sort constraint are not selected. If a goal has an empty non-constraint part all literals in the sort constraint are selected.

Due to Theorem 6.3.15 basic sorted paramodulation on the equational part of goals terminates. The next step is to solve the non-constraint monadic part of goals. Similar to the proof of Theorem 6.3.15 we argue that there are only finitely many applications of Basic Paramodulation Left on the non-constraint monadic part using variable abstraction.

Finally, goals with empty non-constraint part are actually monadic types. Since only Sort Constraint Resolution is applicable to monadic types we may discard all sorted equations from the saturated theory $\mathcal{E}'$. Consequently, $\mathcal{E}'$ becomes a shallow sort theory and we may apply Theorem 4.3.10 to show termination. ■

According to the proof of Theorem 4.3.12, by Proposition 6.3.12 and by Corollary 6.3.13, we obtain the following complexity result. Note that satisfiability tests on monadic variable types do not involve any superposition/paramodulation inferences but only computations with respect to the linear shallow sort theory within a linear shallow sorted equational theory.

Corollary 6.3.18
The satisfiability problem of the theory of monadic variable types over (alternating) linear shallow sorted equational theories is EXPTIME-complete.

The above corollary implies the EXPTIME-hardness of the satisfiability problem not only of the theory of monadic variable types but also of the theory of (general) monadic equational types over (alternating) non-linear shallow sorted equational theories.

Corollary 6.3.19
The satisfiability problem of the theory of monadic equational (variable) types over (alternating, non-linear) shallow sorted equational theories is EXPTIME-hard.

In the proof of Proposition 7.1.5, we will show the EXPTIME-completeness of the satisfiability problem of the theory of (general) monadic equational types over linear shallow typed equational theories. We obtain as an immediate corollary that the satisfiability problem of the theory of monadic equational types over (alternating) linear shallow sorted equational theories is EXPTIME-complete. The EXPTIME-completeness of the same problem for non-linear theories remains open.

Corollary 6.3.20
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) linear shallow sorted equational theories is EXPTIME-complete.

Sorted superposition/paramodulation implements the test of E-unifiability with respect to (sorted) monadic Horn theories. The E-unifiability test of two terms induces repeated tests of *well-sortedness*, as well as tests of *emptiness of sorts* and *sorted unification* (Weidenbach 1996*a*) on the sort constraint literals. More precisely, the saturation by sorted superposition/paramodulation of a shallow sorted equational theory with a negated ground monadic equational type corresponds to an attempt to solve the word problem given by the equations in the type where the sort constraint of the type represents a test of well-sortedness. The same saturation with a negated non-ground monadic equational type corresponds to an E-unifiability test of the equations in the type where the sort constraint of the type encodes emptiness tests of sorts and sorted unification problems.

We shall exploit the complexity results for sorted unification with respect to shallow non-equational sort theories to derive similar results on the number of most general unifiers of an equational problem with respect to shallow sorted equational theories. The following theorem is an easy extension of Theorem 4.3.15 from shallow sort theories to shallow sorted equational theories.

Theorem 6.3.21 (Weidenbach (1998))
Let $\mathcal{E}$ be a (non-linear) shallow sorted equational theory. Let C be a clause of the form $S_1(t_1),\ldots,S_n(t_n) \parallel \rightarrow$. We can derive a clause $T_1(y_1),\ldots,T_k(y_k) \parallel \rightarrow$ from C and $\mathcal{E}$ by Sort Constraint Resolution if and only if the sorted unification problem $x_1 = t_1,\ldots,x_n = t_n$ has a "well-sorted" most general unifier with respect to the shallow sort theory contained in $\mathcal{E}$ where each x_i is new and has the sort S_i.

Nieuwenhuis (1996) has already demonstrated that there are simply exponentially many minimal (irreducible) unifiers of an equational problem with respect to a finite set of Horn clauses that is saturated under basic paramodulation. The idea is to show that saturation implicitly generates a polynomially branching narrowing tree whose overall size is then simply exponential. Using the same argument, it follows that saturation under basic sorted paramodulation of a sorted disequation $\parallel s \approx t \rightarrow$ and a saturated shallow sorted equational theory $\mathcal{E}$ produces at most simply exponentially many sorted clauses of the form $S_1(t_1),\ldots,S_n(t_n) \parallel \rightarrow$ in the size of $s \approx t$ and $\mathcal{E}$.

Recall that Theorem 4.3.16 states that sorted unification in (non-linear) shallow sort theories is NP-complete and that the number of "well-sorted" most general unifiers is simply exponential in the size of a shallow sort theory and the sorted unification problem. Thus from Theorem 6.3.21, 4.3.16, and 6.3.15 we obtain that there are at most simply exponentially many "well-sorted" most general unifiers for a unification problem with respect to a finitely saturated shallow sorted equational theory.

Corollary 6.3.22
The number of "well-sorted" most general unifiers with respect to a finitely saturated shallow sorted equational theory $\mathcal{E}$ is simply exponential.

6.4 Semi-linear Approximations

The theory of monadic equational types over shallow sorted equational theories is decidable. In analogy to the transformation of semi-linear sort theories into essentially equivalent shallow sort theories, we can improve the result for shallow sorted equational theories using an extension of Semi-linear Flattening to sorted Horn clauses with equality called *Semi-linear Equational Flattening.* The transformation introduces a new monadic predicate for each occurrence of a proper non-variable subterm in a term declaration or a sorted equation where the number of new symbols is linear in the number of function symbols which occur in the equational theory. Semi-linear Equational Flattening replaces in a top-down manner each occurrence of a non-variable proper subterm t_i by a variable x where an additional sort constraint on x restricts the ground instances of x to the "well-sorted" ground instances of t. Given a semi-linear sorted equational theory, the minimal model of the resulting shallow sorted equational theory is essentially equivalent to the minimal model of the original equational theory while for arbitrary sorted equational theories the minimal model of the result is an upper approximation of the minimal model of the original. Note that we do not consider the adaption of Flattening, which is a proper instance of Semi-linear Flattening, to *Equational Flattening.*

Definition 6.4.1
The following abstraction is called *Semi-linear Equational Flattening*:

$$\text{Abstract}\frac{\Psi, \Psi' \parallel \rightarrow L[t]_{p_1}}{\begin{array}{c}\Psi' \parallel \rightarrow S_t(t)\\ S_t(x), \Psi, \Psi'' \parallel \rightarrow L[p_1, \dots , p_n/x]\end{array}}$$

where (i) L is either a monadic atom or an equation, (ii) Ψ, Ψ' is solved, (iii) t is a non-variable subterm at position p_1 with $|p_1| = 2$, (iv) the positions $p_1, \dots , p_n$ refer to all positions q of t in L with $|q| = 2$, (v) Ψ' is maximal such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t)$, (vi) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (vii) S_t is a new monadic predicate, and (viii) x is a new variable.

Note that Semi-linear Equational Flattening differs from Semi-linear Flattening only in case (i) which includes, in addition to monadic atoms, also equations. In a similar way, Flattening carries over to Equational Flattening. In the sequel, a monadic predicate S_t refers to the new predicate which has been introduced by Semi-linear Equational Flattening for a non-variable term t. The newly introduced declarations represent, from an automata-theoretic point of view, tree automata which recognize in a state S_t exactly the "well-sorted" ground instances of the original term t. The following proposition suggests that Semi-linear Equational Flattening may be applied to arbitrary sorted equational theories. However, minimal models are preserved only for semi-linear sorted equational theories. For arbitrary sorted equational theories the transformation yields a sorted equational theory whose minimal model is an upper approximation of the minimal model of the original theory.

Proposition 6.4.2
Let $\mathcal{E}$ be a sorted equational theory. Exhaustive application of Semi-linear Equational Flattening to $\mathcal{E}$ terminates and results in a (non-linear) shallow sorted equational theory.

Proof The proof is similar to the proof of the Proposition 4.4.3. ■

In the sequel we call a sorted equational theory which has been obtained by an exhaustive application of (Semi-linear) Equational Flattening to a sorted equational theory $\mathcal{E}$ the (*semi-linear*) *flat approximation of* $\mathcal{E}$. The following lemma on the minimal model properties of the new predicates introduced by Semi-linear Equational Flattening corresponds to the technical Lemma 4.4.7 for Semi-linear Flattening.

Lemma 6.4.3
Let $\mathcal{E}$ be a sorted equational theory and let $\mathcal{E}_{SF}$ be the (semi-linear) flat approximation of $\mathcal{E}$. Let I be the minimal model of $\mathcal{E}_{SF}$. Let t be a non-variable term for which a new monadic predicate symbol S_t has been introduced by (Semi-linear) Equational Flattening. Let C be the immediate conclusion $\Psi \parallel \; \rightarrow S_t(t)$ of (Semi-linear) Equational Flattening on t. Then for all ground atoms $S_t(s)$ which are true in I there is a ground substitution σ such that (i) $I \models s \approx t\sigma$ and (ii) $\Psi\sigma$ is true in I.

Proof The proof is similar to the proof of Lemma 4.4.4 and Lemma 4.4.7. ∎

The Semi-linear Equational Flattening abstracts sorted equational theories such that the minimal models of the approximations are upper approximations of minimal models of the original equational theories.

Proposition 6.4.4
Let $\mathcal{E}$ be a sorted equational theory and let $\mathcal{E}_F$ be the (semi-linear) flat approximation of $\mathcal{E}$. The minimal model $T^{\mathcal{E}_F}$ is an upper approximation of the minimal model $T^{\mathcal{E}}$.

Proof The proof is similar to the proof of Proposition 4.4.8. ∎

Semi-linear Equational Flattening yields exact approximations for semi-linear sorted equational theories.

Proposition 6.4.5
Let $\mathcal{E}$ be a linear (semi-linear) sorted equational theory and let $\mathcal{E}_F$ be the (semi-linear) flat approximation of $\mathcal{E}$. The minimal model of $\mathcal{E}_F$ is equivalent to the minimal model of $\mathcal{E}$ up to the new atoms introduced by (Semi-linear) Equational Flattening.

Proof The proof is similar to the proof of Proposition 4.4.9. ∎

It turns out that the theory of monadic equational types over semi-linear sorted equational theories is decidable. Semi-linear Equational Flattening is an effective abstraction from semi-linear sorted equational theories to non-linear shallow sorted equational theories for which satisfiability of monadic equational types has been shown to be decidable.

Theorem 6.4.6
Let $\mathcal{E}$ be a semi-linear sorted equational theory. Then the theory $\mathcal{F}_{\mathcal{E}}$ of monadic equational types over $\mathcal{E}$ is decidable.

Proof Let $\mathcal{E}_F$ be the semi-linear flat approximation of $\mathcal{E}$. By Proposition 6.4.2 $\mathcal{E}_F$ can be effectively computed where $\mathcal{E}_F$ is a shallow sorted equational theory. Due to Theorem 6.3.17 the theory of monadic equational types over shallow sorted equational theories is decidable. Let $\exists\, x_1, \ldots, x_n\, (\Psi)$ be a monadic equational type in $\mathcal{F}_{\mathcal{E}}$ where $\{x_1, \ldots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ holds if and only if $T^{\mathcal{E}} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ holds and, by Proposition 6.4.5, if and only if $T^{\mathcal{E}_F} \models \exists\, x_1, \ldots, x_n\, (\Psi)$

holds under the proviso that Ψ does not contain any new atoms introduced by Semi-linear Equational Flattening. It follows that the theory of monadic equational types over semi-linear sorted equational theories is decidable. ■

The following corollary is an immediate consequence of Corollary 6.3.19.

Corollary 6.4.7
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) semi-linear sorted equational theories is EXPTIME-hard.

The following corollary is an immediate consequence of Corollary 6.3.20 and Proposition 6.4.2.

Corollary 6.4.8
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) linear sorted equational theories is EXPTIME-complete.

We restrict our attention to saturation-based decision procedures and conclude that Semi-linear Equational Flattening is a suitable concept for effective soft typing with respect to arbitrary static/dynamic sorted equational theories and monadic equational types.

Corollary 6.4.9
Let N_0 be a set of clauses. The (semi-linear) flat approximation of the static sorted equational theory of N_0 is a decidable static approximation for monadic equational types. Let N_0, N_1, N_2, $\ldots$ be a fair theorem proving derivation. The sequence of (semi-linear) flat approximations of the dynamic sort theories of each N_i is a decidable dynamic approximation for monadic equational types.

6.5 Undecidability

The theory of monadic equational types over semi-linear sorted equational theories has been shown to be decidable, which includes the decidability of the E-unifiability problem with respect to this class of theories. In this section, we discuss the limits of effective E-unification and demonstrate that already the word problem with respect to the so-called *pseudo-linear equational theories* is undecidable. A term t is pseudo-linear whenever each non-linear variable x in t occurs at the same depth in t, c.f. Chapter 2 for a formal definition. Intuitively, pseudo-linearity generalizes semi-linearity such that each pair of distinct paths to the non-linear occurrences of a variable have to be of the same length and not, as for semi-linearity, have also to be labeled by the same symbols. The extension of pseudo-linearity to equations and equational theories is straightforward. Note that every semi-linear theory is pseudo-linear but not vice versa. The equations $f(h(x), g(x)) \approx g(g(x))$ and $f(h(x), g(x), y) \approx y$ are both pseudo-linear and form a pseudo-linear equational theory which is not semi-linear. The equation $f(h(x), g(x)) \approx g(x)$ is neither semi-linear nor pseudo-linear.

Oyamaguchi (1990) has demonstrated that the word problem for right-ground term rewriting systems is undecidable whereas the word problem in left-linear and right-ground term rewriting systems is decidable in polynomial time. In the undecidability proof for right-ground systems, rewrite rules which simulate a transition of one configuration of a Turing machine to the next contain non-linear variable occurrences at different depth. Recall that non-linear variable occurrences at different depth are excluded from semi-linear and pseudo-linear theories. We follow a similar idea in the following proof by an encoding of the transitions of a Turing machine in a pseudo-linear term rewriting system R over a signature which contains unary function symbols and constant symbols only (string rewriting system or Semi-Thue system). The rewriting rules are all of the form $f(g(x)) \Rightarrow f'(g'(x))$. The reduction of the halting problem (blank tape accepting problem) is completed by adding certain left-linear (right-ground) rewrite rules to R. We obtain an orthogonal (left-linear, no critical pairs) term rewriting system R' for which the according equational theory (symmetric closure) $\mathcal{E}$ simulates exactly the computations of the original Turing machine. Note that $\mathcal{E}$ is close to the simplest case of pseudo-linear non-shallow equational theories. In this way, Theorem 6.5.5 characterizes a gap between semi-linear and pseudo-linear equational theories with respect to our combination of saturation and abstraction of Sections 6.3 and 6.4. In contrast to pseudo-linear equational theories, Weidenbach (1996*b*) shows that sorted unification with respect to pseudo-linear sort theories (the theory of monadic types) as a generalization of semi-linear sort theories is still decidable.

Definition 6.5.1 (Turing Machine)
A *Turing machine* $\mathcal{M}$ is a 7-tuple $(Q, \Sigma, \Sigma_{in}, \delta, q_0, b, F)$ where (i) Q is a finite set of *states*, (ii) Σ is a finite set of *tape symbols* disjoint from Q, (iii) Σ_{in} is a subset of Σ called the set of *input symbols*, (iv) δ is a mapping from $Q \times \Sigma$ to subsets of $Q \times \Sigma \times \{\texttt{left}, \texttt{right}\}$, called the *transition function of* $\mathcal{M}$, (v) q_0 is the *initial state of* $\mathcal{M}$, (vi) $b \in \Sigma$ is a tape symbol called *blank*, and (vii) $F \subseteq Q$ is the set of *final states*. $\mathcal{M}$ is called *deterministic* if δ is functional, i.e. the range of δ is a singleton set. An *instantaneous description* or ID of $\mathcal{M}$ is any string $\alpha q \beta$ where $q \in Q$ and $\alpha, \beta \in \Sigma^*$. A *move of* $\mathcal{M}$ is a pair (v, w) of ID's where either $v = \alpha a q \beta$ and $w = \alpha q' a' \beta$ if $(q', a', \texttt{left}) \in \delta(\texttt{q}, \texttt{a})$ or $v = \alpha a q c \beta$ and $w = \alpha a' c q' \beta$ if $(q', a', \texttt{right}) \in \delta(\texttt{q}, \texttt{a})$ for all $c, a, a' \in \Sigma$, $q, q' \in Q$, and $\alpha, \beta \in \Sigma^*$. The binary relation of all moves of $\mathcal{M}$ is denoted by $\vdash_{\mathcal{M}}$, and its transitive and reflexive closure by $\vdash^*_{\mathcal{M}}$.

The intended meaning of an ID of the form $\alpha a q \beta$ of $\mathcal{M}$ is to give a complete description of the execution state of $\mathcal{M}$ called the *configuration of* $\mathcal{M}$: q is the state of $\mathcal{M}$, α corresponds to the contents of the tape of $\mathcal{M}$ from the left edge of the tape to the symbol a pointed to by the head, and β denotes the rest of the contents of the tape to the right of the head.

Definition 6.5.2 (Turing Machine Language)
Let $\mathcal{M}$ be a Turing machine. The *language accepted by* $\mathcal{M}$ is the set $L(\mathcal{M})$ of strings given as:

$$L(\mathcal{M}) = \{w \in \Sigma^*_{in} \mid q_0 w \vdash^*_{\mathcal{M}} \alpha q \beta \text{ with } q \in F \text{ and } \alpha q \beta \text{ is an ID}\}$$

We introduce the notion of a *valid computation* of a Turing machine which is non-restrictive in the sense that any Turing machine can be effectively transformed into a Turing machine that performs valid computations only. In particular, case (iii) can be guaranteed by additional transition rules that replace, after reaching a final state, all non-blank symbols on the tape by b and then proceed to a final state.

Definition 6.5.3 (Valid Computation)
Let $\mathcal{M}$ be a Turing machine. A *valid computation of* $\mathcal{M}$ is a non-empty sequence of strings $(w_1, \ldots, w_n)$ where (i) each w_i is an ID of $\mathcal{M}$ of the form $b^\infty\Sigma^*Q\Sigma^*b^\infty$ (b^∞ denotes an infinite string of the blank symbol b), (ii) w_1 is the *initial* ID of the form $b^\infty q_0 v b^\infty$ with $v \in \Sigma_{in}^*$, (iii) w_n is a *final* ID of the form $b^\infty q b^\infty$ with $q \in F$, and (iv) $w_i \vdash_{\mathcal{M}} w_{i+1}$ for all i with $1 \le i < n$.

We use the following obvious connection of valid computations of a Turing machine $\mathcal{M}$ and the language accepted by $\mathcal{M}$.

Lemma 6.5.4
Let $\mathcal{M}$ be a Turing machine. Then there is a valid computation of $\mathcal{M}$ with initial ID $b^\infty q_0 v b^\infty$ if and only if $v \in L(\mathcal{M})$.

We call the problem of deciding whether the language accepted by a Turing machine contains the empty word or not the *blank tape accepting problem of Turing machines*. If, for a Turing machine $\mathcal{M}$, the language $L(\mathcal{M})$ accepted by $\mathcal{M}$ contains the empty word, then we say that $\mathcal{M}$ *accepts the blank tape.* This problem is well-known to be undecidable by a reduction of the halting problem of Turing machines.

Theorem 6.5.5
The word problem with respect to pseudo-linear equational theories is undecidable.

Proof We reduce the blank tape accepting problem of Turing machines. Let $\mathcal{M}$ be a deterministic Turing machine which performs only valid computations. Without loss of generality we assume that $F = \{q_\mathrm{f}\}$ and that $\delta(q_\mathrm{f}, a) = \emptyset$ for all tape symbols $a \in \Sigma$, i.e. a move is not possible after $\mathcal{M}$ reaches the final state q_f. For the construction of the pseudo-linear equational theory $\mathcal{E}$, let $\Sigma_\mathcal{E}$ be the signature $\Sigma \uplus \{a_q \mid a \in \Sigma, q \in Q\} \uplus \{\epsilon, g_0, h\}$ where ϵ is a constant symbol and the other function symbols are unary. The symbols a, b, and c denote unary function symbols in the rest of this proof. We use string notation to denote the (ground) terms over $\Sigma_\mathcal{E}$, i.e. $a_1a_2\ldots a_n(x)$ and $a_1a_2\ldots a_n(\epsilon)$ $(a_1a_2\ldots a_n)$ denote $a_1(a_2(\ldots a_n(x)))$ and $a_1(a_2(\ldots a_n(\epsilon)))$, respectively.

Given the signature $\Sigma_\mathcal{M} = \Sigma_\mathcal{E} \setminus \{g_0, h\}$, the configurations of $\mathcal{M}$ are represented by terms of the form $b^*\Sigma_\mathcal{M}^*b^*$. The symbols of the form a_q represent both the state q and the position of the head which is pointing to a. More precisely, a configuration $b^\infty\alpha a q \beta b^\infty$ of $\mathcal{M}$ with $\alpha, \beta \in \Sigma^*$, $a \in \Sigma$, and $q \in Q$ is represented by a string (term) of the form $b^*\alpha a_q \beta b^*$.

We construct a semi-Thue system R according to the transition function δ of $\mathcal{M}$ over the signature $\Sigma_\mathcal{M}$. Let R be the smallest set of rewrite rules such that for each

$a, a', c \in \Sigma$ and each $q, q' \in Q$ either (i) $ca_q(x) \Rightarrow c_{q'}a'(x) \in R$ if $\delta(q,a) = \{(q', a', \texttt{left})\}$ or (ii) $a_qc(x) \Rightarrow a'c_{q'}(x) \in R$ if $\delta(q,a) = \{(q', a', \texttt{right})\}$. It follows that $\mathcal{M}$ accepts the blank tape if and only if there is a term s of the form $b^*b_{q_0}b^*$ and a term t of the form $b^*b_{q_\mathrm{f}}b^*$ with $s \overset{*}{\Rightarrow}_R t$.

Let R_1 be the ground term rewriting system which consists of the rewrite rules $g_0(\epsilon) \Rightarrow bg_0(\epsilon)$, $g_0(\epsilon) \Rightarrow b_{q_0}h(\epsilon)$, $h(\epsilon) \Rightarrow bh(\epsilon)$, and $h(\epsilon) \Rightarrow \epsilon$. The construction of R_1 implies that the ground term $g_0(\epsilon)$ can be rewritten by R_1 to any ground term of the form $b^*b_{q_0}b^*$. Let R_2 be the left-linear (right-ground) term rewriting system which consists of the rewrite rules $b_{q_\mathrm{f}}(x) \Rightarrow b_{q_\mathrm{f}}(\epsilon)$ and $bb_{q_\mathrm{f}}(x) \Rightarrow b_{q_\mathrm{f}}(\epsilon)$. Then exactly the ground terms of the form $b^*b_{q_\mathrm{f}}\Sigma_{\mathcal{E}}^*$ can be rewritten by R_2 to the ground term $b_{q_\mathrm{f}}(\epsilon)$. Let R' be the term rewriting system $R \cup R_1 \cup R_2$. It follows that $\mathcal{M}$ accepts the blank tape if and only if $g_0(\epsilon) \overset{*}{\Rightarrow}_{R'} b_{q_\mathrm{f}}(\epsilon)$. Note that $\mathcal{M}$ accepts the blank tape if and only if there is a rewriting sequence of the form $g_0(\epsilon) \overset{*}{\Rightarrow}_{R_1} u \overset{*}{\Rightarrow}_R v \overset{*}{\Rightarrow}_{R_2} b_{q_\mathrm{f}}(\epsilon)$ where u and v are ground terms over the signature $\Sigma_{\mathcal{E}}$. Moreover, observe that R' is indeed orthogonal (left-linear, no critical pairs) since $\mathcal{M}$ is deterministic and by the assumption that after reaching the final state q_f no further moves are possible. The orthogonality of R' implies that R' is confluent such that any rewriting sequence $g_0(\epsilon) \overset{*}{\Rightarrow}_{R'} b_{q_\mathrm{f}}(\epsilon)$ can be reordered into a rewriting sequence of the above form.

We construct $\mathcal{E}$ as the symmetric closure of the rewrite rules of R', i.e. $\mathcal{E}$ is the set $\{s \approx t \mid s \Rightarrow t \in R'\}$ of equations. Observe that $\mathcal{E}$ is in fact a pseudo-linear equational theory. The ground term $b_{q_\mathrm{f}}(\epsilon)$ is irreducible with respect to R' which implies, again using the orthogonality of R', that $\mathcal{M}$ accepts the blank tape if and only if $g_0(\epsilon) \overset{*}{\Leftrightarrow}_{\mathcal{E}} b_{q_\mathrm{f}}(\epsilon)$. ■

This reduction also proves that $(\overset{*}{\Leftrightarrow}_{\mathcal{E}})(L)$ is not necessarily recognizable by a tree automaton when L is recognizable and $\mathcal{E}$ is a pseudo-linear equational theory. As a corollary from the previous theorem, we obtain that the E-unifiability problem with respect to pseudo-linear (string) equational theories in which all equations have the form $f(g(x)) \approx f'(g'(x))$ (no ground equations) is undecidable. The word problem in such a system is decidable, since equations of the above type are length-preserving.

Corollary 6.5.6
Given a signature Σ which contains only unary function symbols and constant symbols, the E-unifiability problem with respect to pseudo-linear equational theories over Σ in which all equations are of the form $f(g(x)) \approx f'(g'(x))$ is undecidable.

Proof Given a Turing machine $\mathcal{M}$, we assume the same conditions as in the proof of Theorem 6.5.5. Moreover, the rewrite system R is constructed in the same way as above. Note that R is a semi-Thue system in which all rewrite rules are of the form $f(g(x)) \Rightarrow f'(g'(x))$. Let R_1 be the term rewriting system (semi-Thue system) which consists of the rewrite rules $g_0b(x) \Rightarrow bg_0(x)$ and $g_0b_{q_0}(x) \Rightarrow b_{q_0}b(x)$. Given a substitution σ, the construction of R_1 implies that a term $g_0b(y)\sigma$ can always be reduced by R_1 to a term of the form $b^*bb_{q_0}b(y')$ whenever the variable y is instantiated by σ to a term of the form $b^*b_{q_0}(y')\sigma'$ where σ' is an arbitrary substitution. Let R_2 be the term rewriting system (semi-Thue system) which contains only the rewrite rule $bb_{q_\mathrm{f}}(x) \Rightarrow b_{q_\mathrm{f}}b(x)$ such that only a term of the form $b^*bb_{q_\mathrm{f}}(z')$ can be reduced by R_2 to a term $b_{q_\mathrm{f}}bb^*(z')$. Let R' be the term rewriting system $R \cup R_1 \cup R_2$. Then, by a similar reasoning as above, we have that $\mathcal{M}$ accepts the blank tape if and only if there is a substitution σ such that $g_0b(y)\sigma \overset{*}{\Leftrightarrow}_{\mathcal{E}} b_{q_\mathrm{f}}b(z)\sigma$. ■

6.6 Standard Equational Theories

The E-unifiability problem with respect to the so-called *standard theories* or *standard equational theories* has been shown to be decidable by Nieuwenhuis (1996). Recall Definition 6.2.5 of standard equations and standard equational theories. The syntactic characterization of standard equational theories includes non-linear shallow variable occurrences which cause substantial difficulties when using automata-theoretic concepts as we already demonstrated in Section 6.2. Nieuwenhuis (1996) has partly overcome these problems by the saturation-based methods of basic superposition/paramodulation. Standard equations may have non-linear shallow variables which are, however, restricted to dedicated positions for each function symbol in the signature. From a technical point of view, these restrictions avoid overlappings of non-linear shallow variables and more complex terms.

With (Semi-linear) Equational Flattening of (semi-linear) sorted equational theories we have demonstrated that these restrictions can also be dropped. However, semi-linear sorted equational theories do not subsume standard equational theories. There is one form of standard equations which is not embedded by the semi-linear case: the form $f(\dots, g(x), \dots) \approx x$ where g is a unary function symbol with additional restrictions on the positions of linear terms and non-linear shallow variables in other equations, c.f. case (ii) of Definition 6.2.5. Obviously, the subterm $g(x)$ cannot be transformed into a sort declaration. However, we can show that E-unification in those theories can still be decided by basic sorted paramodulation. The following discussion mainly follows the previous argumentation on shallow and semi-linear sorted equational theories though the proofs are much more involved. We slightly generalize the form of invariants of the saturation process. The definition of shallow terms has to be adapted such that not only variables but also *constants as arguments* of shallow terms are allowed. In the sequel, we will use this generalized definition of shallow terms.

Definition 6.6.1 (Semi-shallow Sorted Equation)
We call a sorted equation $\Psi \parallel \to f(t_1, \dots, t_n) \approx x$ *semi-shallow* if for all i with $1 \leq i \leq n$ the term t_i is either a variable or a term of the form $g(x)$ where g is a unary function symbol. If there is any $g(x)$ among the t_i then all other t_j which are variables are distinct from x.

Note that several occurrences of a subterm $g(x)$ with a unique function symbol g are possible in a semi-shallow sorted equation. The following definition of *semi-shallow sorted equational theories* explicitly excludes this case. However, a semi-shallow sorted equation with several occurrences of a subterm of the form $g(x)$ can easily be transformed into a semi-shallow sorted equation with only one occurrence of $g(x)$ and a semi-linear sorted equation. Consider the semi-shallow equation $f(g(x), g(x)) \approx x$. We transform the equation into a semi-shallow equation $f'(g(x)) \approx x$ and a semi-linear (shallow) equation $f(x,x) \approx f'(x)$. The two new equations are logically equivalent to the former equation if f' is a fresh function symbol. Nevertheless, we omit a formal definition of the transformation here to simplify technical matters. Note that, in general, the transformation does not work to extend the class of standard equational theories using the approach of Nieuwenhuis (1996). Suppose, for example, that $lin(f) = \{1\}$ and $sh(f) = \{2\}$. In this case, an equation $f(x,x) \approx f'(x)$ is not allowed since the first argument contains at a linear position argument the variable x which is not a linear term in the equation. In other words, in standard equational theories we cannot freely introduce new non-linear shallow equations due to the restrictions of shallow and linear position arguments.

Definition 6.6.2 (Semi-shallow Sorted Equational Theory)
A sorted equational theory $\mathcal{E}$ is called *semi-shallow* if all declarations in $\mathcal{E}$ are shallow and all sorted equations in $\mathcal{E}$ are shallow or semi-shallow where each semi-shallow sorted equation $\Psi \parallel \rightarrow f(t_1, \dots, t_n) \approx x$ in $\mathcal{E}$ contains at most one t_i of the form $g(x)$.

With the distinction of shallow and semi-shallow sorted equational theories we show that the technical difficulties in (Nieuwenhuis 1996, Nieuwenhuis 1998) are due to the somewhat artificial definition of semi-shallow equations of the form $f(\dots, g(x), \dots) \approx x$ with the occurrences of terms of the form $g(x)$. The decidability proof of shallow sorted equational theories is substantially easier than the proof for semi-shallow theories.

In order to show that the generalized fragment is still closed under basic sorted paramodulation we have to treat certain equations which contain variable disjoint sides carefully, e.g., the equation $f(x,y) \approx g(z)$. Equations of this form cannot be oriented by any admissible ordering and thus have to be applied in both directions. Saturation may not terminate when such equations are superposed on subterms of the form $g(x)$ in an equation $f(\dots, g(x), \dots) \approx x$. Arbitrary deep term structures can be generated by subsequent paramodulation steps. However, it is possible to transform such equations dynamically during the saturation process. For example, the equation $f(x,y) \approx g(z)$ can be transformed into equations $f(x,y) \approx a$ and $g(z) \approx a$ where a is a new constant symbol. As a consequence we have to show that only a finite number of new symbols can be generated.

Consider the simplest equation $x \approx y$. If this equation is derived we can just stop saturation since the Herbrand universe collapses in this case. For more complicated sorted equations with variable disjoint sides and solved sort constraint, e.g., $\Psi \parallel \rightarrow x \approx t$ and $x \notin vars(t)$, we can still deduce that either Ψ is empty or the Herbrand universe collapses since $x \notin vars(\Psi)$. As we have mentioned above, a critical situation may come up when such equations are superposed onto a subterm $g(x)$. In general, this may happen using sorted equations of the form $\Psi \parallel \rightarrow x \approx t$ and also $\Psi \parallel \rightarrow g(x) \approx t$ where $x \notin vars(t)$. Importantly, equations of the form $\Psi \parallel \rightarrow f_1(x_1, \dots, x_n) \approx f_2(y_1, \dots, y_m)$ where $\{x_1, \dots, x_n\} \cap \{y_1, \dots, y_m\} = \emptyset$ and $n, m > 1$ do not have to be considered. Paramodulation by those equations onto a subterm $g(x)$ is not possible.

Consider the transformation of a sorted equation $S(x) \parallel \rightarrow g(x) \approx f(y,y)$ into two equations $S(x) \,\|\, \to g(x) \approx a$ and $S(x) \,\|\, \to f(y,y) \approx a$. Such a transformation step is called *splitting*. The splitting of similar unsorted equations has already been used in (Nieuwenhuis 1996). If the constant symbol a is a new symbol splitting clearly preserves (un-)satisfiability. However, (un-)satisfiability is also preserved if the constant a is being reused if a has been introduced before for the same solved sort constraint $S(x)$. The situation for a sorted equation C of the form $\Psi \,\|\, \to x \approx t$ is easier. We may replace C by a sorted equation C' of the form $\Psi \,\|\, \to x \approx a$ where a is a unique constant symbol which can be reused for all sorted equations of the form of C. The derivation of a sorted equation D of the form $\Psi \,\|\, \to t \approx a$ is not required in this case since C' is already a generalization of D. Consider, for example, the sorted equation $S(y) \,\|\, \to x \approx f(y,y)$. The equation is true whenever either the sort S is empty or the Herbrand universe contains exactly one element. Splitting introduces with a an explicit name for this element.

Splitting terminates for clause sets which contain only finitely many sorts. There are two important observations. For finitely many sorts there are an exponential number of intersections of sorts. Thus variables can have only finitely many different sorts. Shallow equations in which both sides have root symbols with arity greater than one are not split. For if one side contains a newly introduced constant symbol as an argument, splitting would generate new constant symbols which depend on previously introduced symbols. For example, splitting of a sorted equation $S(x) \,\|\, \to f(x,a) \approx h(y,z)$ generates a sorted equation $S(x) \,\|\, \to f(x,a) \approx b$ where a is a constant symbols previously introduced by splitting and b is a new constant symbol. In this way splitting would not terminate in general.

Definition 6.6.3
Let N be a set of clauses. A splitting step is defined as the simplification of a clause in N by the rule *UniversalSplit*:

$$\text{Simplify}\frac{\Psi \,\|\, \to x \approx t}{\Psi \,\|\, \to x \approx a}$$

where (i) the sort constraint Ψ is solved in the premise, (ii) t is a non-variable shallow term, (iii) $x \notin \mathit{vars}(t)$, and (iv) a is a unique constant symbol reserved for UniversalSplit, as well as by the rule *UnarySplit*:

$$\text{Simplify}\frac{\Psi \,\|\, \to g(x) \approx t}{\begin{array}{c}\Psi \,\|\, \to g(x) \approx a \\ \Psi \,\|\, \to t \approx a\end{array}}$$

where (i) the sort constraint Ψ is solved in the premise, (ii) t is a non-variable shallow term, (iii) g is a unary function symbol, (iv) $x \notin \mathit{vars}(t)$, and (v) if there is a clause $\Psi \,\|\, \to g(x) \approx b$ in N then let $a\!:= b$ or, otherwise, let a be a fresh constant symbol.

Proposition 6.6.4
Let Σ be a finite signature and let N be a set of clauses over Σ of the form:

$$\Psi \,\|\, \to x \approx a$$

and the form:

$$\Psi \parallel \rightarrow g(x) \approx a$$

where the sort constraint Ψ is solved, $a \in \Sigma$ is a constant, and $g \in \Sigma$ is a unary function symbol. Then the set N is finite up to variable renaming and Sort Condensing.

Proof Let n be the number of different monadic predicate symbols occurring in clauses of N. Then there are 2^n different sort constraints on one variable up to Sort Condensing. Consequently, the set N contains in the order of 2^n different clauses up to variable renaming and Sort Condensing. ■

In the following proposition we show that splitting can only be applied finitely many times. Sorted equations are split only if one side is an (unsorted) variable x (by UniversalSplit) or a term $g(x)$ (by UnarySplit) where g is unary and x is a (sorted) variable. More complex sorted equations $f(s_1, \dots, s_n) \approx h(t_1, \dots, t_m)$ are not split if f and h are n-ary functions with $n > 1$ and the s_i and t_j are variables or constants. Recursive occurrences of formerly introduced constants would lead to an infinite splitting process. Recall that UniversalSplit reuses the same constant symbol in each application. A constant symbol introduced by UnarySplit can be seen as a name for a particular intersection of sorts on one variable. In this sense constants are reused by UnarySplit whenever the same intersection of sorts arises several times. Finally, splitting is applied only to sorted equations in which both sides are shallow. Obviously, there are only finitely many different shallow sorted terms over a finite signature.

Proposition 6.6.5
Let N be a set of clauses over a finite signature. Then splitting can be applied only finitely many times in any derivation of basic sorted paramodulation with splitting.

Proof Only the rules UniversalSplit and UnarySplit may introduce new constant symbols. We discuss the proof for UnarySplit. The proof for the rule UniversalSplit is a special case of the proof for UnarySplit. We assume that all sort constraints are minimal in the sense that no further Sort Condensing is possible. Let C be a clause $\Psi \parallel \rightarrow g(x) \approx t$ from N where the conditions of UnarySplit hold for C. Suppose that there is no clause $\Psi \parallel \rightarrow g(x) \approx b$ in N. In this case UnarySplit introduces a new constant symbol a and transforms C to $C' = \Psi \parallel \rightarrow g(x) \approx a$. Note that in general the sort constraint Ψ is not in solved form in C' anymore since the equation does not contain the variables of t.

Suppose the sort constraint of C' has been simplified to Ψ' where Ψ' does not need to be in solved form. The constant a may be viewed as a label for Ψ' and g, not for Ψ and g. Thus in a subsequent application of UnarySplit to a clause $\Psi \parallel \rightarrow g(x) \approx t'$ a new constant has to be introduced. However, an application of UnarySplit to a clause $\Psi' \parallel \rightarrow g(x) \approx t'$ can reuse the constant a.

By Proposition 6.6.4 there are only finitely many clauses of the form $\Psi \parallel \rightarrow g(x) \approx a$. Thus UnarySplit introduces only finitely many new constant symbols. Note that there are only finitely many different shallow sorted terms over a finite signature up to variable renaming and Sort Condensing. Thus UnarySplit can even be applied only finitely many times. ■

Proposition 6.6.6
Let N and N' be two sets of clauses such that the set N' is the result of a transformation of N by the rules UniversalSplit and UnarySplit. Then N is satisfiable if and only if N' is satisfiable.

Proof Let N be a set of clauses and let C be the clause $\Psi \,\|\, \to x \approx t$ in N where the conditions of UniversalSplit hold for C, in particular, $x \notin \mathit{vars}(\Psi)$ for otherwise Ψ would not be solved. Let D be the conclusion $\Psi \,\|\, \to x \approx a$ of an application of UniversalSplit to C. Let N' be the set $\{N \setminus C\} \cup \{D\}$.

Suppose I is a model of N. To show that D is true in I we distinguish two cases. Suppose that σ is a ground substitution where $\Psi\sigma \subseteq I$ and $x \notin \mathit{Dom}(\sigma)$. Then $x\tau \approx t\sigma \in I$ for all ground substitutions τ since C is true in I and $x \notin \mathit{vars}(t)$. In particular, we have that $a \approx t\sigma \in I$. By the transitivity and symmetry of $\approx$ it follows that $x\tau \approx a$ and thus $D\sigma\tau$ are true in I for all ground substitutions τ. On the other hand, suppose that for a ground substitution σ with $x \notin \mathit{Dom}(\sigma)$ there is a sort constraint literal $A \in \Psi$ where $A\sigma \notin I$. Then $D\sigma\tau$ is also true in I for all ground substitutions τ. Consequently, I is a model of N'. The other direction is similar.

Let N be a set of clauses and let C be the clause $\Psi \,\|\, \to g(x) \approx t$ in N where the conditions of UnarySplit hold for C. Let C' and D be the conclusions $\Psi \,\|\, \to g(x) \approx a$ and $\Psi \,\|\, \to t \approx a$, respectively, of an application of UnarySplit to C. Let N' be the set $\{N \setminus C\} \cup \{C', D\}$.

Suppose I is a model of N. If there is no clause $\Psi \,\|\, \to g(x) \approx b$ in N then a is a fresh constant symbol and we may construct I' by adding, for all ground substitutions σ such that $\Psi\sigma \subseteq I$, the ground instances $g(x)\sigma \approx a$ and $t\sigma \approx a$ to I. Note that for any ground substitution σ with $\Psi\sigma \subseteq I$ we have that $g(x)\sigma \approx t\sigma$ is true in I since C is true in I. Thus I' is a model of N'.

Let C'' be a clause $\Psi \,\|\, \to g(x) \approx b$ in N. In this case a has been chosen to be equal to b. Since C'' is true in I the clause C' is also true in I. To show that D is true in I we distinguish two cases. Suppose that σ is a ground substitution where $\Psi\sigma \subseteq I$. Then $g(x)\sigma \approx t\sigma \in I$ and $g(x)\sigma \approx b \in I$ since C and C'' are true in I, respectively. By the transitivity of $\approx$ we have that $t\sigma \approx b$ and thus $D\sigma$ are true in I. On the other hand, suppose that for a ground substitution σ there is a sort constraint $A \in \Psi$ where $A\sigma \notin I$. Then $D\sigma$ is also true in I. Consequently, I is a model of N'.

The other direction follows immediately since C is a consequence of a (sound) paramodulation step on a in C' to t using D. ■

Note that both splitting rules are monotone in the sense that the conclusions are always smaller than the premise, assuming that constants are the smallest objects in the term ordering. It follows by the previous result that both splitting rules are admissible simplification rules.

Corollary 6.6.7
We assume a finite signature. The combination of the basic sorted paramodulation calculus with splitting is sound and refutationally complete.

Proof By Proposition 6.6.6 splitting preserves (un-)satisfiability. We may additionally assume that splitting is an admissible simplification rule, see the previous discussion. Due to Proposition 6.6.5 splitting can be applied only finitely many times and thus fairness is not affected. Consequently, the corollary follows from the soundness and refutational completeness of basic sorted paramodulation. ■

6.7 Decidability

In order to show that satisfiability of monadic equational types with respect to semi-shallow sorted equational theories is decidable, we employ basic sorted paramodulation. Similar to the previous decidability proof, the atom ordering must be, in particular, compatible with the subterm property. An atom ordering which is induced by an admissible term ordering is compatible with the subterm property. We assume an admissible term ordering which is compatible with the arity of non-unary function symbols and in which constants are the smallest objects. A term ordering $\succ$ is compatible with the arity of non-unary function symbols whenever for all function symbols f with arity $n > 1$ and for all unary function symbols g we have that $f(x_1, \ldots, x, \ldots, x_n)\sigma \succ g(x)\sigma$ for all ground substitutions σ. We assume admissible literal orderings of Definition 3.1.48 and admissible clause orderings of Definition 3.1.15 and use the equality selection strategy of Definition 6.3.1. The motivation of this strategy is to avoid an increase in term depth during saturation by basic sorted paramodulation.

We assume that the basic restrictions are maintained by some marking mechanism that identifies substitution positions in the clauses. However, basic restrictions outside the sort constraint are only required for clauses of the form $\Psi \,\|\, \rightarrow f(t_1, \ldots, t_m) \approx x$ where $f(t_1, \ldots, t_m) \approx x$ is a semi-shallow equation with at least two occurrences of the subterm $\boxed{g(x)}$. That is, substitution positions have to be maintained only for repeated occurrences of the subterm $g(x)$. For example, a shallow sorted equation $S(g(y)) \,\|\, \rightarrow \boxed{g(y)} \approx y$ may be transformed into $S(g(y)) \,\|\, \rightarrow g(y) \approx y$ whereas $g(y)$ in a semi-shallow sorted equation of the form $S(g(y)) \,\|\, \rightarrow f(\boxed{g(y)}, \boxed{g(y)}) \approx y$ must be marked to indicate that it occurs at a substitution position. If substitution positions are maintained by equality constraints, we may use a constraint inheritance strategy that propagates equality constraints eagerly into the clause part unless a semi-shallow sorted equation is generated. The constraint propagation corresponds to a weakening of the equational constraint which is always possible, c.f. (Nieuwenhuis & Rubio 1995).

In the following lemma we do not explicitly include constants as arguments of (semi-)shallow terms in the definition of the invariant, although constants may also occur in this way during saturation. However, as long as constants are the smallest objects in the term ordering, constants can only be replaced by constants using paramodulation. Therefore, we skip this case to simplify technical matters.

Lemma 6.7.1
Let $\mathcal{E}$ be a semi-shallow sorted equational theory. Then $\mathcal{E}$ can be finitely saturated by basic sorted paramodulation, splitting, Sort Condensing, and Intersection Condensing. The productive clauses in the saturated set form a semi-shallow sorted equational theory which consists of (non-linear) shallow term declarations, trivial subsort declarations, and semi-shallow sorted equations with a solved sort constraint.

Proof The closure process is parameterized by an admissible term ordering $\succ$ with the additional proviso that constants are the smallest symbols with respect to $\succ$ and that for all function symbols f with arity $n > 1$ and for all unary function symbols g we have that $f(x_1, \dots, x, \dots, x_n)\sigma \succ g(x)\sigma$ for all ground substitutions σ. We assume that splitting is applied with the highest priority, i.e. a sorted equation is always split before any other rule is applied to this equation. We show that the class of clauses of the following form is closed under basic sorted paramodulation with splitting:

1. $S_1(s), \dots, S_n(s), T_1(t), \dots, T_m(t), \Psi \parallel \;\rightarrow L$ where n and m are possibly zero, L is either a monadic atom $S(u)$ or an equation $u \approx v$, s and t are non-variable shallow terms, u and v are shallow terms, and Ψ contains only literals of the form $S(x)$ or $S(a)$. Note that Ψ does not have to be solved.

2. $\Psi \parallel \;\rightarrow f(x_1, \dots, g(x), \dots, x_n) \approx x$ where n is possibly zero, g is a unary function symbol, $g(x)$ occurs only once, and Ψ is solved.

3. $T_1(t), \dots, T_n(t), \Psi \parallel \;\rightarrow f(t_1, \dots, t_m) \approx x$ where n is possibly zero, g is a unary function symbol, $f(t_1, \dots, t_m) \approx x$ is a semi-shallow equation with at least two occurrences of the subterm $\boxed{g(x)}$, t is a non-variable shallow term, Ψ contains only literals of the form $S(y)$ or $S(a)$. Note that Ψ does not have to be solved.

Each clause of the original semi-shallow equational theory belongs to Category 1 or Category 2 and has a solved sort constraint. The predicates in these clauses are all predicates of the original theory.

Assuming a finite signature there is only a bounded number of clauses which belong to each of the three categories because the depth of all these clauses as well as the length of variable chains between its literals is bounded. Hence, there are only finitely many different clauses of this form up to Sort and Intersection Condensing and variant clauses. If the saturation process produces only clauses of this form, it will terminate.

If a rule in Category 1, 2, or 3 has unsolved sort constraints then we can perform a Sort Constraint Resolution inference. The result is a clause of the same category. The equality selection strategy prefers a literal $S(t)$ with a non-variable term t, i.e. potential occurrences of a literal $S(x)$ with $x \notin vars(L)$ are not selected in the presence of t. Otherwise, there are several ways to apply Basic Paramodulation Right on clauses with solved sort constraints:

1. The Basic Paramodulation Right of a clause C_1 in Category 1 on a clause C_2 in Category 1 results in a clause C_3 of the same category shown in the proof of Lemma 6.3.7. This result still holds for terms which are shallow in the sense that they may contain not only variables but also constants as arguments since constants are the smallest objects in the term ordering. Note that the invariant in Lemma 6.3.7 may contain only one non-variable t in the sort constraint in contrast to clauses of Category 1. However, using Sort Constraint Resolution such unsolved sort constraints can be transformed into the form of the sort constraint that appears in the invariant of Lemma 6.3.7.

2. The Basic Paramodulation Right of a clause C_1 in Category 1 on a clause C_2 in Category 2 results in a clause C_3 of Category:

   (a) 3 if C_1 contains a non-collapsing equation and is applied at topmost position.

   (b) 2 if C_1 is a non-collapsing sorted equation of the form $\Psi \,||\, \rightarrow g(x) \approx h(x)$ and is applied on non-topmost position. Note that an equation of the form $f(s_1, \ldots, x, \ldots, s_k) \approx g(x)$ where $k > 1$ cannot be applied because it is oriented from left to right due to the assumption that the term ordering $\succ$ is in a sense compatible with the arity of function symbols.

   (c) 1 if C_1 is a sorted equation of the form $\Psi \,||\, \rightarrow x \approx t$ or $\Psi \,||\, \rightarrow g(x) \approx t$ where $x \notin vars(t)$ and is applied at non-topmost position. Splitting ensures that t is always a constant. Note that the equation in C_3 may contain constants as arguments. However, we do not consider this case explicitly here to simplify technical matters. Informally, constants can only be paramodulated to other constants since constant symbols are assumed to be the smallest objects in the term ordering. Thus constants may be seen as variables in this case.

   (d) 1 if C_1 contains a collapsing equation, regardless of topmost or non-topmost application.

3. The Basic Paramodulation Right of a clause C_1 in Category 2 on a clause C_2 in Category 1 results in a clause C_3 of Category:

   (a) 3 if C_2 contains a non-collapsing equation. If the equation in C_3 does not contain any non-variable marked subterm then C_3 belongs to Category 1.

   (b) 1 if C_2 contains a collapsing equation or a sort declaration.

4. The Basic Paramodulation Right of a clause C_1 in Category 2 or 3 on a clause C_2 in Category 2 or 3 results in a clause C_3 of Category 1. Note that if both C_1 and C_2 belong to Category 2 then two different non-variable terms may occur in the sort constraint of the resulting clause C_3 of Category 1. For example, if $C_1 \approx S_1(y) \,||\, \rightarrow f(g(x), y) \approx x$ and $C_2 = S_2(z) \,||\, \rightarrow f(z, h(z')) \approx z'$ then $C_3 =$

$S_1(h(z')), S_2(g(x)) \parallel \; \rightarrow z' \approx x$. Another example explains the restriction of semi-shallow theories where only one occurrence of a term $g(x)$ is allowed in each semi-shallow equation. Suppose $C_1 = \parallel \rightarrow f(g(x), g(x)) \approx x$ and $C_2 = \parallel \rightarrow g(h(y)) \approx y$ where C_1 is obviously semi-shallow but is not allowed in the original clause set. Then we may derive $C_3 = \parallel \rightarrow f(y, g(h(y))) \approx h(y)$ which is also not semi-shallow.

5. The Basic Paramodulation Right of a clause C_1 in Category 3 on a clause C_2 in Category 1 or vice versa results in a clause C_3 of Category:
   (a) 3 if C_2 contains a non-collapsing equation. If the equation in C_3 does not contain any marked subterm of the form $g(x)$ then C_3 belongs to Category 1.
   (b) 1 if C_2 contains a collapsing equation or a sort declaration.

It is important to see that clauses of Category 3 which contain several occurrences of a subterm $g(x)$ appear only during the saturation process. The occurrences of the $g(x)$ in those clauses are marked due to basic restrictions. In other words, a paramodulation step on a subterm $g(x)$ is only possible in clauses of Category 2. Those clauses either belong to the original theory or are conclusions of case 2b. ■

The complexity of saturation by basic sorted paramodulation with splitting remains simply exponential since any inference which involves semi-shallow sorted equations may only derive polynomially more clauses compared to inferences from (non-linear) shallow sorted equations. This observation and Proposition 6.3.12 imply that saturation by basic sorted paramodulation with splitting of an (alternating) semi-shallow sorted equational theory can be done in simply exponential time.

Proposition 6.7.2
Let $\mathcal{E}$ be an (alternating) semi-shallow sorted equational theory. Then $\mathcal{E}$ can be finitely saturated by basic sorted paramodulation, splitting, Sort Condensing, and Intersection Condensing in simply exponential time with respect to $size(\mathcal{E})$ resulting in a sorted equational theory $\mathcal{E}'$ such that $size(\mathcal{E}')$ is at most simply exponentially larger than $size(\mathcal{E})$.

Recall the subset construction in Section 4.3 which involves the inference rule State Union and the simplification rule De-Alternation. The Proposition 4.3.9 states that any alternating (non-linear) shallow sort theory $\mathcal{S}$ without non-trivial subsort declarations can be transformed into a non-alternating theory $\mathcal{S}'$ such that the minimal model of $\mathcal{S}$ is essentially equivalent to the minimal model of $\mathcal{S}'$. In particular, De-Alternation encodes the alternation over one variable into a single state. We may use the same technique to obtain the following corollary.

Corollary 6.7.3
Let $\mathcal{E}$ be an (alternating) semi-shallow sorted equational theory without non-trivial subsort declarations and let $\mathcal{P}$ be the set of predicate symbols which occur in $\mathcal{E}$. Then $\mathcal{E}$ can be transformed into a semi-shallow sorted equational theory $\mathcal{E}'$ in which all term declarations are non-alternating and all sorted equations have non-alternating sort constraints such that

(i) $T^{\mathcal{E}}(P_1)\cap\ldots\cap T^{\mathcal{E}}(P_n) = T^{\mathcal{E}'}(S_{\{P_1,\ldots,P_n\}})$ for any set of predicate symbols $P_1,\ldots,P_n \in \mathcal{P}$ and (ii) $size(\mathcal{E}')$ is at most simply exponentially larger than $size(\mathcal{E})$ with respect to the cardinality of $\mathcal{P}$.

Semi-shallow sorted equational theories over a finite signature can be finitely saturated by basic sorted paramodulation and splitting. Thus the decidability proof of E-unification with respect to semi-shallow sorted equational theories can be established analogously to the proof for shallow sorted equational theories of Theorem 6.3.15. In particular, there are only finitely many applications of Basic Paramodulation Left (and Equality Resolution) on some given (dis-)equation with respect to a clause set that is finitely saturated by basic sorted paramodulation. The subsequent applications of Sort Constraint Resolution have already been shown to be terminating.

Theorem 6.7.4
Unifiability with respect to finitely saturated semi-shallow sorted equational theories is decidable.

Proof The result follows from the proof of Theorem 6.3.15 which can easily be extended to semi-shallow sorted equational theories. ■

The next theorem summarizes the discussion and states that the theory of monadic equational types over semi-shallow sorted equational theories is decidable. Any semi-shallow sorted equational theory may be finitely saturated by basic sorted paramodulation such that the candidate model of the saturation is the minimal model of the theory. By a subsequent saturation with basic sorted paramodulation between the saturated theory and a negated monadic equational type we can effectively compute the satisfiability of the type with respect to the theory. A negated monadic equational type is a negative clause and thus any clause with an unsolved sort constraint may safely be removed from the saturated theory.

Theorem 6.7.5
Let $\mathcal{E}$ be a semi-shallow sorted equational theory. The theory of monadic equational types over $\mathcal{E}$ is decidable.

Proof The proof is similar to the proof of the Theorem 6.3.17. ■

Following the proof of Theorem 4.3.12, by Proposition 6.7.2 and Corollary 6.7.3, we obtain the following complexity result. Note that satisfiability tests on monadic variable types do not involve any paramodulation inferences but only computations with respect to the linear shallow sort theory within a linear semi-shallow sorted equational theory.

Corollary 6.7.6
The satisfiability problem of the theory of monadic variable types over (alternating) linear semi-shallow sorted equational theories is EXPTIME-complete.

The above corollary implies the EXPTIME-hardness of the satisfiability problem not only of the theory of monadic variable types but also of the theory of (general) monadic equational types over (alternating) semi-shallow sorted equational theories.

Corollary 6.7.7
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) semi-shallow sorted equational theories is EXPTIME-hard.

In the proof of Proposition 7.1.5, we will show the EXPTIME-completeness of the satisfiability problem of the theory of (general) monadic equational types over linear shallow typed equational theories. We obtain as a corollary that the satisfiability problem of the theory of monadic equational types over (alternating) linear semi-shallow sorted equational theories is EXPTIME-complete. The EXPTIME-completeness of the same problem for non-linear theories remains open.

Corollary 6.7.8
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) linear semi-shallow sorted equational theories is EXPTIME-complete.

We shall exploit the complexity results for sorted unification with respect to shallow non-equational sort theories to derive similar results on the number of "well-sorted" most general unifiers of an equational problem with respect to semi-shallow sorted equational theories. The following theorem is an easy extension of Theorem 4.3.15 from shallow sort theories to semi-shallow sorted equational theories similar to Theorem 6.3.21.

Theorem 6.7.9 (Weidenbach (1998))
Let $\mathcal{E}$ be a semi-shallow sorted equational theory and let C be a negative clause of the form $S_1(t_1), \ldots, S_n(t_n) \,\|\, \rightarrow$. We can derive a clause $T_1(y_1), \ldots, T_k(y_k) \,\|\, \rightarrow$ from C and $\mathcal{E}$ by Sort Constraint Resolution if and only if the sorted unification problem $x_1 = t_1, \ldots, x_n = t_n$ has a "well-sorted" most general unifier with respect to the shallow sort theory contained in $\mathcal{E}$ where each x_i is new and has the sort S_i.

The Theorem 4.3.16 states that sorted unification in (non-linear) shallow sort theories is NP-complete and that the number of "well-sorted" most general unifiers is simply exponential in the size of a shallow sort theory and the sorted unification problem. Thus from Theorem 6.7.9, 4.3.16, and 6.7.4 we obtain that there are at most simply exponentially many "well-sorted" most general unifiers for a unification problem with respect to a finitely saturated semi-shallow sorted equational theory.

Corollary 6.7.10
The number of "well-sorted" most general unifiers with respect to a finitely saturated semi-shallow sorted equational theory $\mathcal{E}$ is simply exponential.

6.8 Semi-standard Approximations

The theory of monadic equational types over semi-shallow sorted equational theories is decidable. Similar to the transformation of semi-linear sorted equational theories into essentially equivalent shallow sorted equational theories, we can improve this result by a modification of Semi-linear Equational Flattening called *Semi-standard Equational Flattening*. The transformation introduces a new monadic predicate for each occurrence of a proper non-variable subterm in a term declaration or a sorted equation where the number of new symbols is linear in the number of function symbols which occur in the equational theory. The difference to Semi-linear Equational Flattening is that shallow occurrences of unary subterms of the form $g(x)$ in a so-called *semi-standard sorted equation* of the form $\Psi \parallel \rightarrow f(\ldots, g(x), \ldots) \approx x$ are not transformed. Semi-standard Equational Flattening replaces in a top-down manner the remaining occurrences of a non-variable proper subterm t_i by a variable x where an additional sort constraint on x restricts the ground instances of x to the "well-sorted" ground instances of t. Given a so-called *semi-standard sorted equational theory*, the minimal model of the resulting semi-shallow sorted equational theory is essentially equivalent to the minimal model of the original equational theory while for arbitrary sorted equational theories the minimal model of the result is an upper approximation of the minimal model of the original.

Definition 6.8.1 (Semi-standard Sorted Equation)
We call a sorted equation $\Psi \parallel \rightarrow f(t_1, \ldots, t_n) \approx x$ *semi-standard* if $f(t_1, \ldots, t_n)$ is semi-linear and there is one unary symbol g such that $t_i = g(x)$ for all t_i with $x \in \mathit{vars}(t_i)$.

Definition 6.8.2 (Semi-standard Sorted Equational Theory)
A sorted equational theory $\mathcal{E}$ is called *semi-standard* if all declarations in $\mathcal{E}$ are semi-linear and all sorted equations in $\mathcal{E}$ are semi-linear or semi-standard where each semi-standard sorted equation $\Psi \parallel \rightarrow f(t_1, \ldots, t_n) \approx x$ in $\mathcal{E}$ contains at most one t_i of the form $g(x)$.

The class of semi-standard sorted equational theories strictly subsumes the class of standard equational theories (Nieuwenhuis 1996), c.f. Definition 6.2.5. In particular, despite the generalization to semi-linear variable occurrences and the sort information on the variables, the partitioning of the argument positions of each function symbol into shallow and linear position arguments is omitted.

Definition 6.8.3
The following abstraction is called *Semi-standard Equational Flattening*:

$$\text{Abstract}\frac{\Psi, \Psi' \parallel \rightarrow L[t]_{p_1}}{\begin{array}{c}\Psi' \parallel \rightarrow S_t(t) \\ S_t(x), \Psi, \Psi'' \parallel \rightarrow L[p_1, \ldots, p_n/x]\end{array}}$$

where (i) L is either a monadic atom or an equation, (ii) Ψ, Ψ' is solved, (iii) t is a non-variable subterm at position p_1 with $|p_1| = 2$, (iv) the positions $p_1, \ldots, p_n$ refer to all positions q of t in L with $|q| = 2$, (v) Ψ' is maximal such that $\mathit{vars}(\Psi') \subseteq \mathit{vars}(t)$,

(vi) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (vii) S_t is a new monadic predicate, (viii) x is a new variable, and (ix) if L is a semi-standard equation $s \approx y$ then y does not occur in t.

Semi-standard Equational Flattening differs from Semi-linear Equational Flattening only in the additional case (ix) which prohibits the transformation of subterms of the form $g(x)$ in sorted equations of the form $\Psi \,\|\, \rightarrow f(\dots, g(x), \dots) \approx x$. In the sequel, a monadic predicate S_t refers to the new predicate which has been introduced by Semi-standard Equational Flattening for a non-variable term t. The newly introduced declarations represent, from an automata-theoretic point of view, a tree automaton which recognizes in a state S_t exactly the "well-sorted" ground instances of the original term t. The following proposition suggests that Semi-standard Equational Flattening may be applied to arbitrary sorted equational theories. However, minimal models are preserved only for semi-standard sorted equational theories. For arbitrary sorted equational theories the transformation yields a sorted equational theory whose minimal model is an upper approximation of the minimal model of the original theory.

Proposition 6.8.4
Let $\mathcal{E}$ be a sorted equational theory. Exhaustive application of Semi-standard Equational Flattening to $\mathcal{E}$ terminates and results in a (non-linear) semi-shallow sorted equational theory.

Proof The proof is similar to the proof of the Proposition 4.4.3. Note that an occurrence of $g(x)$ in a semi-standard equation $f(\dots, g(x), \dots) \approx x$ is not abstracted. ■

In the sequel we call a sorted equational theory which has been obtained by an exhaustive application of Semi-standard Equational Flattening to a sorted equational theory $\mathcal{E}$ the *semi-standard flat approximation of* $\mathcal{E}$. The corollary below on the minimal model properties of the new predicates introduced by Semi-standard Equational Flattening follows from the technical Lemma 4.4.7 for Semi-linear Flattening.

Corollary 6.8.5
Let $\mathcal{E}$ be a sorted equational theory and let $\mathcal{E}_{SF}$ be the semi-standard flat approximation of $\mathcal{E}$. Let I be the minimal model of $\mathcal{E}_{SF}$. Let t be a non-variable term for which a new monadic predicate symbol S_t has been introduced by Semi-standard Equational Flattening. Let C be the immediate conclusion $\Psi \,\|\, \rightarrow S_t(t)$ of Semi-standard Equational Flattening on t. Then for all ground atoms $S_t(s)$ which are true in I there is a ground substitution σ such that (i) $I \models s \approx t\sigma$ and (ii) $\Psi\sigma$ is true in I.

Proof The proof is similar to the proof of Lemma 4.4.4. See also Lemma 4.4.7. ■

The Semi-standard Equational Flattening abstracts sorted equational theories such that the minimal models of the resulting approximations are upper approximations of minimal models of the original equational theories.

Proposition 6.8.6
Let $\mathcal{E}$ be a sorted equational theory and let $\mathcal{E}_F$ be the semi-standard flat approximation of $\mathcal{E}$. The minimal model of $\mathcal{E}_F$ is an upper approximation of the minimal model $\mathcal{E}$.

Proof The proof is similar to the proof of Proposition 4.4.8. ■

Semi-standard Equational Flattening yields exact approximations for semi-standard sorted equational theories.

Proposition 6.8.7
Let $\mathcal{E}$ be a semi-standard sorted equational theory and let $\mathcal{E}_F$ be the semi-standard flat approximation of $\mathcal{E}$. The minimal model of $\mathcal{E}_F$ is equivalent to the minimal model $\mathcal{E}$ up to the new atoms introduced by Semi-standard Equational Flattening.

Proof The proof is similar to the proof of Proposition 4.4.9. ■

The following theorem states that the theory of monadic equational types over semi-standard sorted equational theories is decidable. Semi-standard Equational Flattening is an effective abstraction from semi-standard sorted equational theories to non-linear semi-shallow sorted equational theories for which satisfiability of monadic equational types has been shown to be decidable. Note that the class of semi-standard sorted equational theories strictly subsumes the class of standard equational theories (Nieuwenhuis 1996), c.f. Definition 6.2.5.

Theorem 6.8.8
Let $\mathcal{E}$ be a semi-standard sorted equational theory. Then the theory $\mathcal{F}_{\mathcal{E}}$ of monadic equational types over $\mathcal{E}$ is decidable.

Proof Let $\mathcal{E}_F$ be the semi-standard flat approximation of $\mathcal{E}$. By Proposition 6.8.4 $\mathcal{E}_F$ can be effectively computed where $\mathcal{E}_F$ is a semi-shallow sorted equational theory. Due to Theorem 6.7.5 the theory of monadic equational types over semi-shallow sorted equational theories is decidable. Let $\exists\, x_1, \ldots, x_n\, (\Psi)$ be a monadic equational type in $\mathcal{F}_{\mathcal{E}}$ where $\{x_1, \ldots, x_n\}$ is the set of free variables in Ψ. Note that $\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ holds if and only if $T^{\mathcal{E}} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ holds and, by Proposition 6.8.7, if and only if $T^{\mathcal{E}_F} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ holds under the proviso that Ψ does not contain any new atoms introduced by Semi-standard Equational Flattening. It follows that the theory of monadic equational types over semi-standard sorted equational theories is decidable. ■

The following corollary is an immediate consequence of Corollary 6.7.7.

Corollary 6.8.9
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) semi-standard sorted equational theories is EXPTIME-hard

The following corollary is an immediate consequence of Corollary 6.7.8 and Proposition 6.8.4.

Corollary 6.8.10
The satisfiability problem of the theory of monadic equational (variable) types over (alternating) linear semi-standard sorted equational theories is EXPTIME-complete.

We restrict our attention to saturation-based decision procedures and conclude that Semi-standard Equational Flattening is a suitable concept for effective soft typing with respect to arbitrary static/dynamic sorted equational theories and monadic equational types.

Corollary 6.8.11
Let N_0 be a set of clauses. The semi-standard flat approximation of the static sorted equational theory of N_0 is a decidable static approximation for monadic equational types. Let N_0, N_1, N_2, ... be a fair theorem proving derivation. The sequence of semi-standard flat approximations of the dynamic sort theories of each N_i is a decidable dynamic approximation for monadic equational types.

Chapter 7

Typed Equational Theories

We study so-called *typed equational theories* which are a particular form of monadic Horn theories with equality. A typed equational theory can be seen as a type theory where positive occurrences of equations are possible. In particular, a typed equational theory consists of type declarations and so-called *typed equations.* A type declaration strictly generalizes sort declarations such that the non-constraint part of the antecedent may contain monadic literals. Similarly, typed equations correspond to sorted equations with additional monadic literals in the antecedent. In this way, typed equational theories are for the type theories (logic programs) what sorted equational theories are for sort theories, a generalization to equality to a certain degree (positive occurrences). An appropriate representation of type declarations and typed equations are sorted clauses of the form $\Psi \,\|\, \Theta \rightarrow S(t)$ and $\Psi \,\|\, \Theta \rightarrow s \approx t$, respectively, where Ψ is the sort constraint and Θ contains arbitrary monadic atoms.

Definition 7.0.1 (Typed Equation)
A sorted clause $\Psi \,\|\, \Theta \rightarrow s \approx t$ is called a *typed equation* if $\Psi \,\|\, \rightarrow s \approx t$ is a sorted equation and Θ contains monadic atoms only. A typed equation $\Psi \,\|\, \Theta \rightarrow s \approx t$ is called *shallow* (*linear*, *semi-linear*) if the equation $s \approx t$ is shallow (linear, semi-linear).

Definition 7.0.2 (Typed Equational Theory)
A *typed equational theory* $\mathcal{E}$ is a finite set of typed equations and type declarations. $\mathcal{E}$ is called *shallow* (*linear*, *semi-linear*) if the typed equations and type declarations in $\mathcal{E}$ are shallow (linear, semi-linear).

Note that, in general, a sorted equational theory is a typed equational theory but not vice versa. We demonstrate that linear shallow typed equational theories are decidable and that arbitrary monadic Horn theories with equality can be effectively transformed into (linear shallow) typed equational theories. *Equational Type Abstraction* transforms a monadic Horn theory $\mathcal{H}$ into a typed equational theory $\mathcal{E}$ in such a way that the minimal model $T^{\mathcal{E}}$ is an upper approximation of $T^{\mathcal{H}}$. There is, however, a trade-off for decidability as linearity has to be imposed on the positive literals in each clause, similar to the situation of type theories (without equality), c.f. Chapter 5.

Definition 7.0.3
The following abstraction is called *Equational Type Abstraction*:

$$\text{Abstract}\frac{S_1(x_1), \ldots, S_n(x_n), \Theta, \Gamma \to L}{S_1(x_1), \ldots, S_n(x_n) \parallel \Theta \to L}$$

where (i) L is either a monadic atom or an equation, (ii) $S_1(x_1), \ldots, S_n(x_n)$ is maximal such that it contains only monadic atoms of the form $T(x)$ with $x \in \mathit{vars}(L)$, and (iii) Γ contains all non-monadic atoms and equations in the sort constraint of the premise.

Recall that Equational Sort Abstraction may remove certain monadic atoms which occur negatively in the antecedent of a clause. Equational Type Abstraction improves the approximation obtained by Equational Sort Abstraction such that all monadic atoms in the antecedent are kept in the abstracted clause.

Proposition 7.0.4
Let $\mathcal{H}$ be a monadic Horn theory. The result of an exhaustive application of Equational Type Abstraction to $\mathcal{H}$ is a typed equational theory $\mathcal{E}$ where the minimal model $T^{\mathcal{E}}$ of $\mathcal{E}$ is an upper approximation of the minimal model $T^{\mathcal{H}}$ of $\mathcal{H}$.

Proof By the construction of Equational Type Abstraction it follows immediately that any exhaustive application of the rule is finite and results in a typed equational theory. The minimal model $T^{\mathcal{E}}$ is in fact an upper approximation of $T^{\mathcal{H}}$ since all clauses with positive literals are abstracted into $\mathcal{E}$ while only negative literals are removed. ■

In the sequel, we call a typed equational theory $\mathcal{E}$ which has been obtained by an exhaustive application of Equational Type Abstraction to the static or dynamic monadic Horn theory $\mathcal{H}$ of a set N of clauses the *typed equational theory of* N. In order to emphasize that $\mathcal{E}$ has been obtained from $\mathcal{H}$ we may also say that $\mathcal{E}$ is the *static* or *dynamic typed equational theory of* N, respectively. The following corollary is an immediate consequence of Proposition 7.0.4 and Corollary 4.0.7.

Corollary 7.0.5
Let N_0 be a set of clauses. The static typed equational theory of N_0 is a static approximation for monadic equational types. Let N_0, N_1, N_2, ... be a fair theorem proving derivation. The sequence of dynamic typed equational theories of each N_i is a dynamic approximation for monadic equational types.

7.1 Decidability

In order to show that satisfiability of monadic equational types with respect to linear shallow typed equational theories is decidable, we employ basic sorted paramodulation, Type Simplification, Sort Condensing, and Intersection Condensing. Note that Type Simplification moves the non-constraint part of a clause of the form $\Psi \parallel S_1(x_1), \ldots, S_n(x_n) \to \Pi$

to the sort constraint part. In the context of basic sorted paramodulation, the sort constraint essentially represents a particular part of a clause in which each term occurs at a substitution position, i.e. each term in the sort constraint is blocked for paramodulation. Since $S_1(x_1), \dots, S_n(x_n)$ contains only variable positions, Type Simplification does not affect the refutational completeness and, in particular, does not cause any lifting problems. Similar to the previous decidability proofs, the atom ordering must be compatible with the subterm property. An atom ordering which is induced by an admissible term ordering is compatible with the subterm property. We assume admissible literal orderings of Definition 3.1.48 and admissible clause orderings of Definition 3.1.15 and use the equality selection strategy of Definition 6.3.1. The motivation of this strategy is to prohibit an increase in term depth during a saturation process by basic sorted paramodulation.

We only consider sets of sorted clauses which have been obtained by a transformation of arbitrary sets of clauses using Equational Type Abstraction. Any term in the sort constraint of a sorted clause obtained this way is a variable. In other words, any clause set obtained by Equational Type Abstraction is a schematic set. The saturation of linear shallow typed equational theories by the basic sorted paramodulation calculus is a terminating process. We assume that Type Simplification, Sort Condensing, and Intersection Condensing are eagerly applied. Sort and Intersection Condensing are compatible with the particular redundancy concept which is required for the basic/sorted versions of the calculi.

Lemma 7.1.1
Let $\mathcal{E}$ be a linear shallow typed equational theory. Then $\mathcal{E}$ can be finitely saturated by basic sorted paramodulation, Type Simplification, Sort Condensing, and Intersection Condensing. The productive clauses in the saturated set form a linear shallow sorted equational theory which consists of linear shallow term declarations, trivial subsort declarations, and linear shallow sorted equations with a solved sort constraint.

Proof The idea of the proof is to show that (i) there is a certain subclass of linear shallow monadic Horn theories with equality which includes $\mathcal{E}$ and which is closed under basic sorted paramodulation and (ii) the conclusion of each inference by Sort Constraint Resolution or Basic Paramodulation Left/Right is either (ii.a) strictly smaller than the main premise with respect to the lexicographic combination of the multiset of all term depths, the number of non-substitution positions, and the number of monadic atoms in the sort constraint and the antecedent, or else (ii.b) a clause of the form $T_1(t), \dots, T_n(t), S_1(x_1), \dots, S_m(x_m) \,\|\, \rightarrow L$ where t is a non-variable linear shallow term and L is either a monadic atom $S(t)$ or a disjoint equation $s \approx t$ with s is a linear shallow term. Note that there are only finitely many clauses of the form of case (ii.b) over a finite signature up to Sort Condensing, Intersection Condensing, and variant clauses. For (i) we shall argue that the following class of clauses is closed under basic sorted paramodulation and Type Simplification.

$$\Psi_1, \dots, \Psi_n, S_1(x_1), \dots, S_m(x_m) \,\|\, \Theta \rightarrow L$$

where n and m are possibly zero, each Ψ_i is of the form $T_1(t_i), \dots, T_{k_i}(t_i)$ where each t_i is a non-variable term, Θ contains monadic atoms of the form $S(t)$ with t is a non-variable term, L is either a monadic atom $S(u)$ or an equation $u \approx v$, and u and v are linear shallow terms. Note that the constraint $S_1(x_1), \ldots, S_m(x_m)$ does not have to be solved. We call a clause of the above form the *invariant*.

The class of invariants includes any linear shallow typed equational theory. Any inference that involves an invariant as the side premise is only possible if $n = 0$, Θ is empty, and the sort constraint of the invariant is solved, i.e. if the invariant is a linear shallow term declaration, a trivial subsort declaration, or a linear shallow sorted equation with a solved sort constraint. Note that the eager application of Type Simplification guarantees that a clause of the form $\Psi \,\|\, S_1(x_1), \ldots, S_m(x_m) \to \Pi$ is simplified to a clause of the form $\Psi, S_1(x_1), \ldots, S_m(x_m) \,\|\, \to \Pi$. Thus an invariant that is involved as the main premise may only be reduced to another invariant by a linear shallow term declaration, a trivial subsort declaration, or a linear shallow sorted equation with a solved sort constraint which implies (i) by a case analysis similar to the proof of Lemma 6.3.7.

In order to show (ii) we use variable abstraction. Recall that using basic sorted paramodulation we may abstract from the right hand sides of the rewrite rules. That is, not only variable positions can be marked as substitution positions but also the position of the redex when the right hand side has been inserted. In the proof of Lemma 5.2.5, Sort Constraint Resolution (Type Resolution) has already been shown to be monotone in the sense of case (ii). Note that Type Resolution is generalized by (Basic) Paramodulation Left.

Consider a Sort Constraint Resolution inference from an invariant and a linear shallow term declaration. If the invariant is a non-trivial subsort declaration or a disjoint collapsing/universal sorted equation then the conclusion is a clause of the form of case (ii.b). In all other cases, the conclusion is a clause with a strictly smaller multiset of all term depths which implies (ii.a). On the other hand, an inference from any invariant and a trivial subsort declaration decreases the number of monadic atoms in the sort constraint by one.

Suppose that C is the main premise of the form of the invariant in a Basic Paramodulation Left inference where Φ denotes the sort constraint of C, $\Theta = S(u[s']), \Theta'$, and s' does not occur at a substitution position. By the equality selection strategy $S(u[s'])$ is selected. Let C' be the side premise $\Psi \,\|\, \to s \approx t$ where Ψ is solved and $s \approx t$ is a linear shallow equation. We may assume that s is not a variable x, for otherwise either a selected literal $S(x)$ occurs in Ψ, or else a clause $\|\, \to x \approx y$ can be derived from C'. The conclusion D is a clause $\Psi\sigma, \Phi \,\|\, S(u[\boxed{t\sigma}]), \Theta' \to L$ of the form of the invariant. Note that since s is a linear shallow non-variable term, the most general unifier between s and s' can be represented by a minimal substitution σ with $Dom(\sigma) \subseteq vars(s)$. Note also that the number of non-substitution positions in D is strictly smaller than in C. If $u[t\sigma]$ is a variable and Θ' contains only monadic atoms of the form $S(x)$ then D will be simplified by Type Simplification to the clause $\Psi\sigma, \Phi, S(u[t\sigma]), \Theta' \,\|\, \to L$.

Suppose that C is the main premise of the form of the invariant in a Basic Paramodulation Left inference where Φ denotes the sort constraint of C and $\Theta = S(u), \Theta'$. By the equality selection strategy $S(u)$ is selected. Let C' be the side premise $\Psi \,\|\, \to S(s)$ where Ψ is solved and s is a linear shallow term. The conclusion D is a clause $\Psi\sigma, \Phi \,\|\, \Theta' \to L$

of the form of the invariant. Similarly as above, since s is linear shallow and u is not a variable, the most general unifier between s and u can be represented by a minimal substitution σ with $Dom(\sigma) \subseteq vars(s)$. Note again that the number of non-substitution positions in D is strictly smaller than in C.

Suppose that C is the main premise of the form of the invariant in a Basic Paramodulation Right inference where Φ is the solved sort constraint of C, i.e., in particular, $n = 0$, and Θ is empty. Furthermore, suppose that L is of the form $L[s']$ where s' does not occur at a substitution position. Let C' be the side premise $\Psi \,\|\, \rightarrow s \approx t$ where Ψ is solved and $s \approx t$ is a linear shallow equation. The conclusion D is a clause $\Psi\sigma, \Phi \,\|\, \rightarrow L[\boxed{t\sigma}]$ of the form of the invariant where $n \leq 1$. Note that $L[t\sigma]$ is again a linear shallow atom or equation, c.f. Lemma 6.3.7. Note also that since s is linear shallow, the most general unifier between s and s' can be represented by a minimal substitution σ with $Dom(\sigma) \subseteq vars(s)$. Note again that the number of non-substitution positions in D is strictly smaller than in C. However, in this case, we even do not need the variable abstraction argument for termination since L also contains only linear shallow terms. That is, termination also follows by a combinatorial argument on the distinct forms of linear shallow terms. In sum, we have that (ii) holds.

The saturated set may still contain clauses with an unsolved sort constraint and/or non-empty antecedents. However, these clauses cannot be productive which implies that the set of productive clauses are term declarations, trivial subsort declarations, and sorted equations with a solved sort constraint which form a linear shallow sorted equational theory. ■

The following proposition states that, similar to (non-linear) shallow sorted equational theories, saturation by basic sorted paramodulation transforms a linear shallow typed equational theory in simply exponential time into an equivalent linear shallow sorted equational theory.

Proposition 7.1.2
Let $\mathcal{E}$ be a linear shallow typed equational theory. Then $\mathcal{E}$ can be finitely saturated by basic sorted paramodulation, Type Simplification, Sort Condensing, and Intersection Condensing in simply exponential time with respect to $size(\mathcal{E})$ resulting in a typed equational theory $\mathcal{E}'$ such that $size(\mathcal{E}')$ is at most simply exponentially larger than $size(\mathcal{E})$.

Proof We follow the proof of Proposition 5.2.6. It is important to see that the number of clauses may increase at most simply exponential where each new clause is polynomially bound in size. Given a linear shallow typed equational theory $\mathcal{E}$ over a signature Σ, we may assume that Σ contains only the function symbols which occur in clauses in $\mathcal{E}$. Note that the saturation by sorted paramodulation is similar to the saturation of a linear shallow type theory by type resolution. We only have to consider the additional inferences which involve sorted equations.

Let m_f be the number of function symbols in Σ. Let m be the maximal size $size(\Theta)$ of the clauses of the form $\Psi \,\|\, \Theta \rightarrow L$ in $\mathcal{E}$, let k_a be the maximal arity among all arities of the function symbols in Σ, and let q be the number of distinct monadic predicate symbols which occur in the clauses in $\mathcal{E}$. We observe that the size of the non-constraint antecedents in the clauses does not grow during saturation. More precisely, a Basic Paramodulation Left inference is possible only once on each subterm which occurs in the non-constraint antecedent part of a clause. Thus there are at most linearly many derivable clauses in the size m with respect to a single linear shallow sorted equation and $m * c * m_f * k_a * 2^q$ derivable clauses with respect to all possible linear shallow sorted equations over Σ and the q monadic predicates where c is some constant which refers to the distinct forms of equations. The number of clauses derivable by Basic Paramodulation Right inferences can be estimated in a similar way. ■

Similar to the saturated (non-linear) shallow sorted equational theories which are obtained from saturation by sorted superposition/paramodulation, a saturated typed equational theory can be further transformed by State Union and De-Alternation into an equivalent linear shallow sorted equational theory in which all term declarations are non-alternating and all sorted equations have non-alternating sort constraints by Corollary 6.3.13. Unifiability with respect to finitely saturated linear shallow typed equational theories is decidable following similar techniques which have been used for (non-linear) shallow sorted equational theories. Note that the productive clauses in the saturated set form a linear shallow sorted equational theory.

Theorem 7.1.3
Unifiability with respect to finitely saturated linear shallow typed equational theories is decidable.

Proof Two arbitrary terms u and v are unifiable if and only if we can derive the empty clause from the saturated theory and the goal clause $\| u \approx v \rightarrow$. Since the linear shallow typed equational theory is saturated, no inferences inside the theory need to be considered. Furthermore, the goal is purely negative, so we can delete all clauses with an unsolved sort constraint or a non-empty antecedent from the saturated theory. Since the result is actually a linear shallow sorted equational theory, we may proceed exactly as in the proof of Theorem 6.3.15. ■

The next theorem summarizes the discussion and states that the theory of monadic equational types over linear shallow typed equational theories is decidable. Any linear shallow typed equational theory may be finitely saturated by basic sorted paramodulation such that the candidate model of the saturation is the minimal model of the theory. By a subsequent saturation with basic sorted paramodulation between the saturated theory and a negated monadic equational type we can effectively compute the satisfiability of the type with respect to the theory. A negated monadic equational type is a negative clause and thus any clause with an unsolved sort constraint or a non-empty antecedent may safely be removed from the saturated theory. For the satisfiability test of monadic equational types with respect to a linear shallow typed equational theory, we arrive at the same test with respect to linear shallow sorted equational theories.

Theorem 7.1.4
Let $\mathcal{E}$ be a linear shallow typed equational theory. The theory $\mathcal{F}_{\mathcal{E}}$ of monadic equational types over $\mathcal{E}$ is decidable.

Proof The proof is similar to the proof of the Theorem 6.3.17. By Lemma 7.1.1 $\mathcal{E}$ can be finitely saturated. Moreover, since a monadic equational type is encoded as a negative clause we can delete all clauses with unsolved sort constraints or non-empty antecedents from the saturated set. The result is a linear shallow sorted equational theory. ■

By observing that a linear shallow typed equational theory together with a negated monadic equational type can still be saturated in simply exponential time, we obtain that the satisfiability problem of the theory of monadic equational types over linear shallow typed equational theories is EXPTIME-complete.

Proposition 7.1.5
The satisfiability problem of the theory of monadic equational (variable) types over linear shallow typed equational theories is EXPTIME-complete.

Proof The EXPTIME-hardness follows from the EXPTIME-completeness of the satisfiability problem of the theory of monadic variable types over (alternating) linear shallow sorted equational theories. We show that the problem is in EXPTIME. Given a linear shallow typed equational theory $\mathcal{E}$ and a monadic equational type $\exists\, x_1, \ldots, x_n\, (\Psi)$ from the theory over $\mathcal{E}$, we add the (expansion) clause $\Psi \,\|\, \rightarrow S_{\mathrm{f}}(x)$ to $\mathcal{E}$ where x is a new variable and transform the result in simply exponential time into a linear shallow sorted equational theory $\mathcal{E}'$ due to Proposition 7.1.2 and Corollary 6.3.13. We may assume that S_{f} does not occur in $\mathcal{E}$. It is important to see that Equality Resolution involves only a polynomial unifiability test on the equations in Ψ. Hence, we may apply the previously mentioned proposition and corollary in this more general setting.

By Proposition 4.2.5, the non-equational part of $\mathcal{E}'$ can be transformed in polynomial time into an equivalent tree automaton $\mathcal{A}$ where we assume that S_{f} is the only final state of $\mathcal{A}$. Then $\mathcal{E} \models \exists\, x_1, \ldots, x_n\, (\Psi)$ if and only if $L(\mathcal{A}) \neq \emptyset$. Note that the non-emptiness test of tree automata is decidable in linear time. ■

From Corollary 6.3.22, we obtain that there are at most simply exponentially many "well-sorted" most general unifiers for a unification problem with respect to a finitely saturated linear shallow typed equational theory.

Corollary 7.1.6
The number of "well-sorted" most general unifiers with respect to a finitely saturated linear shallow typed equational theory $\mathcal{E}$ is simply exponential.

7.2 Linear Approximations

The theory of monadic equational types over linear shallow typed equational theories is decidable. In correspondence to the transformation of type theories into shallow type theories, we can improve this result using an extension of Typed Flattening to sorted monadic Horn clauses with equality called *Typed Equational Flattening.* The transformation introduces a new monadic predicate for each occurrence of a proper non-variable subterm in a type declaration or a typed equation where the number of new symbols is linear in the number of function symbols which occur in the equational theory. Typed Equational Flattening replaces in a top-down manner each occurrence of a non-variable proper subterm t_i by a variable x where an additional sort constraint on x restricts the ground instances of x to the "well-sorted" ground instances of t. The minimal model of the resulting shallow typed equational theory is an upper approximation of the minimal model of the original theory. Typed Equational Flattening abstracts from non-linear non-shallow occurrences of variables.

Definition 7.2.1
The following abstraction is called *Typed Equational Flattening*:

$$\text{Abstract}\ \frac{\Phi, \Psi, \Psi' \parallel \Theta \to L[t]_p}{\begin{array}{r}\Psi' \parallel \Theta \to S_t(t)\\ S_t(x), \Phi, \Psi, \Psi'' \parallel \Theta \to L[p/x]\end{array}}$$

where (i) L is either a monadic atom or an equation, (ii) p is a position with $|p| = 2$, (iii) Φ contains all new atoms of the form $S_{t'}(y)$, (iv) Ψ, Ψ' is maximal such that Ψ, Ψ' is solved in the premise, (v) t is a non-variable term, (vi) Ψ' is maximal such that $vars(\Psi') \subseteq vars(t)$, (vii) $\Psi'' \subseteq \Psi'$ is maximal such that Ψ'' is solved in the conclusion, (viii) S_t is a new monadic predicate, and (ix) x is a new variable.

In the sequel, a monadic predicate S_t refers to the new predicate which has been introduced by Typed Equational Flattening for a non-variable term t. The newly introduced declarations represent, from an automata-theoretic point of view, tree automata which recognize in a state S_t exactly the "well-sorted" ground instances of the original term t. The following proposition suggests that Typed Equational Flattening may be applied to arbitrary typed equational theories. The transformation yields a theory whose minimal model is an upper approximation of the minimal model of the original theory.

Proposition 7.2.2
Let $\mathcal{E}$ be a typed equational theory. Exhaustive application of Typed Equational Flattening to $\mathcal{E}$ terminates and results in a (non-linear) shallow typed equational theory.

Proof The proof is similar to the proof of Proposition 5.3.3. ■

In the sequel we call a typed equational theory which has been obtained by an exhaustive application of Typed Equational Flattening to a typed equational theory $\mathcal{E}$ the *flat* *approximation of $\mathcal{E}$*. Typed Equational Flattening does not *linearize* the shallow occurrences of non-linear variables in the positive atoms or equations in each clause of a theory. However, for the decidability of shallow typed equational theories, it is difficult to avoid the *Equational Linearization* of the positive atoms and equations. Note that the sorted unification problem with respect to arbitrary sort theories, which is undecidable, can be reduced to the same problem already with respect to non-linear shallow type theories. Observe that Equational Linearization improves upon a naive renaming by additional "sort" information for each renamed variable. However, Equational Linearization differs from Linearization in Definition 5.3.8 only in case (i).

Definition 7.2.3
The following abstraction is called *Equational Linearization*:

$$\text{Abstract}\ \frac{\Psi, \Psi' \parallel \Theta \to L[x]_p}{\begin{array}{c} S_{x_1}(x_1), \dots, S_{x_n}(x_n), \Psi, \Psi' \parallel \Theta \to L[p_1/x_1, \dots, p_n/x_n] \\ \Psi' \parallel \Theta \to S_{x_1}(x) \\ \vdots \\ \Psi' \parallel \Theta \to S_{x_n}(x) \end{array}}$$

where (i) L is either a monadic atom or an equation, (ii) x is a non-linear variable in L at position p, (iii) the positions $p_1, \dots, p_n$ refer to all other positions of x in L, (iv) Ψ is maximal such that $x \notin \mathit{vars}(\Psi)$, (v) each S_{x_i} is a new monadic predicate, and (vi) the $x_1, \dots, x_n$ do not occur in the premise.

The following proposition suggests that Equational Linearization may be applied to arbitrary typed equational theories. However, for a *linear* flat approximation, we prefer to apply Typed Equational Flattening first, for, in general, a better approximation may be obtained.

Proposition 7.2.4
Let $\mathcal{E}$ be a typed equational theory. Exhaustive application of Equational Linearization to $\mathcal{E}$ terminates and results in a linear typed equational theory.

Proof The proof is similar to the proof of Proposition 5.3.9. ■

In the sequel we call a typed equational theory which has been obtained by an exhaustive application of Typed Equational Flattening to a typed equational theory $\mathcal{E}$ followed by an exhaustive application of Equational Linearization to $\mathcal{E}$ the *linear flat approximation of $\mathcal{E}$*. The following lemma on the minimal model properties of the new predicates introduced by Typed Equational Flattening corresponds to the technical Lemma 5.3.10 for Typed Flattening/Linearization.

Lemma 7.2.5
Let $\mathcal{E}$ be a monadic Horn theory without equality and let $\mathcal{E}_F$ be the linear flat approximation of $\mathcal{E}$. Let I be the minimal model of $\mathcal{E}_F$. Let t be a non-variable term for which a

new monadic predicate symbol S_t has been introduced by Typed Equational Flattening. Let C be the immediate conclusion $\Psi, \Theta \,\|\, \rightarrow S_t(t)$ of Typed Equational Flattening on t where Ψ is maximal such that Ψ is solved. Then for all ground atoms $S_t(s)$ which are true in I there is a ground substitution σ such that (i) $I \models s = t''\sigma$ where t'' is the linear renaming of t and (ii) for all variables $x \in \mathit{vars}(t'')$ there is a ground substitution λ such that $(\Psi, \Theta)\sigma|_x\lambda$ is true in I.

Proof The proof is similar to the proof of Lemma 5.3.10. ■

Typed Equational Flattening combined with Equational Linearization abstract typed equational theories such that the minimal models of the resulting approximations are upper approximations of minimal models of the original theories.

Proposition 7.2.6
Let $\mathcal{E}$ be a typed equational theory and let $\mathcal{E}_F$ be the linear flat approximation of $\mathcal{E}$. The minimal model $T^{\mathcal{E}_F}$ is an upper approximation of the minimal model $T^{\mathcal{E}}$.

Proof The proof is similar to the induction proof of Proposition 4.4.8 since the hypothesis extends to include also the non-constraint antecedent part of the clauses in $\mathcal{E}$. This part is completely inherited by Typed Equational Flattening. The prerequisites for an appropriate proof are contained in the proof of part (i) of Lemma 5.3.10. ■

We restrict our attention to saturation-based decision procedures and conclude that Typed Equational Flattening/Linearization is a suitable concept for effective soft typing with respect to arbitrary static/dynamic typed equational theories and monadic equational types.

Corollary 7.2.7
Let N_0 be a set of clauses. The linear flat approximation of the static typed equational theory of N_0 is a (decidable) static approximation for monadic equational types. Let N_0, N_1, N_2, $\ldots$ be a fair theorem proving derivation. The sequence of linear flat approximations of the dynamic typed equational theories of each N_i is a (decidable) dynamic approximation for monadic equational types.

Chapter 8

Conclusions

The purpose of this thesis is (i) to obtain a comprehensive understanding of *semantically* guided theorem proving and (ii) to provide decidable fragments of first-order logic (with equality) which are suitable for an *effective* approximation of proof guidance by semantics.

The main contribution with respect to (i) is the general framework of *soft typing for clausal inference systems*. Soft typing controls theorem proving derivations from clausal inference systems by a *blocking* mechanism according to the validity/satisfiability of the clauses with respect to certain model hypotheses. Any clausal inference system which enjoys the so-called (*strong*) *reduction property* is compatible with this concept. The strong reduction property also implies the compatibility with a general concept of *simplification* based on a certain notion of *redundancy*. These concepts play a central role in all modern automated theorem proving systems.

We have shown that soft typing is a suitable concept for semantically guided theorem proving in terms of (refinements of) resolution and superposition (with selection) (Bachmair & Ganzinger 1994). Soft typing for semantic resolution explains the refutational completeness of semantic resolution (Slagle 1967) and clarifies the theoretical background of *ordered semantic hyper-linking* (Plaisted 1994, Plaisted & Zhu 1997). In particular, we have demonstrated that the saturation criterion of ordered semantic hyper-linking is an instance of the saturation criterion of soft typing for *semantic* resolution. Soft typing for *ordered* resolution, on the other hand, covers the theoretical aspects of the SATCHMO theorem prover (Manthey & Bry 1988) and its efficient counterpart, the *model generation theorem prover* MGTP (Hasegawa, Fujita & Koshimura 1997).

The effectiveness of soft typing for ordered resolution and superposition requires decidable approximations of certain model hypotheses. We propose the use of sets of Horn clauses to represent (approximations of) certain model hypotheses for sets of clauses in the spirit of Frühwirth et al. (1991). Approximations are obtained by inferring *automatically* abstractions from sets of clauses that result in sets of Horn clauses. The satisfiability of clauses with respect to the approximations implies then the satisfiability in the original model hypotheses.

As the main contribution for (ii) we have demonstrated the decidability of the satisfiability problem of the class of *monadic equational types* with respect to the following equational theories. Monadic equational types are existentially closed conjunctions of monadic atoms and equations.

- *semi-linear sorted equational theories* (Jacquemard et al. 1998*a*) which strictly embed the (non-linear) shallow equational theories in (Comon et al. 1994),
- *semi-standard sorted equational theories* (Jacquemard et al. 1998*b*) which strictly generalize the standard (equational) theories in (Nieuwenhuis 1996), and
- *linear shallow typed equational theories* which extend the monadic Horn theories in (Weidenbach 1999) by equality.

All theories are syntactically characterized as certain sets of Horn clauses (with equality) which allows the automatic abstraction of arbitrary sets of clauses into the theories. The satisfiability problem has been shown to be EXPTIME-complete for linear (semi-standard) sorted equational theories and linear shallow typed equational theories and EXPTIME-hard for (non-linear) semi-linear and semi-standard sorted equational theories. Note that the E-unifiability problem is a special case of the satisfiability problem of (monadic) equational types. In contrast to the decidability of sorted unification in pseudo-linear *sort* theories (Weidenbach 1996*b*), we have shown that the word problem is undecidable already in pseudo-linear *equational* theories.

We have also reported on the complexity of the satisfiability problem of monadic types with respect to sort theories and so-called type theories. In particular, we have shown the EXPTIME-completeness for alternating linear (shallow) sort theories and linear shallow type theories and the EXPTIME-completeness of the non-emptiness problem of semi-linear sort theories. Moreover, the satisfiability problem with respect to alternating non-linear shallow and semi-linear sort theories has been shown to be EXPTIME-hard whereas the inclusion in EXPTIME remains open. It is not clear whether the combination of alternation and non-linearity involves additional exponential increase in complexity.

8.1 Future Work

Soft typing for clausal inference systems is a general framework to incorporate semantics into automated theorem proving processes. The impact of soft typing seems to be determined by two aspects:

1. Does the abstracted theory describe a non-trivial structure?
2. If so, does this structure sufficiently reflect the structure of the original problem?

Efficiency. From a theoretical point of view, we have addressed the first question by demonstrating the decidability of non-trivial (equational) theories which can automatically be obtained by an effective abstraction from arbitrary sets of clauses (with equality). The

Problem	Status	Derived	Kept	Deleted using $\mathcal{S}$
5-1-154-3-2	satisfiable	929	233	7
4-7-2-2-0	satisfiable	1156	400	369
4-9-2-2-0	unsatisfiable	139	61	37
8-1-10-4-2	satisfiable	867	276	349
8-1-8-4-2	satisfiable	111	49	42
5-2-80-3-2	unsatisfiable	32	12	0
5-1-85-3-2	unsatisfiable	15	15	0

Table 8.1: Experiments with static sort theories

current status of implementation is that soft typing with respect to *static* sort theories has been implemented in the theorem prover Spass (Weidenbach et al. 1998). A static theory is obtained once from the original problem set at the beginning of a theorem proving derivation and remains admissible throughout the derivation.

Table 8.1 provides experimental evidence about the effects of blocking based on a static sort theory $\mathcal{S}$ (Ganzinger et al. 1997). Spass has been tested on several hundred knowledge bases written in $\mathcal{ALC}$. The language $\mathcal{ALC}$ is a notational variant of multi-modal propositional logic K that can be translated into first-order logic such that resolution with subsumption and condensing becomes a decision procedure for the resulting fragment of first-order clause logic (Schmidt 1997). Hence, any resolution-based prover with subsumption and condensing can be used as a decision procedure for this class.

The deletion of clauses which have a blocked monadic type with respect to $\mathcal{S}$ is one of the key techniques to make resolution an efficient decision procedure on these problems. The table shows the names of several representative problems, the number of derived clauses, kept clauses, and the number of clauses deleted as a result of a blocked monadic type in $\mathcal{S}$. These clauses cannot be deleted by usual redundancy criteria. The problems become significantly harder without the deletion of clauses which have a blocked monadic type (at least a factor of 2, except for the "easy" final two examples).

We also introduced a theoretical concept for *dynamic* theories which have to be revised at every step in a theorem proving derivation. Neither a dynamic version of the sort theories nor the decidable equational theories have been implemented yet. At least the implementation of dynamic versions of the (equational) theories requires the analysis of efficient representations. An implementation of soft typing with respect to (dynamic versions of) the (equational) theories may require a more involved representation by, e.g., *binary decision diagrams* (Bryant 1986, Bryant 1992), to deal with the exponential time complexity.

Constructor-based equational theories. The equations in semi-linear (sorted) equational theories involve non-linear occurrences of variables only at the same depth. This requirement can be relaxed using so-called constructor-based alphabets. A constructor-based alphabet is a signature in which function symbols and so-called constructors are distinguished. The occurrences of function symbols and constructors in the equations of a constructor-based equational theory are then subject to certain restrictions. Decidable constructor-based equational theories, as investigated by, e.g., Fassbender & Maneth (1996) and Limet & Réty (1997), are possible candidates for effective soft typing.

Non-monadic predicates. Beside equational theories, we may also consider decidable fragments of first-order logic which involve non-monadic predicates. For example, we have studied first-order fragments which are obtained by the (relational) translation of propositional modal logics. In (Ganzinger, Hustadt, Meyer & Schmidt 1999) we have shown how ordered chaining (Bachmair & Ganzinger 1995) may be used to obtain resolution decision procedures for the relational translation of a range of multi-modal propositional logics with transitive accessibility relations such as the logics K4 and S4. However, for soft typing, suitable abstractions of arbitrary sets of clauses into the decidable fragments are required which, in particular, preserve certain validity and/or satisfiability properties.

Andréka, van Benthem & Németi (1996) proposed another interesting decidable fragment of first-order logic called the *guarded fragment* which also captures a large portion of multi-modal propositional logics. In (Ganzinger, Meyer & Veanes 1999) we have studied certain two-variable variants of the guarded fragment which still enjoy a sufficient expressiveness and, unlike the full guarded fragment (Grädel 1998), remain decidable if additional closure properties of the accessibility relation, such as transitivity, are involved. In particular, we have shown, using Rabin's tree theorem (Rabin 1969), that the *monadic* two-variable guarded fragment with binary relations that are transitive, symmetric and/or reflexive, is decidable. Any non-monadic predicate in the monadic fragment appears as a so-called *guard*. The monadic two-variable guarded fragment strictly embeds the first-order fragment considered in (Ganzinger, Hustadt, Meyer & Schmidt 1999).

Two open problems arise in order to exploit these results for soft typing. Similar to the previously mentioned fragment, suitable abstractions have to be studied to obtain the monadic two-variable guarded fragment (with additional relations) from arbitrary sets of clauses under the proviso of certain model-theoretic aspects. Secondly, the current decidability proof does not give a clear complexity bound. A careful analysis may reveal a simpler and sufficient representation which allows more efficient, probably resolution-based decision procedures for the decidable fragment. However, there are a lot more possibilities to obtain effective soft typing which cannot be covered here.

Adequacy. The second question whether a particular approximation is actually appropriate to describe the underlying mathematical structure of a clause set, is another open problem at least in the context of automatic detection of adequate approximations. It is difficult to recognize mathematical structures by syntactic properties of their formalization. An indirect solution to this problem could be to relax the abstraction in a way that the resulting approximation is closer to the original. An efficient inference system in combination with powerful simplification techniques may still terminate with a high probability on a given class of approximations, although this class may be undecidable.

Bibliography

Andréka, H., van Benthem, J. & Németi, I. (1996), Modal languages and bounded fragments of predicate logic, ILLC Research Report ML-1996-03, University of Amsterdam.

Apt, K. R. (1990), Logic programming, *in* J. van Leewen, ed., 'Handbook of Theoretical Computer Science', Vol. B: Formal Models and Semantics, The MIT Press, New York, N.Y., chapter 10, pp. 493–574.

Baader, F. & Nipkow, T. (1998), *Term rewriting and all that*, 1st ed. edn, Cambridge University Press, Cambridge, UK.

Bachmair, L. & Ganzinger, H. (1991), Perfect model semantics for logic programs with equality, *in* K. Furukawa, ed., 'Logic Programming : 8th International Conference ICLP '91', MIT Press, Cambridge, MA, pp. 645–659.

Bachmair, L. & Ganzinger, H. (1994), 'Rewrite-based equational theorem proving with selection and simplification', *Journal of Logic and Computation* **4**(3), 217–247. Revised version of Technical Report MPI-I-91-208, 1991.

Bachmair, L. & Ganzinger, H. (1995), Ordered chaining calculi for first-order theories of binary relations, Technical Report MPI-I-95-2-009, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Bachmair, L. & Ganzinger, H. (1997), Strict basic superposition and chaining, Technical Report MPI-I-97-2-011, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Bachmair, L. & Ganzinger, H. (1998*a*), Equational reasoning in saturation-based theorem proving, *in* W. Bibel & P. Schmitt, eds, 'Automated Deduction: A Basis for Applications', Vol. I, Kluwer, chapter 11, pp. 353–397.

Bachmair, L. & Ganzinger, H. (1998*b*), Strict basic superposition, *in* C. Kirchner & H. Kirchner, eds, 'Proceedings of the 15th International Conference on Automated Deduction, CADE-15', Vol. 1421 of *LNCS*, Springer-Verlag, Lindau, Germany, pp. 160–174.

Bachmair, L., Ganzinger, H., Lynch, C. & Snyder, W. (1992), Basic paramodulation and superposition, *in* D. Kapur, ed., 'Proceedings of the 11th International Conference on Automated Deduction, CADE-11', Vol. 607 of *LNCS*, Springer-Verlag, Saratoga Springs, NY, pp. 462–476.

Bachmair, L., Ganzinger, H., Lynch, C. & Snyder, W. (1995), 'Basic paramodulation', *Information and Computation* **121**(2), 172–192. Revised version of Technical Report MPI-I-93-236, 1993.

Bancilhon, F., Maier, D., Sagiv, Y. & Ullman, J. D. (1986), Magic sets and other strange ways to implement logic programs, *in* 'Symposium on Principles of Database Systems, PODS-86', ACM Press, pp. 1–15.

Beeri, C. & Ramakrishnan, R. (1991), 'On the power of magic', *Journal of Logic Programming* **10**(2), 255–299.

Bogaert, B. & Tison, S. (1992), Equality and disequality constraints on direct subterms in tree automata, *in* A. Finkel & M. Jantzen, eds, 'Proceedings of the Symposion on Theoretical Aspects of Computer Science, STACS-92', Vol. 577 of *LNCS*, Springer-Verlag, pp. 161–172.

Bryant, R. (1986), 'Graph-based algorithms for boolean function manipulation', *IEEE Transactions on Computers* **35**(8), 677–691.

Bryant, R. E. (1992), 'Symbolic boolean manipulation with ordered binary-decision diagrams', *ACM Computing Surveys* **24**(3), 293–318.

Caferra, R. & Peltier, N. (1995*a*), Decision procedures using model building techniques, *in* H. Kleine Büning, ed., 'Computer Science Logic : International workshop (CSL-9); Proceedings of the Conference Computer Science Logic, CSL-95', Vol. 1092 of *LNCS*, Springer-Verlag.

Caferra, R. & Peltier, N. (1995*b*), Extending semantic resolution via automated model building: applications, *in* C. S. Mellish, ed., 'Proceedings of the 14th International Joint Conference on Artificial Intelligence', Morgan Kaufmann, pp. 328–334.

Caferra, R. & Zabel, N. (1990), Extending resolution for model construction, *in* 'Proceedings of JELIA-90, Workshop on Logics in AI', Vol. 478 of *LNCS*, Springer-Verlag, pp. 153–170.

Caferra, R. & Zabel, N. (1992), 'A method for simultaneous search for refutations and models by equational constraint solving', *Journal of Symbolic Computation* **13**(6), 613–641.

Cartwright, R. & Fagan, M. (1991), Soft typing, *in* B. Hailpern, ed., 'Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation', ACM Press, Toronto, ON, Canada, pp. 278–292.

Cartwright, R. & Felleisen, M. (1996), 'Program verification through soft typing', *ACM Computing Surveys* **28**(2), 349–351.

Chandra, A. K., Kozen, D. C. & Stockmeyer, L. J. (1981), 'Alternation', *Journal of the ACM* **28**(1), 114–133.

Charatonik, W., McAllester, D., Niwinski, D., Podelski, A. & Walukiewicz, I. (1998), The horn mu-calculus, *in* 'Proceedings of the 13th IEEE Symposium on Logic in Computer Science, LICS-98', IEEE Computer Society Press, pp. 58–69.

Charatonik, W. & Podelski, A. (1998), Set-based analysis of reactive infinite-state systems, *in* B. Steffen, ed., 'Proceedings of the 1st International Conference on Tools and Algorithms for the Construction and Analysis of Systems', Vol. 1384 of *LNCS*, Springer-Verlag, pp. 358–375.

Chu, H. & Plaisted, D. A. (1994*a*), 'Model finding in semantically guided instance-based theorem proving', *Fundamenta Informaticae* **21**(1), 221–235.

Chu, H. & Plaisted, D. A. (1994*b*), Semantically guided first-order theorem proving using hyper-linking, *in* A. Bundy, ed., 'Proceedings of the 12th International Conference on Automated Deduction, CADE-12', Vol. 814 of *LNCS*, Springer-Verlag, Nancy, France, pp. 192–206.

Comon, H. (1995), Sequentiality, second order monadic logic and tree automata, *in* 'Proceedings of the 10th IEEE Symposium on Logic in Computer Science, LICS-95', IEEE Computer Society Press, pp. 508–517.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S. & Tommasi, M. (1997), 'Tree automata techniques and applications', Available on: `http://www.grappa.univ-lille3.fr/tata`.

Comon, H. & Delor, C. (1994), 'Equational formulae with membership constraints', *Information and Computation* **112**, 167–216.

Comon, H., Haberstrau, M. & Jouannaud, J. P. (1994), 'Syntacticness, cycle-syntacticness and shallow theories', *Information and Computation* **111**(1), 154 – 191.

Dershowitz, N. & Jouannaud, J.-P. (1990), *Rewrite Systems*, Handbook of Theoretical Computer Science, North-Holland, pp. 243–309.

Devienne, P., Talbot, J.-M. & Tison, S. (1997), Set-based Analysis for Logic Programming and Tree Automata, *in* P. V. Hentenryck, ed., 'Proceedings of the 4th International Static Analysis Symposium', Vol. 1302 of *LNCS*, Springer-Verlag, pp. 127–140.

Doner, J. (1970), 'Tree acceptors and some of their applications', *Journal of Computer and System Sciences* **4**, 406–451.

Fassbender, H. & Maneth, S. (1996), A strict border for the decidability of e-unification for recursive functions, *in* M. Hanus & M. Rodríguez-Artalejo, eds, 'Algebraic and Logic Programming (ALP-5) : 5th international conference, Aachen, Germany, September 25-27, 1996', Vol. 1139 of *LNCS*, Springer-Verlag, Berlin.

Fermüller, C. & Leitsch, A. (1996), 'Hyperresolution and automated model building', *Journal of Logic and Computation* **6**(2), 173–203.

Frühwirth, T., Shapiro, E., Vardi, M. Y. & Yardeni, E. (1991), Logic programs as types for logic programs, *in* A. R. Meyer, ed., 'Proceedings of the 6th IEEE Symposium on Logic in Computer Science, LICS-91', IEEE Computer Society Press, pp. 300–309.

Ganzinger, H., Hustadt, U., Meyer, C. & Schmidt, R. (1999), A resolution-based decision procedure for extensions of K4, *in* 'Proceedings of AIML-98'. To appear.

Ganzinger, H., Meyer, C. & Veanes, M. (1999), The two-variable guarded fragment with transitive relations, *in* 'Proceedings of the 14th IEEE Symposium on Logic in Computer Science, LICS-99', IEEE Computer Society Press, pp. ??–?? Accepted.

Ganzinger, H., Meyer, C. & Weidenbach, C. (1997), Soft typing for ordered resolution, *in* W. McCune, ed., 'Proceedings of the 14th International Conference on Automated deduction, CADE-14', Vol. 1249 of *LNCS*, Springer-Verlag, pp. 321–335.

Gelernter, H. (1960), Realization of a geometry theorem-proving machine, *in* 'Information Processing. Proceedings of the International Conference on Information Processing UNESCO, Paris, June 15-20, 1959', Oldenbourg-Verlag, München;Paris;London, pp. 273–282.

Giunchiglia, F. & Walsh, T. (1989), Abstract theorem proving, *in* N. Sridharan, ed., 'Proceedings of the 11th International Joint Conference on Artificial Intelligence', Morgan Kaufmann, Detroit, MI, USA, pp. 372–377.

Giunchiglia, F. & Walsh, T. (1992), 'A theory of abstraction', *Artificial Intelligence* **57**, 323–389.

Giunchiglia, F. & Walsh, T. (1993), 'The inevitability of inconsistent abstract spaces', *Journal of Automated Reasoning* **11**(1), 23–41.

Grädel, E. (1998), On the restraining power of guards. To appear in *Journal of Symbolic Logic*.

Hasegawa, R., Fujita, H. & Koshimura, M. (1997), MGTP: A model generation theorem prover: Its advanced features and applications, *in* D. Galmiche, ed., 'Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods', Vol. 1227 of *LNCS*, Springer-Verlag, pp. 1–15.

Hasegawa, R., Inoue, K., Ohta, Y. & Koshimura, M. (1997), Non-horn magic sets to incorporate top-down inference into bottom-up theorem proving, *in* W. McCune, ed., 'Proceedings of the 14th International Conference on Automated Deduction, CADE-14', Vol. 1249 of *LNCS*, Springer-Verlag, Townsville, North Queensland, Australia, pp. 176–190.

Hermann, M. & Salzer, G. (1996), Workshop on term schematizations and their applications, *in* '13th International Conference on Automated Deduction, New Brunswick, NJ (USA)', URL = http://logic.tuwien.ac.at/cade13/.

Jacquemard, F. (1996), Decidable approximations of term rewriting systems, *in* H. Ganzinger, ed., 'Proceedings of the 7th International Conference on Rewriting Techniques and Applications, RTA-96', Vol. 1103 of *LNCS*, Springer-Verlag, New Brunswick, NJ, USA, pp. 362–376.

Jacquemard, F., Meyer, C. & Weidenbach, C. (1998*a*), Unification in extensions of shallow equational theories, *in* 'Proceedings of the 9th International Conference on Rewriting Techniques and Applications, RTA-98', Vol. 1379 of *LNCS*, Springer-Verlag, Tsukuba, Japan, pp. 76–90.

Jacquemard, F., Meyer, C. & Weidenbach, C. (1998*b*), Unification in extensions of shallow equational theories, Technical Report MPI-I-98-2-002, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Joyner Jr., W. H. (1976), 'Resolution strategies as decision procedures', *Journal of the ACM* **23**(3), 398–417.

Kaji, Y., Toru, F. & Kasami, T. (1997), 'Solving a unification problem under constrained substitutions using tree automata', *Journal of Symbolic Computation* **23**, 79–117.

Leitsch, A. (1997), *The resolution calculus*, EATCS texts in theoretical computer science, Springer-Verlag, Berlin.

Limet, S. & Réty, P. (1997), E-unification by means of tree tuple synchronized grammars, *in* M. Dauchet & M. Bidoit, eds, 'Proceedings of the 7th International Conference TAPSOFT'97', Vol. 1214 of *LNCS*, Springer-Verlag, pp. 429–440.

Loveland, D. W. (1969), 'A simplified format for the model elimination theorem proving procedure', *Journal of the ACM* **16**(3), 349–363.

Loveland, D. W., Reed, D. W. & Wilson, D. S. (1995), 'Satchmore: Satchmo with relevancy', *Journal of Automated Reasoning* **14**, 325–351.

Manthey, R. & Bry, F. (1988), Satchmo: A theorem prover implemented in prolog, *in* E. L. R. Overbeek, ed., 'Proceedings of the 9th International Conference on Automated Deduction, CADE-9', Vol. 310 of *LNCS*, Springer-Verlag, pp. 415–434.

Nieuwenhuis (1998), 'Decidability and complexity analysis by basic paramodulation', *INFCTRL: Information and Computation* **147**.

Nieuwenhuis, R. (1996), Basic paramodulation and decidable theories (extended abstract), *in* 'Proceedings of the 11th IEEE Symposium on Logic in Computer Science, LICS-96', IEEE Computer Society Press, pp. 473–482.

Nieuwenhuis, R. & Rubio, A. (1992), Basic superposition is complete, *in* B. Krieg-Brückner, ed., 'ESOP'92, 4th European Symposium on Programming', Vol. 582 of *LNCS*, Springer-Verlag, pp. 371–389.

Nieuwenhuis, R. & Rubio, A. (1995), 'Theorem proving with ordering and equality constrained clauses', *Journal of Symbolic Computation* **19**(4), 321–352.

Ohta, Y., Inoue, K. & Hasegawa, R. (1998), On the relationship between non-horn magic sets and relevancy testing, *in* C. Kirchner & H. Kirchner, eds, 'Proceedings of the 15th International Conference on Automated Deduction, CADE-15', Vol. 1421 of *LNCS*, Springer-Verlag, pp. 333–348.

Otto, F., Narendran, P. & Dougherty, D. J. (1995), Some independence results for equational unification, *in* J. Hsiang, ed., 'Proceedings of the 6th International Conference on Rewriting Techniques and Applications, RTA-95', Vol. 914 of *LNCS*, Springer-Verlag, Kaiserslautern, Germany, pp. 367–381.

Oyamaguchi, M. (1990), 'On the word problem for right-ground term-rewriting systems', *The Transactions of the IEICE* **73**(5), 718–723.

Plaisted, D. A. (1980), Abstraction mappings in mechanical theorem proving, *in* 'Proceedings of the 5th International Conference on Automated Deduction, CADE-5', Vol. 87 of *LNCS*, Springer-Verlag, Les Arcs, France, pp. 264–280.

Plaisted, D. A. (1981), 'Theorem proving with abstraction', *Artificial Intelligence* **16**, 47–108.

Plaisted, D. A. (1986), Abstraction using generalization functions, *in* J. H. Siekmann, ed., 'Proceedings of the 8th International Conference on Automated Deduction, CADE-8', Vol. 230 of *LNCS*, Springer-Verlag, Oxford, UK, pp. 365–376.

Plaisted, D. A. (1994), Ordered semantic hyper-linking, Technical Report MPI-I-94-235, Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Plaisted, D. A., Alexander, G. D., Chu, H. & Lee, S.-J. (1992), Conditional term rewriting and first-order theorem proving, *in* M. Rusinowitch & J.-L. Rémy, eds, 'Conditional Term Rewriting Systems, Third International Workshop', Vol. 656 of *LNCS*, Springer-Verlag, Pont-à-Mousson, France, pp. 257–271.

Plaisted, D. A. & Zhu, Y. (1997), Ordered semantic hyper linking, *in* 'Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)', AAAI Press, pp. 472–477.

Rabin, M. O. (1969), 'Decidability of second-order theories and automata on infinite trees', *Transactions of the American Mathematical Society* **141**, 1–35.

Reiter, R. (1976), 'A semantically guided deductive system for automatic theorem proving', *IEEE Transactions on Computers* **C-25**(4), 328–334.

Robinson, J. (1965), 'A machine-oriented logic based on the resolution principle', *Journal of the ACM* **12**(1), 23–41.

Schmidt, R. A. (1997), Resolution is a decision procedure for many propositional modal logics, Technical Report MPI-I-97-2-002, Max-Planck-Institut für Informatik, Saarbrücken, Germany. Submitted for publication in *J. Automat. Reasoning.* The extended abstract version is to appear in Kracht, M., de Rijke, M., Wansing, H. and Zakharyaschev, M. (eds) (1997), *Advances in Modal Logic 1996*, CSLI Publications.

Seidl, H. (1994), 'Haskell overloading is DEXPTIME-complete', *Information Processing Letters* **52**(2), 57–60.

Slagle, J. R. (1967), 'Automatic theorem-proving with renamable and semantic resolution', *Journal of the ACM* **14**, 687–697.

Tarski, A. (1955), 'A fixed point theorem and its applications', *Pacific J. Math.* pp. 285–309.

Thatcher, J. & Wright, J. (1968), 'Generalized finite automata theory with an application to a decision problem of second-order logic', *Mathematical Systems Theory* **2**(1), 57–81.

Uribe, T. E. (1992), Sorted unification using set constraints, *in* D. Kapur, ed., 'Proceedings of the 11th International Conference on Automated Deduction, CADE-11', Vol. 607 of *LNCS*, Springer-Verlag, Saratoga Springs, NY, pp. 163–177.

Veanes, M. (1997), On Simultaneous Rigid E-Unification, PhD thesis, Computing Science Department, Uppsala University.

Wang, T. C. (1985), Designing examples for semantically guided hierarchical deduction, *in* A. Joshi, ed., 'Proceedings of the 9th International Joint Conference on Artificial Intelligence', Morgan Kaufmann, Los Angeles, CA, pp. 1201–1207.

Wang, T. C. & Bledsoe, W. W. (1987), 'Hierarchical deduction', *Journal of Automated Reasoning* **3**(4), 35–71.

Weidenbach, C. (1996*a*), Computational Aspects of a First-Order Logic with Sorts, Dissertation, Technische Fakultät der Universität des Saarlandes, Saarbrücken, Germany.

Weidenbach, C. (1996*b*), Unification in pseudo-linear sort theories is decidable, *in* M. McRobbie & J. Slaney, eds, 'Proceedings of the 13th International Conference on Automated Deduction, CADE-13', Vol. 1104 of *LNCS*, Springer-Verlag, pp. 343–357.

Weidenbach, C. (1998), Sorted unification and tree automata, *in* W. Bibel & P. H. Schmitt, eds, 'Automated Deduction - A Basis for Applications', Vol. 1 of *Applied Logic*, Kluwer, chapter 9, pp. 291–320.

Weidenbach, C. (1999), Towards an automatic analysis of security protocols in first-order logic, *in* H. Ganzinger, ed., 'Proceedings of the 16th International Conference on Automated Deduction, CADE-16', LNCS, Springer-Verlag.

Weidenbach, C., Meyer, C., Cohrs, C., Engel, T. & Keen, E. (1998), 'Spass v0.77', *Journal of Automated Reasoning* **21**(1), 113–113.

Index