Peter Baumgartner

Theory Reasoning in Connection Calculi

August 25, 1998

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Preface

Certainly, the ability to draw inferences is a central operation in any Artificial Intelligence (AI) system. Consequently, *automated deduction* is one of the most traditional disciplines in AI. One core technique, the *resolution principle* [Robinson, 1965b] nowadays seems to be even standard knowledge of any computer scientist.

Although resolution is particularly well-suited for implementation on a computer, it soon became clear after its invention that even more elaborate techniques are needed to cope with the tremendous search space. One of these techniques is *theory reasoning*, where the idea is to incorporate specialised and *efficient* modules for handling general domain knowledge into automated reasoning. These theory reasoning modules might, for instance, simplify a goal, compute solutions, lookup facts in a database, etc. Also, the most suitable knowledge representation formalism might be used for this.

This book is on extensions of calculi for automated theorem proving towards theory reasoning. It focuses on connection methods and in particular on model elimination. An emphasis lies on the combination of such calculi with theory reasoners to be obtained by a new compilation approach.

Several theory-reasoning versions of connection calculi are defined and related to each other. In doing so, *theory model elimination* (TME) will be selected as a base to be developed further. The final versions are search-space restricted versions of total TME, partial TME and partial restart TME. These versions all are *answer*-complete, thus making TME interesting for logic programming and problem-solving applications.

A theory reasoning system is only operable in combination with an (efficient) background reasoner for the theories of interest. Instead of handcrafting respective background reasoners, a more general approach — linearizing completion — is proposed. Linearizing completion enables the *automatic* construction of background calculi suitable for TME-based theory reasoning systems from a wide range of axiomatically given theories, namely Horn theories.

Technically, linearizing completion is a device for combining the unit-resulting strategy of resolution with a goal-oriented, linear strategy à la Prolog in a refutationally complete way.

Finally, an implementation extending the PTTP technique (Prolog based Technology Theorem Proving) towards theory reasoning is described, and the usefulness of the methods developed in this text will be experimentally demonstrated.

This book was written during my employment[1] as a research assistant at the institute for computer science at the University Koblenz-Landau (Germany). It is the revised version of a doctoral dissertation submitted and accepted at this institute.

I would like to thank all those who contributed in this work in one way or the other:

In the first place I am grateful to Ulrich Furbach. He conducted me to scientific work, and he always found time for helpful discussions. His advice and criticisms pushed me ahead quite a few times.

Further I want to thank the other reviewers of my thesis, which are Peter Schmitt (University of Karlsruhe, Germany), David Plaisted (University of North Carolina, USA), as well as Karin Harbusch (University of Koblenz-Landau, Germany), who substituted Prof. Plaisted in my thesis defence talk.

It was really a great pleasure for me to spend the last years in company with nice colleagues. Jürgen Dix, Frieder Stolzenburg und Stefan Brüning (Technische Hochschule Darmstadt, Germany) furthermore read this lengthy text. Several improvements resulted from their suggestions.

Several bright students supported me a lot through implementational work and discussions. Never tired to turn my ideas into running code had been Katrin Erk, Michael Kühn, Olaf Menkens, Dorothea Schäfer and Bernd Thomas.

Koblenz, August 1998 *Peter Baumgartner*

[1] Funded by the DFG (Deutsche Forschungsgemeinschaft) within the research programme "Deduction".

Contents

1. Introduction

1.1 Theory Reasoning: Motivation

Certainly, the ability to draw inferences is a central operation in any Artificial Intelligence (AI) system. Consequently, *automated deduction* is one of the most traditional disciplines in AI. A major goal of research here is to develop general algorithms or semi-decision procedures for the solution of problems formulated in first-order predicate logic. One core technique, the *resolution principle* [Robinson, 1965b] nowadays seems to be even standard knowledge of any computer scientist. Resolution made popular the use of *clause logic* and of *unification* in theorem proving calculi. In particular, the use of unification is a major point, as calculi were predominated at that time by reducing first order logic to propositional logic by more or less blindly guessing ground instantiations of formulae.

But soon after the invention of the resolution principle it became clear that even more elaborate techniques are needed to cope with the tremendous search space. One of these techniques is *theory reasoning*, where the idea is to incorporate specialised and efficient modules for handling general domain knowledge into automated reasoning. These theory reasoning modules might, for instance, simplify a goal, compute solutions, lookup facts in a database, etc. Also, the most suitable knowledge representation formalism might be used for this.

Let us take a more technical point of view: suppose as given a set F of predicate logic formulas and a predicate logic formula G. We would like to know if G follows from the F, i.e. whether $F \models G$ holds. In principle, any complete predicate logic calculus can be used to semi-decide this question. In a refined situation, G is of the form $\exists \boldsymbol{x} G'$ (for some formula G'), and the task is to compute *solutions* for the variables $\boldsymbol{x}$ of the problem G. The calculi developed in this book are *answer complete*, and hence allow to compute, in a sense, all of them.

The emphasis in this book, however, is on *theory reasoning*: In a theorem prover supporting theory reasoning, one can think of F as partitioned in $F = \mathcal{T} \cup F_G$ (read $\mathcal{T}$ as "theory"), and $\mathcal{T}$ would be handled by a specialised "background reasoner $\vdash_{\mathcal{T}}$", while F_G and G would be treated by a general first-order "foreground reasoner $\vdash$". Typically, these parts heavily interact in the combined system $\vdash_{\vdash_{\mathcal{T}}}$ (cf. Figure 1.1).

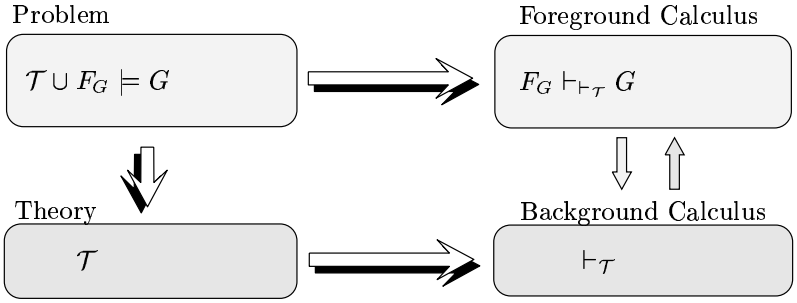


Figure 1.1. Principle of Theory Reasoning

Since predicate logic is such an expressive language, many practically arising problems match the scheme "Does F imply G". But which parts of F should be separated away as a theory $\mathcal{T}$ into the background reasoner? I list some typical instances in order to convey some intuition how this splitting typically is done:

Application area	$F \models G$
Mathematics	*Group Theory*, $\forall x\ x \cdot x = 1 \models \forall x, y\ x \cdot y = y \cdot x$
Knowledge Representation	*T-Box*, A-Box Clauses $\models$ Concept1 $\sqsubseteq$ Concept2
Diagnosis	*N-fault assumption*, System description, Observation $\models \exists$Comp Diagnosis(Comp)
Modal Logic	*Reachability Relation*, Axioms $\models$ Theorem

Here it is not necessary to have a detailed look at F and G. Instead the point to be made is that on the one side F contains *general*, but *domain-specific* knowledge (written in *italic* letters). Usually this constitutes the theory part $\mathcal{T}$. On the other side F contains knowledge F_G dependent from the *concrete* problem G to be solved.

Partitioning F in this way (and not the other way round) is motivated by *reuse*, because the domain-specific knowledge might used in other contexts or problems as well. Thus, some effort for its realization as a background reasoner may pay off well. At best, the background reasoner would be general enough to handle theories from a wide range of problem instances.

For instance, *group theory* can be compiled into a (finite) term-rewriting system by means of the Knuth-Bendix completion procedure [Knuth and Bendix, 1970], and feeding it into an inference mechanism based on narrowing [Hullot, 1980] would constitute a background reasoner for group theory. Now,

a foreground reasoner capable of equality reasoning might replace unification by calls to the background reasoner in order to generate solutions for pure group theory problems (see e.g. [Furbach *et al.*, 1989b]).

For terminological *knowledge representation* languages like $\mathcal{ALC}$ and its descendants (see e.g. [Schmidt-Schauß, 1989]), a lot of work has been spent into the development of *decision* procedures for e.g. concept subsumption, or deciding whether a A-Box is consistent with a T-Box (Figure 1.2 depicts a toy knowledge base for software configuration).

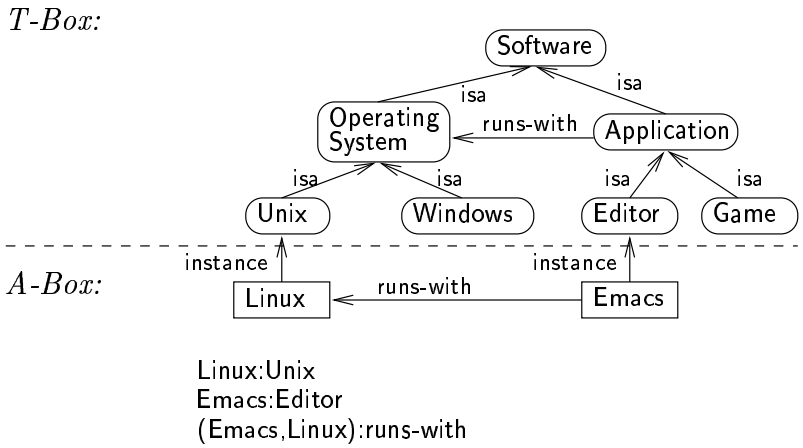


Figure 1.2. A toy knowledge base.

Since concept languages typically are subsets of first-order predicate logic, a concept subsumption task could also be passed to any general first-order theorem prover. However, the termination property typically would be lost then[1]. Hence, it is very appealing to employ a specialised terminological reasoner for this decidable fragment (in Section 4.4.5 such a combination is proposed).

This application area als shows that theory reasoning system can sometimes be also seen as a means to extend the expressivity of an existing reasoner (here: the terminological reasoner) in order to represent knowledge which would not or only in a difficult way be representable otherwise. Using a theory reasoning system, we could, for instance have a scenario like the one in Figure 1.2 above and use the predicate logic formula

$$\forall x, y\ \mathsf{Editor} : x \wedge y : \mathsf{OperatingSystem} \wedge (\mathsf{Emacs}, y) : \mathsf{runs\text{-}with} \rightarrow (x, y) : \mathsf{runs\text{-}with}$$

[1] However, there are – successful – attempts to tune general theorem provers for this task [Paramasivam and Plaisted, 1995].

to express the fact that an operating system y running Emacs also runs all other editors (because Emacs simulates them all). Thus, in such a system the "usual" A-Box language simply consisting of assertions about concept individuals and roles between them would be generalised towards a rule-language over A-Box literals.

In the *diagnosis* domain, following e.g. the consistency-based approach of Reiter [Reiter, 1987], one has a description of a system (for instance, a digital circuit), and an observation about its faulty behaviour. The task of the diagnosis system is to suggest certain components of the device such that a supposed failure of these components is consistent with the observed faulty behaviour. These components are called a *diagnosis*, and usually the task is to compute (subset-)minimal diagnosis. While the minimality property of diagnosis can be formulated in predicate logic, it is preferable to employ a special background reasoner for this, because it can save exponentially many axioms (see [Baumgartner *et al.*, 1997]).

Like in the mathematics example, for certain *modal logics* with corresponding reachability relations for their Kripke semantics, specialised background reasoners can even be *compiled* from the respective axioms for the reachability relations (see Chapter 5 on linearizing completion). By this approach, efficiency can be achieved which would not be possible when using a flat predicate logic formulation (i.e. proofs are often discovered in much shorter time).

To sum up, theory reasoning is an instance of *hybrid reasoning* where the one part is comprised of an (implementation of a) first-order calculus, and the other part implements general knowledge about a special problem domain (also called a *theory*). It is convenient to refer to the former part as the *foreground reasoner* (or "foreground *calculi*" if we refer to a calculus but not a proof procedure or a concrete implementation), and to the latter part as the *background reasoner* (or "background *calculi*"). Whenever this coupling is done in a formal way we will speak of a "theory reasoning system".

For further illustration some more instances of theory reasoning will be mentioned next. Chapter 4.1 contains a more exhaustive overview.

A very prominent example of theory reasoning is *equality handling*. There is a simple way of specifying this theory, namely by stating the axioms of reflexivity, symmetry, transitivity and by stating the substitution axioms of function and predicate symbols. If these formulas are added to the formulas to be proven by the system, the usual inference rules are able to process this theory. A better approach is to supply special inference rules for handling the equality predicate with respect to the equality theory, like e.g. paramodulation [Robinson and Wos, 1969] (Section 4.1.2). This example also serves as another motivation for theory reasoning: the hierarchical structure introduced by theory reasoning often allows a more natural structuring of proofs and presentation at a higher level than it would be possible with a "flat" specification.

Another very well-investigated example for theory-handling is the coupling of a foreground reasoner with efficient unification procedures for dedicated theories (e.g. for *order-sorted* logics [Oberschelp, 1962; Schmitt and Wernecke, 1989] or *equational theories* [Siekmann, 1989]).

Besides such *efficiency* concerns, theory reasoning is also motivated by the possibility of combining systems which handle *different kinds of knowledge*, typically also using different representations thereof. Examples of this kind are hybrid systems, made of a resolution calculus and a taxonomical knowledge representation system, such as KRYPTON [Brachmann *et al.*, 1983].

The spectrum of theories which fit into the framework of theory reasoning is wide. The "upper bound" in order to obtain a complete foreground calculus is given by the class of *universal theories*, i.e. theories that can be axiomatised by a set of formulas that does not contain $\exists$-quantifiers (cf. Section 2.5.3). This restriction is necessary because "unification" is not applicable for existentially quantified variables (see [Petermann, 1992]). Universal theories are expressive enough to formulate e.g. the theory of equality or interesting taxonomic theories.

If the background theory is given in predicate logic form, the restriction to universal theories is not essential. A theory which contains existential quantifiers may be transformed by Skolemization into a universal theory which is equivalent (wrt. satisfiability) to the original one (cf. Section 2.4.1).

1.2 Historical Notes

According to the title of this book, historical remarks can be divided into those on *theory reasoning* and those on *connection calculi*. Let us consider *connection calculi* first.

Connection Calculi

The early roots of connection calculi trace back to Gentzen's famous invention of the sequent calculus [Gentzen, 1939]. The sequent calculus was developed as a method to encode proofs in Gentzen's calculus of natural deduction. To accomplish this an additional syntactical level of "sequents" was introduced which lies above the level of quantifiers and logical connectives. Later, building of work done by Beth (he coined the name "semantic tableaux"), Hintikka and Schütte, R. Smullyan developed his *analytic tableaux* calculus [Smullyan, 1968].

The term "analytic" should be opposed to "generative" and means that in a sense no new *new* formulas are constructed in the course of a derivation; merely, only subformulae of the given formula, instances or negations thereof are handled. Analytic tableaux can be seen as a calculus isomorphic to the sequent calculus, Put a bit loosely, one has to put a tableau upside down

to get a sequent proof. For proof *search* one usually inverts the inference rules of the sequent calculus in order to obtain the same top-down flavour as in analytic tableaux. Smullyan's analytic tableaux have a minimal set of inference rules due to the uniform notation of α, β, γ and δ rules. Unlike the sequent calculus, only (signed) first-order formula are kept in the nodes of the trees, and due to the classification of inference rules and the absence of structural rules, (meta-) proofs become considerably simpler.

First implementations of tableau calculi trace back into the 60s, and probably the most advanced implementation, which also contains a lot of improvements of the basic calculus, is the HARP prover [Oppacher and Suen, 1988].

Of course, the development of the resolution principle [Robinson, 1965b] and in particular the insight in the importance of unification for theorem proving had its influences on tableau calculi as well. Surprisingly, it took quite a long time until unification was brought back into analytic tableaux. As a predecessor work, in [Brown, 1978] unification was used to guide instantiation of sequents (in an incomplete way). Independently from that and also mutually independent, analytic tableaux with unification have been defined in [Reeves, 1987; Schmitt, 1987; Fitting, 1990]. The perhaps most advanced analytic tableaux automated theorem prover is [Beckert *et al.*, 1996].

And what about connection calculi? Connection calculi, also called matrix calculi, were developed independently by W. Bibel [Bibel, 1981; Bibel, 1983; Bibel, 1992] and by P. Andrews [Andrews, 1976].

The central concept underlying connection methods is that of a "spanning mating". A spanning mating for a quantifier-free matrix (e.g. a conjunction of clauses) contains for every "path" through the matrix a complementary set of literals from that path, and hence indicates unsatisfiability of the matrix. According to Bibel [Bibel, 1987], any systematic approach for the construction of spanning matrices qualifies as a "connection calculus".

The close relationship between (clausal) connection calculi and (clausal) tableau calculi became obvious in 80s only. There is agreement that tableau calculi can be seen as connection calculi, and, of course, both include unification. The construction of a closed tableau represents at the calculus level the set of all (or sufficiently many) paths through a matrix and thus yields a spanning mating. The converse, however, need not necessarily be the case: it is conceivable that connection calculi can be developed which search for a spanning mating in a completely different way than tableau calculi do.

In this book, *model elimination* [Loveland, 1968] is in the centre of interest. So what about this calculus? By imposing a certain restriction on the link structure in a tableau calculus, and restricting to clausal input formulas, one immediately obtains a calculus which slightly generalises Loveland's original formulation of model elimination. Chapter 3 contains more about the relation between tableau calculi, connection calculi and model elimination. Model elimination is very closely related to SL-Resolution [Kowalski and Kuehner,

1971] (but it came earlier than SL-Resolution); the formulation of a "linear" connection calculus first appeared in [Bibel, 1982b].

The connection between model elimination and tableau calculi was not obvious at all for a long time and was recognised as late as in the 80 only; making the relationships between model elimination, connection calculi and tableau calculi precise and defining a unifying formal treatment can be attributed to Reinhold Letz from the intellectics research group at the Technical University Munich, Germany. At that time the group was led by Wolfgang Bibel, and the major effort was the design of the model elimination — i.e. connection method — theorem prover SETHEO [Letz *et al.*, 1992], which is today's most developed implementation of that calculus.

It might be appropriate to mention the *connection graph calculus* [Kowalski, 1974] here, as it has a "connection" in the name. Indeed, this paper seems to have introduced the term *connection*, and it has the same meaning as in connection methods. However, the connection graph calculus conceptually is *not* a tableau calculus. It is a generative calculus and it is a certain restricted resolution calculus.

Theory Reasoning

As mentioned above, soon after the development of the resolution principle it became clear that refinements are needed for efficiency reasons. Possibly the first attempt from the viewpoint of theory reasoning was to treat *equality* by a special inference rule. In [Robinson and Wos, 1969] the paramodulation inference rule was introduced; it formalises the idea of replacing "equals by equals" within the resolution calculus. A different approach following soon, which is based on more macroscopic inferences, is E-resolution [Morris, 1969].

A different approach to handle equations was proposed by Plotkin [Plotkin, 1972], where he proposed to build-in certain equational axioms (such as commutativity, associativity) into the unification procedure. Such a unification procedure would then replace an "ordinary" unification procedure in resolution. See [Siekmann, 1989] for an overview over "unification theory".

The idea of explicitly coupling a foreground calculus to a background calculus in an even more general way was formulated in the context of connection calculi in [Bibel, 1982a]. The motivation for doing so was to achieve a better structuring of knowledge. This was exemplified with a coupling of the connection calculus to database systems. In the KRYPTON system [Brachmann *et al.*, 1983], the semantic net language KL-ONE is used as a theory defining language, which is combined with a theory resolution theorem prover. The LOGIN logic programming language [Ait-Kaci and Nasr, 1986] generalises unification towards feature unification, which is a language to represent concept hierarchies in the tradition of KL-ONE with functional roles. These description languages thus generalise order-sorted logics, which was first defined and provided with a clear semantics by A. Oberschelp [Oberschelp,

1962]. Surprisingly, it took quite some time until sorts were brought into resolution [Walther, 1983].

The explicit term *theory reasoning* seems to be introduced by Mark Stickel within the resolution calculus [Stickel, 1985]. This paper also contains a classification of theory reasoning, namely "total vs. wide" theory reasoning and "partial vs. total" theory reasoning (cf. Section 4.1). By doing so, an appropriate generalisation of many special instances of theory reasoning known at that time was developed. In particular, all the approaches listed so far are captured. Indeed, [Stickel, 1985] contains an impressive list of instances of theory reasoning (cf. also Section 4.1.2 below).

A different line of development is *constraint reasoning*. The motivation is similar to theory reasoning — coupling specialised background reasoners to logic. The essential difference (to my understanding) is this: theory reasoning means to call a background reasoner with a goal expression, which shall return a (not necessarily unique) solution, i.e. a substitution[2]. In contrast to this *early* computation of solutions, constraint reasoning means to allow postponing the computation of solutions. In principle there is no need to compute a concrete solution at all. Instead it suffices to show that a solution *exists*. Due to this, constraint reasoning is slightly more general than theory reasoning, as calculi need not be based on Herbrand's theorem. On another side, constraint reasoning is more specific than theory reasoning as in constraint reasoning the foreground theory must be a conservative extension of the background theory (see Section 4.1.2).

Historically, constraint reasoning was developed within logic programming languages, i.e. Horn clause logic [Jaffar and Lassez, 1987]. The most general resolution approach is due to Bürckert [Bürckert, 1990]. For model elimination it was developed in [Baumgartner and Stolzenburg, 1995].

1.3 Further reading

Concerning the history of automated deduction, I recommend the source book [Siekmann and Wrightson, 1983]. It contains classical papers written between 1957 and 1970.

Between 1992 and 1998 the German research council, the DFG (Deutsche Forschungsgemeinschaft), funded the research programme "Deduction". The research results are layed down by the participants in a three volume book [Bibel and Schmitt, 1998]; it summarises the research on deduction carried out in the mentioned period in Germany. In particular, Volume I contains chapters on tableaux calculi, connection calculi and refinements, as well as chapters on theory resoning.

[2] possibly augmented by a condition (residue) under which the substitution is indeed a solution.

A good introductory text on first order tableau methods is [Fitting, 1990]. The handbook [D'Agostino *et al.*, 1998] describes tableau methods both for classical and non-classical logics; it also contains a more detailled historical overview.

There are various journals and conferences which cover automated deduction in general and, in particular, connection methods and theory reasoning. The probably most well-known journal is the *Journal of Automated Reasoning* (Kluwer), and the probably most important conferences are those on *Analytic Tableaux and Related Methods* and the *Conference on Automated Deduction* (both published by Springer in the LNAI series).

1.4 Overview and Contributions of this Book

The purpose of this section is to give an overview of the rest of this book, thereby emphasising where I see the original contributions to the field. A detailed discussion of related work will be carried out in the main parts where appropriate.

The general motivation for my work, and in particular for this book, is the desire to construct an *efficient* reasoning system for first order logic. The basic assumption is that *theory reasoning* is an appropriate tool for accomplishing this goal. Further, the primary calculus of interest is model elimination[3].

In this text, I try to address as many issues as possible which are relevant for building a theory-reasoning-based theorem-prover. Obviously, due to the great variety of methods and tools available in the field, there is a need for biasing the investigations. Nevertheless, I think that within the chosen specialisation (to be described soon) the most important issues are addressed. This concerns questions

- on the *theoretical background* of first-order logic,
- on *foreground reasoners* (or calculi) to consider for theory reasoning,
- on the *interface* to background reasoners,
- on the construction of suitable (wrt. the chosen foreground calculus) *background reasoners*,
- and on the implementation and practical feasibility of such systems.

These items shall be described in more detail now.

[3] The reason for this choice is my personal scientific development in research groups biased towards connection methods. I do not claim that these type of calculi are superior to, for instance, resolution calculi. But neither would I claim the contrary. Both have their pros and cons, are established in the literature, and in my opinion, there is no evidence given so far that one method is superior to the other.

Theoretical Background

The theoretical background is presented in Chapter 2 and contains standard notions such as the syntax and semantics of first-order logic, as well as basic semantical facts about clause logic and theory reasoning.

Foreground Calculi

In accordance with most research on automated theorem proving we will restrict ourselves to clause logic as input language. Then, suitable foreground calculi are, for instance, resolution type calculi [Robinson, 1965b], or analytic calculi within the family of connection methods [Bibel, 1987][4] such as model elimination (ME) [Loveland, 1968] or the connection calculus (CC) by Eder [Eder, 1992].

Chapter 3 introduces the deductive formalism used in this book and recapitulates the latter calculi in a uniform notation.

General theory reasoning versions of CC and ME will be presented and gradually refined in Chapter 4. By the term "general" I mean that no special properties of a particular theory is made use of. In fact, the only restriction is that they are "universal", i.e. that they can be axiomatised by formula containing no existential quantifiers (Definition 2.5.5 makes this precise). Further, rather high-level principles for the interaction between the foreground and the background calculi are employed.

Theory versions of both CC and ME are established in the literature. For CC, the primary source is the work done by Petermann [Petermann, 1992; Petermann, 1993a], and for ME it is my own work [Baumgartner, 1991b; Baumgartner, 1992a; Baumgartner, 1994]. In order to investigate the differences between these seemingly very similar calculi, the clausal tableau calculus introduced in Chapter 3 is extended towards theory reasoning (Section 4.2 and Section 4.3). This sets up a common framework to compare the theory reasoning versions of CC and ME, which are obtained by further specialisation. This turned out to be necessary, because for the theory-reasoning versions of these calculi some notable differences arise which are not an issue for the non-theory versions; the most important one concerns the different literals kept in the branches of ME vs. CC. ME needs strictly less of these literals, and the conclusion will be that theory ME is the "weaker" calculus, i.e. it can be simulated stepwise by CC.

The consequences of this result are described in Section 4.3.3. In brief, CC allows for shorter refutations, but ME has less local search space. Having a weak calculus is also important for theoretical reasons, since its completeness implies the completeness of all "stronger" calculi (the reader might want to

[4] According to Bibel [Bibel, 1987] any *calculus* which employs the idea of systematically investigating paths through clause sets for contradictory literals (i.e. connections) qualifies to be a connection *method*. More technically, according to Bibel [Bibel, 1987] any calculus based on Lemma A.1.5 is a connection calculus.

have a look at Figure 4.1 on page 66 which contains all the calculi treated in this text and their relationships). As a consequence of the completeness of ME it can easily be shown, that a non-determinism hidden in the CC inference rule "extension step" is fortunately a "don't care" nondeterminism ("order of extending clauses", page 100).

Due to its "weakness", it is ME which is considered for further refinements. These refinements are *regularity* ("no subgoal needs to be proven repeatedly"), *factorisation*, the *independence of the computation rule* (i.e., the next subgoal may be chosen don't-care nondeterministically), and, most important for problem solving applications, the computation of *answers* to queries for programs. It should be noted that answer completeness includes the notion of refutational completeness traditionally used in automated theorem proving.

Interface to the Background Reasoner

For search space reasons one very carefully has to address the question of *how* to interface the foreground reasoner to the background reasoner. We consider both the *total* and *partial* variants of theory reasoning for this (the overview in Section 4.1 explains these notions). Both principles are rather different and deserve investigation.

The variant for ME which was used for the comparison to CC is refined in Section 4.4 to the final *total* theory reasoning version, and it is refined in Sections 4.5 and 4.6 towards the final *partial* theory reasoning version.

For the total version, the interface to the background reasoner is specified by a "complete and most general set of theory refuters" ($MSR_{\mathcal{T}}$, Def. 4.4.2), a concept which includes the usual concept of a most general unifier as a special case. $MSR_{\mathcal{T}}$s have to be defined very carefully. In particular, it is necessary to use the concept of "protected variables", which means that the substitutions from a $MSR_{\mathcal{T}}$ must never introduce a new variable which is contained in such a set of protected variables.

In sum, for total theory reasoning, the background reasoner can be thought of as given as a black box, which only has to fulfil some semantic properties. These properties will be defined in such a way that they admit an answer completeness result for the combined calculus. This is the main result of Section 4.4.

The purely semantic approach to *total* theory reasoning is justified by the existence of numerous domain-dependent reasoning systems which can be used immediately for total theory reasoning. On the other hand, *partial* theory reasoning is an attractive principle as well, in particular for model elimination (Section 4.1.1 gives reasons). However, partial theory reasoning is much more tricky than total theory reasoning, as it requires a tighter and symmetric coupling of the two reasoners. It is unclear how to use, say a given AC-unification algorithm in a meaningful way for partial theory reasoning.

Therefore, it seems advantageous to give up the semantic view and propose in Section 4.5.1 a syntactical framework instead. It is called "theory inference rules", and can be seen as a specification language for permissible partial theory inferences. A general completeness criterion for background reasoners following this scheme is defined, and the resulting calculus is proven as answer-complete. This is the main result of Section 4.5.

In Section 4.6 a variant of model elimination — restart model elimination — is presented which supports the "usual" procedural reading of clauses. As was argued for in [Baumgartner and Furbach, 1994a], this variant is interesting in at least two respects: it makes model elimination available as a calculus for non-Horn logic programming and it enables model elimination to perform proofs in a natural style by case analysis. Of course, the concern here is "lifting" restart model elimination towards theory reasoning. Since the primary interest is in partial theory reasoning, it is thus the *partial restart theory model elimination* which is discussed and proven answer-complete.

Background Reasoner

The *background reasoner* carries out deductions in the domain-dependent part of the specification. These can be rather course-grained deductions, as in the case of total theory reasoning, or be rather simple ones, as is the idea of partial theory reasoning.

In automated theorem proving the problem specifications typically contain axioms which can naturally be separated off as a theory (e.g. equality, orderings, set theory, group theory etc.). Since such theories are typically "reusable" for many problems, it is worth spending some effort in developing efficient theory reasoners for them.

But how can this be achieved? One way is to analyse the theory in question and hand-craft a respective theory reasoner. Another, more general way is to design a "compiler" which can automatically transform theory axioms into background reasoners. It is the latter way which is pursued here.

From the viewpoint of theory reasoning, the well-known Knuth-Bendix completion method can be seen as such a compiler. However, Knuth-Bendix completion works for equational theories only, and it is unclear how to build-in "rewriting" into a linear calculus such as model elimination.

Therefore, I propose a more general compilation technique which is compatible with model elimination. It is called "linearizing completion" and is treated in depth in Chapter 5. It is general, as it can deal with any Horn theory, and it leads to efficient partial theory reasoners.

Implementation

In order to assess its practical usability, the framework "partial theory model elimination plus linearizing completion" was implemented by students in our

research group. Chapter 6 reports on this implementation and on practical experiments. The results show that the suggested methodology significantly improves the power of model elimination-based theorem proving.

2. Logical Background

I have tried to make the text self-contained. To this end I will recapitulate in this chapter the necessary background from first order logic to a certain degree. However, I cannot go much into the details here, and also I had to omit certain topics. In general, the text is biased towards our theorem proving needs, and in particular towards clause logic.

For a more elaborate discussion of the logical background I refer the reader to [Enderton, 1972] and [Gallier, 1987]. An introduction to logics with an emphasis on theorem proving can be found in [Chang and Lee, 1973] and in [Loveland, 1978]. An introduction to first order logics under a broader viewpoint of computation is [Lalement, 1993].

The plan of this chapter is as follows: After having defined some preliminary notions, we will turn towards the *syntax* and *semantics* of first order logic (Sections 2.2 and 2.3, respectively).

After that, predicate logic will be specialized towards *clause logic*. Clause logic is the dominant logic used in automated theorem proving and also in this text. The concluding section then turns towards *theories*. There, usual definitions (such as "logical consequence-ship") are generalized towards theories. Also, a generalized Herbrand-theorem will be proven.

2.1 Preliminaries

Multisets

Multisets are like sets, but allow for multiple occurrences of identical elements. Formally, a multiset N over a set X is a function $N : X \mapsto \mathbb{N}$. $N(x)$ is intended as the "element count" of x. We often say that x "occurs $N(x)$ times in N". The multiset union $N \cup M$ of two multisets is described by the equation $(N \cup M)(x) = N(x) + M(x)$, the multiset difference by $(N - M)(x) = N(x) \dot{-} M(x)$, and the intersection is given by $(N \cap M)(x) = min(N(x), M(x))$. Finally, two multisets are equal, $N = M$ iff $N(x) = M(x)$ for every $x \in X$.

We will use set-like notation with braces '$\{\!|$' and '$|\!\}$' for multisets. For example the set for which $N(a) = 3$ and $N(b) = 1$ and $N(x) = 0$ for all other

values can be written as $\{\!|a, a, a, b|\!\}$. If a *set* is used below where a *multi*set were required, then the type conversion is done in the obvious way.

As usual, we write $x \in N$ iff $N(x) > 0$. As a non-standard notion, we say that multiset M is an *amplification* of multiset N, written $M \sqsupseteq N$, iff $M(x) \geq N(x) > 0$ for every $x \in M$. That is, an amplification M of N is obtained by repeating some of N's elements. Finally, we say that Multiset N *contains no duplicates* iff $N(x) = 1$ for every $x \in N$.

Orderings

We recall some definitions and facts about orderings. In [Dershowitz, 1987] a detailed survey can be found.

A partial strict ordering[1] $\succ$ on a set X (of terms, for our purpose) is called *well-founded* if there is no infinite (endless) sequence $s_1 \succ s_2 \succ \ldots s_n \succ \ldots$ of elements from X. An ordering on terms[2] is said to be *monotonic* iff $s_1 \succ s_2$ implies $f(\ldots s_1 \ldots) \succ f(\ldots s_2 \ldots)$.

As usual we define $s \prec t$ iff $t \succ s$, $s \preceq t$ iff $s \prec t$ or $s = t$ and $s \succeq t$ iff $t \preceq s$. Below we will order terms and integers, in which case $=$ means the respective identity relation (although any equivalence relation can be used in principle), however with one exception: *finite multisets* can be thought of as terms constructed from the variadic constructor symbol "$\{\!|.|\!\}$". Henceforth equality of such terms is understood as multiset equality.

Partial orderings can be extended to partial orderings on tuples by comparing their components – as usual – as follows: Assume n sets X_i equipped with n respective partial strict orderings $\succ_i$ as given. Then an n-tuple $s = \langle s_1, \ldots, s_n \rangle$ is *lexicographically greater* than an n-tuple $t = \langle t_1, \ldots, t_n \rangle$, written as $s \succ_{Lex, \succ_1, \ldots, \succ_n} t$ iff $s_i \succ_i t_i$ for some i $(1 \leq i \leq n)$ while $s_j = t_j$ for all $j < i$.

Similarly, for a given set X of terms with well-founded strict ordering $\succ$ we will with $\gg_\succ$ denote the extension of $\succ$ to (finite) multisets over X. $\gg_\succ$ is defined redursively as follows:

$$X = \{\!|x_1, \ldots, x_m|\!\} \gg_\succ \{\!|y_1, \ldots, y_n|\!\} = Y$$
$$\text{if} \quad x_i \succ y_{j_1}, \ldots, y_{j_k} \text{ and } X - \{\!|x_i|\!\} \gg_\succeq Y - \{\!|y_{j_1}, \ldots, y_{j_k}|\!\}$$
$$\text{for some } i,\ 1 \leq i \leq m \text{ and}$$
$$\text{for some } j_1, \ldots, j_k,\ 1 \leq j_1 \leq \ldots \leq j_k \leq n\ (k \geq 0)\ .$$

Informally, $X \gg_\succ Y$ if Y can be obtained from X by successively replacing an element by zero or more (finite) elements strictly smaller in the base relation $\succ$. Notice that the termination condition for this recursive definition is the case where $X - \{\!|x_i|\!\} = Y - \{\!|y_{j_1}, \ldots, y_{j_k}|\!\}$.

---

[1] a *strict ordering* is a transitive and irreflexive binary relation.

[2] *Terms* are defined in Def. 2.2.2 below.

Quite often, we will write $\gg$ instead of $\gg_\succ$ if $\succ$ is clear from the context (and similar for $\succ_{Lex}$).

It is well-known that the orderings $\succ_{Lex}$ and $\gg$ are well-founded provided that the orderings are which they are based upon (see [Dershowitz, 1987] for an overview of orderings). Orderings on multisets can be generalized towards on *nested multisets* [Dershowitz and Manna, 1979], i.e. multisets on nested multisets in a well-founded way.

For termination proofs the *simplification orderings* are of particular interest; a *simplification ordering on a set X of terms* is defined as a monotonic partial strict ordering $\succ$ on X which possesses the *subterm property*, i.e. $f(\ldots s \ldots) \succ s$, and the deletion property, i.e. $f(\ldots s \ldots) \succ f(\ldots\ldots)$. The latter property is required only if f is a variadic function symbol (such as the multiset constructor). Various simplification orderings are known, among them the *recursive path ordering (with and without status)*, the *semantic path ordering* and *lexicographic path ordering* (see again [Dershowitz, 1987] for an overview of orderings). Simplification orderings are interesting for the following reason:

Theorem 2.1.1. *[Dershowitz, 1982; Dershowitz, 1987] Any simplification ordering on a set X of terms is a monotonic well-founded ordering on X.*

It is decidable whether two terms are related in a given simplification ordering. This problem is NP-complete (see e.g. [Nieuwenhuis, 1993]).

2.2 Syntax of First-Order Logic

We will start with a common defintion.

Definition 2.2.1 (First-Order Signature). *A* (first-order)[3] signature Σ *consists of a set of objects, called* symbols *which can be written as the disjoint union of the following sets:*

1. *a denumerable set $\mathcal{F}$ of* function symbols,
2. *a denumerable set $\mathcal{P}$ of* predicate symbols,
3. *a denumerable set $\mathcal{X}$ of* (first-order) variables,
4. *the set $\{\neg, \vee, \wedge, \leftarrow, \rightarrow, \leftrightarrow\}$ of* connectives,
5. *the set $\{\forall, \exists\}$ of* quantifiers, *and*
6. *the set $\{(, , ,)\}$ of* punctuation symbols.

The sets 1–3 may differ from signature to signature, while the remaining sets 4–6 are the same in every signature. The former ones are therefore called the parameters *of the signature, and it is sufficient to refer to a signature uniquely as a triple $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$.*

[3] In definitions parenthesis indicate a long form of the definiens. Usually the short form is used.

We assume a fixed arity associated to every predicate and function symbol. Formally, an arity *is a function mapping $\mathcal{P} \cup \mathcal{F}$ into the non-negative integers. Function symbols of arity 0 are also called* constant (symbols).

The symbols are the building blocks for various kinds of structured objects. In particular we will define *terms, atoms* and *formulas*. The collection of these all together is referred to as the *language of first-order predicate logic*.

Definition 2.2.2 (Term). *The* terms *of a given signature $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ are inductively defined as follows:*

1. *every variable $x \in \mathcal{X}$ is a Σ-term.*
2. *if $f \in \mathcal{F}$ is a function symbol of arity n, and $t_1, \ldots, t_n$ are n Σ-terms then $f(t_1, \ldots, t_n)$ is a Σ-term.*
3. *nothing else is a Σ-term*[4].

The set of Σ-terms is also denoted by $Term(\mathcal{F}, \mathcal{X})$. The set of Σ-terms which do not contain variables of $\mathcal{X}$, i.e. the set of terms which can be obtained due to case 2 alone is denoted by $Term(\mathcal{F})$. The elements of $Term(\mathcal{F})$ are also called ground terms (of Σ).

Convention 2.2.1 (Syntax 1). If in case 2 the function symbol f is a constant symbol we will always write f instead of $f()$. Also, common function symbols of arity 2 will often be written infix. For instance we will write $s + t$ instead of $+(s, t)$. But then, parenthesis will occasionally be necessary for disambiguation. There is no need here to go further into the details of syntax, instead it is taken for granted that the notation will always be in such a way that the structure according to the definition of terms can uniquely be recovered. See [Hopcroft and Ullman, 1979] for details.

Building on terms we can arrange *atoms* and *formulas*:

Definition 2.2.3 (Atom, Formula). *Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature. If $t_1, \ldots, t_n$ are Σ-terms and $P \in \mathcal{P}$ is an n-ary predicate symbol then $P(t_1, \ldots, t_n)$ is an* atomic formula *(short:* atom*). If all $t_1, \ldots, t_n$ are ground terms then $P(t_1, \ldots, t_n)$ is also called a* ground atom. (Well-formed) Σ-Formulas *are inductively defined as follows:*

1. *every atom A is a Σ-formula.*
2. *if F is a Σ-formula then $\neg F$ is a Σ-formula, called* negated formula.
3. *if F_1 and F_2 are Σ-formulas then $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \rightarrow F_2)$ (considered identical to $(F_2 \leftarrow F_1)$) and $(F_1 \leftrightarrow F_2)$ are Σ-formulas. These are called* conjunction, disjunction, implication *and* equivalence *(in this order).*
4. *if F is a Σ-formula and $x \in \mathcal{X}$ is a variable then $\exists x\, F$ and $\forall x\, F$ are Σ-formulas. These are called* existentially quantified formula *and* universally quantified formula*, respectively.*

[4] In subsequent inductive definitions such a phrase will be omitted, but shall be considered as implicitly given.

In the sequel we will often omit the prefix "Σ-" if not relevant or clear from the context. By $\mathcal{PF}(\Sigma)$ we denote the set of all Σ-formulas. If F is a formula and F occurs as part of a formula G then F is called a subformula *of G*[5]*. For instance, A is a subformula of A, and A is also a subformula of $B \wedge \neg A$. A formula is called a* ground formula *iff all atoms occuring in it are ground atoms.*

Note that atoms are not defined inductively. Instead we have simply said explicitly which expressions are atoms.

At the moment formulas (and terms) are not assigned any meaning. Thus it is more appropriate to read the connective "$\wedge$" as "hat" or something similar rather than "and". It is subject to *interpretations* defined below to assign the intended meaning to formulas.

Convention 2.2.2 (Syntax 2). Extending Convention 2.2.1 above we will make use of some notational conventions. Note first that according to their use in the definition the connectives $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ can be thought of as binary operator symbols. Thus it is possible to assert associativity to them and leave parenthesis out in many cases. We have decided to declare each of them as left-associative (although right-associative would work as well). Now, if parenthesis may be left away *binding power* for the connectives has to be used in order to disambiguate expressions like $\forall x P(x) \wedge Q(x)$. As a convention we declare that the binding power of the connectives $\neg, \forall, \exists, \wedge, \vee, \leftrightarrow, \rightarrow$ shall *de*crease in that order. Thus, for instance the expression

$$A \wedge B \wedge C \vee \neg D \wedge P(y) \rightarrow \forall x\, P(x) \wedge Q(x)$$

is translated into the formula

$$(((A \wedge B) \wedge C) \vee (\neg D \wedge P(y))) \rightarrow ((\forall x\, P(x)) \wedge Q(x)) \ .$$

As a further convenience we allow the abbreviation $\forall x_1\ \forall x_2\ \ldots\ \forall x_n\, F$ to $\forall x_1, x_2, \ldots, x_n\, F$. The same convention is used for "$\exists$".

As with function symbols, binary predicate symbols will also be written infix occasionally. Typical examples are "$=$" and "$<$". Putting things together we may for example write

$$\forall \varepsilon\ \varepsilon > 0 \rightarrow \exists n_0\ \forall n\ (n \geq n_0 \rightarrow abs(f(n) - a) < \varepsilon)$$

as a better readable form of the formula

$$\forall \varepsilon\ (> (\varepsilon, 0) \rightarrow \exists\, n_0 \forall n\ (\geq (n, n_0) \rightarrow < (abs(-(f(n), a)), \varepsilon))) \ .$$

[5] Here we have taken the notions "occurence" and "part of" for granted. A formal treatment can be carried out as is done for the respective notions in "derivations" in Chapter 5.

Next we turn to *bound* and *free* occurrences of variables in formulas. This distinction is important since formulas *without* free variables are "self-contained" in the sense that they are meaningful by referring *only* to the symbols occurring in them.

Definition 2.2.4 (Bound and Free Variable Occurrences, Sentence). *Let Σ be a signature. An occurrence of a variable x in a formula F is called* bound in F *if x occurs in some subformula of F of the form $\forall x\, G$ or of the form $\exists x\, G$. Otherwise, this occurrence is called* free. *A formula without occurrences of free variables is called* closed. *A closed formula is also called* sentence. *By $\mathcal{CPF}(\Sigma)$ we denote the set of all closed Σ-formulas.*

It is possible that a variable x occurs at different positions in one single formula both bound and free. For example, in

$$F = P(x) \wedge \exists x\, Q(x, y)$$

the variable x occurs bound and free in F, y occurs free in F and z does not occur in F.

Definition 2.2.5 (Universal and Existential Closure). *If F is a formula we denote by $\forall F$ its* universal closure, *i.e. the sentence $\forall x_1, \ldots, x_x\, F$, where $x_1, \ldots, x_n$ are all variables occurring free in F. The* exitential closure *$\exists F$ is defined analogously.*

We conclude this section with one more common definition.

Definition 2.2.6 (Complement). *Let F be a formula The* complement of *F, written as $\overline{F}$, is defined as follows:*

$$\overline{F} = \begin{cases} G & \text{if } F = \neg G, \text{ for some formula } G \\ \neg F & \text{else} \end{cases} .$$

It is easily verified that $F = \overline{\overline{F}}$.

For a finite set X of formulas we define[6]

$$\overline{X} = \begin{cases} \overline{F} & \text{if } X \text{ is the singleton } \{F\}, \text{ for some formula } F \\ \bigvee_{F \in X} \overline{F} & \text{else} \end{cases} .$$

2.3 Semantics of First-Order Logic

In order to assign meaning to predicate logic formulas the function and predicate symbols have to be interpreted with certain functions and relations, respectively, over some domain (which is also called *universe*). This collection — functions, relations and the universe — is called a *structure*. As an

[6] Note that X has to be finite, because otherwise $\overline{X}$ were no formula.

example think of the domain of natural numbers equipped with the functions and relations of Peano Arithmetic.

An *interpretation* then maps the parameters of a language into their semantic counterpart of a structure. Together with an *assignment* function for the free variables in a formula we have means to *evaluate* the truth value of this formula. All this is introduced next.

Definition 2.3.1 (Structure, Interpretation, Assignment). *A* structure *$\mathcal{S}$ is a triple $\langle \mathcal{U}, F, R \rangle$ where*

1. *$\mathcal{U}$ is a non-empty set, called the* universe *(also called* domain *or* carrier *sometimes),*
2. *F is a set of total functions of given arity on $\mathcal{U}$[7], and*
3. *R is a set of total relations of given arity on $\mathcal{U}$[8].*

The universe of the structure $\mathcal{S}$ is also denoted by $|\mathcal{S}|$.

Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature. A Σ-interpretation *is a triple $\mathcal{I} = \langle \mathcal{S}, \mathcal{I}_\mathcal{F}, \mathcal{I}_\mathcal{P} \rangle$ where*

1. *$\mathcal{S}$ is a structure, and*
2. *$\mathcal{I}_\mathcal{F}$ is a total function mapping each n-ary function symbol from $\mathcal{F}$ to an n-ary function from F, and*
3. *$\mathcal{I}_\mathcal{P}$ is a total function mapping each n-ary predicate symbol from $\mathcal{P}$ to an n-ary relation from R.*

An assignment *$v_\mathcal{X}$* (or *v* for short) *into $\mathcal{U}$ is a partial function mapping $\mathcal{X}$ into $\mathcal{U}$. An assignment $v_\mathcal{X}$ is said to be* suitable *for a given formula (or set of formulas) iff it is defined for every free variable occurring in it. By an* extended Σ-interpretation *we mean a Σ-interpretation $\mathcal{I}$ plus (not necessarily suitable) assignment into $|\mathcal{S}|$, written as a tuple $\langle \mathcal{I}, v_\mathcal{X} \rangle$.*

For brevity we will write the "universe of the interpretation" instead of the "universe component of the structure component of the interpretation" and denote it by $|\mathcal{I}|$.

Example 2.3.1. As an example consider a signature

$$\mathcal{N} = \langle \{zero, succ, plus, times\}, \{equal, lt\}, \{x_1, x_2, x_3, \ldots\} \rangle$$

and the structure $Nat = \langle \mathbb{N}, \{0, s, +, *\}, \{=, <\} \rangle$. In Nat let s denote the successor function, let 0, $+$ and $*$ denote the functions as expected, and let $=$ and $<$ be interpreted as the usual "equality" and "less than" relations. If we assume that the arities are defined suitably, then we might naturally define the following $\mathcal{N}$-interpretation NAT and assignment

$$\begin{array}{llll} \mathcal{S} = Nat & \mathcal{I}_\mathcal{F}(zero) = 0 & \mathcal{I}_\mathcal{P}(equal) = = & v(x_1) = 0 \\ & \mathcal{I}_\mathcal{F}(succ) = s & \mathcal{I}_\mathcal{P}(lt) = < & v(x_2) = 8 \\ & \mathcal{I}_\mathcal{F}(plus) = + & & v(x_3) = 15 \\ & \mathcal{I}_\mathcal{F}(times) = * & & \end{array}$$

[7] An *n-ary operation on $\mathcal{U}$* is a function mapping $\mathcal{U}^n$ into $\mathcal{U}$

[8] An *n-ary relation on $\mathcal{U}$* is a subset of $\mathcal{U}^n$

By definition, functions are nothing but relations with special properties. Thus, in a strict sense it is not necessary to introduce functions explicitly in structures. However this approach would be far less natural.

In the literature one can find several definitions of *structure* and *interpretation*. Our definition is much like in [van Dalen, 1980], however do we not restrict to *finite* R and F. In [Enderton, 1972] a *structure* is our "structure plus interpretation"; "interpretation" has a very different meaning there. Also by some authors the terms "structure" and "interpretation" are used synonymously.

Interpretations assign meaning to the predicate and function symbols. Together with suitable assignments they can be used to evaluate a formula F by first recursively evaluating the parts of the formula, and then computing the value of F as a function of these. The recursion stops at variables or constants whose values can immediately be computed. The next definition expresses this in a precise way:

Definition 2.3.2 (Evaluation). *Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature and $\mathcal{I} = \langle \mathcal{S}, \mathcal{I}_\mathcal{F}, \mathcal{I}_\mathcal{P} \rangle$ be a Σ-interpretation. We define the* value of a term t wrt. an extended interpretation $\langle \mathcal{I}, v_\mathcal{X} \rangle$, $\mathcal{I}_{v_\mathcal{X}}(t)$, *(or $\mathcal{I}_v(t)$ for short) where $v_\mathcal{X}$ is an assignment into $|\mathcal{I}|$ as a function from the Σ-terms into $|\mathcal{I}|$ recursively as follows:*

1. *If $t = x$ for some variable $x \in \mathcal{X}$ then $\mathcal{I}_v(t) = v_\mathcal{X}(x)$.*
2. *If $t = f(t_1, \ldots, t_n)$ for some n-ary function symbol $f \in \mathcal{F}$. Define*

$$\mathcal{I}_v(t) = (\mathcal{I}_\mathcal{F}(f))(\mathcal{I}_v(t_1), \ldots, \mathcal{I}_v(t_n)) \ .$$

Note that this definition includes the case where t is a constant c. Here we find immediately $\mathcal{I}_v(c) = \mathcal{I}_\mathcal{F}(c)$. But note also that $\mathcal{I}_v$ is a partial *function, since in case 1, $\mathcal{I}_v(x)$ is undefined if $v(x)$ is undefined.*

Following the non-recursive definition of atoms we can extend the definition. The value of an Atom $A = P(t_1, \ldots, t_n)$ wrt. $\langle \mathcal{I}, v_\mathcal{X} \rangle$, $\mathcal{I}_{v_\mathcal{X}}(A)$ (or $\mathcal{I}_v(A)$ for short), *where v is supposed to be suitable for A, is defined as*

$$\mathcal{I}_v(A) = \begin{cases} \mathit{true} & \mathit{if}\ \langle \mathcal{I}_v(t_1), \ldots, \mathcal{I}_v(t_n) \rangle \in \mathcal{I}_\mathcal{P}(P) \\ \mathit{false} & \mathit{else} \ . \end{cases}$$

Now let F be a Σ-formula and let v be a suitable assignment for F into $|\mathcal{I}|$. Further extending the definition, we define the value of F wrt. $\langle \mathcal{I}, v \rangle$, $\mathcal{I}_{v_\mathcal{X}}(F)$, (or $\mathcal{I}_v(F)$ for short) *as follows:*

1. *If $F = \neg G$ for some formula G then*

$$\mathcal{I}_v(F) = \begin{cases} \mathit{true} & \mathit{if}\ \mathcal{I}_v(G) = \mathit{false} \\ \mathit{false} & \mathit{else} \ . \end{cases}$$

2. *If $F = G_1 \circ G_2$ for some formulas G_1 and G_2 and $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ then $\mathcal{I}_v(F)$ is determined according to the following truth table:*

G_1	G_2	$G_1 \wedge G_2$	$G_1 \vee G_2$	$G_1 \leftarrow G_2$	$G_1 \leftrightarrow G_2$
false	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*true*	*false*	*false*
true	*false*	*false*	*true*	*true*	*false*
true	*true*	*true*	*true*	*true*	*true*

3. *If* $F = \forall x\, G$ *for some formula* G *then*

$$\mathcal{I}_v(F) = \begin{cases} \mathit{true} & \mathit{if}\ \mathcal{I}_{v[x \leftarrow a]}(G) = \mathit{true}\ \mathit{for\ every}\ a \in |\mathcal{I}| \\ \mathit{false} & \mathit{else}\ . \end{cases}$$

Here $v[x \leftarrow a]$ *is defined to be exactly the same assignment as* v *(possibly) except that* $v[x \leftarrow a]$ *assigns* a *to* x*. In other words, we have to "update" the assignment expressed by* v *at the point* x*. This can formally expressed by defining*

$$v[x \leftarrow a]_{\mathcal{X}}(y) = \begin{cases} a & \mathit{if}\ y = x \\ v_{\mathcal{X}}(y) & \mathit{else}\ . \end{cases}$$

4. *If* $F = \exists x\, G$ *for some formula* G *then*

$$\mathcal{I}_v(F) = \begin{cases} \mathit{true} & \mathit{if}\ \mathcal{I}_{v[x \leftarrow a]}(G) = \mathit{true}\ \mathit{for\ some}\ a \in |\mathcal{I}| \\ \mathit{false} & \mathit{else}\ . \end{cases}$$

It is important to note that evaluation under *suitable* assignments is a *total* function. This is a consequence from the definitions of *interpretation, suitability* and of *structure*. In particular, as a property of the definition of interpretation, all function symbols and predicate symbols from Σ can be mapped into their semantic counterparts $\mathcal{I}_{\mathcal{F}}$ and $\mathcal{I}_{\mathcal{P}}$. Furthermore, the structure definition guarantees that $\mathcal{I}_{\mathcal{F}}$ and $\mathcal{I}_{\mathcal{P}}$ are total. Finally, by the suitability of assignments every free variable in a formula gets a value. As an example for an evaluation consider the formula

$$\mathit{PredExist} = \forall x\, (x > x_1 \rightarrow \exists y\, y + 1 = x)\ .$$

PredExist (quite freely "every number greater than x_1 has a predecessor") evaluates to *true* in the interpretation and assignment given in Example 2.3.1 above.

Convention 2.3.1 (Implicit Signature). In the sequel if we speak of an "interpretation for a formula" or set of formulas we often let the underlying signature Σ implicitly defined by the symbols occurring in it, and interpretations are understood as such Σ-interpretations.

Next we turn to the important notions of *satisfiability* and *validity*.

Definition 2.3.3 (Satisfiability, Validity, Model, etc.). *Let* $\langle \mathcal{I}, v \rangle$ *be an extended interpretation suitable for a formula* F*. We say that* $\langle \mathcal{I}, v \rangle$ satisfies F *iff* $\mathcal{I}_v(F) = \mathit{true}$ *and we write*

$$\langle \mathcal{I}, v \rangle \models F \ .$$

If $\mathcal{I}_v(F) = false$ *we say that* $\langle \mathcal{I}, v \rangle$ falsifies F.

Now let M *be a set of formulas. Generalizing the previous definition we say that* $\langle \mathcal{I}, v \rangle$ satisfies M *iff* $\langle \mathcal{I}, v \rangle \models F$ *for every* $F \in M$. *This is written as* $\langle \mathcal{I}, v \rangle \models M$ *(thus, a (finite!) set of formulas can be thought as a conjunctions of its members). Accordingly,* $\langle \mathcal{I}, v \rangle$ falsifies M *iff* $\mathcal{I}_v(F) = false$ *for some* $F \in M$. F *is said to be a* logical consequence *of* M, $M \models F$ *iff whenever* $\langle \mathcal{I}, v \rangle \models M$ *then also* $\langle \mathcal{I}, v \rangle \models F$, *where* v *is suitable for* $M \cup \{F\}$. *For brevity we will write* $G \models F$ *instead of* $\{G\} \models F$. *We say that* F *and* G *are* logically equivalent, $F \equiv G$, *iff* $F \models G$ *and* $G \models F$.

In the following X *denotes a* Σ*-formula or a set of* Σ*-formulas, and* $\mathcal{I}$ *denotes a* Σ*-interpretation.* X *is called* satisfiable *iff* X *is satisfied by some extended interpretation* $\langle \mathcal{I}, v \rangle$ *with suitable* v. *If* X *is not satisfiable, i.e. if* X *is falsified by every suitable* $\langle \mathcal{I}, v \rangle$ *then* X *is also called* unsatisfiable. $\mathcal{I}$ *is called a* model for X *iff for all suitable assignments* v *it holds* $\langle \mathcal{I}, v \rangle \models X$. *This is written as* $\mathcal{I} \models X$. *If for some suitable assignment* v, *we have* $\mathcal{I}_v(X) = false$ *we say that* $\mathcal{I}$ *is a* countermodel for X. X *is called* valid *iff every* Σ*-interpretation is a model for* X. *This is noted as* $\models X$. *If* X *is a formula it is also called a* tautology.

Recall from the definition of structure above that the universe component is required to be non-empty. Here we can indicate the motivation for this. By the non-emptiness unexpected and contraintuitive consequences can be prevented. For instance, the formula $(\forall x\, P(x)) \rightarrow \exists y\, P(y)$ is false in an interpretation whose structure is empty. However we would expect this formula to be valid.

Next we turn towards important standard results about free variables in formulas. The first result says that the truth-value of a formula depends on the assignment to free variables only.

Theorem 2.3.1. *Suppose* $\langle \mathcal{I}, v_1 \rangle$ *and* $\langle \mathcal{I}, v_2 \rangle$ *are extended interpretations with suitable assignments for a formula* F. *Suppose* v_1 *and* v_2 *agree on the free variables of* F. *Then*

$$\langle \mathcal{I}, v_1 \rangle \models F \qquad \textit{iff} \qquad \langle \mathcal{I}, v_2 \rangle \models F \ .$$

This theorem simplifies the task of computing the truth value of a formula considerably, since only a finite subset of the — in general infinite — assignment has to be considered. The straightforward proof requires structural induction and can be found for a slightly different formulation in [Enderton, 1972]. Since *sentences* do not contain occurrences of free variables we obtain:

Corollary 2.3.1. *For a sentence* S *and interpretation* $\mathcal{I}$, *either*

1. $\langle \mathcal{I}, v \rangle \models S$ *for every assignment* v, *or*
2. $\langle \mathcal{I}, v \rangle \not\models S$ *for every assignment* v.

Thus for sentences the truth value depends alone from an interpretation $\mathcal{I}$. This gives some nice properties. For instance, from $\mathcal{I} \not\models S$ we may infer $\mathcal{I} \models \neg S$ (cf. also Lemma 2.3.1 below). Using this, the definitions of "model" and of negation we can rewrite the previous corollary to

Corollary 2.3.2. *For a sentence S and interpretation $\mathcal{I}$, either $\mathcal{I} \models S$ or $\mathcal{I} \models \neg S$.*

Also we have:

Corollary 2.3.3. *Let M be a set of sentences and X be a sentence or finite set of sentences. Then the following are equivalent:*

1. *$M \cup \overline{X}$ is unsatisfiable.*
2. *$M \models X$.*
3. *for every interpretation $\mathcal{I}$: $\mathcal{I} \models M$ implies $\mathcal{I} \models X$.*

The proof of Corollary 2.3.3 is easy and is done by applying the definitions and Theorem 2.3.1.

The equivalence 1.–2. is the basis for *refutational theorem proving*; it allows the reduction of the task of determing whether a given set X of sentences is a logical consequences of another set M of assumptions to a question of unsatisfiability.

The equivalence 2.–3. states that for logical consequenceship wrt. sentences does not depend on assignments. This does not hold if we do not restrict to sentences; since it can rarely be found in the literature we will give a counterexample here. It shows that we can find a set M and a formula F such that F is a logical consequence of M although not every model of M is also a model of F.

Example 2.3.2. Consider a signature $\Sigma = \langle \{a, b\}, \{=\}, \{x, y, z, u\} \rangle$, where a and b are constants and $=$ is a two-place predicate symbol (written infix). Let $\mathcal{EQ}$ be the following set of formulas ("equivalence relation"):

$$\begin{array}{lrl} \mathcal{EQ}: & \forall x & x = x, \\ & \forall x \forall y & x = y \to y = x, \\ & \forall x \forall y \forall z & x = y \wedge y = z \to x = z \ . \end{array}$$

Suppose, to the contrary that Corollary 2.3.3 also holds for arbitrary formulas. Then $\mathcal{EQ} \cup \{u = a\} \models u = b$ iff every model for $\mathcal{EQ} \cup \{u = a\}$ is also a model for $u = b$. We show that the "if" direction leads to a contradiction (the "only-if" direction does indeed hold). After slightly rewriting the "if" direction and using the appropriate definitions we have

for all $\mathcal{I}$: (if for all v: $\mathcal{I}_v(\mathcal{EQ} \cup \{u = a\}) = \mathit{true}$
then for all v: $\mathcal{I}_v(u = b) = \mathit{true}$)

implies

for all $\mathcal{I}$, for all v: (if $\mathcal{I}_v(\mathcal{EQ} \cup \{u = a\}) = \mathit{true}$ then $\mathcal{I}_v(u = b) = \mathit{true}$).

By contraposition we obtain the equivalent form

for some $\mathcal{I}$, for some v: $(\mathcal{I}_v(\mathcal{EQ} \cup \{u = a\}) = \mathit{true}$ and $\mathcal{I}_v(u = b) = \mathit{false})$

implies

for some $\mathcal{I}$: (for all v: $\mathcal{I}_v(\mathcal{EQ} \cup \{u = a\}) = \mathit{true}$
and for some v: $\mathcal{I}_v(u = b) = \mathit{false}$)

The premise of this implication can easily be satisfied: take the universe $\{a', b'\}$, interpret the constants a and b with a' and b', respectively, and let the interpretation of "=" be $\{\langle a, a\rangle, \langle b, b\rangle\}$. Finally, with the assignment $v(u) = a$ it is easily verified that the premise holds. It is essential that a and b are interpreted with *different* values. At the conclusion we do not have this freedom (the left side of the conjunction) and a and b have to be interpreted to the same value. But then we have a contradiction to the right side of the conjunction. More precisely we conclude as follows:

Suppose the conclusion is true. Let $\mathcal{I}$ be the interpretation as claimed, $\mathcal{U} = \{a', b'\}$ with $a' \neq b'$ be the universe, and let $\mathcal{I}_=$ be the interpretation of the =-symbol. Suppose a is mapped to a', and b is mapped to b'. Since for all assignments v we are given that $\mathcal{I}_v(\mathcal{EQ} \cup \{u = a\}) = \mathit{true}$ we conclude in particular that $\langle X, a'\rangle \in \mathcal{I}_=$ (by taking $v(u) = X$ for all $X \in \mathcal{U}$) and $\langle b', a'\rangle \in \mathcal{I}_=$ (by taking $v(u) = b'$). However, by symmetry and transitivity of $=$ we find that $\langle X, b'\rangle \in \mathcal{I}_=$ for every $X \in \mathcal{U}$. Thus $\mathcal{I}_v(u = b) = \mathit{true}$ for every v. Thus a v such that $\mathcal{I}_v(u = b) = \mathit{false}$ does not exist. Contradiction.

Convention 2.3.2 (Assignments). For sentences we can slightly simplify matters concerning semantics: recall from Definition 2.2.4 that a sentence is *closed*, i.e. it contains no free variables. As a special case of Theorem 2.3.1 we obtain easily that

$$\langle \mathcal{I}, v_{\mathcal{X}}\rangle \models X \quad \text{iff} \quad \langle \mathcal{I}, \varepsilon\rangle \models X$$

for any extended interpretations $\langle \mathcal{I}, v_{\mathcal{X}}\rangle$ and $\langle \mathcal{I}, \varepsilon\rangle$ for a sentence or set of sentences X, where ε denotes the "empty" assignment, i.e. the assignment which is undefined everywhere. Thus, with the assignment playing no role in evaluating X, we are motivated to define $\mathcal{I}(X)$ as a shorthand notation for $\mathcal{I}_\varepsilon(X)$. Similarly, we will write $\mathcal{I}_{[x \leftarrow a]}$ instead of $\mathcal{I}_{\varepsilon[x \leftarrow a]}$.

The following trivial lemma collects some useful facts for later use. The easy proof makes use of the *totality* of the evaluation functions, as well as the theorem and corollaries of this section.

Lemma 2.3.1. *Let F be a formula, X be a formula or a set of formulas, and let Y be a sentence or a set of sentences.*

1. $\langle \mathcal{I}, v\rangle \not\models F$ *iff* $\mathcal{I}_v(F) = \mathit{false}$
2. $\mathcal{I} \not\models X$ *iff for some assignment v it holds* $\mathcal{I}_v(X) = \mathit{false}$
3. $\not\models X$ *iff for some extended interpretation $\langle \mathcal{I}, v\rangle$ it holds* $\mathcal{I}_v(X) = \mathit{false}$

4. $\mathcal{I} \not\models Y$ *iff* $\mathcal{I}(Y) = \mathit{false}$
5. $\not\models Y$ *iff for some interpretation* $\mathcal{I}$ *it holds* $\mathcal{I}(Y) = \mathit{false}$
6. *Let* M *be a set of sentences. Then* $M \not\models Y$ *iff for some interpretation* $\mathcal{I}$ *it holds* $\mathcal{I}(M) = \mathit{true}$ *and* $\mathcal{I}(Y) = \mathit{false}$

Lemma 2.3.2. *Let* X *be a sentence or a set of sentences. Then* $\mathcal{I} \models X$ *iff for all sentences* S *such that* $X \models S$ *we have* $\mathcal{I} \models S$.

Proof. "$\Rightarrow$": trivial.
"$\Leftarrow$": We prove the contraposition: whenever $\mathcal{I} \not\models X$ then for some sentence S we have (1) $X \models S$ and (2) $\mathcal{I} \not\models S$. Hence suppose $\mathcal{I} \not\models X$. Then by Lemma 2.3.1.4 $\mathcal{I}(X) = \mathit{false}$. Thus, for some sentence $S \in X$ it holds $\mathcal{I}(S) = \mathit{false}$, and by Lemma 2.3.1.4 again $\mathcal{I} \not\models S$ which proves (2). Since $S \in X$ it trivially holds $X \models S$ which proves (1).

2.4 Clause Logic

In automated theorem proving most research is done in a particular simple syntactic setting of formulas, namely *clause logic*. The formulas of clause logic are conjunctions of disjunctions of — possibly negated — atoms, and all variables are understood as universally quantified. It is an advantage of clause logic that it enables the design of simple proof procedures. In particular, due to the universal quantification, *unification* can be used. This was demonstrated for *resolution* in the seminal paper [Robinson, 1965b].

Fortunately, every sentence can be transformed in satisfiability preserving way into clause logic. We will summarize the transformation steps here. A detailed treatment can be found in standard textbooks on automated theorem proving (e.g. [Chang and Lee, 1973]).

2.4.1 Transformation into clause logic

Suppose some first-order signature is given. A sentence F is said to be in *prenex normal form* if it is of the form $F = Q_1 x_1 \cdots Q_m x_m G$, where $Q_i \in \{\forall, \exists\}$ (for $i = 1 \ldots m$) and G is a formula which does not contain any quantifiers. $Q_1 x_1 \cdots Q_1 x$ is also called the *prefix* of F and G is called the *matrix* of F.

Fortunately, it holds that for every sentence F a sentence F' exists such that F' is in prenex normal form and $F \equiv F'$. The proof is constructive and makes use of replacement rules which allow the movement of quantifiers "outside". For instance, if F contains a sentence $\forall x\,(P(x) \vee \forall y\, Q(y))$ this occurrence may be replaced by the sentence $\forall x \forall y\,(P(x) \vee Q(y))$. Note for this particular case that the variables x and y have to be distinct (this can always be achieved by renaming).

The next step in converting a sentence F to clause form consists of transforming the matrix of its prenex normal form F' into *conjunctive normal form*. A formula F is said to be in conjunctive normal form iff it is of the form

$$G_i = G_1 \wedge \cdots \wedge G_n \ ,$$

where every G_i (for $i = 1 \ldots n$) is of the form

$$F = L_{i,1} \vee \cdots \vee L_{i,n_i} \ ,$$

where every $L_{i,j}$ (for $j = 1 \ldots n_i$) is a *literal* (a *literal* is either an atom or a negated atom).

Here we note that the conversion of a matrix to conjunctive normal form can effectively be carried out in an logical equivalence preserving way. Essentially one has to apply various equivalences such as deMorgan's laws and distributivity of "$\wedge$" over "$\vee$" etc. The case of the "$\leftrightarrow$" is a bit problematic in that its elimination may cause exponential growth of the formula. This is the case for the naïve transformation which replaces a subformula $F \leftrightarrow G$ by, say, $(F \rightarrow G) \wedge (G \rightarrow F)$. As an alternative to this, better, polynomial transformations exists (see e.g. [Eder, 1992] for such a transformation).

Hence let F'' be the prenex normal form of F whose matrix is in conjunctive normal form. Next we want to get rid of the existential quantifiers in the prefix of F''. This is done by introducing *Skolem functions* for the existentially quantified variables. More precisely, if F'' is written as

$$F'' = \forall x_1 \cdots \forall x_{l-1} \exists x_l \, Q_{l+1} x_{l+1} \cdots Q_m x_m \; G$$

where $l \in \{1 \ldots m\}$, then F'' is converted to

$$\forall x_1 \cdots \forall x_{l-1} \, Q_{l+1} x_{l+1} \cdots Q_m x_m \; G'$$

where G' is obtained from G by replacing every occurrence of the variable x_l by the term $f(x_1, \ldots, x_{l-1})$. Here f is a $l-1$-ary function symbol — called *Skolem function* — from the given signature, which is "new" to G. That is f must be different from all other function symbols occurring in G. Intuitively, $f(x_1, \ldots, x_{l-1})$ represents the value for x_l which might depend from $x_1, \ldots x_{l-1}$.

It is clear that repeated application of this transformation terminates and finally results in a formula where all existential quantifiers are removed. Such sentences, i.e. sentences whose prefix contains only universal quantifiers, will be referred to as sentences in *Skolem normal form*. Since this can be done in a unique way we will denote "the" Skolem form of a sentence F in prenex form by $Sk(F)$. Similarly, for a set X of sentences we define $Sk(X) = \{Sk(F) \mid F \in X\}$. In order to avoid collisions among Skolem functions across sentences, $Sk(X)$ is built with the understanding that the used Skolem functions to obtain $Sk(F)$ and $Sk(G)$ for $F, G \in X$, $F \neq G$ are different.

In general the resulting sentence (or set of sentences) in Skolem normal form is not logically equivalent to the given one. This can be seen by considering e.g. $F'' = \exists x P(x)$ and a skolemized form $F''' = P(a)$. It is easy to find an interpretation $\mathcal{I}$ such that $\mathcal{I}(F'') = \mathit{true}$ and $\mathcal{I}(F''') = \mathit{false}$. Hence F'' and F''' are not logically equivalent. Instead there holds a weaker result:

Theorem 2.4.1 (Skolem Normal Form). *Let X be sentence in prenex normal or set of sentences in prenex normal form. Then X is satisfiable if and only if $Sk(X)$ is satisfiable.*

The proof can again be found in standard textbooks on logic. We note here only that the existence of the Skolem functions is guaranteed by the axiom of choice.

In automated theorem proving we are mainly interested in establishing the *validity* of a given formula, but not its satisfiability. Fortunately, the preceeding theorem poses no real obstacles for doing so: suppose we want to prove a given conjecture F to be valid. For this consider its negation, $\neg F$. As a special case of Corollary 2.3.3 it holds that F is valid iff $\neg F$ is unsatisfiable *provided that F is a sentence* (i.e. F is a closed formula). Hence the question of validity can be reduced to a question of (un)satisfiability, and in this context the theorem is applicable.

As the only restriction, one has to make a decision of how to treat free variables in the given formula prior to converting it to Skolem normal form. It should be noted that if F is a closed formula then every Skolem normal form is also closed.

In automated theorem proving one usually starts with the conversion of $\neg F$ to Skolem normal form, with the matrix being put into conjunctive normal form. In other words we deal with what is known as *clause logic*.

Definition 2.4.1 (Syntax of Clause Logic). *A closed formula in Skolem normal form whose matrix is in conjunctive normal form is said to be in* clausal form.

A clause *is a finite multiset $\{\!|L_1, \ldots, L_n|\!\}$ of literals*[9] *which is identified with any of the n possible permutations of the disjunction $L_1 \vee \cdots \vee L_n$ ($n \geq 0$). Some notational conventions: if C and D are clauses, then $C \vee D$ is the clause $C \cup D$, and if L is a literal and C is a clause, then $L \vee C$ is the clause $\{\!|L|\!\} \cup C$. In such a context, C is called the* rest clause *of $L \vee C$.*

Furthermore, if F is a formula in clausal form with matrix $F_1 \wedge \cdots \wedge F_m$, F will be identified with the set $\{F_1, \ldots, F_m\}$.

Clauses are usually categorized in the following non-disjoint way: the empty clause *contains no literals, i.e. it is the set $\{\!|\}$; it is also written as "$\Box$". For clauses different to the empty clause we define: a* Horn clause *contains at*

[9] For Resolution calculi clauses often are defined as *sets* rather than as *multi*sets. For connection methods the latter definition seems to be standard (e.g. [Bibel, 1987]). Using multisets also simplifies the lifting proof (Section A.1.1) a bit and allows for a simpler implementation

most one positive literal. A definite clause *contains exactly one positive literal. A* negative clause *contains only negative literals. A* positive clause *contains only positive literals. A* disjunctive clause *contains at least two positive literals. A clause is a* unit clause *iff it contains exactly one literal, otherwise it is a* non-unit clause.

Note that it is always possible to restore from a clause set a corresponding formula in clausal form by building a conjunction from its clauses and adding back the universal quantifiers. Furthermore, all formulas obtainable in this way are logically equivalent. This is due to the associativity and commutativity of "$\wedge$" and "$\vee$". Furthermore, in the prefix the ordering of the $\forall$-quantifiers plays no role. We can even allow common variables across clauses (although the conversion to clausal form will not produce that) because "$\forall$" distributes over "$\wedge$". This will relieve us from inventing ever new variable names in clauses.

Example 2.4.1 (Clausal Form). Consider the signature $\langle \{a, f\}, \{P, Q\}, \{x, y\} \rangle$ and the clause set $\{p(f(x)),\ p(x) \vee q(x),\ \neg q(f(a))\}$. The corresponding clausal form is

$$\forall x\, (p(f(x)) \wedge (\neg p(x) \vee q(x)) \wedge \neg q(f(a)) \ .$$

This clausal form might have been derived from the formula

$$\exists y \forall x\, (p(f(x)) \wedge (p(x) \rightarrow q(x)) \wedge \neg q(f(y)) \ .$$

All definitions and properties obtained so far in general first order logic carry over to clause logic. For instance, a *ground clause set* is a clause set whose members are all ground clauses, which in turn are multisets of possibly negated ground atoms.

2.4.2 Herbrand Interpretations

As mentioned in the previous section, the validity problem for a given formula is transformed into the unsatisfiability problem for its conversion to a clause set. That is, the problem is to prove that the resulting clause set, say M, is false in *all* interpretations over all domains. Since it is impossible to consider all of them it would be most advantageous if there were a canonical structure such that M is unsatisfiable if and only if M is false in all interpretations over this structure. Indeed, there are such canonical interpretations — *Herbrand interpretations*.

Definition 2.4.2 (Herbrand Interpretation 1). *Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature. Without loss of generality $\mathcal{F}$ can be assumed to contain at least one constant symbol. The* Herbrand universe *$\mathcal{U}_\Sigma$ of Σ is defined to be the set $Term(\mathcal{F})$, i.e. the set of all ground terms of Σ. A* Herbrand structure *for Σ is a structure $\mathcal{S}_\Sigma = \langle \mathcal{U}_\Sigma, Ident, R \rangle$, where*[10]

[10] We use usual λ-notation for functions.

$$Ident = \{Ident_f \mid f \in \mathcal{F}\}, \text{ and}$$
$$Ident_f = \lambda x_1.\cdots\lambda x_n.f(x_1,\ldots,x_n)$$
(i.e. $Ident_f$ maps any f-term to "itself")

A Herbrand interpretation *is an interpretation*

$$\mathcal{H}^*_\Sigma = \langle \mathcal{S}_\Sigma, \mathcal{I}_{Ident}, \mathcal{I}_\mathcal{P} \rangle \ ,$$

where $\mathcal{I}_{Ident}(f) = Ident_f$ *for every* $f \in \mathcal{F}$.

The constituents of a Herbrand structure and Herbrand interpretation are determined completely by the underlying signature Σ, except for the interpretations of the predicate symbols $\mathcal{I}_\mathcal{P}$. For instance in Example 2.3.1 we find for the Herbrand universe

$$\mathcal{U}_\Sigma = \{zero, succ(zero), succ(succ(zero)), \ldots, plus(zero, zero),$$
$$plus(succ(zero), zero), \ldots times(zero, zero), \ldots\} \ .$$

Concerning the interpretation of function symbols we have, for instance, $\mathcal{I}_{Ident}(plus) = \lambda x_1, \lambda x_2.plus(x_1, x_2)$. Concerning the interpretation of predicate symbols, $\mathcal{I}_\mathcal{P}$, it has become usual to identify for a given predicate symbol $P \in \mathcal{P}$ its meaning $\mathcal{I}_\mathcal{P}(P)$ with the set of just those ground atoms which are in the relation given by R. More formally this is achieved as follows:

Definition 2.4.3 (Herbrand Base, Herbrand Interpretation 2).
The Herbrand base $\mathcal{HB}(\Sigma)$ *(for a given signature* $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$*) is the set*

$$\mathcal{HB}(\Sigma) = \{P(t_1,\ldots,t_n) \mid P \in \mathcal{P}, t_1,\ldots,t_n \in Term(\mathcal{F})\} \ .$$

From now on a Herbrand interpretation may be represented by some subset $\mathcal{H}_\Sigma \subseteq \mathcal{HB}(\Sigma)$*. The intended Herbrand interpretation in the sense of Def. 2.4.2 then is defined as follows (it suffices to define* R *and* $\mathcal{I}_\mathcal{P}$*):*

$$R = \{\{\langle t_1,\ldots,t_n\rangle \mid P(t_1,\ldots,t_n) \in \mathcal{H}_\Sigma\}_P \mid P \in \mathcal{P}\}$$
$$\mathcal{I}_\mathcal{P}(P) = R_P, \qquad \text{where } R_P \in R \ .$$

In words, a Herbrand interpretation just lists the *true* atomic ground formulas, and all other ground literals are interpreted by *false*. Their falsehood can be concluded since R is assumed to be a collection of total relations. The reader should convince him- or herself that this definition meets the formal requirements. In particular, the set R is indeed a collection of relations over the given Herbrand universe.

In order to carry on Example 2.3.1 we can define a Herbrand interpretation containing

$$\{zero = zero,\ succ(zero) = succ(zero),\ zero < succ(zero),$$
$$zero < succ(succ(zero)), \ldots\} \ .$$

Our interest in Herbrand interpretation is justified by the following central theorem:

Theorem 2.4.2. *A set M of clauses is unsatisfiable if and only if M is false under all Herbrand Σ-interpretations, where Σ is determined by the function and predicate symbols and variables occurring in M.*

Thus, for purposes of automated reasoning (at least if one is interested in establishing unsatisfiability) it suffices to restrict to Herbrand interpretations.

A proof of Theorem 2.4.2 can be sketched as follows: the "only-if" direction is trivial, since if M is falsified by every interpretation, M is also falsified by every Herbrand interpretation. For the "if" direction one proves the contrapositive direction, i.e. if M is satisfiable, say by interpretation I, then there is also a satisfying Herbrand interpretation $\mathcal{I}^*$. This $\mathcal{I}^*$ is constructed in such a way that $\mathcal{I}^*(A) = \mathit{true}$ for every atom A if and only if $\mathcal{I}(A) = \mathit{true}$.

Theorem 2.4.2 does not hold for arbitrary formulas, not even for sentences. For the validity Theorem 2.4.2 we need at least formulas in Skolem normal form. Consider e.g. the formula $F = P(a) \wedge \exists x \neg P(x)$, where a is a constant. It is easy to see that F is satisfiable (take an interpretation whose universe consists of exactly two elements, say 0 and 1, and let $P(0)$ be true and $P(1)$ be false). However, no Herbrand model exists for F.

Recall that the universe of a Herbrand interpretations is committed to the set of (ground) terms of the given signature. Since we will mainly deal with Herbrand interpretations in the sequel such assignments will be central. The next definition introduces this at a more general level (most items are taken from [Lloyd, 1987]):

Definition 2.4.4 (Substitution, Instance, Ground Instance). *Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature. A* substitution (for the variables $\mathcal{X}$) *is a mapping $\sigma_{\mathcal{X}} : \mathcal{X} \mapsto Term(\mathcal{F}, \mathcal{X})$ which is the identity at all but finitely many points. Usually $\sigma_{\mathcal{X}}$ is represented by the finite set of pairs*

$$\{x \leftarrow t \mid \sigma_{\mathcal{X}}(x) = t,\ \sigma_{\mathcal{X}}(x) \neq x\} \quad ,$$

the members of which are called assignments.

A substitution $\sigma_{\mathcal{X}}$ is homomorphically extended to terms as follows:

$$\sigma(t) = \begin{cases} \sigma_{\mathcal{X}}(t) & \text{if } t \in \mathcal{X} \\ f(\sigma(t_1), \ldots \sigma(t_n)) & \text{if } t \text{ is of the form } f(t_1, \ldots, t_n). \end{cases}$$

In the following, by an expression *we mean either a term, a literal, a multiset (recall that clauses are multisets), a sequence of terms or literals or a quantifier-free formula (i.e. a formula built without quantifier). Substitutions are extended to expressions in the obvious way:*

$$\begin{aligned}
\sigma(P(t_1,\ldots,t_n)) &= P(\sigma(t_1),\ldots,\sigma(t_n)) && \textit{for an atom } P(t_1,\ldots,t_n)\\
\sigma(\neg A) &= \neg\sigma(A) && \textit{for an atom or formula } A\\
\sigma\{\!|E_1,\ldots,E_n|\!\} &= \{\!|\sigma(E_1),\ldots,\sigma(E_n)|\!\} && \textit{for multisets of expressions}\\
\sigma(E_1\cdots E_n) &= \sigma(E_1)\cdots\sigma(E_n) && \textit{for sequences of expressions}\\
\sigma(F\circ G) &= \sigma(F)\circ\sigma(G) && \textit{for formulas } F \textit{ and } G, \textit{ where}\\
& & & \circ\in\{\wedge,\vee,\leftrightarrow,\leftarrow,\rightarrow\}
\end{aligned}$$

It is usual and convenient to write $E\sigma$ instead of $\sigma(E)$, where E is an expression. An expression F is an instance *of E iff a substitution σ exists such that $E\sigma = F$. A* ground substitution (for an expression E) *is a substitution σ such that $E\sigma$ is a ground expression. F is called a* ground instance *of E iff F is an instance of E and F is ground (or, equivalently, F is an instance of E by some ground substitution).*

Example 2.4.2. Let $\mathcal{X} = \{x, y, z\}$ and $\neg P(x, g(y))$ be a literal. Applying the substitution $\sigma_{\mathcal{X}} = \{x \leftarrow f(z)\}$ yields $\neg P(f(z), g(y))$. The literal $\neg P(a, g(z))$ is an instance of $\neg P(x, g(y))$ by means of $\tau_{\mathcal{X}} = \{x \leftarrow a,\ y \leftarrow z\}$. However it is not a ground instance.

The following lemma lists some trivial consequences of Definition 2.3.3, Theorem 2.4.2 and of the preceding definition. For the easy proofs one only has to recall that clauses are universally quantified disjunctions of literals. As these facts are used quite often we will give them here explicitly:

Lemma 2.4.1 ([Chang and Lee, 1973]). *1. A ground instance C' of a clause C is satisfied by a Herbrand interpretation $\mathcal{I}$ iff there is a ground literal $L' \in C'$ such that $L' \in \mathcal{I}$, that is $C' \cap \mathcal{I} \neq \{\}$.*

2. A clause C is satisfied by a Herbrand interpretation $\mathcal{I}$ iff every ground instance of C is satisfied by $\mathcal{I}$.

3. A clause C is falsified by a Herbrand interpretation $\mathcal{I}$ iff there is at least one ground instance C' of C such that C' is not satisfied by $\mathcal{I}$.

4. A set S of clauses is unsatisfiable iff for every Herbrand interpretation $\mathcal{I}$, there is at least one ground instance C' of some clause $C \in S$ such that C' is not satisfied by $\mathcal{I}$.

The next theorem is an extremely important "tool" for completeness proofs of first-order calculi. It allows the reduction of the problem of finding a proof of a clause set to the problem of finding a proof of a *finite* set of ground instances of that clause set.

It is according to Skolem, Herbrand and Löwenheim[11]

[11] In the literature one can often find this theorem cited as "Herbrand's theorem". This is not quite accurate since Herbrand has developed a similar theorem which is formulated in a proof-theoretic version, and not in a model-theoretic version as this one.

Theorem 2.4.3 (Skolem, Herbrand, Löwenheim Theorem). *A clause set M is unsatisfiable if and only if some* finite *set M' of ground instances of clauses from M is unsatisfiable.*

For a proof see e.g. [Gallier, 1987] or [Chang and Lee, 1973].

On the one hand, this theorem enables a "ground-proof and lifting technique" for completeness proofs of first-order calculi. This technique will be employed several times throughout this paper and will be described in more detail below.

On the other hand, it can be the base for an implementation according to the following scheme: in order to prove a given clause set M as unsatisfiable one has to systematically enumerate all (finite) sets of all ground instances of all clauses in M, whereby *deciding* the validity of each enumerated set[12]. If M is indeed unsatisfiable, by Theorem 2.4.3 such an unsatisfiable set M' of ground instances will be enumerated eventually. Then deciding M' as unsatisfiable will render M itself as unsatisfiable.

There is a whole class of proof procedures following this line, e.g. the *Davis-Putnam procedure* [Davis and Putnam, 1960]. These procedures, however, were abandoned more or less once the resolution principle was invented [Robinson, 1965b], because it allows us to carry out the proof search directly on clauses with variables by the device of *unification*. Only recently, interest in Herbrand proof procedures came up again. D. Plaisted and his coworkers have developed several variants of the *Hyper-linking* calculus [Lee and Plaisted, 1992; Chu and Plaisted, 1994]. The new idea wrt. Davis and Putnam's procedure is in the way the ground instances are generated. A clever enumeration scheme, joined with an optimized variant of the Davis-Putnam decision procedure for propositional logic allows for the proof of many theorems which are difficult in the field of automated theorem proving.

2.4.3 Unification

As with resolution, most calculi used in automated theorem proving operate on the variable-level and employ *unification* as the basic operation. Our calculi described in subsequent chapters will also do so, although, traditional unification will be replaced by a more general concept ("theory unifiers"). As a base for that, and for the sake of completeness, we will briefly recall some standard definitions. For their place within first-order calculi the reader is referred to Chapter 4 below.

Definition 2.4.5 (Operations on Substitutions, Variant, Unifier). *In the following we mean by an expression either a term or a literal. The* composition *of two substitutions σ and τ is written by juxtaposition $\sigma\tau$. It is*

[12] this can be done effectively since a finite set of (finite) ground clauses essentially is a *propositional* formula which can be decided e.g. by using truth tables.

defined via the equivalence $x(\sigma\tau) = (x\sigma)\tau$. *That is, for every expression* E, $E\sigma\tau = (E\sigma)\tau$.

The empty *substitution* ε *is to be defined the identity function everywhere.*

Let $\mathcal{X}$ *and* V *sets of variables with* $V \subseteq \mathcal{X}$, *and let* σ *be a substitution. The* restriction of σX wrt. V, *written as* $\sigma|V$ *is defined as follows:*

$$(\sigma|V)(x) = \begin{cases} \sigma(x) & \text{if } x \in V \\ x & \text{if } x \notin V \end{cases} .$$

That is, the restriction of a substitution forgets about the assignments outside V.

Occasionally we need the sets

$$\begin{aligned} Dom(\sigma) &= \{x \mid x \in \mathcal{X},\ \sigma(x) \neq x\} && \text{(domain of } \sigma) \\ Cod(\sigma) &= \{t \mid x \in Dom(\sigma),\ \sigma(x) = t\} && \text{(codomain of } \sigma) \\ VCod(\sigma) &= \bigcup_{t \in Cod(\sigma)} Var(t) && \text{(variable codomain of } \sigma) \end{aligned}$$

The function $Var(E)$ *denotes the set of all variables occuring in an expression* E. *It is extended to sets or multisets* S *of expressions by* $Var(S) = \bigcup_{E \in S} Var(E)$. *For substitutions, we define* $Var(\sigma) = Dom(\sigma) \cup VCod(\sigma)$.

We say that a substitution σ *is* away from (a set of variables) V *iff* $VCod(\sigma) \cap V = \emptyset$.

Quite often we are interested in substitutions which coincide with others on a certain set of variables. Hence we define $\sigma = \gamma\ [V]$ *iff* $\sigma|V = \gamma|V$. *A substitution* σ *is said to be* more general on the variables V *than the substitution* δ, *written as* $\sigma \leq \delta\ [V]$, *if for some substitution* γ *it holds* $\sigma\gamma = \delta\ [V]$. *By* $\sigma \leq \delta$ *we mean* $\sigma \leq \delta\ [\mathcal{X}]$.

We say that expressions E *and* F *are* variants *if substitutions* σ *and* δ *exist such that* $E = F\sigma$ *and* $F = E\delta$. *A* renaming substitution *is a substitution* ρ *such that* $Cod(\rho) \subseteq \mathcal{X}$, *i.e.* ρ *replaces variables by variables, and* $\rho(x) = \rho(y)$ *implies* $x = y$ *for all* $x, y \in Dom(\rho)$. *As a consequence, because* ρ *is a bijection in* $\mathcal{X}$, ρ *can be inverted, which is defined to be the substitution* ρ^{-1}, *with* $\rho\rho^{-1} = \varepsilon\ [Dom(\rho)]$.

It holds that E *and* F *are variants iff renaming substitutions* ρ *for* E *and* τ *for* F *exists such that* $F = E\rho$ *and* $E = F\tau$ *[Lloyd, 1987].*

Two expressions E *and* F *are said to be* unifiable *if a substitution* σ *exists such that* $E\sigma = F\sigma$. *In this case* σ *is called a* unifier for E and F; σ *is called a* most general unifier (MGU) *iff* $\sigma \leq \sigma'$ *for every other unifier* σ' *of* E *and* F.

Unification is extended to multisets of literals as follows: a substitution σ *is a unifier for literal multisets* N *and* M *iff* $N\sigma = M\sigma$. *Multiset unification is of type "finitary" (i.e. results in a finite complete set of multiset-MGUs). See [Büttner, 1986] for an unification algorithm.*

Note 2.4.1 (Unification and Variable Restrictions). It can be verified that "$\leq$" is indeed a partial ordering. The variable restriction V in $\sigma \leq \delta\ [V]$ is indeed necessary sometimes, as $\sigma \leq \delta$ might not hold although we would expect so. For instance, if $\sigma = \{x \leftarrow y\}$ and $\delta = \{x \leftarrow a\}$ then a substitution γ does *not* exist such that $\sigma\gamma = \delta$ (on all variables !). The "solution" $y \leftarrow a$ does not work because then

$$\sigma\gamma = \{x \leftarrow a,\ y \leftarrow a\} \neq \{x \leftarrow a\} = \delta\ .$$

However, $\sigma \leq \delta\ [\{x\}]$ holds. Notably, such a restriction to certain variables is *not* necessary in the context of MGUs. That is, if δ is a unifier for expressions E and F, then a MGU σ and a substitution γ always exists such that $\sigma\gamma = \delta$. This is known as the *unification theorem*. A respective algorithm was given first in [Robinson, 1965b]. See [Knight, 1989; J.-P. Jouannaud, 1991] for overviews about unifications.

Obviously, *multiset unification* can be reduced to syntactic unification, because δ is a multiset-MGU for $\{\!|L_1, \ldots, L_n|\!\}$ and $\{\!|K_1, \ldots, K_n|\!\}$ iff δ is a (syntactic) unifier for $c(L_1, \ldots, L_n)$ and $c(K_{\pi(1)}, \ldots, L_{\pi(n)})$, where π is some permutation of $1, \ldots, n$. From this reduction to syntactic unification we thus conclude that for any multiset unifier δ a multiset-MGU σ exists and a substitution γ such that $\sigma\gamma = \delta$ (without variable restriction). However, introducing variable restrictions becomes necessary again in the context of "theory-MGU"s (see Section 4.2.1).

Example 2.4.3 (Unification). The terms $t(x)$ and $t(y)$ are unifiable by MGU $\sigma = \{x \leftarrow y\}$. The unifier $\delta = \{x \leftarrow a,\ y \leftarrow a\}$ is not an MGU because for $\gamma = \{y \leftarrow a\}$ we find $\sigma\gamma = \delta$. The terms $t(a)$ and $t(b)$ are not unifiable.

2.5 Theories

Next we will formalize one of our central concepts, namely the concept of a *theory*. The material presented here is a condensed form of the respective chapters of the textbooks [Enderton, 1972; Heinemann and Weihrauch, 1991; Lalement, 1993].

We will start with a brief motivation. Then we will formally define the notion of a "theory" and will give methods how theories can be defined. Then we turn towards properties theories can have.

In Section 2.3 we have introduced the semantical notion of *validity*, which means that a formula is *true* in all interpretations. However, for a mathematician the question of whether some given formula is valid in this sense is often not so relevant. Instead he asks whether his conjecture is true in *some* dedicated interpretation. For instance, he will be interested in assertions about the *natural numbers* or *real numbers*. Or he asks whether a certain assertion is a logical consequence of some given axioms, e.g. the axioms of group theory. Now, the notion of a *theory* allows the expression of these applications in a convenient way.

Definition 2.5.1 (Contradictory Sets of Sentences, Theory).
Let $M \subseteq \mathcal{CPF}(\Sigma)$ be a subset of the sentences of a given signature. M is called contradictory *iff for some $S \in \mathcal{CPF}(\Sigma)$ both $M \models S$ and $M \models \neg S$. If M is not contradictory it is called* consistent[13] *(iff for every $S \in \mathcal{CPF}(\Sigma)$ not both $M \models S$ and $M \models \neg S$).*

*M is called a Σ-*theory *iff (1) M is consistent and (2) for every $S \in \mathcal{CPF}(\Sigma)$ it holds that $M \models S$ implies $S \in M$*

That is, a theory is a consistent set of sentences which is closed under logical consequence.

Note 2.5.1. The notion of consistency is also used in the literature as a "syntactic" property: then, a formula is consistent if no contradiction is *derivable* (using some calculus) from it. However, in usual calculi by soundness and Gödel-completeness the "syntactic" and our model-theoretic "semantic" notions coincide.

The "syntactic" approach is taken for instance in [Lalement, 1993]. Occasionally one finds that property (1) is omitted, e.g. in [Gallier, 1987; Enderton, 1972]. The difference is not that crucial, since in this case (for a given language Σ) only *one single* contradictory theory exists, namely $\mathcal{CPF}(\Sigma)$. Reason: If a theory $\mathcal{T}$ is contradictory, it is unsatisfiable (by Lemma 2.5.1.1 below). Hence, by definition of logical consequence $\mathcal{T} \models S$ for every $S \in \mathcal{CPF}(\Sigma)$. Thus by closure property (2) $\mathcal{T} = \mathcal{CPF}(\Sigma)$.

It should be noted that concerning property (2) there seems to be an agreement in the literature that it is always part of the definition.

In order to state a positive example for a theory consider the set $\mathcal{T} = \{S \mid \models S\}$ of all tautologies. $\mathcal{T}$ is a theory (it is even the *smallest* theory). In order to see this let $M = \{\}$ in Lemma 2.5.1.2 below.

Lemma 2.5.1 (Definition of Theories). *1. Let $M \subseteq \mathcal{CPF}(\Sigma)$ be a set of sentences. M is consistent iff M is satisfiable.*

2. *Let $Ax \subseteq \mathcal{CPF}(\Sigma)$ be a non-contradictory set of sentences. Then*

$$Cons(Ax) = \{S \in \mathcal{CPF}(\Sigma) \mid Ax \models S\}$$

is a theory (this is the axiomatic method *of defining a theory. Ax is called a* set of axioms *or* axiomatization*).*

3. *Let $\mathcal{I}$ be a Σ-interpretation. Then*

$$Th(\mathcal{I}) = \{S \in \mathcal{CPF}(\Sigma) \mid \mathcal{I} \models S\}$$

is a theory (also called the theory of $\mathcal{I}$*. This is the* model theoretic method *of defining a theory).*

Before we will prove this we need a lemma:

[13] In German: "widerspruchsfrei"

Lemma 2.5.2. *Let $\mathcal{I}$ be a Σ-interpretation and $M \subseteq \mathcal{CPF}(\Sigma)$. Then $\mathcal{I} \models M$ iff $\mathcal{I} \models Cons(M)$.*

Proof.
$\mathcal{I} \models M$
iff $\mathcal{I} \models S$ for every sentence S with $M \models S$ (by Lemma 2.3.2)
iff $\mathcal{I} \models \{S \mid M \models S\}$
iff $\mathcal{I} \models Cons(M)$

Proof. (Lemma 2.5.1) 1. "$\Leftarrow$": We prove the contraposition: if M is contradictory then for some $S \in \mathcal{CPF}(\Sigma)$ it holds both $M \models S$ and $M \models \neg S$. Now, if M were satisfiable then for some model $\mathcal{I}$ of M we would have both $\mathcal{I} \models S$ and $\mathcal{I} \models \neg S$ which is impossible. Hence M is unsatisfiable.
"$\Rightarrow$": We prove the contraposition: if M is unsatisfiable then it holds trivially $M \models S$ for every $S \in \mathcal{CPF}(\Sigma)$. So in particular $M \models \neg S$. Hence M is contradictory.

2. In order to see that $Cons(Ax)$ is a theory we have to show that (1) $Cons(Ax)$ is consistent and that (2) obeys the closure property.

Ad (1): Ax is consistent (as given) iff Ax is satisfiable (by this lemma item 1. iff $Cons(Ax)$ is satisfiable (by Lemma 2.5.2) iff $Cons(Ax)$ is consistent (by this lemma item 1. again).

Ad (2): $Cons(Ax) \models S$ iff $Ax \models S$ (consequence of Lemma 2.5.2) iff $S \in Cons(Ax)$ (by definition).

3. Clearly $Th(\mathcal{I})$ is consistent since no interpretation can satisfy both F and $\neg F$. For the closure property suppose, to the contrary, that for some sentence S we have $Th(\mathcal{I}) \models S$ (*) but $S \notin Th(\mathcal{I})$. From $S \notin Th(\mathcal{I})$ it follows by definition $\mathcal{I} \not\models S$ (**). On the other side, $Th(\mathcal{I})$ consists precisely of those sentences being *true* in $\mathcal{I}$, i.e. $\mathcal{I} \models Th(\mathcal{I})$ holds trivially. But then using (*) we conclude $\mathcal{I} \models S$ which contradicts to (**).

Property 1 gives an alternative characterisation of noncontradictoriness. Satisfiability of a given set of formulas may be much easier to establish than noncontradictoriness. For example, every clause set consisting of definite clauses is satisfiable (and hence consistent). This can be seen by taking the interpretation that assigns *true* to every positive literal and *false* to every negative literal.

The properties 2 and 3 give methods how theories can be defined. For this we will supply examples.

2.5.1 Sample theories

Let us motivate by some examples the methods of Lemma 2.5.1 of defining theories. We will concentrate on theories which are of real interest for the working mathematician. For some of these we will show in Chapter 5 how to automatically derive an efficient background calculus from their axiomatization.

These examples are centered around "equality", that is, the underlying signature contains a special 2-ary predicate symbol — usually "="— which is to be interpreted as an equality relation. One way to achieve this is to add the *equality axioms* to a given set of formulas. These shall be defined next:

Definition 2.5.2 (Equality Axioms). *Let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature.*

1. *The set $FSUB(\Sigma) \subseteq \mathcal{PF}(\Sigma)$ of* functional substitution axioms for Σ *is the following set*[14]*:*

$$FSUB(\Sigma)\text{: } \forall x_1, \ldots, x_n, y_1, \ldots, y_n : x_1 = y_1 \wedge \cdots \wedge x_n = y_n \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$$

for every n-ary function symbol $f \in \Sigma$

2. *The set $PSUB(\Sigma) \subseteq \mathcal{PF}(\Sigma)$ of* predicate substitution axioms for Σ *is the following set:*

$$PSUB(\Sigma)\text{: } \forall x_1, \ldots, x_n, y_1, \ldots, y_n : x_1 = y_1 \wedge \cdots \wedge x_n = y_n \wedge P(x_1, \ldots, x_n) \rightarrow P(y_1, \ldots, y_n)$$

for every n-ary predicate symbol $P \in \Sigma$

3. *The* equality axioms for Σ, *$EAX(\Sigma)$ is the set*

$$EAX(\Sigma) = \mathcal{EQ} \cup FSUB(\Sigma) \cup PSUB(\Sigma)$$

Concerning the set $\mathcal{EQ}$ (equivalence relation) see Example 2.3.2 above.

Now we will define some theories.

The Theory of Equality. Let Σ be a signature. Since $EAX(\Sigma)$ is satisfiable (take an interpretation which assigns *true* to every Atom) and hence consistent, the set

$$Cons(EAX(\Sigma)) = \{S \in \mathcal{CPF}(\Sigma) \mid EAX(\Sigma) \models S\}$$

is by Lemma 2.5.1.2 a theory, called the *theory of equality (for Σ)*. By overloading of notation, we denote it by $\mathcal{E}(\Sigma)$ or simply $\mathcal{E}$.

Equational Theories. Let Σ be a signature. The bare theory of equality can be extended to an *equational theory*. Equational theories are usually identified with an axiom set E of equations[15] which are implicitly considered as universally quantified. To be precise, the equational theory of E is the set

$$Cons(E \cup EAX(\Sigma)) = \{S \in \mathcal{CPF}(\Sigma) \mid E \cup EAX(\Sigma) \models S\} \ .$$

An important instance are AC-theories, that is,

$$E = \{f(x, f(y, z)) = f(f(x, y), z),\ f(x, y) = f(y, x)\}$$

for a given 2-ary function symbol f. See [Siekmann, 1989] for an overview over this area, which is also called *unification theory*.

[14] In order to be completely accurate the variables occuring in these formulas must be contained in $\mathcal{X}$. This will be presupposed here and in similar situations below.

[15] A (Σ-)equation is a Σ-Atom whose predicate symbol is "=".

Group theory. Let $\mathcal{G} = \langle \{=\}, \{e, \circ\}, \{x, y, z, \dots\} \rangle$ be an signature. Let $GA \subseteq \mathcal{PF}(\mathcal{G})$ be the following set:

$$\begin{array}{lrrcl} GA\text{:} & \forall x \forall y \forall z & (x \circ y) \circ z & = & x \circ (y \circ z) \\ & \forall x & e \circ x & = & x \\ & \forall x \exists y & y \circ x & = & e \ . \end{array}$$

The elements of GA are called *group axioms*. Then, building on 1., *group theory* is the theory

$$GrTh = Cons(GA \cup EAX(\mathcal{G})) \ .$$

Note that GA is *not* an equational theory because of the existential quantifier in the third line. However, it can be converted into an equational theory by means of a Skolem function.

Orderings. The following are relevant axioms for orderings:

$$\begin{array}{lll} O\text{:} \quad \forall x \forall y & x < y \ \wedge \ y < x \ \rightarrow \ x < z & \text{(Transitivity)} \\ \forall x & \neg(x < x) & \text{(Irreflexivity)} \\ \forall x \forall y & x < y \ \vee \ x = y \ \vee \ y < x & \text{(Trichotomy)} \\ \forall x \forall z & (x < z \ \rightarrow \ \exists y (x < y \ \wedge \ y < z) & \text{(Density)} \\ \forall x \exists y & y < x & \text{(No left endpoint)} \\ \forall x \exists y & x < y & \text{(No right endpoint)} \ . \end{array}$$

Transitivity and Irreflexivity alone axiomatize *strict (partial) orderings*, below referred to by $\mathcal{SO}$. If we add trichotomy (and equality) to $\mathcal{SO}$ we obtain strict *total* (or *linear*) orderings. All axioms together constitute *dense linear orderings*.

Arithmetic. Let $\mathcal{N}$ and NAT as in Example 2.3.1. Then, according to Lemma 2.5.1.3 the set

$$Th(NAT) = \{S \in \mathcal{CPF}(\mathcal{N}) \ \mid \ NAT \models S\}$$

is the theory of NAT, also called *arithmetic*.

Peano Arithmetic. In close relationship to the preceding example let $\mathcal{N}'$ be the signature $\langle \{0, s, +, *\}, \{=, <\}, \{x, y, z, \dots\} \rangle$. *Peano Arithmetic* is the following theory $Cons(PA)$:

$$\begin{array}{lll} PA\text{:} \quad \forall x & \neg(s(x) = 0) & \\ \forall x \forall y & s(x) = s(y) \ \rightarrow \ x = y & \\ \forall x & x + 0 = x & \\ \forall x \forall y & x + s(y) = s(x + y) & \\ \forall x & x * 0 = 0 & \\ \forall x \forall y & x * s(y) = x * y + x & \\ \forall x & \neg(x < 0) & \\ \forall x \forall y & x < s(y) \leftrightarrow (x < y \ \vee \ x = y) & \\ & (\Phi(0) \ \wedge \ \forall x \, (\Phi(x) \rightarrow \Phi(s(x)))) \rightarrow \forall x \, \Phi(x) & \end{array}$$

The last line is not a formula but an *axiom scheme* (the induction axiom scheme). The expression $\Phi(x)$ stands for any formula containing a free variable x. Given a signature with countably many variables, there are countably many instances of the scheme. Thus, PA is an infinite set of formulas.

A weaker theory with fewer axioms can be obtained by replacing the induction scheme with the trichotomy axiom (cf. the previous example). This formula cannot be proved from the other ones. The resulting system is called "elementary arithmetic". An even simpler theory called "Presburger Arithmetic" is obtained from elementary arithmetic by removing the axioms for multiplication.

2.5.2 Properties of Theories

For our interest — theorem proving modulo a built-in theory — it is most important to know about the decidability properties of the theory in question to be used as a background theory. For instance, (and in preview to subsequent sections) in order to get a complete calculus for the combined theory the background theory must at least be semi-decidable[16]. Further, if one wants to check a given proof in such a calculus for correctness the theory must even be decidable.

Definition 2.5.3 (Complete Theory). *A theory $\mathcal{T}$ is called* complete *iff for every sentence $S \in \mathcal{CPF}(\Sigma)$ it holds $S \in \mathcal{T}$ or $\neg S \in \mathcal{T}$.*

Note that a theory cannot contain both, S and $\neg S$ for some sentence S. Hence a complete theory contains exactly one of S, $\neg S$ for every sentence S.

The *model theoretic* way to define a theory is always complete: let $\mathcal{I}$ be an interpretation (over some signature). Then the theory of $\mathcal{I}$, $Th(\mathcal{I})$ is complete, since by totality of the evaluation function either a sentence S is *true* in $\mathcal{I}$ or it is *false* in $\mathcal{I}$ (and hence $\neg S$ is *true* in $\mathcal{I}$). In the former case we have $S \in \mathcal{I}$, while in the latter case we have $\neg S \in \mathcal{I}$. Thus, for instance, the theory of arithmetic ($Th(NAT)$) is complete.

The *axiomatic* way to define a theory is not necessarily complete. For instance, for the atom A neither $\models A$ nor $\models \neg A$. Hence the theory $Cons(\{\})$ of all tautologies is not complete.

---

[16] The notion of a *recursive* relation or set will not be formally defined in this book. The same holds for the informal counterpart of a *decidable* relation or set. We adopt the standard viewpoint (see [Enderton, 1972]) of a set to be decidable iff there is a (terminating and correct) algorithm for its characteristic function (i.e. the set-membership predicate). In our context, a theory $\mathcal{T}$ is decidable iff an algorithm to determine whether $F \in \mathcal{T}$ or $F \notin \mathcal{T}$ exists, for arbitrary given sentence F. A set which is not decidable is also called *undecidable*, and it is *semi-decidable* iff there is a (correct) effective procedure for its characteristic function and which terminates for any member of the set. The formal counterpart to semi-decidability is *recursive enumerability*.

In this book we are mainly concerned with the axiomatic method of defining theories. Interesting properties evolve for theories whose axioms are decidable. Then we arrive at *axiomatizable theories*:

Definition 2.5.4. *A theory $\mathcal{T}$ is* axiomatizable *iff there is a decidable set Ax of sentences such that $\mathcal{T} = Cons(Ax)$. $\mathcal{T}$ is* finitely axiomatizable *iff $\mathcal{T} = Cons(Ax)$ for some finite set Ax of sentences.*

Note that a theory being axiomatizable (i.e. it admits a decidable set of axioms) does not necessarily imply that the theory itself is decidable. For instance, Peano Arithmetic, elementary arithmetic, group theory (all defined in the previous Section 2.5.1) and predicate calculus (the theory $Cons(\{\})$) all are axiomatizable yet not decidable. Elementary arithmetic is even *essentially undecidable* (proved by Church) which means that it remains undecidable for every (consistent) extension of its axiomatisation.

On the other hand, Presburger Arithmetic, dense linear orderings and *commutative* group theory (i.e. for the group operator it also holds $\forall x, y x \circ y = y \circ x$) is decidable (thus group theory is not essentially undecidable). See [Rabin, 1977] for methods of how to obtain decision procedures. For instance, the *quantifier elimination method* means to show that for any formula F there exists a quantifier-free formula G such that $\mathcal{T} \models (F \leftrightarrow G)$. For this it suffices to consider only sentences of the form $\exists(L_1 \wedge \cdots \wedge L_n)$ with the L_is being literals. If this holds, we can then in many model theoretically given theories $\mathcal{T}$ simply evaluate G in order to decide if F is a member of T.

For decision procedures based on theorem proving techniques, sometimes *resolution* can be used (see [Joyner, 1976; Tammet, 1992]). Other examples for decidable theories are some equational theories, e.g. associative theories, associative-commutative theories and many others (see again [Siekmann, 1989] for an overview). These theories are of particular interest within automated theorem proving as quite efficient decision procedures, or even unification algorithms are known. The same holds for interesting classes of *taxonomical* theories (see again [Tammet, 1992]).

The following theorem lists computability results for axiomatizable theories:

Theorem 2.5.1 (Computability Results for Theories).

1. *A theory is axiomatizable iff it is semi-decidable.*
2. *An axiomatizable and complete theory is decidable.*

Proof. 1. "⇒" Let $\mathcal{T}$ be the given theory. Since $\mathcal{T}$ is axiomatizable a decidable set Ax of sentences such that $\mathcal{T} = Cons(Ax)$ exists.

We give a procedure for enumerating the members of the theory, which clearly constitutes a semi-decision procedure: take any deduction complete calculus for first order logic (e.g. Natural Deduction) and enumerate all tautologies. This is possible due to Gödels completeness theorem. Whenever a

sentence of the form $(F_1 \wedge \cdots \wedge F_n) \rightarrow F$ is enumerated, decide whether $\{F_1, \ldots, F_n\} \subseteq Ax$. If so, F is a member of the theory, and this procedure will produce every member. Notice that for this direction it suffices even that Ax is semi-decidable.
"$\Leftarrow$" Let $F_1, F_2, \ldots, F_n, \ldots$ be an enumeration of $\mathcal{T}$. Such an enumeration exsist because $\mathcal{T}$ is semi-decidable. Then Ax can be constructed as follows:

$$Ax = \{\ F_1$$
$$F_2 \wedge F_2$$
$$F_3 \wedge F_3 \wedge F_3$$
$$\vdots$$
$$\underbrace{F_n \wedge \cdots \wedge F_n}_{n \text{ times}}$$
$$\vdots \qquad\qquad \}$$

Obviously, Ax is logical equivalent to $\mathcal{T}$. Furthermore, for every F_i there is by construction only a *finite* number of formulas in Ax which are smaller (in a suitable complexity measure, e.g. by counting the formulas' symbols) than F_i. Hence, for a given formula F we have to enumerate Ax only so far until all formulas which are less than or equal (in this complexity measure) are generated. We can decide whether F is contained in this finite set. Hence Ax is an axiomatization for $\mathcal{T}$.

2. Let $\mathcal{T}$ be the given axiomatizable and complete theory. In order to decide whether $S \in \mathcal{T}$ for some sentence S the following algorithm can be used: by completeness either $S \in \mathcal{T}$ or $\neg S \in \mathcal{T}$. Hence by the procedure given in 1., either S or $\neg S$ will be generated after a finite number of steps. In the former case we have $S \in \mathcal{T}$, while in the latter case we have $S \notin \mathcal{T}$

The above mentioned decidability results can be rephrased in the terminology of theories by using Theorem 2.5.1. For instance, the axiom system for elementary arithmetic is finite and hence decidable. Thus elementary arithmetic is axiomatizable. But it must be incomplete, since otherwise, by Theorem 2.5.1.2, elementary arithmetic would be decidable. Since elementary arithmetic is even essentially undecidable, every axiomatizable extension must be incomplete (this result is also known as Gödels first incompleteness theorem). As a consequence, Peano arithmetic is also incomplete (because it is an extension of elementary arithmetic). This means that there are sentences S which are *true* or *false* in *Nat* (Example 2.3.1) but neither S nor $\neg S$ are contained in Peano arithmetic.

For a positive example we can mention the theory of dense linear orderings without endpoints: it can be shown that this theory is complete, and with being axiomatizable it is by Theorem 2.5.1.2 also decidable. The same line of reasoning holds for the theory of natural numbers with successor and equality. The abovementioned quantifier elimination method is applicable in this case.

The relationships among the discussed concepts as well as some trivial ones are summarized in Figure 2.1.

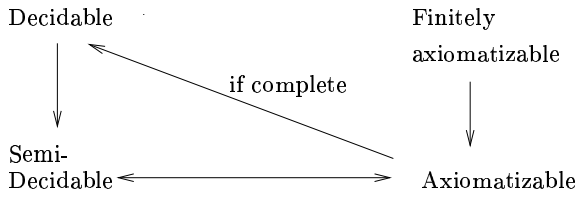


Figure 2.1. Relationships among theories.

2.5.3 Universal Theories and Herbrand Theory Interpretations

Next we will introduce a special class of theories, namely *universal theories*. Roughly, these are theories which can be axiomatized by a set of sentences which do not contain any existential quantifiers. This class of *universal theories* is important for us because a theory-version of the Skolem-Herbrand-Löwenheim theorem (Theorem 2.4.3) holds for it (thus, also theorem proving in the context of theories can be based on ground instances). Of course, this all will be made precise below.

Let us first introduce universal theories.

Definition 2.5.5 (Universal Theories). *A Σ-sentence F in prenex normal form (cf. Section 2.4.1) is called a* universal (Σ-)sentence *iff it is of the form $F = \forall x_1 \cdots \forall x_m\, G$, where G is the matrix of F*[17]*. A (Σ-)theory $\mathcal{T}$ is called* universal *iff $\mathcal{T} = Cons(Ax)$ for some axiom set Ax of universal Σ-sentences.*

Example 2.5.1. Consider again the sample theories of Section 2.5.1. The **theory of equality** is universal by virtue of the equality axioms, $EAX(\Sigma)$. **Group theory** is not universal due to the axiom $\forall x \exists y\, y \circ x = e$. However, if we replace this axiom by $\forall x\, i(x) \circ x = e$ where i is a new Skolem function, we arrive at a universal theory which is satisfiable if and only if group theory is (cf. Section 2.4.1).

Orderings: Strict partial orderings are universal, while dense linear orderings are not, etc.

Before we come to the above-mentioned Herbrand theorem we will introduce some notation which will be very convenient in the following. It is motivated by the standard situation in theorem proving with built-in theories, where

[17] Note that a sentence in Skolem normal form (Section 2.4.1) is a universal sentence.

one has given a (semi-) decision procedure for a theory $\mathcal{T}$ (e.g. group theory), a clause set M of hypothesis (e.g. stating the commutativity of the group operation) and a particular theorem Q to be proved. The question then is whether Q is a logical consequence of M with respect to $\mathcal{T}$. Let us define this more formally.

Definition 2.5.6 ($\mathcal{T}$-Interpretation, -Satisfiability, -Validity, -Model). *Let $\mathcal{T}$ be a Σ-theory. A Σ-interpretation $\mathcal{I}$ is called a* $\mathcal{T}$-(Σ-)interpretation *iff $\mathcal{I} \models \mathcal{T}$. A* Herbrand $\mathcal{T}$-Σ-interpretation *is a $\mathcal{T}$-Σ-interpretation which is also a Herbrand interpretation (Def. 2.4.3).*

Now let M be a set of Σ-sentences. We extend Definition 2.3.3 towards theories in the following way.

A $\mathcal{T}$-Σ-interpretation $\mathcal{I}$ is called a $\mathcal{T}$-(Σ-)model for M *iff $\mathcal{I} \models M$.*

M is called $\mathcal{T}$-(Σ-)satisfiable *iff $\mathcal{I} \models M$ for some $\mathcal{T}$-Σ-interpretation I. Otherwise M is called $\mathcal{T}$-(Σ-)unsatisfiable.*

M is called $\mathcal{T}$-(Σ-)valid iff $\mathcal{I} \models M$ for every $\mathcal{T}$-Σ-interpretation $\mathcal{I}$. This is written as $\models_{\mathcal{T}} M$

Let X be a Σ-sentence or a set of Σ-sentences. X is called a logical $\mathcal{T}$-consequence of M, *written as $M \models_{\mathcal{T}} X$ iff for every $\mathcal{T}$-Σ-interpretation I: $\mathcal{I} \models M$ implies $\mathcal{I} \models X$.*

These notions shall also be defined with respect to Herbrand interpretations by referring to Herbrand $\mathcal{T}$-Σ-interpretations instead of $\mathcal{T}$-Σ-interpretations.

The next proposition rephrases some model-theoretic facts in the current terminology. It demonstrates that everything is defined as one would expect naturally.

Proposition 2.5.1 (Model-theoretical Facts About Theories). *Let $\mathcal{T}$ be a Σ-theory, M be a set of Σ-sentences and X be a Σ-sentence or a finite set of Σ-sentences.*

1. *Let $Taut = Cons(\emptyset)$ (wrt. the signature Σ), i.e. the set of all $\sum$-tautologies. Then $M \models_{Taut} X$ iff $M \models X$.*
2. *$\emptyset \models_{\mathcal{T}} X$ iff $\models_{\mathcal{T}} X$.*
3. *Suppose $\mathcal{T} = Cons(Ax)$ for some $Ax \subseteq \mathcal{CPF}(\Sigma)$. Then the following are equivalent:*
   a) *$M \cup \overline{X}$ is $\mathcal{T}$-unsatisfiable (cf. Definition 2.2.6).*
   b) *$M \models_{\mathcal{T}} X$.*
   c) *$M \cup \mathcal{T} \models X$.*
   d) *$M \cup Ax \models X$.*
   e) *$M \cup Ax \cup \overline{X}$ is unsatisfiable.*

Item 1 shows that logical consequenceship without theories is a special case of the theory-version. Item 2 shows that validity is defined as expected. Item 3(a) shows that proving consequenceship can be reduced to proving unsatisfiability. Items 3(a)–(e) show how theory-reasoning can be reduced to non-theory reasoning.

Proof. **1.** $M \models_{Taut} X$
iff $M \cup \emptyset \models X$ (by item 3, equivalence (a)-(c))
iff $M \models X$.

2. $\emptyset \models_{\mathcal{T}} X$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I} \models \emptyset$ implies $\mathcal{I} \models X$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I} \models X$ ($\mathcal{I} \models \emptyset$ is vacuously true)
iff $\models_{\mathcal{T}} X$.

3. "(a) iff (b)":
$M \cup \overline{X}$ is $\mathcal{T}$-unsatisfiable
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I}(M \cup \overline{X}) = \mathit{false}$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I}(M) = \mathit{false}$ or $\mathcal{I}(\overline{X}) = \mathit{false}$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I}(M) = \mathit{true}$ implies $\mathcal{I}(\overline{X}) = \mathit{false}$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I}(M) = \mathit{true}$ implies $\mathcal{I}(X) = \mathit{true}$
(by definition of "$\overline{\ }$").
iff $M \models_{\mathcal{T}} X$.

3. "(b) iff (c)":
$M \models_{\mathcal{T}} X$
iff for every $\mathcal{T}$-interpretation $\mathcal{I}$: $\mathcal{I} \models M$ implies $\mathcal{I} \models X$
iff for every interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{T}$ implies ($\mathcal{I} \models M$ implies $\mathcal{I} \models X$)
(by definition of $\mathcal{T}$-interpretation)
iff for every interpretation $\mathcal{I}$: ($\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models M$) implies $\mathcal{I} \models X$)
iff for every interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{T} \cup M$ implies $\mathcal{I} \models X$
iff $\mathcal{T} \cup M \models X$ (by Corollary 2.3.3).

3. "(c) iff (d)": First we need the following *fact:* for every interpretation $\mathcal{I}$:

$$\mathcal{I} \models \mathcal{T} \cup M \text{ iff } \mathcal{I} \models Ax \cup M \ .$$

Proof of fact: "⇒": Suppose, to the contrary, that $\mathcal{I} \models \mathcal{T} \cup M$ but $\mathcal{I} \not\models Ax \cup M$. Then by Lemma 2.3.1.4 it holds $\mathcal{I}(Ax \cup M) = \mathit{false}$ which means (1) $\mathcal{I}(M) = \mathit{false}$ or (2) $\mathcal{I}(Ax) = \mathit{false}$. Since $\mathcal{I} \models \mathcal{T} \cup M$ is given, (1) is impossible. Hence (2) holds. But then, by Lemma 2.5.2, $\mathcal{I}(Cons(Ax)) = \mathit{false}$. With $Cons(Ax) = \mathcal{T}$ given, this is equivalent to $\mathcal{I}(\mathcal{T}) = \mathit{false}$. On the other side, from $\mathcal{I} \models \mathcal{T} \cup M$ as given we conclude $\mathcal{I}(\mathcal{T}) = \mathit{true}$ which is a contradiction. Hence the assumption must have been wrong, and thus the "⇒"-direction holds. The "⇐"-direction is proved analogously.

Now we can prove the equivalence "(c) iff (d)":
$\mathcal{T} \cup M \models X$
iff for every interpretation $\mathcal{I}$: $\mathcal{I} \models \mathcal{T} \cup M$ implies $\mathcal{I} \models X$
iff for every interpretation $\mathcal{I}$: $\mathcal{I} \models Ax \cup M$ implies $\mathcal{I} \models X$ (by the *fact*)
iff $Ax \cup M \models X$ (by Corollary 2.3.3).

3. "(d) iff (e)": immediate from Corollary 2.3.3.

For later use we need the following lemma:

Lemma 2.5.3 (Instantiation lemma). *Let M be a set of Σ-sentences, F be a quantifier free Σ-formula and γ be a substitution. Then $M \models_{\mathcal{T}} \forall F$ implies $M \models_{\mathcal{T}} \forall(F\gamma)$, where $\forall G$ denotes the universal closure of formula G.*

Proof. Suppose, to the contrary, that $M \models_{\mathcal{T}} \forall F$ but $M \not\models_{\mathcal{T}} \forall(F\gamma)$. Hence, by Def. 2.3.1, Def. 2.3.2 and Convention 2.3.2), for some $\mathcal{T}$-model $\mathcal{I}$ for M and some $a_1, \ldots, a_n \in |\mathcal{I}|$ we find $\mathcal{I}_{[x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n]}(F\gamma) = \mathit{false}$.

The substitution γ can be written as

$$\gamma = \{y_1 \leftarrow t_1, \ldots, y_m \leftarrow t_m\} \ .$$

Let $b_i = \mathcal{I}_{[x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n]}(t_i)$. Using induction over n and the structure of F and the term structure of t_j, it can be shown that

$$\mathcal{I}_{[x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n]}(F\gamma) = \underbrace{\mathcal{I}_{[x_1 \leftarrow a_1, \ldots, x_n \leftarrow a_n, y_1 \leftarrow b_1, \ldots, y_m \leftarrow b_m]}}_{=:\mathcal{I}'}(F) \ .$$

That is, we now have a $\mathcal{T}$-interpretation $\mathcal{I}'$, which is still a $\mathcal{T}$-model for M such that $\mathcal{I}'(F) = \mathit{false}$. This, however, contradicts the given assumption that $M \models_{\mathcal{T}} \forall F$.

Next we turn to a theory-version of Herbrand's theorem (Theorem 2.4.3).

Theorem 2.5.2 (Skolem-Herbrand-Löwenheim Theorem). *Let $\mathcal{T}$ be a universal Σ-theory. A clause set M over the signature Σ is $\mathcal{T}$-Σ-unsatisfiable if and only if some finite set M' of ground instances of clauses from M is $\mathcal{T}$-Σ-unsatisfiable.*

Proof. Proof idea: due to Proposition 2.5.1 and the condition that $\mathcal{T}$ be a universal theory the theorem can be reduced to its non-theory version.

For the proof we recall first that $\mathcal{T}$ being universal means that a set $Ax \subseteq \mathcal{CPF}(\Sigma)$ of universal sentences existssuch that $Cons(Ax) = \mathcal{T}$. By the algorithm sketched in Section 2.4.1 every member of Ax can be converted into clause form. Let Ax_C be the set resulting from this transformation. It is most important to note that Ax is universal, which implies that this transformation does not introduce new Skolem functions. Thus Ax_C is still a set of Σ-sentences, and furthermore, as a property of this transformation it holds that Ax and Ax_C are logically equivalent wrt. Σ-interpretations (the transformation only has to express all connectives in terms of $\wedge$ and $\vee$).

Using the set-notation for clauses, Ax_C is a set of set of clauses. In order to convert Ax_C into a logical equivalent clause set C we define

$$C = \bigcup_{c \in Ax_C} c \ .$$

It is easy to see that C and Ax_C are logically equivalent. This comes from the fact that both, the members of Ax_C and the clauses in every clause set $c \in Ax_C$ are connected by "and".

Thus, in sum, we conclude that Ax is logical equivalent to C wrt. Σ-interpretations and turn to the proof of the theorem:

M is $\mathcal{T}$-Σ-unsatisfiable
iff $M \cup Ax$ is Σ-unsatisfiable
(by Proposition 2.5.1.3, equivalence (a)–(e), setting $X = \emptyset$ there)
iff $M \cup C$ is Σ-unsatisfiable (by the above)
iff $M' \cup C'$ is Σ-unsatisfiable
where M' and C' are finite sets of ground instances
(wrt. Σ) of clauses from M and C, respectively (by Theorem 2.4.3)
iff $M' \cup C$ is Σ-unsatisfiable (by Theorem 2.4.3 again)
iff $M' \cup Ax$ is Σ-unsatisfiable (by the above)
iff M' is $\mathcal{T}$-Σ-unsatisfiable
(by Proposition 2.5.1.3, equivalence (a)–(e), setting $X = \emptyset$ there)

In order to see the demand that $\mathcal{T}$ is a universal theory, consider e.g. the non-universal theory $\mathcal{T}$ which is axiomatized by $X = \{\exists x\, P(x)\}$. Assume that Σ contains a single constant symbol a. Then the clause set $M = \{\forall y\, \neg P(y)\}$ clearly is Σ-unsatisfiable (because for every $\mathcal{T}$-Σ-interpretation we must have $\mathcal{I}_P(t) = \mathit{true}$ for some $t \in |\mathcal{I}|$, and this serves as a counterexample for $\{\forall y\, \neg P(y)\}$). On the other hand, there is only one possibility to get a set M' of ground instances of clauses from M, namely $M' = \{\neg P(a)\}$. However, M' is not $\mathcal{T}$-Σ-unsatisfiable because one can easily think of a $\mathcal{T}$-interpretation which satisfies M' (take e.g. $|\mathcal{I}| = \{0, 1\}$, $\mathcal{I}_P = (\lambda x.x = 0)$, $\mathcal{I}_F(a) = 1$).

The problem here is, informally, that in building ground instances we do not have the skolem function symbols at our disposal, which would allow to access the term whose existence is claimed in the theory.

In order to generalize the theorem to non-universal theories one would have to take ground instances wrt. a signature Σ' which is obtained from Σ by adding the Skolem functions coming in by Skolemizing the theories' axioms Ax.

3. Tableau Model Elimination

Model elimination [Loveland, 1968] is a calculus, which is the base of numerous proof procedures for first order deduction. There are high speed theorem provers, like METEOR [Astrachan and Stickel, 1992] or SETHEO [Letz *et al.*, 1992]. The implementation of model elimination provers can take advantage of techniques developed for Prolog. For instance, SETHEO compiles the input clause set into a generalized WAM architecture. Stickel's Prolog technology theorem proving system (PTTP, [Stickel, 1988]) uses Horn clauses as an intermediate language, which can even be processed by conventional Prolog systems [Stickel, 1989].

Such implementational aspects are dealt with in Chapter 6. The purpose of the present chapter is to introduce the data structures and operations of model elimination more abstractly, and to describe some of the basic properties of model elimination calculi. This will serve us as a basis for the subsequent theory-extensions in Chapter 4.

As a starting point we use a model elimination calculus that differs from the original one presented in [Loveland, 1968; Loveland, 1978]; it is described in [Letz *et al.*, 1992] as the base for the prover SETHEO. In [Baumgartner and Furbach, 1993] this calculus is discussed in detail by presenting it in a consolution style [Eder, 1991] and comparing it to various other calculi. In [Baumgartner and Furbach, 1994a] a variant ("restart model elimination") exhibiting a totally different search space was defined.

3.1 Clausal Tableaux

The most intuitive way to introduce model elimination is to think of it as a procedure for manipulating trees in the style of semantic tableaux [Smullyan, 1968].

Definition 3.1.1 (Literal Tree, Branch, etc.). *A* literal tree *(for a given signature) is a pair $\langle t, \lambda \rangle$ consisting of a finite, ordered tree t and a labeling function λ, which assigns a literal to every non-root node of t.*

A branch (of length n, of a given tableau T) *is a sequence $N_0 \cdot N_1 \cdots N_n$ ($n \geq 0$) of nodes in T (written, as indicated, by separating its elements by "·") such that N_0 is the root of T, N_i is the immediate predecessor of N_{i+1}*

for $0 \leq i < n$, and N_n is a leaf of T. The function Leaf(b) gives the label of the leaf of a branch with length $n > 0$, i.e. Leaf$(N_0 \cdots N_n) = \lambda(N_n)$

The branch literal sequence *of branch $b = N_0 \cdots N_n$ is the literal sequence Litseq$(b) = \lambda(N_1) \cdots \lambda(N_n)$, and the* branch literals *of b is the set Lit$(b) = \{\lambda(N_1), \ldots, \lambda(N_n)\}$*

We want the fact that a branch contains a contradiction to be immediately reckognizable. For this purpose we allow a branch to be labeled as either open *or* closed. *A literal tree is* closed *if each of its branches is closed, otherwise it is* open.

If δ is a substitution, then $T\delta$ denotes the literal tree which is obtained from literal tree T by application of δ to the labels of all nodes of T. That is, if $T = \langle t, \lambda \rangle$ then $T\delta = \langle t, \lambda' \rangle$, where $\lambda'(N) = (\lambda(N))\delta$ for every node N in T.

Equality on literal trees is defined as follows: First define $T_1 \subseteq T_2$ iff for every branch b_1 in T_1 a branch b_2 in T_2 exists such that Litseq(b_1) = Litseq(b_2) and b_1 is closed iff b_2 is closed. Then define $T_1 = T_2$ iff $T_1 \subseteq T_2$ and $T_2 \subseteq T_1$.

Model elimination can be thought of as a calculus which constructs tableaux which "contain" clauses in the following sense:

Definition 3.1.2 (Clausal Tableaux, ME Tableaux). *[Letz* et al., *1992]*
The successor sequence *of a node N in an ordered tree t is the sequence of nodes with immediate predecessor N, in the order given by t.*

A (clausal) tableau *T for a set of clauses M is a literal tree $\langle t, \lambda \rangle$ in which, for every maximal successor sequence $N_1, \ldots, N_n$ in t labeled with literals $K_1, \ldots, K_n$, respectively, there is a substitution σ and a clause $L_1 \vee \cdots \vee L_n \in M$ with $K_i = L_i\sigma$ for every $1 \leq i \leq n$. $K_1 \vee \cdots \vee K_n$ is called a* tableau clause *and the elements of a tableau clause are called* tableau literals.

A tableau is called a model elimination tableau (ME tableau) *iff each inner node N, except the root node, has a leaf node N' among its immediate successor nodes such that $\lambda(N) = \overline{\lambda(N')}$. This condition is called the* link condition.

It is widely used jargon to call a pair of literals which can be made complementary by some substitution a connection.

By the previous definition ME tableaux are introduced as static objects. We wish to construct such tableaux in a systematic way. This is accomplished by, for instance, the model elimination calculus. In order to preview the calculus with a simple example consider the clause set

$$\{\{P, Q\}, \{\neg P, Q\}, \{\neg Q, P\}, \{\neg P, \neg Q\}\},$$

Since this clause set is unsatisfiable, model elimination should be able to find a proof (also called a *refutation*). Such a refutation is depicted in Figure 3.1. It is obtained by successive fanning with clauses from the input set, until every branch of the resulting tableau is closed.

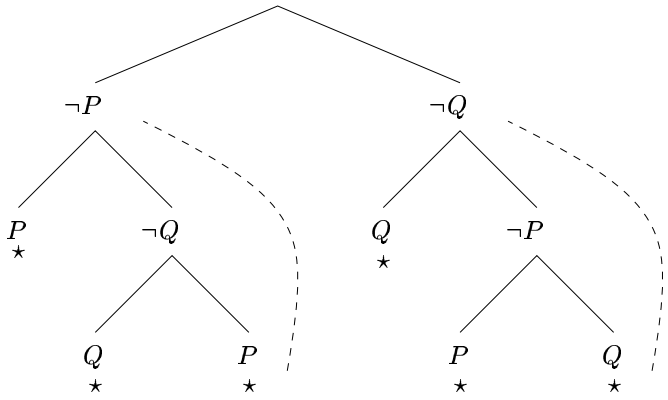


Figure 3.1. A closed model elimination tableau.

In an *initialization step* some input clause is taken and fanned below the root node. Subsequent fanning with a new clause below the leaf of a so far constructed tableau is called an *extension step*. In an extension step one has to obey the condition that the (former) leaf is complementary to one of the newly fanned literals, and the branch ending in this literal is marked with a "$\star$" as closed. By this restriction, the link condition stated in Definition 3.1.2 is realized.

In the non-ground case employing a (most general) unifier in order to establish the complementary pair is allowed; the unifier then is applied to the whole resulting tableau.

Besides the "extension step" it might be necessary to use another inference rule, called the "reduction step": a reduction step allows to close a branch (i.e. to mark it as "closed") due to an inner literal that is complementary to the leaf literal (again, possibly by application of some substitution).

The reader familiar with the tableau literature will notice that Definition 3.1.1 differs with respect to the notion of a "closed branch" from the standard definition. The standard definition says that a branch b is closed iff it contains a complementary pair of literals, i.e. For instance, iff $L \in b$ and $\overline{L} \in b$, for some literal L.

However, this condition is not suitable to lift to theory reasoning, because then we would have to say that a branch is closed iff its literals are $\mathcal{T}$-complementary (cf. Definition 4.2.2). However, as opposed to *syntactical* complementarity, $\mathcal{T}$-complementarity is a *semantical* property and in general is undecidable (see Note 4.2.2). As a consequence, it could not be decided if a given tableau is closed or open, which clearly is highly undesirable. Therefore, we decided to deviate from the standard definition, and to distinguish between the mere *presence* of a $\mathcal{T}$-complementary branch and the *detection* of its $\mathcal{T}$-complementarity by marking it as closed.

Note 3.1.1 (Link Condition and Proof Confluence.). It must be emphasized that the presence of the link condition is a central property for model elimination: if the link condition is not present, then any literal along the branch (or even none) can be used to establish the connection. This freedom guarantees the property of *proof confluence* [Bibel, 1987; Letz, 1993], which has the important consequence that every tableau derived so far can be continued to a refutation, if one exists at all. The resulting calculus is best called a *connection calculus* (cf. Def. 3.2.3 below) due to the striking similarities with the connection calculi defined in [Bibel, 1987; Eder, 1992]. Proof confluent calculi are attractive because of the possibility to design proof procedures without backtracking over the generated tableaux.

On the other hand, when the link condition is present, proof confluence is lost, but the local search space is smaller. With regard to a proof procedure, the link condition allows us to guide the search for an extending clause around the leaf literal (for instance, by term indexing the inference rules); in this case, efficient implementations can be built on top of PROLOG by means of the PTTP-technique (Prolog Technology Theorem Proving, [Stickel, 1989; Stickel, 1990a], see also Chapter 6).

This note should not imply that one approach is necessarily superior to the other. Instead, the purpose is to point out that a small change in the calculus definition may have dramatic impacts on its properties. A more detailed comparison between these calculi variants, also taking resolution into account is contained in [Baumgartner and Furbach, 1993].

3.2 Inference Rules and Derivations

We are going to formally define the inference rules of model elimination next. Due to the variety of calculi treated in this text, we find it advantageous to use an economical notation for literal trees, namely *branch sets*, and define calculi inference rules as transformations of these.

Definition 3.2.1 (Representation of Tableaux). *Let $T = \langle t, \lambda \rangle$ be a tableau. The* branch representation *of T is the triple $\langle \mathcal{B}, \lambda, o \rangle$, where $\mathcal{B} = \{p \mid p \text{ is a branch of } t\}$ is the* branch set *of T and o is an* ordering function, *mapping any maximal subset $S_q = \{q \cdot N_1, \ldots, q \cdot N_n\} \subseteq \mathcal{B}$ to a sequence of nodes, such that $o(S_q) = N_1, \ldots, N_n$ iff $N_1, \ldots, N_n$ is a maximal successor sequence of T. In other words, o reflects the order of the nodes in T.*

Let T be a tableau with branch representation $\langle \mathcal{B}, \lambda, o \rangle$. In the sequel we will most often identify T with $\langle \mathcal{B}, \lambda, o \rangle$, or even with $\mathcal{B}$ alone, and let λ and o be implicitly given. Branch sets are typically denoted by the letters $\mathcal{P}, \mathcal{Q}, \ldots$.

We find it convenient to write $T = (\mathcal{P}, \mathcal{Q})$ and mean that $\mathcal{B} = \mathcal{P} \cup \mathcal{Q}$ (disjoint union is intended here). Similarly, $(p, \mathcal{Q})$ means $(\{p\}, \mathcal{Q})$, where p is a branch. Going further, we allow a branch $N_0 \cdot N_1 \cdots N_n \in \mathcal{B}$ to be represented by the literal *sequence $L_1 \cdots L_n$, with the understanding that*

$\lambda(N_1)\cdots\lambda(N_n) = L_1\cdots L_n$. *Also mixed forms are acceptable, as in* $N_0 \cdot N_1\cdots N_{n-1}\cdot L_n$.

For better readability we will supply brackets around branches, as in $[L_1\cdots L_n]$, $[p\cdot q]$ *(the branch to be obtained by appending node sequences* p *and* q*) or* $[p\cdot L]$. *We adopt the convention that "*$[p]$*" means a branch in a tableau, whereas "*p*" means the literal multiset of its labels.*

In order to indicate that a branch is closed, we will append a "$\times$*" to it, as in* $[L_1\cdots L_n]\times$. *If no "*$\times$*" is present, the branch is open.*

In order to define extension steps in a convenient way, we define the extension of branch p with clause C, $[p]\circ C$ *as an abbreviation for the branch set* $\{[p\cdot L] \mid L\in C\}$. *That is,* $(\mathcal{P}, p\circ C)$ *is the branch set* $\mathcal{P}\cup\{[p\cdot L] \mid L\in C\}$ *and if* C *is the empty clause* $\Box$, *then* $(\mathcal{P},\ p\circ C) = \mathcal{P}$.

Note that the lengthening of branch p in tableau T by $|C|$ new successor nodes and labeling them with the literals from C, respectively, can be rephrased in branch set notation by saying that tableau $(p\circ C), \mathcal{Q}$ is obtained from $T = (p, \mathcal{Q})$. It is obvious that the new tableau $(p\circ C, \mathcal{Q})$ exists (more precisely: that a tableau corresponding to the suggested branch set representation exists), provided that a tableau corresponding to T exists and that *new* nodes (wrt. T) for the extension are used, which further are pairwise disjoint and the ordering function o is defined to reflect the required ordering. We will take it for granted that the tableau described below by branch sets always exists.

Definition 3.2.2 (Tableau Inference Rule). *A* (tableau) inference rule *is an expression*

$$\frac{\mathcal{P} \qquad \mathcal{C}_1 \quad \cdots \quad \mathcal{C}_n}{\mathcal{P}'}\ \mathsf{N}$$

where N *is the name of the inference rule,* $\mathcal{P}$,$\mathcal{P}'$ *are schematic expressions standing for branch sets (for some tableau), and* $\mathcal{C}_1,\ldots,\mathcal{C}_n$ $(n\geq 0)$ *are schematic expressions standing for clauses; Inference rules are possibly constrained by some further informally stated conditions. We say that branch set* $\mathcal{P}'$ *is* obtained *from branch set* $\mathcal{P}$ *and a multiset of clauses* $\{\!|C_1,\ldots,C_n|\!\}$ *by an* N*-step, written as an* N-inference

$$\mathcal{P}\vdash_{\{\!|C_1,\ldots,C_n|\!\}}\mathcal{P}'$$

iff $\mathcal{P}$, $C_1,\ldots,C_n$ *and* $\mathcal{P}'$ *are instances of the respective components* $\mathcal{P}$, $\mathcal{C}_1,\ldots,\mathcal{C}_n$ *and* $\mathcal{P}'$ *of the inference rule* N *and the given constraints hold. Below we will further decorate the "*$\vdash$*"-relation suitably in order to make more relevant constraint parameters explicit.*

A calculus $\mathcal{N}$ *consists of a set of inference rules.*

As an example consider model elimination:

Definition 3.2.3 (Model Elimination, Connection Calculus). *The inference rules* extension step *and* reduction step *(*ME-Ext, Red*) are defined as follows:*

Extension Step:

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L \vee R}{([p \cdot K \cdot L]\times,\ [p \cdot K] \circ R,\ \mathcal{Q})\sigma}\ \textsf{ME-Ext}$$

where σ *is a unifier for* K *and* $\overline{L}$*, i.e.* $K\sigma = \overline{L}\sigma$

(Characterizing property for ME-Ext, "link condition")

Reduction Step:

$$\frac{[p \cdot K], \mathcal{Q}}{([p \cdot K]\times, \mathcal{Q})\sigma}\ \textsf{Red}$$

where σ *is a substitution such that* $K\sigma = \overline{L}\sigma$ *for some* $L \in p$*.*

The calculus model elimination (ME) *consists of the inference rules* ME-Ext *and* Red*.*

The inference rule CC-Ext *("CC" standing for "Connection Calculus") is defined as* ME-Ext*, except that instead of the branch expression* $[p \cdot K]$ *the branch expression* $[p \cdot K \cdot p']$ *is used. That is, the link condition is weakened to allow a connection to any ancestor literal of* K*. The* connection calculus (CC)[1] *consists of the inference rules* CC-Ext *and* Red*.*

Thus, ME-Ext derives from a branch set and a given clause a new branch set by unifying the leaf of the selected branch p with some literal L from the given clause. For instance, from (singleton) branch set $[Q(x,y) \cdot P(x)]$ and clause $\neg P(f(z)) \vee R(z)$ derive (using $\sigma = \{x \leftarrow f(z)\}$) the branch set

$$[Q(f(z),y) \cdot P(f(z)) \cdot \neg P(f(z))]\times,\ [Q(f(z),y) \cdot P(f(z)) \cdot R(z)]\ .$$

Notice that the substitution σ is applied to the whole branch set; this must be done for soundness reasons. The link condition for ME tableaux is achieved by a respective condition in the ME-Ext inference rule. Deviating from that, CC allows a connection from the extending clause into any branch literal.

It is easily verified that the CC and ME inference rules transform clausal tableaux into clausal tableaux (and not just literal trees). Further note that closed branches are neither extended nor reduced.

In both ME-Ext and Red the substitution σ is a unifier for the involved literals (cf. Definition 2.4.5). As for the resolution calculus, it suffices to restrict to *most general unifiers*. This (well-known) result will also be obtained as an instance of a more general result of the theory version of model elimination.

[1] As it is used in this text; our formulation is close to the one in [Eder, 1992].

It remains to define derivations; as with inference rules, we prefer to give a "reusable" definition which also applies for the theory reasoning variants. Fortunately, all model elimination inference rules follow a certain schema, which can also be fixed now:

Definition 3.2.4 (Derivation, Refutation). *Let $\mathcal{N}$ be a calculus. Suppose that all inference rules of $\mathcal{N}$ are of the form*

$$\frac{[p], \mathcal{Q} \qquad L_1 \vee R_1 \cdots L_n \vee R_n}{(\mathcal{Q}', \mathcal{Q})\sigma} \ \mathsf{N}.$$

In order to make p and σ explicit, we will write $\mathcal{N}$-inferences (cf. Definition 3.2.2) as

$$([p],\ \mathcal{Q}) \vdash_{[p],\sigma,E} (\mathcal{Q}',\ \mathcal{Q})\sigma \ .$$

The branch p is called the selected branch*, the clauses $E = \{\!|L_1 \vee R_1, \ldots, L_n \vee R_n|\!\}$ are called* extending clauses *and each literal L_i ($i \in \{1, \ldots, n\}$) is called* extending literal*; the R_is are called* rest clauses *and their literals are referred to as* rest literals*. In branch $[p \cdot L]$, the literals of p are also referred to by the term* ancestor literals *(or* ancestor nodes*, or* ancestor context*). As a convenience we allow to omit σ in the index of $\vdash$ if $\sigma = \epsilon$.*

A $\mathcal{N}$ derivation *of a branch set $\mathcal{P}_n$ from a clause set M (called* input clause set*) with length n is a sequence ($n \geq 1$)*

$$\mathcal{P}_1 \vdash_{[p_1],\sigma_1,E_1} \mathcal{P}_2 \cdots \mathcal{P}_{n-1} \vdash_{[p_{n-1}],\sigma_{n-1},E_{n-1}} \mathcal{P}_n$$

such that the following holds:

1. *$\mathcal{P}_1 = [L_1], \ldots, [L_m]$, where $L_1 \vee \cdots \vee L_m \in M$, i.e. $\mathcal{P}_1$ is a clausal tableau (in branch set notation) with tableau clause $L_1 \vee \cdots \vee L_m$. The clause $L_1 \vee \cdots \vee L_m$ is also called the* start clause *in this context, and $\mathcal{P}_1$ is also called the* initial branch set *of the derivation.*
2. *For $i = 2 \ldots n$, $\mathcal{P}_{i-1} \vdash_{[p_{i-1}],\sigma_{i-1},E_{i-1}} \mathcal{P}_i$ for some inference rule from $\mathcal{N}$. Further, the extending clauses E_i each are new and pairwise variable disjoint variants of clauses from M.*

The combined substitution $\sigma_1 \cdots \sigma_{n-1}$ is referred to as the substitution computed by the $\mathcal{N}$ derivation*. If $n = 1$ then the computed substitution is defined to be the "empty" substitution ε.*

A $\mathcal{N}$ refutation of *M is a derivation of the form just defined where $\mathcal{P}_n$ is a closed tableau, i.e. a branch set where every branch is marked as closed.*

Calculus $\mathcal{N}$ is refutationally complete *iff for every minimal $\mathcal{T}$-unsatisfiable clause set M there exists a $\mathcal{N}$ refutation.*

The need to take new variants along a derivation is obvious, because otherwise extension steps might not be applicable due to coincidence of variable names, preventing unification succeeding.

In order to get familiar with this definition we give in Figure 3.2 a refutation corresponding to the ME tableaux in Figure 3.1; the start clause is $\neg P \vee \neg Q$:

Open branches	*Closed branches*
$[\neg P]$, $[\neg Q]$	
$\downarrow$ *extension of* $[\neg P]$ *with clause* $P \vee \neg Q$	
$[\neg P \neg Q]$, $[\neg Q]$	$[\neg P P]\times$
$\downarrow$ *extension of* $[\neg P \neg Q]$ *with clause* $Q \vee P$	
$[\neg P \neg Q P]$, $[\neg Q]$	$[\neg P P]\times$, $[\neg P \neg Q Q]\times$
$\downarrow$ *reduction of* $[\neg P \neg Q P]$	
$[\neg Q]$	$[\neg P P]\times$, $[\neg P \neg Q Q]\times$, $[\neg P \neg Q P]\times$
$\downarrow$ *extension of* $[\neg Q]$ *with clause* $Q \vee \neg P$	
$[\neg Q \neg P]$	$[\neg P P]\times$, $[\neg P \neg Q Q]\times$, $[\neg P \neg Q P]\times$, $[\neg Q Q]\times$
$\downarrow$ *extension of* $[\neg Q \neg P]$ *with clause* $P \vee Q$	
$[\neg Q \neg P Q]$	$[\neg P P]\times$, $[\neg P \neg Q Q]\times$, $[\neg P \neg Q P]\times$, $[\neg Q Q]\times$, $[\neg Q \neg P P]\times$
$\downarrow$ *reduction of* $[\neg Q \neg P Q]$	
	$[\neg P P]\times$, $[\neg P \neg Q Q]\times$, $[\neg P \neg Q P]\times$, $[\neg Q Q]\times$, $[\neg Q \neg P P]\times$ $[\neg Q \neg P Q]\times$

Figure 3.2. A ME refutation corresponding to the tableau in Figure 3.1.

ME and Resolution

There is a close relationship between ME and resolution. More precisely, each ME derivation can be mapped *stepwise* into a linear resolution [Loveland, 1970] derivation. In this mapping, extension steps correspond to resolution steps of the current clause and an input clause, and reduction steps correspond to ancestor resolution steps. This was observed already in [Loveland, 1978], and a rigorous proof was given in [Baumgartner and Furbach, 1993]:

Theorem 3.2.1 (Resolution Simulates ME). *[Baumgartner and Furbach, 1993] Let M be a clause set and let $\mathcal{P}_1 \vdash \cdots \vdash \mathcal{P}_n$ be a ME refutation of M such that in every branch set $\mathcal{P}_i$ ($i = 1 \ldots n-1$) a longest branch is selected. Then a resolution[2] derivation $(C_1, \ldots, C_k)$, for some $k \leq n$, from M and a substitution γ exists such that $C_k\gamma \subseteq O(\mathcal{P}_n)$, where $O(\mathcal{P}) = \{K \mid [p \cdot K] \in \mathcal{Q}\}$ is the set of "open leaves" in branch set $\mathcal{P}$.*

[2] see [Chang and Lee, 1973] for a definition of "resolution".

3.2.1 Answers

In the introduction, theory reasoning was motivated by a *problem solving* application, the eight queens problem. Of course, when formulated in predicate logic and fed into a theorem prover the bare fact that a refutation is found is rather useless. Instead, one would prefer an *answer* to the problem, i.e. a description of the solution. Further, the answer should be meaningful and exclude trivial cases ("queen i is on field 1 or on field 2 or ... or on field 64").

We will thus extend the framework defined so far by the possibility to compute *answer substitutions*.

Definition 3.2.5 (Program, Query, Answers, Answer Completeness). *A* program P *is a $\mathcal{T}$-satisfiable set of clauses. A* query *(over given signature Σ) is an expression $\leftarrow Q$, where Q is a conjunction of literals; a query $\leftarrow Q$ is identified with the clause $\overline{L_1} \vee \cdots \vee \overline{L_n}$, where $Q = L_1 \wedge \cdots \wedge L_n$.*

If $\Phi_1, \ldots, \Phi_m$ are substitutions for $Var(\leftarrow Q)$ then the set $\{Q\Phi_1, \ldots, Q\Phi_m\}$ is called an answer *(for $\leftarrow Q$); in case $m = 1$ it is called a* definite *answer. An answer is called a* correct answer for program P and query $\leftarrow Q$ *iff*

$$P \models_{\mathcal{T}} \forall(Q\Phi_1 \vee \cdots \vee Q\Phi_m).$$

Now let P be a program, $\leftarrow Q$ be a query and let $\mathcal{D}$ be a $\mathcal{N}$ refutation of $P \cup \{\leftarrow Q\}$ with start clause $\leftarrow Q$. Let $\mathcal{D}$ be written as

$$\mathcal{D} = (\mathcal{P}_1 \vdash_{[p_1],\sigma_1,E_1} \mathcal{P}_2 \cdots \mathcal{P}_{n-1} \vdash_{[p_{n-1}],\sigma_{n-1},E_{n-1}} \mathcal{P}_n).$$

Define

$$Answer(\mathcal{D}) = \{Q\sigma_1 \cdots \sigma_{n-1}\} \cup \bigcup_{i=1}^{n-1} Queries(E_i)\sigma_i \cdots \sigma_{n-1}, \text{ where}$$

$$Queries(E) = \{Q' \mid \ \leftarrow Q' \in E \text{ and } \leftarrow Q' \text{ is a variant of } \leftarrow Q\}$$

The set $Answer(\mathcal{D})$ collects the instances of (the variants of) the literals of the query $\leftarrow Q$ used along $\mathcal{D}$. The first element stems from the usage as start clause, while the other elements stem from the usage as extending clauses.

$\mathcal{D}$ will also be called an $\mathcal{N}$ refutation of P and $\leftarrow Q$ with computed answer *$Answer(\mathcal{D})$.*

Calculus $\mathcal{N}$ is answer complete *iff for every correct answer $\{Q\Phi_1, \ldots, Q\Phi_m\}$ for program P and query $\leftarrow Q$ an $\mathcal{N}$ refutation of P and $\leftarrow Q$ exists with computed answer $\{Q_1, \ldots, Q_l\}$ such that the latter subsumes the former, i.e.*

$$\{Q_1, \ldots, Q_l\}\delta \subseteq \{Q\Phi_1, \ldots, Q\Phi_m\}, \quad \text{for some substitution } \delta.$$

Unfortunately we cannot demand that $l \leq m$. For instance, it might be that $\{P(a), P(b)\}$ is a correct answer, but the shortest computed answer is $\{P(x), P(y), P(b)\}$.

Computing answers is more difficult than refutational theorem proving. This was demonstrated in [Baumgartner *et al.*, 1995] using puzzle examples. The additional complexity comes in by refusing to accept non most-general answers. For instance, in the mentioned puzzle domain, there is a case where only a *definite* answer is meaningful, but hard to find, whereas a trivial answer, showing that the problem has a solution, is discovered very early in the proof search.

Answer completeness implies refutational completeness (but not necessarily vice versa), because a successful answer computation for P and $\leftarrow Q$ serves as a proof of the $\mathcal{T}$-unsatisfiability of $P \cup \{\leftarrow Q\}$. Hence we will give answer completeness results for our calculi below.

3.2.2 Clausal Tableaux vs. Branch Sets

This section discusses the usage of various data structures available for model elimination.

Note 3.2.1 (Loveland's Model Elimination). The original model elimination calculus [Loveland, 1968; Loveland, 1978] uses *chains* as the primary data structure. A *chain* is a finite sequence of literals $a_1 b_1 \cdots a_n b_n$ where each a_i stands for a sequence of literals written in brackets, i.e. $a_i = [a_{i,1}] \cdots [a_{i,k_i}]$, and each b_i stands for a sequence of literals $b_i = [b_{i,1}] \cdots [b_{i,l_i}]$. The chains used in Loveland's ME can be simulated by certain literal trees where every inner node lies on one single branch of the tableau, and the closed branches are omitted. Figure 3.3 gives an idea of the transformation (see [Baumgartner and Furbach, 1993] for a more detailed treatment).

In our terminology, Loveland's model elimination is fixed to a computation rule where always some longest branch in the current tableau is selected for an extension or reduction step. In fact, the possibility to use *any* longest branch is mirrored by using ordering rules for extension clauses. Since we want to allow maximal flexibility in the construction of a refutation (cf. the *independence of the computation rule* in Section 3.3 below) and for ease of presentation we will dispense with the chain notation and formulate our calculi within a tableau setting.

In [Baumgartner and Furbach, 1993] we related several calculi (resolution, connection calculi, variants of model elimination) using the framework of *consolution* [Eder, 1992]. The primary data structure of the consolution calculus are multisets of branches, where a branch is a sequence of literals. As done in [Baumgartner and Furbach, 1993], it is straightforward to rephrase tableau model elimination within such a branch-set setting[3]. In order to do

[3] More precisely, the branch sets in [Baumgartner and Furbach, 1993] encode the *open* branches of a corresponding ME tableau; the closing of a branch in the tableau framework corresponds to branch deletion in the branch setting. In order to get a more restricted calculus (cf. Section 3.3) and for our purposes of theory

Chain $a_1 b_1 \cdots a_n b_n$ transforms to:

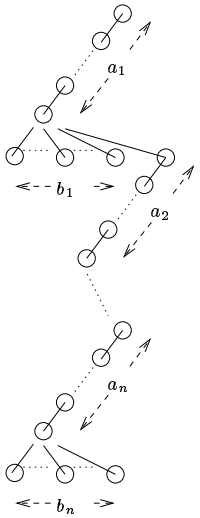


Figure 3.3. Mapping of chains to literal trees.

so, and as carried out in Definition 3.2.1, we think of a tableau as given as the set $\{b_1, \ldots, b_m\}$ of all of its branches. Obviously, with respect to labels, $\{b_1, \ldots, b_m\}$ is isomorphic to the *multi*set $\{\!|Litseq(b_1), \ldots, Litseq(b_n)|\!\}$ of literal sequences. Then, the inference rules of model elimination are to be defined as transformations on such branch multisets, in such a way that the resulting branch sets are isomorphic to the tableau resulting in the tableau-setting. Such a presentation for the case of theory model elimination was given in [Baumgartner, 1994].

Note the subtle difference to the approach as introduced above, where due to *notational conventions* we can write $\{[L_1^1 \cdots L_{n_1}^1], \ldots, [L_1^m \cdots L_{n_m}^m]\}$ and mean a tableau T, represented as a branch set $\{[N_1^1 \cdots N_{n_1}^1], \ldots, [N_1^m \cdots N_{n_m}^m]\}$ with attached labeling function $\lambda(N_j^i) = L_j^i$. In contrast to that, the branch multiset approach dispenses with trees and labeling functions. That is, T would be represented as $\{\!|[L_1^1 \cdots L_{n_1}^1], \ldots, [L_1^m \cdots L_{n_m}^m]|\!\}$.

Since both approaches are easily shown to be equivalent, the question arises which one to prefer. On the one hand, the tableau view is clearly the more "natural" one, due to its graphical presentation and long tradition of tableau calculi. On the other hand, in my opinion, the branch set notation is more suited for formal proofs.

Fortunately, as the discussion should have revealed, there is no need to stick to one view and to dispense with the other one. Instead, the inference rules can always be given either interpretation. For matters of presentation,

---

reasoning, however, it turned out to be most advantageous to keep the closed branches explicitly.

we will most times use the tableau view, and for the formal proofs (in Appendix A) we will use the branch set view.

3.3 Improvements

When experimenting with practical implementations, such as the PROTEIN system (Chapter 6), it soon becomes obvious that certain improvements are absolutely essential. We will briefly introduce the three most important ones ("Independence of the computation rule", "Regularity", "Factorization"). All of them are widely used in model elimination based theorem proving (cf. [Astrachan and Stickel, 1992; Letz *et al.*, 1994]). Other useful improvements not covered here are *caching/lemmaizing* [Astrachan and Stickel, 1992] and *folding up/folding down* [Letz *et al.*, 1994]. The latter generalizes factorization. In [Baumgartner and Brüning, 1997] it is shown how "subsumption" (see e.g. [Chang and Lee, 1973]) can be used for model elimination tableaux, thereby generalizing the regularity restriction.

3.3.1 Independence of the Computation Rule

So far, the ME-Ext- and Red-inference rules operate on *some* selected branch. This would mean for implementations that choosing the selected branch is subject to backtracking. Clearly we would like to avoid this if possible. Indeed we have a free choice regarding the selected branch. As a further advantage of such a result, the selected branch can be chosen heuristically. Occasionally, *factoring* can be applied more successfully (see [Letz *et al.*, 1994]) if such a "subgoal reordering" is allowed.

For the case of non-theory model elimination this was shown in [Letz, 1993], and the generalization to theory reasoning will be shown below. In order to formalize this, we borrow the notion of "computation rule" from logic programming ([Lloyd, 1987]):

Definition 3.3.1 (Computation Rule). *A* computation rule *is a function c which maps a tableau to one of its open branches. Thus a computation rule can be used in derivations to determine the selected branch for the next inference step; we say that a derivation is a* derivation via c *iff the selected branch in every of its inference steps is determined by c.*

For example, if a Prolog-like computation rule is desired, then always the leftmost (open) branch is to be selected.

As noted above, we would like to allow any computation rule for derivations. In other words, we are interested in a "strong" completeness result which says that any computation rule yields a complete calculus. The formal statement of this *independence of the computation rule* requires concepts not yet introduced; it is to be found in the appendix (Proposition A.1.3).

3.3.2 Regularity

The *regularity check* for tableau ME says that it is never necessary to construct a tableau where a literal occurs twice (or even more often) along a branch. Expressed operationally, it says that it is never necessary to repeat in a derivation a previously derived subgoal (viewing open leaves as subgoals). For a semantic interpretation take the view that a branch constitutes a partial model, and any clause containing a literal from the branch would be satisfied by this interpretation. Hence this clause need not be considered for "eliminating" the interpretation given by the branch.

A nice property of regularity is that it constitutes the base for a decision procedure for propositional logic. Regularity is easy to implement (at least approximative) and it is one of the more effective restrictions for model elimination procedures. Our practical experiments (Section 6) strongly support this claim for the case of theory reasoning.

Hence we define:

Definition 3.3.2 (Regularity). *A branch is called* regular *iff all the literals occurring in it are pairwise distinct (i.e. the branch contains no duplicate literals). A branch set is* regular *iff each of its branches is regular. A derivation (in any version defined so far and below, except partial theory restart model elimination in Section 4.6) is called* regular *iff every branch set of the derivation is regular.*

Recall from the definition, that ME keeps the closed branches (i.e. they are not removed). This is a difference to the branch set oriented calculi of e.g. [Eder, 1992] and [Baumgartner and Furbach, 1993] where closed branches are deleted. Also in [Loveland, 1978], in our terminology, closed branches are no longer considered. However, if closed branches are kept, they are subject to the regularity check. In sum, the definition of regularity implies that in a regular derivation every closed branch *remains* regular even after further instantiation.

Regularity is a further restriction of the "equal ancestor check" found occasionally, which means regularity only with respect to open branches. Unfortunately, regularity is sometimes too restrictive and is not compatible with some other refinements. For instance, [Letz *et al.*, 1994] report on an incompatibility between their enforced folding up rule (a kind of generalized factorization) with "strong" regularity and arbitrary computation rules.

For later use we state the following:

Proposition 3.3.1. *Let* $[p]$ *be a branch and* $\sigma_1, \ldots, \sigma_n$ *be substitutions, and suppose that* $[p\sigma_1 \cdots \sigma_n]$ *is regular. Then also any prefix* $[p\sigma_1 \cdots \sigma_m]$ *(*$0 \leq m \leq n$*) is regular.*

Proof. Suppose, to the contrary, $[p\sigma_1 \cdots \sigma_m]$ is not regular for some m. This means that p contains literals L_1 and L_2 such that $L_1\sigma_1 \cdots \sigma_m = L_2\sigma_1 \cdots \sigma_m$. Since substitutions are functions, then also

$$L_1\sigma_1 \cdots \sigma_m\sigma_{m+1} \cdots \sigma_n = L_2\sigma_1 \cdots \sigma_m\sigma_{m+1} \cdots \sigma_n \ .$$

Hence $[p\sigma_1 \cdots \sigma_n]$ is not regular either, which plainly contradicts the assumption.

3.3.3 Factorization

The well-known *factorization* inference rule in resolution calculi can be adapted to model elimination. In the tableau setting of model elimination, factorization allows us to close a branch, say with leaf literal L, if some brother node of an ancestor node of L, say K, is unifiable with L. The unifier is applied to the resulting tableau. In our notation this reads as follows:

$$\frac{[q \cdot p \cdot L], [q \cdot K], \mathcal{Q}}{([q \cdot p \cdot L]\times, [q \cdot K], \mathcal{Q})\sigma} \quad \mathsf{Fact}$$

where σ is a unifier for L and K.

It is important to note that $[q \cdot K]$ is an *open* branch. This is a sufficient criterion in order to avoid mutually dependent closing of branches using factorization, which would destroy soundness (see [Letz *et al.*, 1994] for a detailed discussion).

Adding Fact to the set of inference rules of ME trivially preserves completeness, because Fact need not be used in a refutation. Being thus an optional[4] inference rule, Fact *in*creases the local search space. Nethertheless, it can be advantageous to employ Fact because it may shorten refutations considerably. To see this, notice that there is no need to search for a refutation of the branch closed by the Fact rule.

An interesting special case is $\sigma = \varepsilon$. Thus, L and K are identical. Let us call such steps *weak factorization steps*. It can be required that in derivations weak factorization steps are applied whenever possible. Completeness is preserved, because this strategy can be expressed as a special computation rule, namely one which always selects $[q \cdot K]$ (in terms of the inference rule above) in favor of $[q \cdot p \cdot K]$.

In implementations it is essential to employ this rule, as it neither increases the local search space nor requires longer refutations.

Factorization for model elimination is due to Loveland (see [Loveland, 1978]). In fact, Loveland's *weak* model elimination calculus includes the weak factorization rule and its mandatory application in derivations; his non-weak version also contains the factorization rule.

[4] Recall that in resolution calculi factorization is a *mandatory* inference rule.

3.3.4 Reduction Steps

Since reduction steps introduce additional non-determinism in proof procedures, it would be useful to have a criterion to avoid unnecessary reduction steps. A trivial result is that reduction steps are unnecessary, because not even possible, if the input set is a Horn clauses set and the start clause in the tableau is negative (in any case, if an input set admits a refutation, it also admits a refutation with purely negative clause).

The justifying observation is that along derivations only open branches consisting entirely of negative literals are possible. Hence no complementary literals can arise in an open branch, and so no reduction steps can be carried out at all. In this case, model elimination behaves much like SLD-Resolution (see [Lloyd, 1987]).

A non-trivial result obtained in [Antoniou and Langetepe, 1994] says that reduction steps are unnecessary for Horn-input sets even if the start clause is not a completely negative clause.

Some results are known for the non-Horn case: Plaisted [Plaisted, 1990] introduced the *positive refinement* which states that only reduction steps back to *positive* literals are needed. This result was refined later in [Baumgartner and Brüning, 1997] in that only reduction steps to positive literals stemming from *disjunctive* clauses[5] are needed; notice that this result generalizes the just-mentioned result in [Antoniou and Langetepe, 1994].

Another (well-known) improvement is obtained in analogy to *weak* factorization steps as described below. The *weak* version of Red requires that σ be the empty substitution. As for weak factorization, it can be required that weak Red steps are applied whenever possible. For the completeness, the same argumentation is used.

This concludes our description of basic improvements. Due to their practical importance, we are interested in respective improvements for the theory version of ME. Considerable effort will be necessary to achieve this, because only the weak versions of factorization and reduction steps trivially lift to the theory case (once the "independence of the computation rule" is established for the theory version).

[5] By a *disjunctive clause* we mean a clause which contains at least *two* positive literals.

4. Theory Reasoning in Connection Calculi

In this chapter general theory reasoning versions of a connection calculus (CC) and model elimination (ME) will be introduced and gradually refined. Here, the term "general" means that no special properties of a particular theory is made use of. Instead, rather high-level principles for the interaction between the foreground and the background calculi are employed.

The structure of this chapter is the following: we start with an overview on basics of theory reasoning (Section 4.1). As a calculus we use here a simple theory-version of a connection calculus. This calculus is then formalized in Section 4.2. It will be discussed critically and refined to a first version of theory model elimination (Section 4.3). Sections 4.4 and 4.5 then present the most refined versions of a total and a partial variant of theory model elimination, respectively. Finally, Section 4.6 deals with a variant of partial theory model elimination which does not need contrapositives, i.e. which uses only the positive literals of input clauses for extension steps. Figure 4.1 might be helpful for orientation.

4.1 Basics of Theory Reasoning

The purpose of this section is to introduce a conceptually simple, preliminary version of theory model elimination — a *theory connection calculus* — and to discuss some basic properties of theory reasoning. It will be the basis for more refined *theory model elimination* calculi.

As mentioned previously, theory reasoning is motivated by the possibility of taking advantages of dedicated, efficient mechanisms for reasoning within the given theory. We initiate a running example for this chapter which also illustrates the advantages for theory reasoning.

Example 4.1.1 (Strict Orderings). Consider the following theory[1] $\mathcal{SO}$ of strict orderings:

$$\mathcal{SO}: \quad \forall x,y,z\ x < y \wedge y < z \rightarrow x < z \qquad \text{(Trans)}$$
$$\forall x \quad \neg(x < x)\ . \qquad \text{(Irref)}$$

[1] We will often identify a *theory* with an *axiom set* for it.

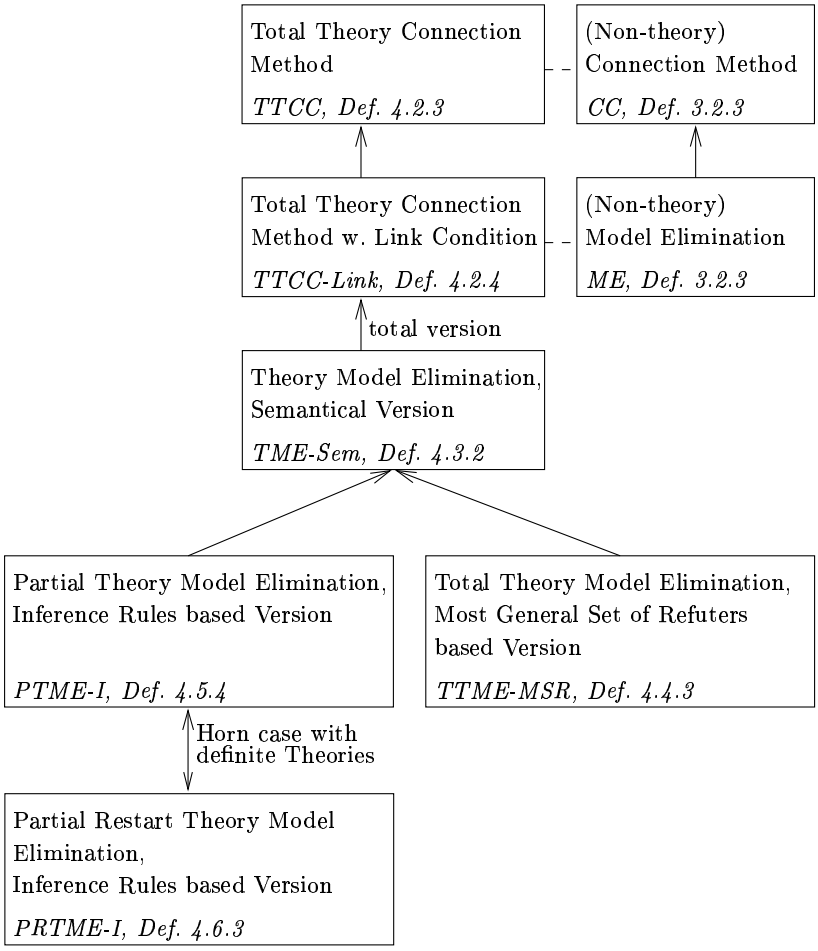


Figure 4.1. "Weaker-than" relation between the calculi of Chapter 4. Calculus A is *weaker than* calculus B ($A \rightarrow B$) iff each inference step of calculus A is also an inference step in calculus B. Dashed arrows indicate that the connected calculi are the same in case of the empty theory. It is therefore the weakest variants which will be proven complete, because their completeness implies the completeness of the stronger variants.

Now suppose the set of (ground) unit clauses M_n

$$\neg(a_1 < a_n), \qquad a_1 < a_2, \ldots, \quad a_{n-1} < a_n$$

to be given. It is easy to find a ME refutation of $\mathcal{SO} \cup M_n$, starting with, for instance, the goal clause $\neg(a_1 < a_n)$. Notice, however, that there are infinitely many derivations, starting from $\neg(a_1 < a_n)$, even when using the (Trans) clause alone: for any $m \geq 1$ there is a derivation of a tableau with open branches ending in the leaves

$$\neg(a_1 < x_1), \quad \neg(x_1 < x_2), \cdots, \quad \neg(x_{m-1} < x_m), \quad \neg(x_m < a_n) \ .$$

The situation is even worse if no refutation exists. For instance, if $M' = M_n \setminus \{a_1 < a_2\}$ then an ME proof procedure will not terminate, although this particular problem class is decidable. Clearly, such cases should be avoided; theory reasoning offers a solution for it.

The idea of coupling a foreground calculus to a background calculus was formulated in the context of connection calculi in [Bibel, 1982a]. The motivation for doing so was to achieve a better structuring of knowledge. This was exemplified with a coupling of the connection calculus to database systems. The explicit term *theory reasoning* seems to be introduced by Mark Stickel within the general, non-linear resolution calculus [Stickel, 1985]. This paper also contains a classification of theory reasoning, as well as completeness criteria. Since then, theory reasoning was ported to many calculi. In [Baumgartner, 1992b] I showed that total theory resolution is compatible with ordering restrictions. Theory reasoning was defined for matrix methods in [Murray and Rosenthal, 1987], for the connection graph calculus in [Ohlbach, 1987] and for connection calculi in [Petermann, 1992]. An improved version for semantic tableaux is presented in [Beckert and Pape, 1996].

All these calculi are not goal-oriented and hence are essentially different from model elimination, thus requiring new efforts for the model elimination case. For model elimination a first theory-reasoning version is defined and proven complete in [Baumgartner, 1992a]. Later I further refined it [Baumgartner, 1993; Baumgartner, 1994]. It is essentially this refinement which will be discussed in detail in Sections 4.4, 4.5 and 4.6 below. Closely related to our theory model elimination calculus is the theory connection calculus with link condition in [Petermann, 1993a]. We will discuss the differences as the text proceeds and summarise the differences in Section 4.4.4.

4.1.1 Total and Partial Theory Reasoning

According to [Stickel, 1985], theory reasoning comes in two variants: *total* and *partial* theory reasoning[2]. *Total* theory reasoning lifts the idea of finding syntactical complementary literals in inferences to a semantic level.

Total Theory Reasoning. Let us illustrate this general mechanism in the case of the theory connection calculus. In essence, this requires only slight changes in the definition "extension step". Consider the left tableau in Figure 4.2 and the branch with leaf $b < c$. In order to carry out a theory extension step additional literals have to be gathered to constitute a contradictory set with respect to the theory. In the example, we select $a < b$ from the ancestor context and additional literals $c < d$, $d < e$, $e < a$ from input clauses (given schematically in the middle of Figure 4.2). Then the whole set $\{\!|a < b, b < c, c < d, d < e, e < a|\!\}$, called *key set* $\mathcal{K}$ in this context, is passed to the background reasoner. The background reasoner in turn should discover that the key set is contradictory in the given theory $\mathcal{SO}$. To be more precise, the background reasoner is supposed to return a substitution σ — called $\mathcal{T}$-refuter — such that $\mathcal{K}\sigma$ is $\mathcal{T}$-complementary (cf. Def. 4.2.2), meaning that the existential closure of $\mathcal{K}\sigma$ is $\mathcal{T}$-unsatisfiable.

Finally, the remaining literals of the involved clauses are fanned below the selected branch as new proof obligations, and the $\mathcal{T}$-refuter σ is applied to the resulting tableau (cf. the open branches in the right tableau in Figure 4.2, σ is the empty substitution in this case). Since the key set $\{\!|a < b, b < c, c < d, d < e, e < a|\!\}$ is contradictory, the branch containing it (the leftmost branch in the right tableau in Figure 4.2) can be marked with a "×" as closed.

Another theory, where this scheme can be used is decidable terminological theories; in this case, a key set $\mathcal{K}$ is to be determined, and the background reasoner has to find a $\mathcal{T}$-refuter σ such that the existential closure of $\mathcal{K}\sigma$ is unsatisfiable wrt. the $\mathcal{T}$-Box (cf. also Section 4.4.5 below); more problematic theories are those, whose unifiability problem is undecidable. In this case one has to interleave *two* enumeration procedures, which gives rise to the practical problem *how* to interleave them. However, it should be noted that due to the semi-decidability of first-order logic, the background reasoner can be designed as a semi-decision procedure.

The background reasoner for total theory reasoning can be designed as a black box, in particular if the underlying theory is decidable. This is a strength and a weakness at the same time. It is a strength, because both, the foreground calculi and the background calculi can be designed independently from each other. And it is a weakness, because it might be difficult to

[2] Stickel [Stickel, 1985] further distinguishes between *narrow* and *wide* theory reasoning. While the former is characterized by key sets (cf. the text below) consisting of *single literals*, the latter allows *disjunction of literals*. In this text only narrow theory reasoning is considered.

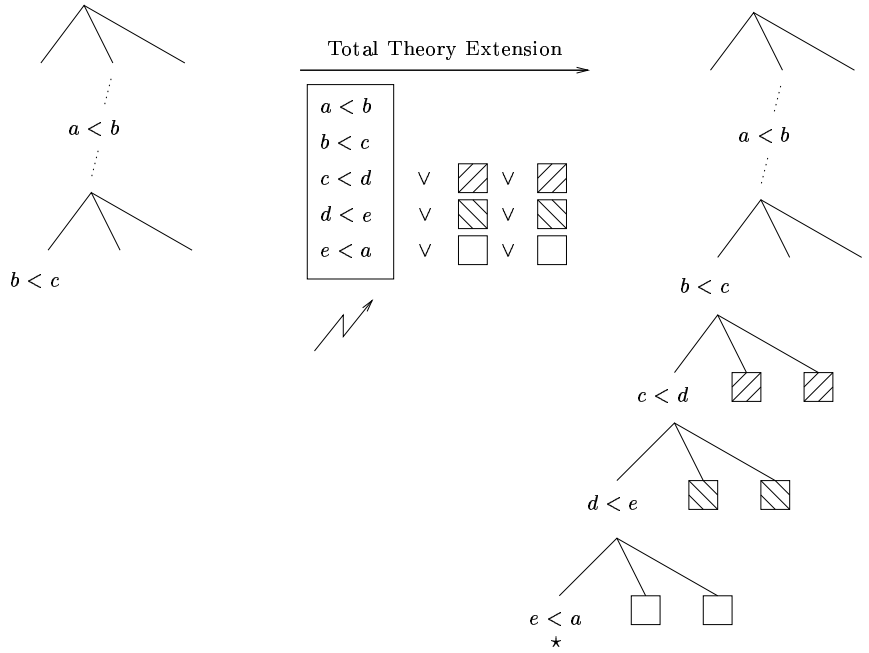


Figure 4.2. A *total* theory extension step in the theory connection calculus.

control the amount of search in the background reasoner. If in the suggested scenario the foreground calculi guesses a key set which admits *no* solution (substitution), or a solution which cannot lead to the closure of the whole tableau under construction, a lot of fruitless search will be performed by the background reasoner. Due to the undecidability of the underlying problem[3] it is unclear when to interrupt the proof attempt of the background reasoner. Further, if the background reasoner has no state information, a lot of potentially useful information derived in that proof search will be lost.

For illustration consider again the theory $\mathcal{SO}$ and clause set M_n from Example 4.1.1. Suppose the refutation starts with $\neg(a_1 < a_n)$, and further suppose that the key sets are guessed by increasing size, which seems to be a plausible strategy. Then $\Sigma_{i=0}^{n}\binom{n}{i} = 2^n$ calls to the background reasoner will result, and only the very last one with M_n as key set is successful. Even worse, a backtracking oriented proof procedure, which forgets during backtracking the generated tableau, will possibly repeatedly solve this key set, even if the final tableau contains only one instance of the problem.

I speculate that calculi with saturating proof procedures, such as resolution, are advantageous if total theory steps are complex, because then at least the re-computations caused by backtracking are avoided.

This assessment should not imply that total theory reasoning cannot be done in an efficient way, in particular for tableau calculi. Instead it should only be pointed out that there are considerations which let the pure "black box approach" look problematic. This problem was also recognized in [Beckert and Pape, 1996]. They offer a solution for the case of equality, using incremental mixed rigid/universal unification and preserving state information of the unification algorithm when branches are extended.

Partial Theory Reasoning. Another framework to solve these problems is *partial* theory reasoning. Instead of having to discover a contradictory set in one single "big" total step, the contradictory set is tried to be discovered in a sequence of better manageable, *decidable*, smaller steps. In order to realize this, the result of such a step is stored as a new proof obligation, called the "residue"[4]. Hopefully, the residue marks an advance in the computation of the contradictory set. It can also be seen as an "interrupted" total step, whose current state is stored in the residue. Semantically, the negated residue states a condition under which the key set becomes contradictory.

In the example, the background reasoner might be passed a key set consisting of the branch literal $a < b$, the leaf literal $b < c$ together with the literals $c < d$ and $d < e$ taken from input clauses; then it computes the residue $a < e$ and returns it to the foreground reasoner (Figure 4.3). The

[3] Namely, the existence of a $\mathcal{T}$-refuter for a given literal set, cf. Note 4.2.2.

[4] In fact, we will later define (Def. 4.3.1) the residue to consist of a clause and a substitution, where the substitution is to be applied to the resulting tableau. This generalizes the concept of $\mathcal{T}$-refuters from above.

fact that the residue is a logical consequence of the key set is mirrored by the fanning of the residue below the branch containing the key set.

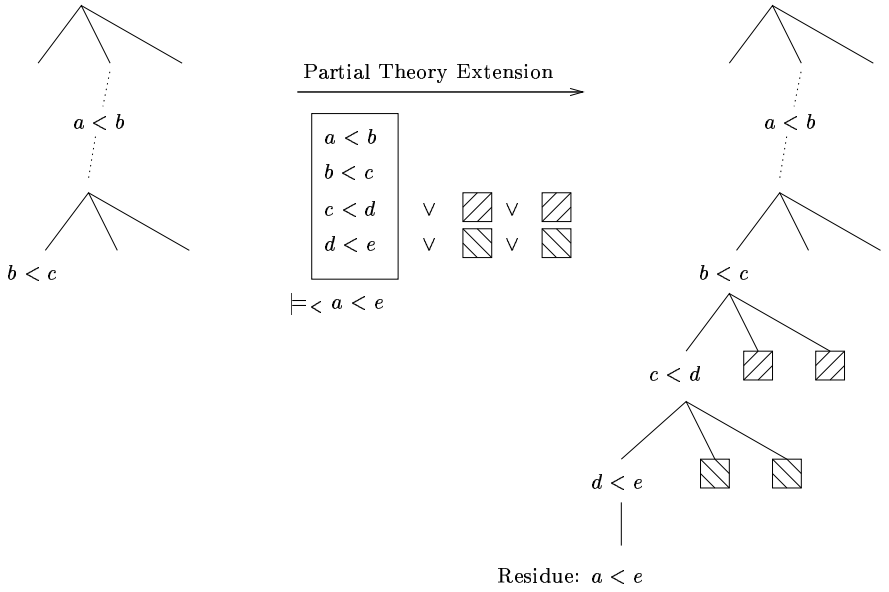


Figure 4.3. A *partial* theory extension step in the theory connection calculus.

Clearly, this partial version alone does not suffice since it never closes a branch. Instead the total extension step is allowed as well, however with the intention that they are applied in a limited way (which depends on the theory).

Compared to total theory reasoning, partial theory is more flexible; the foreground reasoner can take more information — namely the residues – into account when deciding how to assemble the next key set being passed to the background reasoner. Further, due to the "independence of the computation rule", the focus of attention in the proof search can be shifted in a flexible way, e.g. one could always select a branch whose leaf (which might be a residue literal) promises a small search space.

The partial variant is of particular interest for us because there is a technique of automatically constructing background reasoners for partial theory model elimination. This technique, called *linearizing completion*, will be presented in Chapter 5 below. In our example, when applied to the theory of strict ordering, linearizing completion discovers a reasoner in which both total and partial inferences are made up of key sets with at most *two* literals. The example theory $\mathcal{SO}$ and the clause set M_n will have one single *deterministic* refutation, whose (single) open leaves correspond to a "chaining" sequence

of transitivity of the form

$$\neg(a_1 < a_n),\ \neg(a_2 < a_n), \ldots ,\ \neg(a_{n-1} < a_n)\ ,$$

where the last element is an immediate contradiction to $a_{n-1} < a_n$, which is contained in M_n.

4.1.2 Instances of Theory Reasoning

Theory reasoning is a very general scheme and thus has many applications and instances. The classical paper [Stickel, 1985] contains an early list, and general overviews on theory reasoning can be found in [Baumgartner and Petermann, 1998]; an overview with emphasis on equality is given in [Beckert, 1998].

In the last decade, many background reasoners for specialized (classes of) theories have been developed. It is beyond the scope of this text to give a comprehensive overview of all of them. Instead we will try to briefly review some of them, thereby classifying them in terms of "total" and "partial" theory reasoning, and supplying pointers to the literature. Figure 4.4 shows a classification.

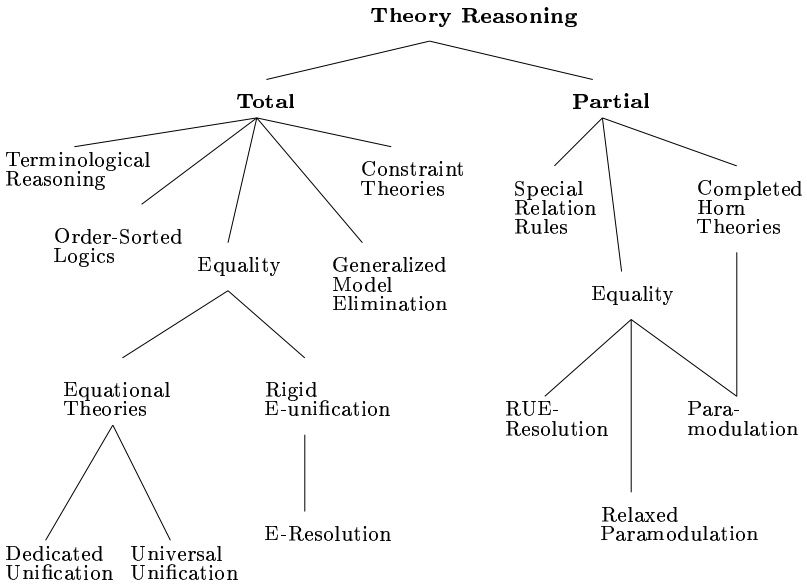


Figure 4.4. A Classification of Theory Reasoning. A connected to B above means that A is an instance of B.

Equality. Possibly the most prominent example for theory reasoning is the theory of equality (cf. Section 2.5.1). It is a research topic of its own. See [Plaisted, 1993] for an in-depth overview.

In its most general setting equality literals may occur both positive and negative in input clauses. Equality can be treated as total theory reasoning. The first approach in this direction was to use *E-Resolution* [Morris, 1969] within the resolution calculus. There, the idea of the systematic search for the key set for inferences is to use *paramodulation* (see below). The major difference to paramodulation calculi proper is that the paramodulants are not contained explicitly in the resolvents.

In [Baumgartner, 1991a] semantic trees have been used to prove the completeness of a resolution calculus with equality with essentially two inference rules, one being "total" similar to E-resolution, and the other being "partial" in that lemma equations are derived.

For analytic calculi, such as connection methods , total equality reasoning is called "rigid E-unification" [Gallier *et al.*, 1992]; [Plaisted, 1995] contains a very readable exposition of the problem and recent results. Rigid E-unification is used within a general tableau calculus in [Beckert and Hähnle, 1992], and it is used within a connection calculus in [Petermann, 1994]. Quite a different approach is suggested in [Degtyarev and Voronkov, 1995a]: In their equality elimination method, a proof consists of conceptually two parts: in the first part, they try to solve the equational part of the input clause set. For this, the basic superposition calculus is used. The resulting clause set then is to be refuted using another calculus, such as the inverse method or a connection method.

I refer to Note 4.2.4 below for a more detailed discussion of rigid E-unification.

Paramodulation [Robinson and Wos, 1969] was invented as an inference rule within the resolution calculus; it allows the derivation from key set[5] $\{P[s],\ l \doteq r\}$ one derives the residue $(P[r])\sigma$, where σ is a MGU for s and l. Thus, paramodulation is an instance of partial theory reasoning. Paramodulation for a connection method is described in [Petermann, 1991b], and Paramodulation for tableau calculi is described in [Fitting, 1990].

Most work on improving paramodulation has been carried out within the resolution framework. In [Peterson, 1983] it is shown that the functional reflexive axioms, i.e. axioms of the form $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ for every n-ary function symbol f, are not needed. The completeness proof presupposes an ordering on terms which is order-isomorphic to ω. A significant improvement was obtained by Hsiang and Rusinowitch, who showed that resolution without functional reflexive axioms is still complete when using more practical reduction orderings [Hsiang and Rusinowitch, 1986; Hsiang and Rusinowitch, 1991].

[5] As usual, the notation $P[s]$ means that the term s occurs at some specific position in the literal P.

Finally, in [Bachmair *et al.*, 1992] compatibility of paramodulation (without functional reflexive axioms) with the "basic" restriction is shown, meaning that paramodulation is needed into only those subterms whose positions are given in the input clause set.

In [Lynch, 1995] an improved version of paramodulation which runs in polynomial time for certain classes is described; the *simultaneous paramodulation* calculus in [Benanav, 1990] avoids many redundant clauses by requiring that paramodulation inferences are carried out simultaneously to *all* occurrences of the target term of the into-clause. Interestingly, the lifting lemma holds for this case.

The paramodulation inference rule can be further restricted by using term orderings for certain purposes: first, term orderings are used to select — usually only maximal — literals inside clauses for inferences; second, paramodulation is restricted to operating in a non order-increasing way; finally, clauses can be simplified to smaller ones in the given term ordering. Resolution calculi of this type (often called "superposition calculi") are described in [Zhang and Kapur, 1988; Bachmair and Ganzinger, 1990; Nieuwenhuis and Rubio, 1995; Bonacina and Hsiang, 1995]. [Bonacina and Hsiang, 1994] classifies strategies with respect to parallelization.

All these refinements of paramodulation were suggested for *non* goal-oriented resolution calculi. Similar improvements (although no simplification, so far) for paramodulation like inferences for *non* goal-oriented tableaux (or connection methods) have been suggested in [Plaisted, 1995; Voronkov and Degtyarev, 1996]. As always in tableaux, the rules have to be defined "rigid", i.e. no copies of the literals in branches are allowed.

Paramodulation for model elimination (which *is* goal-oriented) was described by Loveland [Loveland, 1978]. Unfortunately, none of the above refinements developed for resolution and tableau calculi — no need for functional reflexive axioms, the basic restriction and ordering refinements — can be used immediately in a complete way for goal-oriented calculi such as model elimination ([Furbach *et al.*, 1989a]; see also Section 5.7.1 for a discussion). A typical pathological example which demonstrates the need for functional reflexive axioms consists of solving the goal $\neg P(x,x)$ in presence of the unit clauses $P(f(a),f(b))$ and $a = b$. Although $\mathcal{E}$-unsatisfiable, there is no refutation in, say, model elimination with paramodulation. However, having the functional reflexive axiom $f(x) = f(x)$ at our disposal allows us to instantiate the goal $\neg P(x,x)$ towards $\neg P(f(x),f(x))$ and find a refutation then. Clearly, the usage of functional reflexive axioms should be avoided because they allow for arbitrary instantiation and hence blow up the search space dramatically.

In order to overcome this problem, several modifications of the paramodulation inference rule have been proposed which avoid the functional reflexive axioms but are still compatible with goal oriented calculi see [Snyder, 1991; Moser, 1993]). If the given equations form a canonical rewrite system the calculus of [Plaisted and Greenbaum, 1984] can be used.

More general approaches are based on the inference rule *relaxed paramodulation* [Snyder, 1991]. Relaxed paramodulation would solve the sample problem by decomposing ("top-unifying") the pair $\neg P(x,x)$, $P(f(a),f(b))$ into the subgoals $\neg(x = f(a))$ and $\neg(x = f(b))$ which can be solved without functional reflexive axioms. The general idea is to delay the unification of the arguments of the paramodulating term as new proof obligations[6].

A complete set-of-support resolution calculus based on relaxed paramodulation is described in [Snyder and Lynch, 1991]. The approach of [Moser *et al.*, 1995] improves on this result by additionally incorporating term-ordering constraints. Remarkably, this is done as an extension to model elimination, which is even more goal-oriented than set-of-support resolution. In order to obtain a complete calculus, however, they have to carry out all superposition inferences among positive equations as a separate construction.

Finally, RUE-Resolution [Digricoli and Harrison, 1986] is an approach conceptually "somewhere in the middle" between the fine-grained paramodulation and the coarse-grained E-resolution inference rules. RUE-Resolution was ported to a connection calculus in [Petermann, 1991b].

RUE-Resolution is conceptually related to relaxed paramodulation (see above). As in relaxed paramodulation, resolving $\neg P(x,x)$ and $P(f(a),f(b))$ by the RUE-Resolution inference rule yields the new proof obligation $\neg f(a) = x \vee \neg f(b) = x$. Such derived clauses made up of inequalities are called *disagreement sets*. RUE-Resolution is defined in such a way that without equality literals it reduces to resolution. There is a second inference rule, NRF, which transforms an inequality literal $s \neq t$ in a clause into a disagreement set after first applying a substitution σ to the whole clause.

There are various versions of RUE-Resolution defined, which differ in the admissibility of inference steps depending on certain restrictions ("viability") of the disagreement sets. The "open" form of RUE-Resolution is complete [Digricoli and Harrison, 1986], but suffers from the drawback that it is not specified how the disagreement set and the substitution σ in the NRF rule are computed. Thus, this gives rise to an infinite search space. Unfortunately, more restricted versions are incomplete, at least if the functional reflexive axioms (cf. above) are absent [Bonacina and Hsiang, 1992]. On the other side, adding functional reflexive axioms seems to cast a shadow on the whole approach.

Another approach is to "transform away" the axioms of equality of the input clause set. This means that the transformed set does not contain the equality axioms, and is E-satisfiable iff the original clause set (with equality axioms) is satisfiable. The transformed clause set is suspected to behave better (e.g. less search space) than the original clause set. For instance, Brand's *modification method* [Brand, 1975], when interpreted from the viewpoint of paramodulation, achieves that paramodulation into variables is needed only

[6] The same idea has been proposed for Narrowing procedures (see [Martelli *et al.*, 1986; Dershowitz and Sivakumar, 1988; Hölldobler, 1989]).

rarely[7], and paramodulation below variables is not needed at all. Since the modification method works as a preprocessor to the theorem prover, proper, it is a way to at least partially solve the addressed problems arising with goal-oriented calculi.

A more recent improvement of Brand's modification method was presented in [Bachmair and Ganzinger, 1998a]. The experiments reported there were carried out with the PROTEIN prover (Chapter 6) and showed good results.

A transformational approach for a restricted case, when the input clause set is Horn and the equations form a canonical rewrite system, has been described in [Baumgartner, 1990]. Essentially, the transformation yields an operational treatment for the equations comparable to Narrowing [Hullot, 1980].

Equational Theories. Equational theories are given by a set of universally quantified equations E (cf. Section 2.5.1). From the viewpoint of theory reasoning, equation solving in the theory E, i.e. E-unification [Siekmann, 1989], is actually a special case of total equality reasoning, in that no positive equality literals may occur in input clauses. But then, unlike in the general case, the theory is the *same* for every unification problem. In terms of tableaux: the theory is the same in every branch to be closed. This allows the design of efficient E-unification algorithms which can replace standard unification. There are sound and complete E-unification algorithms for single theories (e.g. associative-commutative unification [Fages, 1984]), and there are universal unification procedures for classes of equational theories (see [Gallier and Snyder, 1990; J.-P. Jouannaud, 1991] for surveys).

Reasoning with equational theories was considered already in 1972 [Plotkin, 1972] for the resolution calculus. For a connection calculus see [Petermann, 1991a]. The mixed universal and rigid E-unification algorithm in [Beckert, 1994] allows combination of equational theories with equations in input clauses (variables stemming from the latter be treated as "rigid" (defined in Note 4.2.4 below) along the branch where the equation comes up).

Terminological Reasoning. While research in equality reasoning is mainly concerned with the development of *efficient* inference rules, there is as well a concern in the field of knowledge engineering of keeping *different kinds of knowledge* apart. Such knowledge representation systems originated with the KL-ONE system [Brachman and Schmolze, 1985]. Nowadays numerous works on defining *concept* languages with well-understood semantics exist (see e.g. [Hollunder, 1990]). In [Paramasivam and Plaisted, 1995] it is shown how theorem provers can be used for subsumption and classification tasks in such languages.

A common feature of such *terminological reasoning systems* [Schmidt-Schauß and Smolka, 1991] is the separation of knowledge into a terminological

[7] Namely to those variables which appear at the right side of an equation, as x in $f(x, y) = x$.

part (called T-Box) and an assertional part (called A-Box). The knowledge about classes of individuals and relationships between these classes is stored in the T-Box, and the knowledge concerning particular individuals can be described in the A-Box.

One of the most prominent examples of those approaches is KRYPTON [Brachmann *et al.*, 1983], where the semantic net language KL-ONE is used as a theory defining language, which is combined with a theorem prover for predicate logic. This system is based on the theory resolution calculus [Stickel, 1985]. In [Kerber *et al.*, 1993] a resolution framework for hybrid reasoning is proposed. Essentially, their theory is given as a set of ground unit literals. Consequently, their resolution inference can be seen as an instance of total narrow theory resolution where every key set consists of exactly one literal.

In [Baumgartner and Stolzenburg, 1995] a model elimination calculus with constraints was instantiated with a constraint solver for a terminological language. In [Hanschke and Hinkelmann, 1992] T-Box reasoning was extended by a rule formalism. In Section 4.4.5 below, a combination of model elimination with terminological reasoning will be presented.

With respect to classification, all these systems are instances of total theory reasoning.

(Order-)Sorted Logics. Sorts appear naturally in various problem domains, such as "mathematics" ("integers", "reals", etc.) or when reasoning about program semantics (e.g. "environments", "variables", "values" etc.). Semantically, sorts divide the universe of discourse into possible non-disjoint parts. Typically, sorts are attached to variables, thus restricting their assignments to values of the respective universe. On the operational side, sort declarations, i.e. the definition of the sorted signature and sort order relationships, can often be compiled into efficient unification algorithms (see e.g. [Schmidt-Schauß, 1989]; the overview [J.-P. Jouannaud, 1991] also covers the sorted case). Roughly, if $x_1 : t_1$ means that the variable x_1 is of type t_1, then the unification of the variables $x_1 : t_1$ and $x_2 : t_2$ succeeds if t_1 and t_2 have a common subsort, which is the sort of the unified variables.

By sorted unification drastic search space pruning can be achieved. This was observed for the resolution calculus (with equality) by Walther [Walther, 1987]. A sorted tableau calculus is described in [Schmitt and Wernecke, 1989]. A general technique for obtaining completeness results for sorted calculi was suggested by Frisch [Frisch, 1991]. C. Weidenbach proposes giving up the "static" view of sorts — all the sort declarations are known in advance of the proof — and to identify sorts with unary predicates which may also come up dynamically ([Weidenbach, 1993] during resolution derivations; the approach was ported to tableaux in [Weidenbach, 1995]).

As with equality, sort information can occasionally be "transformed away". For the case of disjoint subsorts, the sort structure can be coded as terms, and standard unification can be used for sort reasoning [Schmitt and Wernecke, 1989]; U. Furbach [Furbach, 1991] proposes translating the sort

structure into an equational theory, which can be directed into a canonical rewrite system.

Equivalences. Reasoning with equivalences, i.e. formulas of the form $L \equiv K$ where L and K are literals, shares many properties with equational reasoning. In [Socher-Ambrosius, 1990] the paramodulation inference rule was reformulated for this setting. Additionally, it was shown that orientable equivalences can be used as rewrite rules for simplification. In [Socher, 1992] and [Brüning, 1995] (using a connection calculus) it was suggested that one generate equivalences dynamically; for this, it requires recognition of a cyclic sequence $K_0 \to K_1, \ldots, K_{m-1} \to K_m, K_m \to K_0$ of two-literal clauses constructed during proof search. It is easy to see that $K_i \equiv K_j$ for $0 \leq i, j \leq m$. Then, in inferences these equivalences are chained in order to prove connections as theory-complementary (in the theory given by the equivalences). In other words, this is an instance of total theory reasoning.

Other inference rules for reasoning with equivalences have been proposed in [McMichael, 1990] and [Lee and Plaisted, 1989].

Procedural attachments. In [Myers, 1994] replacement of ground terms by equivalent ones is proposed in order to speed up computations. A similar idea is developed in [Sikka and Genesereth, 1994]. A prototypical example would be to replace arithmetic based on successor-arithmetic by built-in arithmetic. The approach of U. Furbach [Furbach, 1991] includes this possibility as a special case. He proposes having for functions (1) an axiomatic description, and (2) a representation of its inverse (if existent) and (3) a simplification procedure. While (1) and (2) are used for equation *solving*, the alternative (3) can be used for replacing ground terms by simpler ones.

Compiled Theories. By this notion, I mean any systematic technique of transforming a given theory into dedicated inference rules. For instance, *Z-Resolution* [Dixon, 1973] allows us to build in a theory consisting of two literal clauses only. A more recent improvement was given in [Ohlbach, 1990].

A system with dedicated inference rules for reasoning with (total) strict orderings (cf. the informal introduction to theory reasoning in this section) was described in [Hines, 1992]. Hines also designed a prover with inference rules for set inclusion ($\subseteq$) [Hines, 1990]. In [Bachmair and Ganzinger, 1994] and [Bachmair and Ganzinger, 1998b] it is demonstrated that the "chaining" inference rule of [Hines, 1992] for orderings can be obtained by instantiating a more general inference rule for transitive relations (within the superposition calculus).

As a general method, in [Murray and Rosenthal, 1987] a matrix method with built-in theories is presented. Unlike in Stickel's general theory resolution [Stickel, 1985], the theory is *not* considered as a black box. Instead the theory, say $\mathcal{T}$, is supposed to be defined by a set of clauses. They propose closing $\mathcal{T}$ under application of binary resolution, modulo subsumption. The resulting — in general infinite — system $\mathcal{T}^*$ is used for total theory extension steps

by simultaneously resolving away all literals from a clause from $\mathcal{T}^*$ against given literals in input clauses.

The *(extended) special relation rules* [Manna and Waldinger, 1986; Manna *et al.*, 1991] are inference rules which are derived from certain axiomatically given properties of relations. More specifically, *monotonicity properties* are declarations of properties of relations "$\prec_1$" and "$\prec_2$", which determine the conditions (including polarity) under which subterm replacements can be carried out. Alternatively, these declarations can be described (disregarding polarities) by axioms of the following form:

$$\begin{array}{ll} \textit{if} \quad u \prec_1 v & \textit{if} \quad u \prec_1 v \\ \textit{then} \quad r(\ldots u \ldots) \prec_2 r(\ldots v \ldots) & \textit{then} \quad r(\ldots v \ldots) \prec_2 r(\ldots u \ldots) \ . \end{array}$$

This scheme covers many interesting relations, such as the axioms (schemes) "symmetry", "transitiviy", "P-substituivity" and "F-substituivity" which comprise — short of "reflexivity" — the axioms of equality (cf. Def. 2.5.2). For instance, "symmetry", i.e. *if* $u = v$ *then* $v = u$, is immediately seen to be an instance of the right scheme above. "Transitivity" is covered by instantiating in the right scheme "$\prec_2$" with logical implication, "$\rightarrow$". Examples of other theories expressible in this language are ordering relations and subset relations.

It is intended to read these axioms operationally in the way suggested by the notation. The thus derived "special relation (SR) inference rules" are embedded into a resolution calculus. Unfortunately, the completeness of this system is still open [Manna *et al.*, 1991].

Finally, the method of *linearizing completion* [Baumgartner, 1996] takes as input a Horn clause theory and derives as output a possibly infinite set of inference rules (Section 4.5) which can be used as a background reasoner within partial theory model elimination. This method will be described in detail in Chapter 5. There, the relation to the other "compiled theory" approaches will be given.

Constraint Reasoning. Theory reasoning is related to *constraint reasoning*. Like theory reasoning, it is a general framework to instantiate with domain specific problem solvers. In the following we will refer to Bürckert's version of constraint resolution [Bürckert, 1991], which is slightly more general than Frisch's version [Frisch, 1991]. Constraint reasoning was investigated for model elimination in [Baumgartner and Stolzenburg, 1995].

Clauses are separated in two parts, written as $R \mid C$, where R is a clause in the usual sense, and C is a formula ("constraint") whose free variables are quantified in R.

A *constraint theory* is given by a class of theories in the usual sense. Thus, if the class is a singleton (i.e. contains only one theory), and the clauses are relativized in the usual way (i.e. $R \mid C$ is read as the sentence $\forall(C \rightarrow R)$) then we have the same setup as in theory reasoning calculi. Semantically, a

clause set is unsatisfiable wrt. a constraint theory iff it is unsatisfiable in each theory of the class.

The resolution inference rule is adapted to accumulate (conjoin) the constraint parts of the parent clauses. A refutation consists of a derivation of finitely many (if the theory class is countable) empty clauses $\square \mid C_1, \ldots \square \mid C_n$ such $\exists(C_1 \vee \cdots \vee C_n)$ is valid in the constraint theory. If the theory class is a singleton, then only one empty clause need be derived.

On the one hand, constraint reasoning is more general than theory reasoning, as constraints may be treated lazily. That is, no theory unifiers need to be computed during proof search. Instead it suffices to establish the satisfiability of the accumulated theory unification problems, as indicated.

On the other hand, constraint reasoning is more specialized than theory reasoning, as in constraint reasoning the foreground theory, say M, must be a conservative extension of the background theory $\mathcal{T}$ (that is, an interpretation I is a model of $\mathcal{T} \cup M$ iff I is a model of $\mathcal{T}$ alone). This is achieved in [Bürckert, 1991] by requiring that the signatures of the clause language and the constraint language are disjoint. This leads to weird situations sometimes. Take for instance the theory of equality. Since E-interpretations define the semantics of every predicate symbol of the signature, every literal must move into the constraint part. Thus, *every* input clause $L_1 \vee \cdots \vee L_n$ has to be transformed into $\square \mid \exists(\neg L_1 \wedge \cdots \wedge \neg L_n)$ in a first step. Now, the entire reasoning is moved to the constraint solving part.

4.2 A Total Theory Connection Calculus

We are going to a more formal treatment of the calculus described in the previous section. Since in Sections 4.4 and 4.5 below more refined calculi (theory model elimination) will be defined, the question arises whether the rather detailed treatment of the thus preliminary total theory connection calculus is justified. There are the following answers to this question:

- The somewhat complicated theory model elimination calculus is best explained as a refinement of the theory connection calculus.
- Some concepts can be introduced and discussed using the simpler theory connection calculus, thus possibly making the presentation more readable.
- The theory connection calculus is established in the literature and thus deserves attention.
- A critical analysis of the theory connection calculus requires setting up the stage to some detail (the critique will be stated in Section 4.3.1).

We proceed by first lifting "unifiers" to the theory level, and then turn towards the calculus proper.

4.2.1 Theory Refuting Substitutions

Definition 4.2.1 ($\mathcal{T}$-Complementarity).
A literal multiset $M = \{\!|L_1, \ldots, L_n|\!\}$ is called $\mathcal{T}$-complementary *iff the existential closure*[8] $\exists(L_1 \wedge \ldots \wedge L_n)$ *is $\mathcal{T}$-unsatisfiable. M is called* minimal $\mathcal{T}$-complementary *iff M is $\mathcal{T}$-complementary and all proper subsets are not $\mathcal{T}$-complementary.*

The notion of $\mathcal{T}$-complementarity generalizes the standard notion of "complementarity" (two literals are complementary iff one of them is the negation of the other) towards theories. Obviously, for the empty theory, minimal $\mathcal{T}$-complementary literal sets consist of exactly two complementary literals.

Note 4.2.1 (Alternate characterization). By using the definitions and Proposition 2.5.1 one easily obtains that the literal multiset $M = \{\!|L_1, \ldots, L_n|\!\}$ is $\mathcal{T} - complementary$ iff $\models_{\mathcal{T}} \forall(\overline{L_1} \vee \cdots \vee \overline{L_n})$.

Note 4.2.2 (Undecidability of $\mathcal{T}$-Complementarity). In general, the problem whether a given literal set M is $\mathcal{T}$-complementary is undecidable. It is surprising that this even holds if we restrict to theories which are axiomatized by a *single* clause and to *ground* sets M. This follows easily from the result in [Schmidt-Schauß, 1988] which says that for two clauses D and C it is undecidable whether $\forall D \models \forall C$. In [Marcinkowski and Pacholski, 1992] this result is strengthened towards Horn-Clauses.

Thus we consider universal theories $\mathcal{T}$ consisting of one single Horn clause only. It holds by the results of Chapter 2: $\mathcal{T} \models \forall C$ iff $\mathcal{T} \cup \{\exists \neg C\}$ is unsatisfiable iff $\mathcal{T} \cup \{\neg sk(C)\}$ iff unsatisfiable (for any Skolem form $sk(C)$ of C) iff $\mathcal{T} \cup \{\overline{L_1}, \ldots, \overline{L_n}\}$ is unsatisfiable, where $sk(C)$ is the (ground) clause $L_1 \vee \cdots \vee L_n$, iff $\{\overline{L_1}, \ldots, \overline{L_n}\}$ is $\mathcal{T}$-unsatisfiable iff $\{\overline{L_1}, \ldots, \overline{L_n}\}$ is $\mathcal{T}$-complementary (because for ground sets these notions are the same). Hence, with the original problem of logical consequence of Horn clauses being undecidable, $\mathcal{T}$-complementarity is undecidable as well.

The same holds for derived notions below. For instance, the existence of a substitution σ such that M is $\mathcal{T}$-complementary (a "$\mathcal{T}$-refuter") is undecidable as well, because this problem class includes the $\mathcal{T}$-complementarity problems.

Building on Definition 4.2.1, the standard notion of a unifier is replaced by the following concept (called $\mathcal{T}$-unifier in [Baumgartner and Petermann, 1998]):

Definition 4.2.2 ($\mathcal{T}$-Refuter [Baumgartner *et al.*, 1992]). *A substitution σ is a called a* $\mathcal{T}$-refuter *for a literal multiset M iff $M\sigma$ is $\mathcal{T}$-complementary. A $\mathcal{T}$-refuter for M is called* minimal *iff $M\sigma$ is minimal $\mathcal{T}$-complementary (i.e. iff $M\sigma$ is $\mathcal{T}$-complementary, but there is no proper subset of $M\sigma$ which is $\mathcal{T}$-complementary).*

[8] Def. 2.2.5.

Our definition of $\mathcal{T}$-refuter generalizes the concept *rigid E-unifier* [Gallier *et al.*, 1987] to more general theories than equality (rigid E-unification is discussed in Note 4.2.4). A dual notion, "unifier with respect to $\mathcal{T}$-complementary literal sets", has been used within an affirmative setting [Petermann, 1992].

As for non-theory calculi we are interested in restrictions on the set of possible $\mathcal{T}$-refuters to be considered for a given literal set. This will be introduced below ("most general sets of $\mathcal{T}$-refuters", Section 4.4.1) but for the moment this concept is not needed.

Example 4.2.1 ($\mathcal{T}$-Refuters). Consider the theory $\mathcal{E}$ of equality. The set

$$M = \{\!|P(x),\ y = f(y),\ \neg P(f(z))|\!\}$$

is clearly $\mathcal{E}$-unsatisfiable when read as a set of unit clauses. However it is not $\mathcal{E}$-complementary, since e.g. the ground instance $P(a) \wedge (a = f(a)) \wedge \neg P(f(b))$ of

$$\exists x, y, z\ (P(x) \wedge (y = f(y)) \wedge \neg P(f(z)))$$

is not $\mathcal{E}$-unsatisfiable. But the substitution $\sigma = \{x \leftarrow z, y \leftarrow z\}$ is a $\mathcal{E}$-refuter since the formula $\exists z\ (P(z) \wedge z = f(z) \wedge \neg P(f(z)))$ obtained from $M\sigma$ is $\mathcal{E}$-unsatisfiable.

A word of warning might be appropriate:

Note 4.2.3 (Warning). The previous example (Example 4.2.1) might suggest that $\mathcal{T}$-complementarity of a set M of Σ-literals is equivalent to saying that $M\gamma$ is $\mathcal{T}$-unsatisfiable for every ground substitution γ employing Σ-terms. This, however, is not true. Take e.g. $M = \{\!|p(x),\ \neg p(a)|\!\}$ and the "empty" theory. Clearly, $\exists x p(x) \wedge \neg p(a)$ is *not* unsatisfiable, although for every ground substitution γ — there is only one, namely $\gamma = \{x \leftarrow a\}$ — we have that

$$\{\!|p(x),\ \neg p(a)|\!\}\{x \leftarrow a\} = \{\!|p(a),\ \neg p(a)|\!\}$$

is $\mathcal{T}$-unsatisfiable.

The problem here is the same as in clause logic (cf. Theorem 2.4.2 and the discussion following it), namely the presence of an existential quantifier. In order to treat it correctly, one would have to replace the variables by *new* Skolem constants. This operation preserves $\mathcal{T}$-unsatisfiability.

More precisely: suppose $\mathcal{T}$ is a universal Σ-theory (cf. Def. 2.5.5). Hence $\mathcal{T} = Cons(Ax)$ for some axiom set Ax of universal Σ-sentences. Let $L'_1 \wedge \ldots \wedge L'_n$ be a Skolem form (Section 2.4.1) of $\exists(L_1 \wedge \ldots \wedge L_n)$, where $\{L_1, \ldots, L_n\} = M$. Assuming the set of constant symbols in Σ large enough (w.l.o.g. this can always be achieved), we may assume that the replacement of variables by Skolem constants is done in such a way that only constants are introduced which occur neither in M nor in Ax. Then it holds

M is $\mathcal{T}$-complementary
iff $\exists(L_1 \wedge \ldots \wedge L_n)$ is $\mathcal{T}$-unsatisfiable
iff $\{\exists(L_1 \wedge \ldots \wedge L_n)\}$ is $\mathcal{T}$-unsatisfiable
iff $\{\exists(L_1 \wedge \ldots \wedge L_n)\} \cup Ax$ is unsatisfiable (by Prop. 2.5.1, (a)-(e))
iff $\{(L'_1 \wedge \ldots \wedge L'_n)\} \cup Ax$ is unsatisfiable (by Theorem 2.4.1)
iff $(L'_1 \wedge \ldots \wedge L'_n)$ is $\mathcal{T}$-unsatisfiable (by Prop. 2.5.1, (a)-(e))
iff M' is $\mathcal{T}$-unsatisfiable
where $M' = \{L'_1, \ldots, L'_n\}$.

Thus, in sum, $\mathcal{T}$-satisfiability is preserved by Skolemizing away the existentially bound variables. An alternate way would be to simply strip the existential quantifier and consider the variables as new constant symbols.

Note 4.2.4 (Rigid E-unifiers). This latter terminology — treating variables as new constants — is used in the context of *rigid E-unification* [Gallier *et al.*, 1987; Gallier *et al.*, 1992]. A standard definition (e.g. [Beckert and Pape, 1996]) is to define a rigid E-unifier for a given set $M = \{s_1 = t_1, \ldots, s_n = t_n\}$ of Σ-equations and goal Σ-equation $G = (s = t)$ as any substitution σ such that

$$M\sigma \models_{\mathcal{E}(\Sigma')} \forall((s = t)\sigma) \ , \tag{4.1}$$

where Σ' is like Σ, except that the Σ-variables $Var(M\sigma)$ are new constant symbols in Σ' ($\mathcal{E}(\Sigma')$ is the theory of equality for Σ', cf. Section 2.5.1). Note that $(s = t)\sigma$ still might contain Σ-variables. Now, since $M\sigma$ is a set of Σ'-ground literals, and the introduced constants are "new", it is straightforward to show (use the previous Note 4.2.3) that Equation 4.1 is equivalent to the condition that σ is a $\mathcal{E}(\Sigma)$-refuter for $M \cup \{\!|\neg(s = t)|\!\}$. Thus, rigid E-unification fits well into our framework.

Interestingly, rigid E-unification is *decidable*. More precisely, decision procedures (see [Gallier *et al.*, 1992; de Kogel, 1995; Beckert, 1994]) exist which compute for any *rigid E-unification problem* $\langle M, G\rangle$ a *finite* and "complete" set of equivalent rigid E-unifiers. For instance, if $M = \{f(a) = a\}$ and $G = (x = a)$ then $\sigma = \{x \leftarrow a\}$ is the single rigid E-unifier computed. Finiteness is possible by discarding unifiers which are equivalent modulo $M\sigma$. In the example, any substitution $\sigma_n = \{x \leftarrow f^n(a)\}$ is a rigid E-unifier for G but can be discarded.

When building in equality into connection methods, one usually has to solve *simultaneous* Rigid-E-unification problems. This means to find for given sets $\{M_1, \ldots, M_n\}$ and equations $\{s_1 = t_1, \ldots, s_n = t_n\}$ a substitution σ such that

$$M_i\sigma \models_{\mathcal{E}(\Sigma')} \forall((s_i = t_i)\sigma) \qquad (\text{for } i = 1, \ldots, n).$$

Such a problem arises if a tableau with n branches is constructed, because then one has to find σ such that all branches are closed simultaneously.

Unlike the non-simultaneous rigid E-unification problem, the simultaneous variant is *un*decidable [Degtyarev and Voronkov, 1995b]. Hence, any decision procedure must be necessarily incomplete. The problem is in the combination of n finite complete sets of rigid E-unifiers, because these are relative to n *different* equational theories $M_i\sigma$.

There are several approaches to cope with this problem: one approach is to give up decidability and to enumerate rigid E-unifiers. This is the approach taken in [Petermann, 1994]. Petermann enumerates rigid E-unifiers modulo a theory which is *the same* for every problem, namely the empty theory. Thus, in the example, the complete set of rigid E-unifiers is the infinite set $\{\sigma, \sigma_1, \ldots, \sigma_n, \ldots\}$.

Another approach is to restrict to *decidable* cases. In [Plaisted, 1995] decidability is shown for the case of unit equations, Horn equations, and "splittable" theories, where a clause set can be partitioned by splitting such that any positive equation occurs only in unit clauses.

A more general approach is to rely on the *in*complete simultaneous rigid E-unification algorithms, which can still be used to obtain a complete calculi. This was shown in [Voronkov and Degtyarev, 1996]. Completeness is recovered by allowing multiple variants ("amplifications") of clauses along the tableau branches, i.e. the M_is have to be enriched by sufficiently many variants. The procedure given there has the attractive property that the σ_i's can be computed branch-local, i.e. without taking other σ_j's ($j \neq i$ into account).

This note is continued in Note 4.2.6 below.

For later use we collect some facts about $\mathcal{T}$-refuters now:

Proposition 4.2.1 (Facts about $\mathcal{T}$-refuters). *Let M be a literal set and γ be a substitution.*

1. *If M is $\mathcal{T}$-complementary then also $M\gamma$ is $\mathcal{T}$-complementary.*
2. *Suppose γ is a minimal $\mathcal{T}$-refuter for M, and suppose that $\sigma \leq \gamma[Var(M)]$ also is a $\mathcal{T}$-refuter for M. Then σ is also a minimal $\mathcal{T}$-refuter.*

Proof. Concerning 1, by Note 4.2.1, $M = \{\!|L_1, \ldots, L_n|\!\}$ is $\mathcal{T}$-complementary iff $\models_{\mathcal{T}} \forall(\overline{L_1} \vee \cdots \vee \overline{L_n})$. By Lemma 2.5.3 it follows $\models_{\mathcal{T}} \forall((\overline{L_1} \vee \cdots \vee \overline{L_n})\gamma)$, which is the same as $\models_{\mathcal{T}} \forall(\overline{L_1\gamma} \vee \cdots \vee \overline{L_n\gamma})$. By Note 4.2.1 again, then $\{\!|L_1\gamma, \ldots, L_n\gamma|\!\} = M\gamma$ is $\mathcal{T}$-complementary.

Concerning 2, we have to show that for each strict subset $N \subset M$, $N\sigma$ is not $\mathcal{T}$-complementary. Suppose, to the contrary, that for some strict subset $N \subset M$ we have that $N\sigma$ is $\mathcal{T}$-complementary. Since $M\gamma$ is given as minimal $\mathcal{T}$-complementary, $N\gamma$ is not $\mathcal{T}$-complementary. It holds $\gamma|\,Var(M) = \sigma\delta|\,Var(M)$ for some substitution δ. With $Var(N) \subseteq Var(M)$ it follows $N\gamma = N\sigma\delta$. Thus, by (1), $N\sigma$ is not $\mathcal{T}$-complementary either, which contradicts the choice of $N\sigma$.

Note 4.2.5 (Minimality). The converse of Proposition 4.2.1.2 does not hold in general. In order to see this, let "=" denote an equivalence relation (cf.

Example 2.3.2). Clearly, $\{x = y,\ \neg(y = x)\}$ is minimal unsatisfiable in that theory, however, because of reflexivity, its instance $\{a = a,\ \neg(a = a)\}$ is *not* minimal unsatisfiable.

For the non-theory case the converse *does* hold, because obviously any minimal unsatisfiable set contains exactly two literals.

4.2.2 Definition of a Total Theory Connection Calculus

Now we are in a position to formally define the above introduced calculus.

Definition 4.2.3 (Total Theory Connection Calculus (TTCC)). *Let $\mathcal{T}$ be a universal theory. The inference rule* total theory connection calculus extension step *is defined as follows (cf. Figure 4.5):*

TTCC-Ext*:*

$$\frac{[p],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{([p] \circ R_1,\ [p \cdot L_1] \circ R_2,\ \ldots,\ [p \cdot L_1 \cdots L_{n-1}] \circ R_n,\ [p \cdot L_1 \cdots L_{n-1} \cdot L_n]\times,\ \mathcal{Q})\sigma}$$

where σ is a $\mathcal{T}$-refuter for $p \cup \{L_1, \ldots, L_n\}$

(characterizing property for TTCC-Ext)

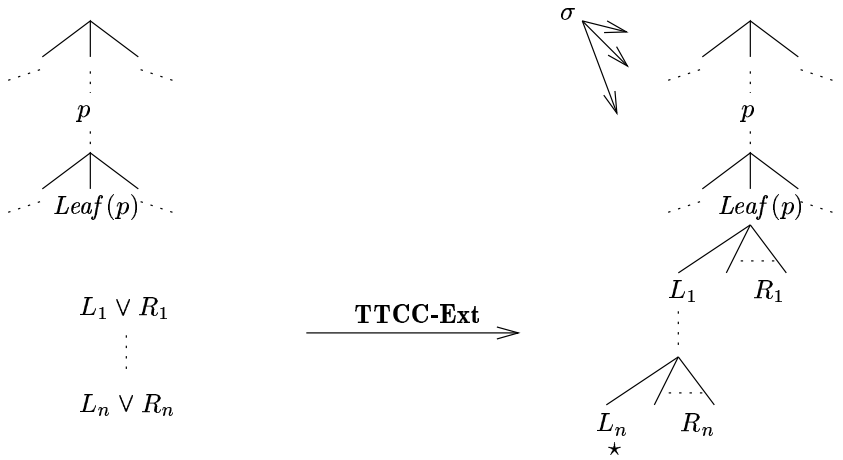


Figure 4.5. Total theory extension step in the theory connection calculus.

The calculus total theory connection calculus, TTCC*, consists of the single inference rule* TTCC-Ext.

The characterizing property for TTCC-Ext suffices to guarantee the soundness of the calculus.

Note 4.2.6 (Early vs. late branch closure). An alternative way to define the theory connection calculus can be sketched as follows: recall that TTCC-Ext steps use a $\mathcal{T}$-refuter σ which is to be applied to the resulting tableau. Alternatively to this "eager" policy, one could defer the computation of the $\mathcal{T}$-refuters. That is, in the course of a derivation, a tableau is constructed without applying any substitution. But then, in order to find a refutation, a "tableau closure rule" has to be applied eventually. By a "tableau closure rule" we mean the possibility to apply some substitution to a given tableau such that all branches become $\mathcal{T}$-complementary (cf. e.g. [Fitting, 1990]). Thus, the closing substitution is computed "late", as the very last step of a refutation. See [Fitting, 1990] for a description of a (non-theory) tableau calculus; [Beckert and Pape, 1996] contains an empirical comparison between the late and early strategies and reports advantages of the "early" over the "late" variant.

We continue on Note 4.2.4 on rigid E-unifiers. As mentioned there, the problem of the simultaneous rigid E-unification arises naturally as the need to find a substitution which closes all branches simultaneously. In other words, branches are closed "late". In practice it is difficult to "decide" whether to spend the resources on trying to find a simultaneous rigid E-unifier, or to further expand the tableau (of course, it is in a strict sense not necessary to compute the closing substitution at all — it suffices to proof the one *exists*. See also the discussion on constraint reasoning above (Section 4.1.2)).

Besides this, another problem for proof procedures is that due to the absence of branch closing substitutions the search is hardly guided. An approach lying "somewhere in the middle" is to basically follow the "late" approach, but also to check in the course of the derivation whether branch closure is possible and to make use of this information.

It is easy to recognize the inference rules of non-theory model elimination and the connection calculus (ME and CC, Definition 3.2.3) as instances of the TTCC-Ext inference rule. This is due to the fact that TTCC-Ext can take an arbitrary (even zero) number of ancestor literals, as is needed to model the extension step, and it can take an arbitrary number of extending literals (even zero), as is needed to model the reduction step. More precisely, recall that an extension step takes a branch $[p]$ and an input clause $L \vee R$ such that for some substitution σ, $K\sigma = \overline{L}\sigma$, where K is the leaf of p. This is equivalent to having that $\exists((K \wedge L)\sigma)$ is unsatisfiable. Clearly, $\exists((p \wedge K \wedge L)\sigma)$ then is unsatisfiable as well. This, however, is just the characterizing property for TTCC-Ext. The case for the reduction step is obtained analogously. Thus, less surprising, ME and CC are both instances of the theory connection calculus.

The converse, however, does not hold for ME. This is because in TTCC the link condition of ME is not enforced (extensions can be done with a connection to any branch literal). Thus, not every TTCC derivation is also

a ME derivation. We would like to establish a link condition for the theory version, in order to lift ME properly to the theory level. This can be done as follows:

Definition 4.2.4 (TTCC-Link). *Let $\mathcal{T}$ be a universal theory. The inference rule* total theory connection calculus extension step with link condition *is the same as* TTCC-Ext *(Def. 4.2.3), except that the characterizing property is changed in the following way:*

TTCC-Link-Ext*:*

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{([p \cdot K] \circ R_1,\ [p \cdot K \cdot L_1] \circ R_2,\ \ldots,[p \cdot K \cdot L_1 \cdots L_{n-1}] \circ R_n,\ [p \cdot K \cdot L_1 \cdots L_{n-1} \cdot L_n]\times,\ \mathcal{Q})\sigma}$$

where σ is a minimal $\mathcal{T}$-refuter for the key set

$$\mathcal{K} = \mathcal{B} \cup \{\!\{K\}\!\} \cup \{\!\{L_1, \ldots, L_n\}\!\},$$

for some subset $\mathcal{B} \subseteq p$.

(characterizing property for TTCC-Link-Ext)

The calculus total theory connection calculus with link condition, (TTCC-Link), *consists of the single inference rule* TTCC-Link-Ext. *The requirement that the leaf of the extended branch, K, is contained in the key set also referred to as the* link condition. *Extending Definition 3.2.4, we will write* TTCC-Link-Ext *inferences as* $([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},\sigma,\{\!\{L_1 \vee R_1,\ldots,L_n \vee R_n\}\!\}} (\mathcal{Q}',\ \mathcal{Q})\sigma$.

As an example consider Figure 4.2 again. The inference step depicted there is a TTCC-Link-Ext step, because the empty substitution is a minimal $\mathcal{T}$-refuter for the key set $\{\!\{a < b,\ b < c,\ c < d,\ d < e,\ e < a\}\!\}$.

The characterizing property of TTCC-Link-Ext is more restrictive than that of TTCC-Ext in that a *minimal* $\mathcal{T}$-refuter has to be used. The price to be paid in order to preserve completeness is that we can no longer insist that the whole branch literals $p \cdot K$ are part of the key set[9]. Instead, only *some* literals are selected. However, as is done in the definition, it can still be required that the leaf K is part of the key set $\mathcal{K}$. Again, this is the essential difference between TTCC and TTCC-Link.

Due to the link condition, it is easily verified for the case of the empty theory the TTCC-Link-Ext inference rule reduces to both ME-Ext and Red. In order to see this, notice that for the empty theory each *minimal* $\mathcal{T}$-complementary multiset consists of *exactly* two complementary literals. Consequently, each key set used in a TTCC-Link-Ext step consists of *exactly* two literals, which are made complementary by the substitution σ of the inference

[9] It is very easy to find a respective counterexample against the completeness of such a restriction.

(using a larger key set would contradict the requirement that σ is a *minimal* $\mathcal{T}$-refuter). Thus, since one literal of the key set is fixed to be the leaf literal, the other one can either stem from an input clause ($n = 1$ in TTCC-Link-Ext) or stem from the branch to be extended ($n = 0$ in TTCC-Link-Ext). In the former case a ME-Ext, and in the latter case a Red step results.

We summarize these observations in the following theorem:

Theorem 4.2.1 (TTCC-Link instantiates to ME). *Let $\mathcal{T}$ be the empty theory. Then any ME derivation is also a TTCC derivation and a TTCC-Link derivation. The converse holds for TTCC-Link, but not for TTCC.*

Note 4.2.7 (Discussion of "minimality" requirement). The restriction to use *minimal* $\mathcal{T}$-refuters in extension steps is a rather strong restriction. There are several alternatives to relieve the background reasoner from computing minimal $\mathcal{T}$-refuters. We will discuss these alternatives now.

One straightforward idea would be to simply drop the minimality requirement. But then, it is easy to see that TTCC-Link is nothing but a notational variant of TTCC: every TTCC-Ext step is also a TTCC-Link-Ext step, and vice versa. Hence, in terms of a proof procedure, the local search space of TTCC-Link-Ext would be as big as that of TTCC-Ext and nothing would be gained.

A more reasonable alternative would be to take properties of the theory into account and to relax the minimality requirement only partially. For instance, think of the theory as instantiated by a terminological database (a $\mathcal{T}$-Box, cf. Section 4.4.5). In an appropriate scenario one would have that e.g. $\{\!|\neg animal(X),\ dog(X)|\!\}$ is minimal $\mathcal{T}$-complementary. We may assume that in the $\mathcal{T}$-Box nothing is said about "ordinary" predicate symbols, such as P. Then it seems natural to relax the minimality requirement for proper $\mathcal{T}$-Box reasoning (because minimal $\mathcal{T}$-complementary literal sets might not be easily identifiable by syntactic means), while keeping the minimality requirement for ordinary literals (because only complementary literals $\{\!|P, \neg P|\!\}$ have to be considered).

Now, either we allow ourselves to guess the key sets, or the key sets are assembled using some different strategy. The first case includes the case of minimal $\mathcal{T}$-complementary sets, thus rendering the whole problem vacuous. A plausible strategy for the second case would be to assemble the *whole* set of (ground) $\mathcal{T}$-Box literals occurring in the input clause set as key sets. While this strategy can be used for TTCC[10], we will show that it leads to incompleteness for TTCC-Link. To this end suppose the following clause set as given:

$$dog(goofy) \leftarrow \qquad \text{(Goofy\_is\_Dog)}$$
$$mouse(mini) \leftarrow P \qquad \text{(Dummy)}$$
$$\leftarrow animal(goofy) \qquad \text{(Query)}$$

[10] Note that the link condition for ordinary literals does not hold then.

Under the stated assumptions, this clause set is $\mathcal{T}$-unsatisfiable. Suppose the initial tableau is constructed with clause (Goofy_is_Dog). But, using the suggested strategy, the key set $\{\!|\neg animal(goofy),\ mouse(mini),\ dog(goofy)|\!\}$ will be assembled in the first TTCC-Link-Ext step. Note that there are only two extension steps, and both leave us with a branch ending in the ordinary leaf literal P. However, this branch cannot be closed, as a literal $\neg P$ is not present. Hence, this strategy is incomplete for TTCC-Link.

This small example demonstrates the difficulties of a mixed minimal/non-minimal $\mathcal{T}$-complementary strategy. Hence, we will further stick to the requirements of *minimal* -refuters. Fortunately, any complete background reasoner is *necessarily* also complete wrt. *minimal* $\mathcal{T}$-refuters. This will be shown in Section 4.4 below.

Note 4.2.8 (Relation to Petermann's "Total Theory Connection Calculus"). In [Petermann, 1993a] a theory connection calculus with link condition is defined, which is essentially the same as TTCC-Link. One difference is that the calculus in [Petermann, 1993a] does not insist on *minimal* $\mathcal{T}$-refuters. In our terminology, it is allowed to take key sets from a sufficiently large ("complete") set of $\mathcal{T}$-connections (a *$\mathcal{T}$-connection* is a set of literals for which a $\mathcal{T}$-refuter exists), which need not necessarily be a minimal key set. The completeness is not affected by this relaxation, because these complete sets of $\mathcal{T}$-connections also include all key sets for which a *minimal* $\mathcal{T}$-refuter exists.

4.2.3 Soundness of TTCC

Since TTCC is the strongest of the investigated calculi (cf. Figure 4.1) we prove the soundness of *this* calculus. The soundness of the other calculi follows easily then. Dually, the completeness of TTCC is obtained as a consequence of a more refined model elimination calculus below.

The key to the soundness theorem is to define the semantics of the tableaux constructed along derivations appropriately. We map a branch set $\mathcal{P} = ([p_1], \ldots, [p_m])$ into a formula as follows:

$$Sem([p_1], \ldots, [p_m]) = Sem([p_1]) \vee \cdots \vee Sem([p_m]), \text{ where}$$
$$Sem([L_1, \ldots, L_n]) = L_1 \wedge \cdots \wedge L_n \ .$$

For convenience, we will write $([p_1], \ldots, [p_m])$ instead of $Sem([p_1], \ldots, [p_m])$, and $[L_1, \ldots, L_n]$ instead of $Sem([L_1, \ldots, L_n])$. Our goal will be to show that for satisfiable input clause set M the relation $M \models_{\mathcal{T}} \forall \mathcal{P}_i$ holds for every derivation $\mathcal{P}_1 \vdash \cdots \vdash \mathcal{P}_l$ $(i = 1, \ldots, l)$. For this, the following lemma is needed.

Lemma 4.2.1. *Let M be a clause set and $([p], \mathcal{Q})$ be a branch set.*

1. *If $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ then $M \models_{\mathcal{T}} \forall([p] \circ C, \mathcal{Q})$ for every variant C of a clause $C' \in M$.*

2. *If* $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ *then* $M \models_{\mathcal{T}} \forall(([p], \mathcal{Q})\sigma)$, *for any substitution* σ.
3. *If* $M \models_{\mathcal{T}} \forall([p]\times, \mathcal{Q})$ *then* $M \models_{\mathcal{T}} \forall(\mathcal{Q})$, *where* $[p]\times$ *is a closed branch.*

For a proof see Appendix A.

Theorem 4.2.2 (Soundness of TTCC). *If a TTCC refutation of a clause set* M *exists then* M *is* $\mathcal{T}$*-unsatisfiable.*

Proof. We show the contraposition. Hence let M be a $\mathcal{T}$-satisfiable clause set. Let $\mathcal{D} = (\mathcal{P}_1 \vdash \cdots \vdash \mathcal{P}_l)$ be a TTCC derivation from M with some start clause from M. By induction on l we show that $M \models_{\mathcal{T}} \forall\mathcal{P}_l$. Once this is shown, it is clear that $\mathcal{D}$ cannot be a refutation (i.e. a derivation where every branch set in $\mathcal{P}_l$ is closed), because, if so, we could repeatedly apply Lemma 4.2.1.3 and delete every branch from $\mathcal{P}_l$ and thus conclude that $M \models_{\mathcal{T}} ()$, where $()$ means the empty branch set. This, however, can only hold if M is $\mathcal{T}$-unsatisfiable (but M is given as $\mathcal{T}$-satisfiable).

Hence we turn to the induction proof.

Induction start ($l = 1$): Hence$Sem(\mathcal{P}_1) = L_1 \vee \cdots L_n$ for some start clause $L_1 \vee \cdots L_n \in M$ (or variant thereof). Hence $M \models_{\mathcal{T}} \forall\mathcal{P}_l$ holds trivially.

Induction step ($l - 1 \to l$): Suppose that $l > 1$ and that $M \models_{\mathcal{T}} \forall\mathcal{P}_{l-1}$ holds. We have to show $M \models_{\mathcal{T}} \forall\mathcal{P}_l$.

When obtaining $\mathcal{P}_l$ from $\mathcal{P}_{l-1}$, the TTCC inference rule (a) just fans say the, m extending clauses below the branch to be extended, giving $\mathcal{P}'_l$, and (b) applies the $\mathcal{T}$-refuter σ, giving $\mathcal{P}_l = \mathcal{P}'_l\sigma$ (cf. Figure4.5). In order to see that $M \models_{\mathcal{T}} \forall\mathcal{P}_l$ we only have to apply Lemma 4.2.1.1 m times, such that the fanning in step (a) is reflected, which gives us $M \models_{\mathcal{T}} \forall\mathcal{P}'_l$; finally, Lemma 4.2.1.2 guarantees $M \models_{\mathcal{T}} \forall\mathcal{P}_l$.

4.3 Theory Model Elimination — Semantical Version

4.3.1 Motivation

The calculus to be developed now is motivated by the following critique of TTCC-Link.

Definite Theories. Recall from Section 3.3.4 that for non-theory model elimination reduction steps can be avoided in some cases. For instance, if the input clause set is a Horn set, reduction steps are not required as they are not even applicable for syntactical reasons.

It would be nice to generalize this observation towards theory reasoning. An important class of theories are *definite theories*, i.e. theories that are axiomatized by a set of definite clauses[11]. Definite theories are for example

[11] Recall that a definite clause contains exactly one positive literal.

equality and partial orderings, but also the empty theory. These examples indicate the importance of definite theories, and so the question arises whether there are optimizations for this case.

Unfortunately, it is not clear whether results from the non-theory case carry over to TTCC-Link. For instance, if the (definite) theory is $\mathcal{T} = \{B \wedge C \rightarrow A\}$, and a TTCC-Link derivation (for appropriate input clause set) starts with the inference

$$[\neg A] \vdash_{[\neg A],\{\neg A,B,C\},\{B,\, C\vee\neg D\}} [\neg A \cdot B \cdot \neg D], [\neg A \cdot B \cdot C]\times \ ,$$

with key set $\{\neg A, B, C\}$, then the subtree below $[\neg A \cdot B \cdot \neg D]$ might well contain reduction steps to B. Notice that this is even the case for Horn input clause sets.

Order of Extending Clauses. The problem addressed here has no counterpart in the non-theory version. It concerns the *order* in which the extending clauses are fanned below the extended branch. The example in the previous paragraph will suffice to illustrate the problem; we will only use input clause $B \vee E$ instead of B. Then the following *two* different TTCC-Link-Ext steps using the same key set, same selected branch and same extending clauses exist:

$$\begin{array}{ll} & [\neg A] \vdash_{[\neg A],\{\neg A,B,C\},\{B\vee E,\, C\vee\neg D\}} [\neg A \cdot E], [\neg A \cdot B \cdot \neg D], [\neg A \cdot B \cdot C]\times \\ \text{vs.} & [\neg A] \vdash_{[\neg A],\{\neg A,B,C\},\{C\vee\neg D,\, B\vee E\}} [\neg A \cdot \neg D], [\neg A \cdot C \cdot E], [\neg A \cdot C \cdot B]\times \end{array}$$

Notice that the leaves of the resulting tableau are the same, but the respective ancestor literals are not. Due to this, it is not obvious whether the order of extension is important to obtain a complete calculus. Clearly, with regard to a proof procedure, it is most desirable to have as much as *don't care nondeterminism* ("*any* ordering is complete") in favor of *don't know nondeterminism* ("*there is* an ordering which is complete"). However, on the other side, there might be a superlinear speedup when using the "right" order of extending clauses (cf. Theorem 4.3.1 below).

Regularity. Recall from Section 3.3 the regularity restriction, which says that in a derivation no branch may contain two identical literals. Unfortunately, I did neither succeed to find a proof showing that regularity can be maintained for TTCC-Link, nor could I find a counterexample. So this question must remain open.

In order to address these problems, I suggest a moderate change in the data structures. These modifications let the "definite theories" problem and the "order of extending clauses" disappear, and positive answers to these problems will be trivial consequences of the completeness theorem. Further, the definition of "regularity" can still be used and is complete (because it yields a weaker version of regularity).

The literal trees used below are no longer *clausal* tableau, in the sense that the input clauses can be identified within the tableau as sibling nodes

(cf. Def. 3.1.2). However, it would be overly pedantic to invent a new name. Hence, we will still use the term "TME tableau", or simply "tableau".

4.3.2 Definition of Theory Model Elimination

The calculi discussed so far used *total* theory reasoning. Besides the announced change to the data structures, *partial* theory reasoning will now be taken care of.

Recall from Section 4.1 that partial theory reasoning is very similar to total theory reasoning, except that instead of closing a branch a residue is to be added. Hence we define:

Definition 4.3.1 ($\mathcal{T}$-Residue [Baumgartner and Petermann, 1998]). *Let $\mathcal{T}$ be a universal theory. Let $\mathcal{K}$ be a literal multiset over given signature Σ (read "$\mathcal{K}$" as "key set"). A* ($\mathcal{T}$-)residue for $\mathcal{K}$ *is a pair $\langle \mathcal{R}, \sigma \rangle$ consisting of a possibly empty Σ-clause $\mathcal{R}$ and a substitution σ such that σ is a $\mathcal{T}$-refuter for $\mathcal{K} \cup \overline{\mathcal{R}}$, where $\overline{\mathcal{R}} = \{\!|\overline{Res} \mid Res \in \mathcal{R}|\!\}$ (that is $(\mathcal{K} \cup \overline{\mathcal{R}})\sigma$ is $\mathcal{T}$-complementary).*

If σ is a minimal $\mathcal{T}$-refuter for $\mathcal{K} \cup \overline{\mathcal{R}}$ then $\langle \mathcal{R}, \sigma \rangle$ is called a minimal *residue for $\mathcal{K}$.*

Any "empty" residue $\langle \Box, \sigma \rangle$ is identified with the substitution σ. Note that in this case σ is a $\mathcal{T}$-refuter for $\mathcal{K}$. Similarly, the residue $\langle \mathcal{R}, \varepsilon \rangle$ is identified with $\mathcal{R}$ alone.

The definition of residue is essentially Stickel's definition [Stickel, 1985], except that we use multisets instead of sets, and that our residue includes a substitution component, which Stickel does not need as he considers the ground case. Also, like Stickel, we insist on the minimality property of residues. Stickel points out in [Stickel, 1985] that allowing extraneous literals in the residue would destroy completeness of theory resolution. In Section 4.5 this condition will be relaxed, but only in such a way that refutational completeness is preserved.

It is easy to show[12] that $\langle L_1 \vee \cdots \vee L_n, \sigma \rangle$ $(n > 0)$ is a residue for $\{\!|K_1, \ldots, K_m|\!\}$ iff $\models_{\mathcal{T}} \forall((K_1 \wedge \cdots \wedge K_m \rightarrow L_1 \vee \cdots \vee L_n)\sigma)$. This formulation is possibly more intuitive than the one in the definition, as it explicates the relation between the key set and residue in an affirmative way as an implication.

As an example for residues consider the set $\mathcal{K} = \{\!|\neg(f(x) = f(y)),\ f(a) = f(b)|\!\}$ and the theory of equality. Then it holds:

[12] Cf. Note 4.2.1

$\langle \mathcal{R},$	$\sigma\rangle$	*Residue for $\mathcal{K}$?*
$\langle \Box,$	$\varepsilon\rangle$	No
$\langle \Box,$	$\{x \leftarrow y\}\rangle$	Yes, not minimal
$\langle \Box,$	$\{x \leftarrow a,\ y \leftarrow b\}\rangle$	Yes, minimal
$\langle \neg(f(a) = f(a)),$	$\varepsilon\rangle$	Yes, not minimal
$\langle \neg(f(b) = f(y)),$	$\{x \leftarrow a\}\rangle$	Yes, minimal
$\langle \neg(f(b) = f(y)) \vee \neg(f(b) = f(y)),$	$\{x \leftarrow a\}\rangle$	Yes, not minimal
$\langle \neg(f(c) = f(y)) \vee \neg(f(b) = f(c)),$	$\{x \leftarrow a\}\rangle$	Yes, minimal

Notice by virtue of the constant c in the last line that the definition of residue does not exclude the case to introduce extraneous symbols not present in the key set, even for minimal $\mathcal{T}$-residues.

Now we can express the changes to the TTCC-Link-Ext inference rule in order to obtain theory model elimination (Figure 4.3.2): the idea is to fan both the residue literals and the rest literals of the extending clause *immediately* below the extended branch.

Definition 4.3.2 (Theory Model Elimination, Semantical Version). *The inference rule* theory model elimination extension step (semantical version) *transforms a tableau (in branch set notation) and some clauses into a tableau as follows (cf. Figure 4.7):*

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{(\mathcal{P},\ \mathcal{Q})\sigma} \text{ TME-Sem-Ext}$$

where

$$\mathcal{P} = \begin{cases} [p \cdot K]\times & \text{if } \mathcal{R} \vee R_1 \vee \cdots \vee R_n = \Box, \\ [p \cdot K] \circ (\mathcal{R} \vee R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

where $\langle \mathcal{R}, \sigma\rangle$ is a minimal $\mathcal{T}$-residue for the key set

$$\mathcal{K} = \mathcal{B} \cup \{\!|K|\!\} \cup \{\!|L_1, \ldots, L_n|\!\},$$

for some subset $\mathcal{B} \subseteq p$.

(characterizing property for TME-Ext)

In case $\mathcal{R} = \Box$, a TME-Sem-Ext *step is called* total, *otherwise* partial.

Extending Definition 3.2.4, we will write TME-Sem-Ext *inferences as*

$$([p], \mathcal{Q}) \vdash_{[p],\mathcal{K},\langle\mathcal{R},\sigma\rangle,\{\!|L_1 \vee R_1, \ldots, L_n \vee R_n|\!\}} (\mathcal{Q}', \mathcal{Q})\sigma \ .$$

The calculus theory model elimination, semantical version (TME-Sem), *consists of the inference rule* TME-Sem-Ext*; in the calculus* total theory model

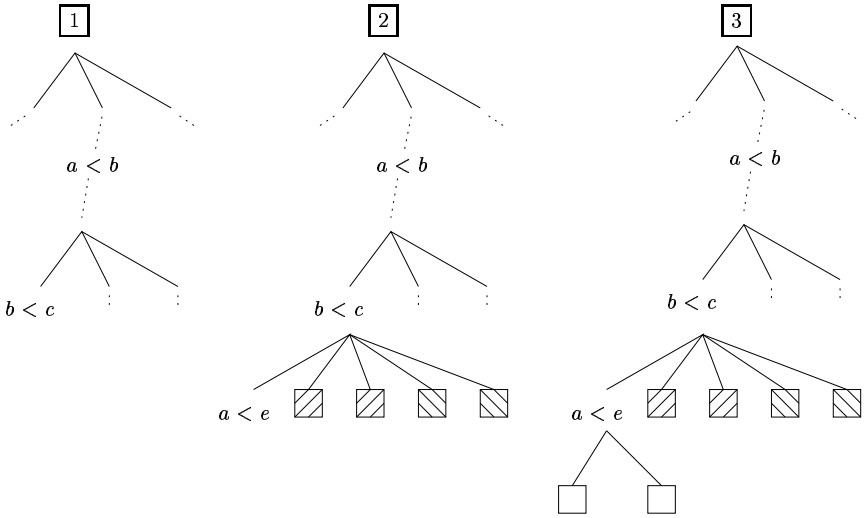


Figure 4.6. Two inference steps of TTCC-Link. The same setup is used as in Figures 4.2 and 4.3.

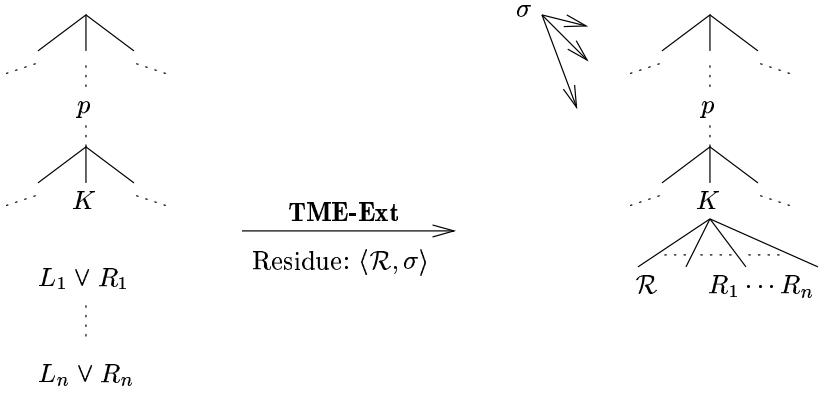


Figure 4.7. A partial theory model elimination extension step.

elimination (TTME-Sem) *only total* TME-Sem-Ext *steps are allowed. The term* partial theory model elimination (PTME-Sem) *is used interchangeably with TME-Sem in order to emphasize that partial steps are allowed.*

In the weak *version of this calculus, we do not insist on minimality of $\mathcal{T}$-residues and only require that in* TME-Sem-Ext *inferences $\langle \mathcal{R}, \sigma \rangle$ is a $\mathcal{T}$-residue for $\mathcal{K}$.*

Notice again that PTME-Sem includes TTME-Sem by the possibility of total steps. Thus, in order to make the PTME-Sem framework meaningful, some ways of restricting the total/partial inferences are necessary. To this end, we will propose the device of "theory inference systems" in Section 4.5.

Notice that unlike in TTCC-Link-Ext inferences in the connection calculus, the extending literals $L_1, \ldots, L_n$ are absent in the subtree below the selected branch. This change requires the case analysis in the TME-Sem-Ext inference rule in order to obtain a proper literal tree.

The partial version "PTCC-Link" of TTCC-Link would be defined in much the same way as was carried out for TME-Sem. This could be done for both, the connection calculus with link condition, and without link condition. We feel, however, that no new insight would be gained from this. Hence it is omitted.

Note 4.3.1 (Minimality). As for TTCC-Link above, the *minimality* requirement for $\mathcal{T}$-residues is also motivated by lifting the link-condition of ME to the theory case. Concerning the partial version, we can relax the minimality requirement in a certain way (Section 4.5) and use the weak version. However, as the discussion in Note 4.4.3 below shows, this is not possible for the total version.

4.3.3 Relation to TTCC-Link

We will now return to the issues addressed in the motivation section above (Section 4.3.1). We start with characterizing the differences to its predecessor TTCC.

Theorem 4.3.1 (TTCC-Link simulates TTME). *For every TTME-Sem derivation $\mathcal{D}$ of a clause set M there is a TTCC-Link refutation $\mathcal{D}'$ of M of the same length which is a stepwise simulation of $\mathcal{D}$. More precisely, if*

$$\mathcal{D} = \mathcal{P}_1 \vdash \mathcal{P}_2 \vdash \cdots \vdash \mathcal{P}_n$$

then a TTCC-Link derivation

$$\mathcal{D}' = \mathcal{P}_1 \vdash \mathcal{P}'_2 \vdash \cdots \vdash \mathcal{P}'_n$$

exists, such that bijective functions exists f_i $(1 \leq i \leq n)$ from the open branches of $\mathcal{P}_i$ onto the open branches of $\mathcal{P}'_i$ (when read as branch sets) such that for every $[b] \in \mathcal{P}_i$ we have

1. $f_i([b]) \supseteq b$, *and*
2. $Leaf([f_i([b])]) = Leaf([b])$.

The converse does not hold. More precisely, there is a theory $\mathcal{T}$ *and a clause set*[13] $M(n)$ *such TTCC-Link admits a refutation of length* $O(n)$*, and the shortest TTME-Sem refutation is of length* $O(2^n)$*.*

Proof. The TTCC-Link refutation starts with the same start clause, and every TTME-Sem-Ext inference (with $\mathcal{T}$-residue $\langle \Box, \sigma \rangle$) can be simulated by a TTCC-Link-Ext inference using the same set of extending clauses, same key set and $\mathcal{T}$-refuter σ, applied to the open branch corresponding to the selected branch in the TTME-Sem refutation. By a "corresponding branch" in the TTCC-Link refutation we mean a branch which contains at inner node positions the same or more literals than a branch in the correspondingly derived TTME-Sem tableau.

More formally, suppose that f_i is already defined appropriately for $\mathcal{P}_i$ and $\mathcal{P}'_i$, and suppose that

$$\underbrace{[p_i], \mathcal{Q}_i}_{=\mathcal{P}_i} \vdash_{[p_i],\mathcal{K}_i,\sigma_i,\{\!|L_1 \vee R_1,\ldots L_n \vee R_n|\!\}} \underbrace{(\mathcal{P},\ \mathcal{Q}_i)\sigma_i}_{=\mathcal{P}_{i+1}} \ ,$$

where

$$\mathcal{P} = \begin{cases} [p_i]\times & \text{if } R_1 \vee \cdots \vee R_n = \Box, \\ [p_i] \circ (R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

The corresponding TTCC-Link-Ext inference then is

$$\underbrace{[p'_i], \mathcal{Q}'_i}_{=\mathcal{P}'_i} \vdash_{p'_i,\mathcal{K}_i,\sigma_i,\{\!|L_1 \vee R_1,\ldots L_n \vee R_n|\!\}}$$
$$\underbrace{([p'_i] \circ R_1,\ [p'_i \cdot L_1] \circ R_2,\ \ldots\ , [p'_i \cdot L_1 \cdots L_{n-1}] \circ R_n,\ [p'_i \cdot L_1 \cdots L_{n-1} \cdot L_n]\times,\ \mathcal{Q}'_i)\sigma_i}_{=\mathcal{P}'_{i+1}}$$

where $[p'_i] = f_i([p_i])$. Obviously, because of $f_i([p_i]) \supseteq p_i$ and $Leaf([f_i([p_i])]) = Leaf([p_i])$ the key set $\mathcal{K}$ of the TTME-Sem-Ext inference can be used in the TTCC-Link-Ext as well. Hence, this inference exists. Finally define

$$f_{i+1}([b]) = \begin{cases} f_i([b]) & \text{for } [b] \in \mathcal{Q}_i \\ [p'_i \cdot L_1 \cdots L_{j-1} \cdot X] & \text{for } [b] = [p_i \cdot X] \in \mathcal{P} \text{ , where} \\ & X \in R_j \text{ with } R_j \neq \Box \ (i \leq j \leq n). \end{cases}$$

Thus, it is only the extending literals which are missing in the TTME-Sem derivation.

[13] By $M(n)$ a function is meant which gives a clause set depending on n.

It is due to this observation that the converse of the theorem does not hold. The presence of an extending literal in a branch can be used in the subtree below it. We construct a clause set in such a way that the usage of such a literal, say L, in TTCC-Link has to be replaced by extending with clause $L \vee R$, where R contains at least two literals. This will cause exponential growth when applied recursively.

The clause set is as follows:

$$R(a) \leftarrow \tag{R_1}$$
$$R(f(a)) \leftarrow \tag{R_2}$$
$$R(f(f(x))) \leftarrow R(f(x)), R(x) \tag{R_3}$$
$$Q(f(x)) \leftarrow P(f(x)), R(f(x)), R(x) \tag{Q}$$
$$P(f^n(a)) \leftarrow \tag{P^n}$$
$$\leftarrow P(f(a)) \tag{G}$$

The notation "$f^n(x)$" means the term for n-fold application of f to x. Of course, the clauses (R_1),(R_2) and (R_3) encode the recursion scheme of the Fibonacci function, which is of exponential complexity wrt. its argument.

As theory we choose:

$$\forall x \, (R(f(x)) \wedge Q(f(x)) \rightarrow P(x)) \ .$$

We start the refutations with start clause (G). The TTME-Sem refutation is completely deterministic, in the sense that for any open branch there exists exactly one set of extending clauses with exactly one key set such that a TTME-Sem-Ext step is applicable.

For TTCC-Link the situation is a bit different. The set of extending clauses and key sets is determined as in the TTME-Sem case. However, there is a choice concerning the order in which the extending clauses are fanned below the extended branch. Figure 4.8 contains a snapshot.

Consider the tableau 1. In one *optimal* policy, which we chose to prove the theorem, extension of branches with leaf $\neg P(f^i(a))$ is carried out with clauses (R_3) and (Q) in that order. This results in a tree shaped as indicated by 2. Notice that all underlined R-literals can be closed by reduction steps, or at the top of the tree by extension with the unit clauses (R_1) and (R_2). To summarize, in this policy there is no fanning below the R-literals, which gives us *constant* breadth (namely 3 or 4) in every level. Since the depth is of order $O(n)$ we conclude that in sum there are $O(n)$ extension steps necessary to find a refutation.

The situation becomes entirely different if a different policy for extension steps is used. This is depicted in tree 3, where extension of branches with leaf $\neg P(f^i(a))$ is carried out with clauses (Q) and (R_3) in that order. Notice that

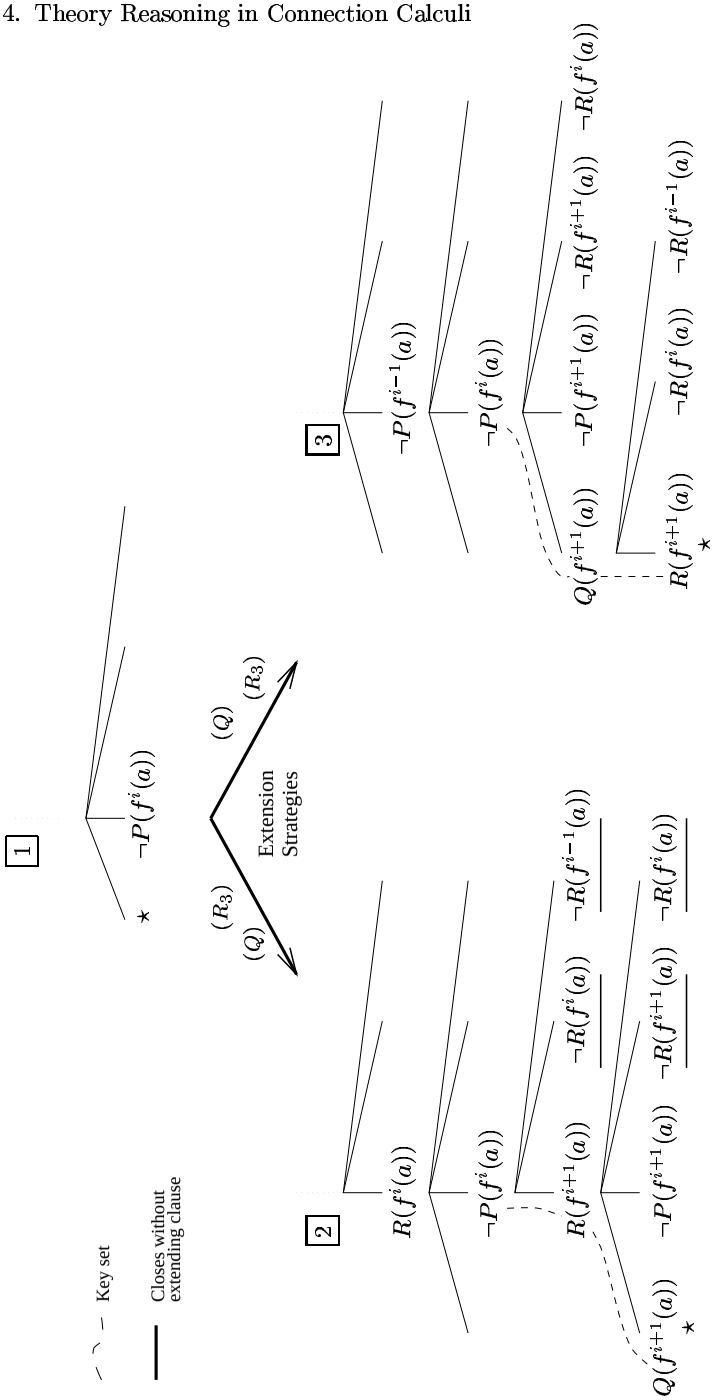


Figure 4.8. A snapshot from the refutation constructed in the proof of Theorem 4.3.1. Starting from tableau 1 two different tableau can be obtained, depending from the order of fanning the extended clauses.

this policy leaves us with negative R literals[14] which cannot be closed using ancestor literals, because no positive R literals are present. Hence, every such negative R literal has to be "solved" by the clauses (R_1), (R_2) and (R_3) which is well-known to be of exponential complexity. In particular, the subproof of the upcoming leaf $\neg R(f^n(a))$ alone requires $O(2^n)$ extension steps.

Since, furthermore, in this policy no extension steps using the Q literals are possible, the refutation behaves as if these Q literals were not present at all. In other words, TTME-Sem will construct precisely the same refutation, except for the absence of these irrelevant Q literals. Thus, a TTME-Sem refutation will also require $O(2^n)$ extension steps.

It is obvious that the proof of this theorem does not depend on whether *total* or *partial* extension steps are considered. The crucial point is the differing ancestor context of TTCC-Link and TTME-Sem. Hence we take it for granted that a respective theorem for the partial variants of the theory connection calculus could be established after proper definition.

Note 4.3.2 (Combining TTCC-Link and TME-Sem). As the previous theorem reveals, there is a tradeoff between TTCC-Link and TME-Sem: TME-Sem needs fewer ancestor literals (recall that wrt. TTCC-Link the extending literals in inference steps are absent) and hence has less local search space than TTCC-Link. On the other side, proofs may become shorter with additional branch literals. This can overweigh the additional local search space. As a drastic example of this, recall the example contrived in the proof of counter direction of Theorem 4.3.1.

As a conclusion I propose a "best of both worlds" approach and to base a proof procedure on a combination of both calculi: from TME-Sem one learns that in inferences the order of extending clauses plays no rôle, giving us a don't-care non-determinism instead of the don't-know nondeterminism of TTCC-Link.

Also from TME-Sem we learn that the extending literals need not be included in extended tableau. On the other hand, from TTCC-Link we learn that they sometimes result in shorter refutations. An approach somewhere in the middle would be to include them, but to restrict the use as substitutes for otherwise equal extending literals in subsequent extension steps. By this, the local search space is not even increased yet saving some extending clauses.

On the other side, however, in experiments with our PROTEIN prover it was my impression that in practice no significant difference arises for most examples.

We are now in a good position to judge whether TME-Sem marks any progress in solving the issues addressed in the beginning of this section.

[14] Here, by an "X literal" we mean a literal of the form $X(f^j(a))$ (or its negation), for some j.

Order of Extending Clauses. Using Theorem 4.3.1 the sample TTCC-Link-Ext inferences on page 91 correspond to the following two TTME-Sem-Ext steps:

$$[\neg A] \vdash_{[\neg A],\{\!|\neg A,B,C|\!\},\{\!|B\vee E,\, C\vee\neg D|\!\}} [\neg A \cdot E], [\neg A \cdot \neg D]$$
$$\text{vs.} \quad [\neg A] \vdash_{[\neg A],\{\!|\neg A,B,C|\!\},\{\!|C\vee\neg D,\, B\vee E|\!\}} [\neg A \cdot \neg D], [\neg A \cdot E] \quad .$$

Notice that the resulting TME tableau are the *same*, because the resulting branch sets, when appropriately ordered (cf. Def. 3.2.1) are the same. It is obvious from the definition of TME-Sem-Ext that this property holds on the general level.

The practical relevance of this observation is that a proof procedure need never backtrack on the ordering of the extending clauses. Again, this is not obvious for TTCC-Link, as switching the order of extending clauses results in branch sets with different ancestor contexts. However, in combination with the simulation result of Theorem 4.3.1, we now learn that also for TTCC-Link the ordering of extending clauses plays no rôle.

Definite Theories. Consider again the example on Page 90 concerning definite theories. The total step given there corresponds to a TTME-Sem-Ext step

$$[\neg A] \vdash_{[\neg A],\{\!|B,C|\!\},\{\!|B,\, C\vee\neg D|\!\}} [\neg A \cdot \neg D] \quad .$$

Notice that now the extending literals are no longer contained as labels of nodes in the resulting TME tableau. We will show that this is not by coincidence. For this, recall that TME-Sem-Ext inferences are based on minimal $\mathcal{T}$-complementary key sets; their structure is characterized by the next proposition:

Proposition 4.3.1. *Let $\mathcal{T}$ be definite theory. Then any minimal $\mathcal{T}$-complementary literal set (or multiset) contains exactly one negative literal.*

Proof: See Appendix A, page 212. The practical relevance of this Proposition is that the search for key sets in TTME-Sem-Ext inferences can be syntactically guided to some degree, even if the input clause set is non-Horn. For the Horn case we can further restrict syntactically:

Proposition 4.3.2. *Let $\mathcal{T}$ be a definite theory, and $\mathcal{D}$ be a PTME-Sem derivation from a Horn clause set M with some negative clause from M as start clause. Then the key set $\mathcal{K}$ in every* TME-Sem-Ext *inference step in $\mathcal{D}$ satisfies the following properties (in terms of Def. 4.3.2):*

1. *The leaf K of the extended branch $[p \cdot K]$ is a negative literal (as well as all other literals in p), and*
2. *$\mathcal{B} = \{\!|\,|\!\}$, i.e. no ancestor literals from $[p]$ are used, and*
3. *$L_1, \ldots, L_n$ are positive literals, and*
4. *$\mathcal{R} = \Box$ or $\mathcal{R}$ consists of negative literals only.*

Thus the search for appropriate key sets is limited to positive literals and one single negative literal, which is given as the leaf; further, as $\mathcal{B} = \{\!\}$, these positive literals are entirely taken from the extending clauses, but not from the selected branch. Notice that this is indeed the result which was desired above.

As a further consequence, "reduction steps" are not possible in case of the empty theory (which is trivially a definite theory). Thus we have a generalization of the corresponding well-known property of non-theory ME.

Proof. Any TME-Sem-Ext step for definite theories applied to a branch with negative leaf must have properties 2, 3 and 4, because otherwise Proposition 4.3.1 would be violated. For the same reason all branch literals $\mathcal{B}$ must be positive. Thus, in order to show that $\mathcal{B} = \{\!\}$ it suffices to show that along $\mathcal{D}$ for every branch set $\mathcal{P}_j$ the following invariant holds: every branch in $\mathcal{P}_j$ is purely negative, i.e. it is of the form $[\neg A_1 \cdots \neg A_n]$. Further, this invariant also gives us that $\mathit{Leaf}([P])$ is negative, as desired.

The invariant is certainly true for the first branch set $\mathcal{P}_1$ corresponding to the the start clause.

Now suppose a TME-Sem-Ext step is applied to some branch $[\neg A_1 \cdots \neg A_n]$. By Proposition 4.3.1 the key set of a theory extension step contains exactly one negative literal, which must be the extended literal $\neg A_n$. The remaining literals L_k are positive and must hence be drawn from input clauses $L_k \vee R_k$. With every literal L_k being the sole positive literal in the clause (we are given that M is Horn) it follows that $[\neg A_1 \cdots \neg A_n]$ is extended only with negative literals. Hence the invariant follows, and thus also the proposition follows.

Regularity. In Section 4.3.1 we motivated the transition from TTCC-Link to TTME-Sem also by the possibility of obtaining a complete calculus with regularity restriction (more precisely, the completeness of TTCC-Link with regularity is still open). Indeed, complete versions of TTME and PTME with regularity exist. For TTME, we will give a direct completeness proof below, which will be used as a base for the completeness proof of a specific version of PTME.

These versions shall be introduced in the next two sections.

4.4 Total Theory Model Elimination — MSR-Version

Following standard results for non-theory model elimination, we would like to carry out derivations employing *most general substitutions*. This is not yet achieved with TME-Sem and thus will be subject of this section. We will consider the *total* version of TME-Sem for this, and the resulting calculus will be called *TTME-MSR* (standing for *T*otal *T*heory *M*odel *E*limination — *M*ost general *S*et of *R*efuters). The necessary changes can essentially be restricted to filtering out non-most general elements from the set of $\mathcal{T}$-refuters for a

given key set. The underlying concept is called a *complete set of $\mathcal{T}$-refuters* and will be treated next. After that, the definition of TTME-MSR will be stated and its answer completeness will be proven. Finally, the approach is illustrated by instantiating the background reasoner with "terminological reasoning".

4.4.1 Complete and Most General Sets of $\mathcal{T}$-Refuters

In non-theory calculi, complementarity of literals is usually established by means of a MGU (most general unifier, cf. Def. 2.4.5). For the sake of efficiency, computing at the most general level is also a desirable goal within theory reasoning. However, in the theory case unifiers need no longer be unique; the concept of an MGU has to be replaced by a more general concept. In the case of purely *equational* theories the concept of "complete set of unifiers" is well-known [Siekmann, 1989; Snyder, 1991].

Since we deal with arbitrary universal theories and not just equational theories we still have to be a bit more general. This concept, which is formulated in a dual way, builds on the concept of $\mathcal{T}$-refuters (Def. 4.2.2) and is called a *complete set of $\mathcal{T}$-refuters*. Further, care must be taken for $\mathcal{T}$-refuters in that they shall not introduce variables which are bound in the context where they are applied. The next definition accounts for this by means of "protected" variables.

Definition 4.4.1 (Complete Set of $\mathcal{T}$-refuters). *A set U of substitutions is a* complete set of $\mathcal{T}$-refuters for M away from a set of variables V *with* $Var(M) \cap V = \emptyset$ *iff*

1. *for all $\sigma \in U$, $M\sigma$ is $\mathcal{T}$-complementary, i.e., σ is a $\mathcal{T}$-refuter for M* (Correctness), *and*
2. *for all $\sigma \in U$, $Dom(\sigma) \subseteq Var(M)$ and $VCod(\sigma) \cap (V \cup Dom(\sigma)) = \{\}$* (Purity), *and*
3. *for all $\mathcal{T}$-refuters σ' for M there is a $\sigma \in U$ such that $\sigma \leq \sigma'$ $[Var(M)]$* (Completeness).

The set U is denoted by $CSR_{\mathcal{T}}(M)[V]$; V is referred to as the set of protected variables.

By a complete $\mathcal{T}$-unification procedure *mean an effective procedure which enumerates a set $CSR_{\mathcal{T}}(M)[V]$ for any finite M and V with $Var(M) \cap V = \emptyset$.*

The necessity for the *correctness* item is obvious. The first conjunct of *purity* restricts the domain of σ to the variables occurring in the problem. This is obviously no real restriction for any $\mathcal{T}$-unification procedure, as assignments of σ outside the scope of the problem M can always be discarded without modifying its effect on M. The second conjunct of purity means that σ will neither introduce variables which are protected by V nor variables which are from the domain of σ. The latter property just characterizes idempotency (see e.g. [Lloyd, 1987]). Thus, we have for any $x \in VCod(\sigma)$ either $x \in Var(M)$

or otherwise $x \notin V$. In the latter case we will say that x is an *extra variable introduced by* σ .

Concerning the *completeness* item, it should be noted that the restriction to $Var(M)$[15] is crucial for the theory-case (again, it is not needed in the non-theory case). For instance, let the theory be given by the clause set $\{\neg p(f(x))\}$. Consider $M = \{\!|p(z)|\!\}$. Clearly, $\sigma = \{z \leftarrow f(a)\}$ is a $\mathcal{T}$-refuter for M. A most general $\mathcal{T}$-refuter is $\sigma' = \{z \leftarrow f(y)\}$. However, it does *not* hold that $\sigma' \leq \sigma$, because we would have to have $\delta = \{y \leftarrow a\}$ in order to have $p(z)\sigma'\delta = p(z)\sigma$. But then, clearly, $y\sigma'\delta \neq y\sigma$ which gives us $\sigma'\delta \neq \sigma$ and, more generally, $\sigma' \not\leq \sigma$. However it holds $\sigma' \leq \sigma\ [Var(M)]$ as expected.

Note 4.4.1 (Need for Protected Variables.). We will argue here for the need to protect variables and prove a calculus not using it as incomplete.

Protected variables are not needed in the case of syntactical unification. The reason is, that the usual unification algorithm will not introduce extra variables. In order to see the importance of protected variables in the theory case consider $M = \{\!|p(x)|\!\}$, and suppose that $p(x)$ occurs in a derivation which has also $q(x, z)$ in its context. Further suppose that a $\mathcal{T}$-refuter $\sigma' = \{x \leftarrow f(a)\}$ is applied in this derivation. This yields $p(f(a))$ and $q(f(a), z)$ wrt. context. For lifting purposes suppose now that $\sigma = \{x \leftarrow f(z)\}$ is a more general $\mathcal{T}$-refuter. However, then we have to find $\delta = \{z \leftarrow a\}$ in order to obtain $\sigma' = \sigma\delta\ [\{x\}]$. Although $p(x)\sigma\delta = p(f(a)) = p(x)\sigma'$, we have, with regard to the context,

$$q(x, z)\sigma\delta = q(f(a), a) \neq q(f(a), z) = q(x, z)\sigma' \quad . \tag{4.2}$$

Thus, we would have had to set the protected variables $V = \{z\}$ and to find a $\mathcal{T}$-refuter from $CSR_{\mathcal{T}}(\{\!|p(x)|\!\})[\{z\}]$.

Note our demand that the variables from $Var(M)$ must — by definition — never be protected. Such a restriction is not present e.g. in [Snyder, 1991]). The advantage of our approach is that the $\mathcal{T}$-refuters are allowed to introduce variables already contained in the problem statement; it relieves the unification algorithm of always unnecessarily inventing new variables. Sometimes this would result in very "unnatural" and unexpected results. For instance, if $p(x)$ and $\neg p(f(y))$ are to be unified syntactically, we would like to have $\sigma = \{x \leftarrow f(y)\}$ as a $\mathcal{T}$-refuter. However, this would not be possible if y is protected. In this case, a suitable $\mathcal{T}$-refuter would be $\sigma' = \{x \leftarrow f(z), y \leftarrow z\}$ (provided that z is not protected). Note that the usual unification algorithm (see e.g. [Lloyd, 1987]) is *not* capable of computing such a substitution and thus cannot be used in applications which need to protect variables in the problem statement.

Substantial problems arise when protection of variables is not used. More specifically, the lifting proof below needs at its very heart equivalences of the form stated in (4.2) and will thus not work with the proposed substitution σ.

[15] Alternatively, the restriction to $Var(\sigma')$ should do as well.

As a consequence an *incomplete* calculus would result. In order to see this, we use the terminology of [Petermann, 1993a]. Finding a refutation then amounts to find a simultaneous $\mathcal{T}$-refuter ϕ for a set $\{\mathcal{K}_1, \ldots, \mathcal{K}_m\}$ of $\mathcal{T}$-connections. We assume that a more general substitution $\sigma \leq \phi\,[Dom(\phi)]$ can be computed incrementally from their m respective sets of most general $\mathcal{T}$-refuters. Now consider the following clause set M and theory $\mathcal{T}$ (also presented as a clause set):

$$M = \{P(x) \vee P(w),\ R(y) \vee Q(z)\}\ ,$$
$$\mathcal{T} = \{\neg P(f(z)) \vee \neg R(f(z)),\ \neg P(f(y)) \vee \neg Q(f(y))\}\ .$$

It is easy to find a (for instance) resolution refutation of $M \cup \mathcal{T}$. Hence M is $\mathcal{T}$-unsatisfiable.

We will now use our notation and attempt to find a TTCC-Link refutation, with the $\mathcal{T}$-refuters determined individually *without protection*. Let $P(x) \vee P(w)$ be the start clause. We have two alternatives for TTCC-Ext-Link inferences:

(a) $[P(x)],\ [P(w)] \vdash_{[P(x)],\{\!|P(x),\ R(y)|\!\},\sigma_a,\{\!|R(y)\vee Q(z)|\!\}} \mathcal{P}_a$ or

(b) $[P(x)],\ [P(w)] \vdash_{[P(x)],\{\!|P(x),\ Q(z)|\!\},\sigma_b,\{\!|R(y)\vee Q(z)|\!\}} \mathcal{P}_b$.

For the key set $\{\!|P(x),\ R(y)|\!\}$ in alternative (a) the substitution $\sigma_a = \{x \leftarrow f(z),\ y \leftarrow f(z)\}$ is a most general and minimal $\mathcal{T}$-refuter, and for alternative (b) the substitution $\sigma_b = \{x \leftarrow f(y),\ z \leftarrow f(y)\}$ is a most general and minimal $\mathcal{T}$-refuter.

Since both cases are totally symmetric, we will consider only alternative (a) now. The resulting branch set then is

$$\mathcal{P}_a = ([P(f(z)) \cdot R(f(z))]\times,\ [P(f(z)) \cdot Q(z)],\ [P(w)])\ .$$

The problem with σ_a is that $VCod(\sigma_a)$ contains the variable z which also occurs in the context outside of the key set. The branch $[P(f(z)) \cdot Q(z)]$ now cannot be closed by using the theory clause $\neg P(f(y)) \vee \neg Q(f(y))$, as would be possible if, say, $\sigma'_a = \{x \leftarrow f(z'),\ y \leftarrow f(z')\}$ were used instead of σ_a[16].

This shows us that a respective refutation of ground instances can not be lifted to a structural identical refutation at the first order level. But, even worse, if the computation of $\mathcal{T}$-refuters is unlucky enough, no refutation will be found at all. For this, let us assume that the branch $[P(f(z)) \cdot Q(z)]$ is selected next (if the other branch $[P(w)]$ were selected, the same problematic

[16] In this example, since $\mathcal{T}$ is given by a clause set, we could use Resolution to compute the $\mathcal{T}$-refuters. Notice that if new variants are taken appropriately, as Resolution would do, then σ'_a would automatically result and the problems would not arise. However, taking such a special setup for computing $\mathcal{T}$-refuters would be unnecessarily restrictive and would not supply a solution if the theory reasoner is given as a "black box".

situation could come up). As extending clause only a variant of $P(x) \vee P(w)$ can be used, say $P(x') \vee P(w')$. If the $\mathcal{T}$-refuter for the respective key set $\{\!|Q(z), P(x')|\!\}$ is selected unluckily enough, namely $\{z \leftarrow f(w'), x' \leftarrow f(w')\}$, this leads to the open branch $[P(f(f(w'))) \cdot Q(f(w')) \cdot P(w')]$. Notice that this branch cannot be closed either and represents a cycle wrt. $[P(x)]$ in the initial branch set, and hence can be repeated infinitely often.

Thus, in sum, no TTCC-Link refutation exists.

In the literature on unification theory one encounters a condition of *minimal* sets of $\mathcal{T}$-refuters ("complete sets of unifiers", as they are called there, see [Siekmann, 1989]). Minimality means to keep only those substitutions which are more general *modulo the given (equational) theory*. However, it may be advisable to leave minimality away, as cases exist where a complete set of *minimal* refuters may not exist (see [Fages and Huet, 1986]).

However, the following weaker property based on "syntactic" most generality safely can be achieved:

Definition 4.4.2 (Most General Set of $\mathcal{T}$-Refuters). *Consider the following property of complete sets of $\mathcal{T}$-refuters:*

$\forall \sigma, \theta \in CSR_{\mathcal{T}}(M)[V]:$ *if* $\sigma \leq \theta\,[Var(M)]$ *then* $\sigma = \theta$

(Most Generality).

Any set $CSR_{\mathcal{T}}(M)[V]$ satisfying this property is also called a most general set of $\mathcal{T}$-refuters for M away from V, *$MSR_{\mathcal{T}}(M)[V]$; its elements are also called* most general $\mathcal{T}$-refuters (for M away from V, *or short* $\mathcal{T}$-MGRs (for M away from V).

Example 4.4.1. In example 4.2.1 above, the substitution $\theta = \{x \leftarrow a,\ y \leftarrow a,\ z \leftarrow a\}$ is also a $\mathcal{E}$-refuter, although, unlike σ, it is not most general.

Note 4.4.2 (Minimality Modulo Equivalence.). The above-mentioned notion of minimality modulo theory-equivalence found in "complete sets of unifiers" is stronger than our syntactical notion. However, unlike in that case, most generality is compatible with the properties of a complete set of $\mathcal{T}$-refuters in the sense that for any $CSR_{\mathcal{T}}(M)[V]$ a $MSR_{\mathcal{T}}(M)[V]$ exists such that the following holds: for every $\theta \in CSR_{\mathcal{T}}(M)[V]$ a $\sigma \in MSR_{\mathcal{T}}(M)[V]$ exists such that $\sigma \leq \theta\,[var(M)]$; $MSR_{\mathcal{T}}(M)[V]$ can be obtained from $CSR_{\mathcal{T}}(M)[V]$ by deleting those substitutions θ which violate the most generality principle. It is easy to show that the resulting set is still a complete set of $\mathcal{T}$-refuters.

Convention 4.4.1 (MSRs and Protected Variables). In practice, the computation of $MSR_{\mathcal{T}}(M)[V]$ can be simplified wrt. extra variables. Recall from the definition, that for any $\sigma \in MSR_{\mathcal{T}}(M)[V]$ and its extra variables $X_\sigma = VCod(\sigma) \setminus Var(M)$ we insist on $X_\sigma \cap V = \emptyset$. This can be achieved by allowing only "new" variables as extra variables[17]; by "new" variables we

[17] E.g. LISP implementations might rely on the `gensym` operation.

mean variables which were not used so far in a derivation, or whatever other context applies. Background reasoners based on Resolution are safe in this respect if allways new variants of the theory clauses are used.

Thus, any algorithm (or enumeration procedure) respecting this principle need not know about a concrete set of protected variables V. This allows us to write $MSR_{\mathcal{T}}(M)$ instead of $MSR_{\mathcal{T}}(M)[V]$ and it will still hold $X_\sigma \cap V = \emptyset$. We will make heavy use of this convention below.

Usually, a $\mathcal{T}$-refuter σ' is applied within a certain variable context W. For lifting purposes it is essential that a corresponding more general $\mathcal{T}$-MGR σ behaves exactly the same on the context W. i.e. that $\sigma\delta = \sigma'\ [W]$ for some δ. The next proposition guarantees the existence of such $\mathcal{T}$-MGRs:

Proposition 4.4.1 (Context Extension of $\mathcal{T}$-MGRs). *Let σ' be a $\mathcal{T}$-refuter for M, and let W be a set of variables. Then substitutions $\sigma \in MSR_{\mathcal{T}}(M)$ with $\sigma \leq \sigma'\ [Var(M)]$ and δ with $\sigma' = \sigma\delta\ [W]$ exist.*

Proof: see Appendix A, page 212.

4.4.2 Definition of TTME-MSR

Now we can rewrite the definition of total theory model elimination to take into account the present notation. An earlier version of this formulation was first suggested in [Baumgartner, 1992a].

Definition 4.4.3 (TTME-MSR). *The inference rule* total theory model elimination $MSR_{\mathcal{T}}$-extension step TTME-MSR-Ext *is the same as* TTME-Sem-Ext *(Definition 4.3.2), except that the characterizing property is changed in the following way:*

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{(\mathcal{P},\ \mathcal{Q})\sigma} \quad \text{TTME-MSR-Ext}$$

where

$$\mathcal{P} = \begin{cases} [p \cdot K]\times & \text{if } R_1 \vee \cdots \vee R_n = \Box, \\ [p \cdot K] \circ (R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

where for some subset $\mathcal{B} \subseteq p$, the key set

$$\mathcal{K} = \mathcal{B} \cup \{\!|K|\!\} \cup \{\!|L_1, \ldots, L_n|\!\},$$

satisfies the following properties:

1. *$MSR_{\mathcal{T}}(\mathcal{K})$ is non-empty, $\sigma \in MSR_{\mathcal{T}}(\mathcal{K})$, and* ($\sigma$ is a solution)
2. *$\mathcal{K}\sigma$ is minimal $\mathcal{T}$-complementary.* (Key set minimality)

(characterizing property for TTME-MSR-Ext)

We will refer to the thus changed inference rule total theory extension step, TTME-MSR-Ext. *Extending Definition 3.2.4, we will write* TTME-MSR-Ext *inferences as* $([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},\sigma,\{\!|L_1 \vee R_1,\ldots,L_n \vee R_n|\!\}} (\mathcal{Q}',\ \mathcal{Q})\sigma$.

A substitution $\sigma \in MSR_{\mathcal{T}}(\mathcal{K})[V]$ *such that* $\mathcal{K}\sigma$ *is minimal* $\mathcal{T}$*-complementary is also called a* minimal most general $\mathcal{T}$-refuter for $\mathcal{K}$ away from V, *or shorter, a* minimal $\mathcal{T}$-MGRs for $\mathcal{K}$ away from V.

The calculus of total theory model elimination, MSR-version (TTME-MSR) *is obtained from TTME-Sem (Definition 4.3.2) by replacing its inference rule* TTME-Sem-Ext *by* TTME-MSR-Ext.

This formulation is stronger than the corresponding one of TTME-Sem, as σ is no longer an *arbitrary* substitution but must be taken from a most general set of refuters. Notice that the conditions 1 and 2 in the characterizing property are compatible with the condition in TTME-Sem-Ext that σ is to be a *minimal residue* for $\mathcal{K}$ (recall from Def. 4.3.1 that we identify the residue $\langle \Box, \sigma \rangle$ with the substitution σ). Hence, TTME-MSR-Ext is a proper restriction of TTME-Sem-Ext.

Note that there are several notions of minimality involved now: the first refers to the "most generality" of MSRs, and the second refers to the size of the key set $\mathcal{K}$. Take, for instance, the key set $\mathcal{K} = \{\!|P(x),\ a = f(y),\ \neg P(f(z))|\!\}$. Then both $\sigma_1 = \{x \leftarrow a,\ y \leftarrow z\}$ and $\sigma_2 = \{x \leftarrow f(z)\}$ are contained in $MSR_{\mathcal{E}}(\mathcal{K})$ and hence are most general. However, attempting to choose σ_2 violates the key set minimality, because deleting $a = f(y)$ from $\{\!|P(f(z)),\ a = f(y),\ \neg P(f(z))|\!\}$ still yields a $\mathcal{T}$-complementary set. Hence, no TTME-MSR-Ext extension step based on $\mathcal{K}$ and σ_2 exists.

Note 4.4.3 (Computing minimal $\mathcal{T}$*-MGRs).* Of course, with $\mathcal{T}$-complementarity being undecidable, *minimal* $\mathcal{T}$-complementarity is undecidable as well. Reason: suppose we had a decision procedure for minimal $\mathcal{T}$-complementarity. Then we could decide the $\mathcal{T}$-complementarity of a given literal set M by deciding the minimal $\mathcal{T}$-complementarity of its (finitely) many subsets, and using the fact that M is $\mathcal{T}$-complementary iff some subset is minimal $\mathcal{T}$-complementary.

Nevertheless, for practical purposes any complete $\mathcal{T}$-unification procedure can be used, because by the completeness property the minimal $\mathcal{T}$-MGRs for a literal set M are included in $MSR_{\mathcal{T}}(M)$. More specifically, it suffices to have an enumeration procedure for (candidates for) key sets, which enumerates at least all key sets admitting a minimal $\mathcal{T}$-MGR, and to interleave it with a complete $\mathcal{T}$-unification procedure. For instance, the linearizing completion technique of Chapter 5 can be used to obtain complete $\mathcal{T}$-unification procedure (See Theorem 5.6.4).

For certain theories one can take advantage of structural information to restrict the candidates for key sets. For instance, for definite theories (such as the empty theory, or equality) any minimal key set consists of precisely one

negative literal (Proposition 4.3.1). A further specialization leads to the case of equational theories:

Example 4.4.2 (Equational Theories). Let Σ be a signature, and let $\mathcal{E}$ be an equational theory over Σ (cf. Section 2.5.1), i.e. a set of equations built from the function symbols and variables of Σ. Since $\mathcal{E}$ is a definite theory, by Proposition 4.3.1 any minimal $\mathcal{T}$-complementary literal multiset contains exactly one negative literal. Further, if we assume that the input clause sets under consideration do not extend $\mathcal{E}$, i.e. there are no positive equations in the input clause set, then obviously any minimal $\mathcal{E}$-complementary literal set is either the singleton $\{\!|\neg(s_1 = t_1)|\!\}$ or consists of two Σ'-literals $\{\!|P(s_1, \ldots, s_n),\ \neg P(t_1, \ldots, t_n)|\!\}$ such that $\mathcal{E} \models s_i = t_i$ (for $i = 1, \ldots, n$). Consequently, the key sets $\mathcal{K}$ for TTME-MSR (Def. 4.4.3) can be restricted to take one of these two forms.

We are interested in certain improvements of this calculus. One is *regularity* ("no duplicate literals along branches") and was introduced above in Section 3.3. Recall from there that regularity is even demanded for closed branches, and since closed branches are never deleted, in a regular derivations every branch *remains* regular, even after further instantiation.

We are going to define an analogous property with respect to the key set minimality:

Definition 4.4.4 (Stability wrt. Minimality for TTME-MSR). *Let*

$$\mathcal{D} = (\mathcal{P}_1 \vdash_{[p_1],\mathcal{K}_1,\sigma_1,E_1} \mathcal{P}_2 \cdots \mathcal{P}_{n-1} \vdash_{[p_{n-1}],\mathcal{K}_{n-1},\sigma_{n-1},E_{n-1}} \mathcal{P}_n)$$

be a TTME-MSR derivation, where E_i denotes the sequence of extending clauses in the i-th extension step. We say that $\mathcal{D}$ is stable wrt. minimality *iff*

for $i = 1, \ldots, n-1$: $\mathcal{K}_i\sigma_i \cdots \sigma_{n-1}$ is minimal $\mathcal{T}$-complementary.

In words, the *stability wrt. minimality* expresses that a once chosen key set $\mathcal{K}_i$, which must be minimal $\mathcal{T}$-unsatisfiable by definition of TTME-MSR-Ext step, *remains* minimal $\mathcal{T}$-unsatisfiable as it is further instantiated along the derivation.

This is a negative example: suppose in a given derivation we have $\mathcal{K}_i = \{\!|z = x,\ \neg(x = y)|\!\}$ and $\sigma_i = \{z \leftarrow y\}$. Using the theory of equality we find that σ_i is a minimal $\mathcal{T}$-MGR for $\mathcal{K}_i$. Now suppose that $\sigma_{i+1} = \{y \leftarrow x\}$. We arrive at $\mathcal{K}_i\sigma_i\sigma_{i+1} = \{\!|x = x,\ \neg(x = x)|\!\}$ which is *not* minimal $\mathcal{T}$-complementary. Hence, this derivation would be not stable wrt. minimality. Note again that stability wrt. minimality is not an issue in syntactic model elimination (cf. Note 4.2.5).

Fortunately, the restriction to derivations which are both stable wrt. minimality and regular is complete. Implementations can take advantage of this fact by remembering all key sets (or selected branches) and setting up constraints expressing their minimality (or expressing their regularity).

Surprisingly, it looks like the proof of the "switching lemma" for theory model elimination (Lemma A.1.11), and hence the "independence of the computation rule" (cf. Section 3.3) cannot be done if stability wrt. minimality cannot be presupposed.

4.4.3 Soundness and Answer Completeness of TTME-MSR

We conclude with the main result of the material developed so far:

Theorem 4.4.1 (Answer Completeness, Soundness of TTME-MSR). *Let $\mathcal{T}$ be a universal theory, c be a computation rule, P be a program and $\leftarrow Q$ be a query (cf. Def. 3.2.5).*

Then, for every correct answer $\{Q\Phi_1, \ldots, Q\Phi_m\}$ for P and $\leftarrow Q$ there is a regular TTME-MSR refutation $\mathcal{D}^c$ of P and $\leftarrow Q$ via c, which is stable wrt. minimality, with computed answer $Answer(\mathcal{D}^c) = \{Q_1, \ldots, Q_l\}$, and such that

$$\{Q_1, \ldots, Q_l\}\delta \subseteq \{Q\Phi_1, \ldots, Q\Phi_m\} \quad \textit{for some substitution } \delta.$$

Conversely, $Answer(\mathcal{D}^c)$ is a correct answer for P.

Proof. We first show the first part, the "completeness" direction. The missing details in the proof can be found in Section A.1.1; here we will glue things together.

The following proof is structured in four parts. This makes it easier to refer to them in later completeness proofs.

Part 1: "Herbrand". Given the correct answer $\{Q\Phi_1, \ldots, Q\Phi_l\}$ we know by definition that $P \models_{\mathcal{T}} \forall(Q\Phi_1 \vee \cdots \vee Q\Phi_l)$. By proposition 2.5.1 we conclude that $P \cup \{\neg\forall(Q\Phi_1 \vee \cdots \vee Q\Phi_l)\}$ is $\mathcal{T}$-unsatisfiable. By transforming this into clausal normal form we get the $\mathcal{T}$-unsatisfiable set of clauses $M' = P \cup \{\leftarrow Q\Phi_1\tau_1, \ldots, \leftarrow Q\Phi_l\tau_l\}$ where each $\tau_i (1 \leq i \leq l)$ substitutes new Skolem constants for the free variables of $Q\Phi_i$.

With the abbreviation $\Phi'_i = \Phi_i\tau_i$, we get a $\mathcal{T}$-unsatisfiable set of clauses

$$M' = P \cup \{\leftarrow Q\Phi'_1, \ldots, \leftarrow Q\Phi'_l\} \ .$$

By the Skolem-Herbrand-Löwenheim theorem for universal theories (Theorem 2.5.2) a $\mathcal{T}$-unsatisfiable ground clause set exists

$$M'' = P' \cup \{\leftarrow Q\Phi'_1, \ldots, \leftarrow Q\Phi'_l\}$$

where P' is a finite set of ground instances of clauses from P. From M'' we select a *minimal* $\mathcal{T}$-unsatisfiable subset

$$M''' = P'' \cup \{\leftarrow Q\Phi'_1, \ldots, \leftarrow Q\Phi'_r\}$$

where $P'' \subseteq P'$, and (without loss of generality) $1 \leq r \leq l$. It must be that $r \neq 0$ because otherwise P'' alone would be $\mathcal{T}$-unsatisfiable, contradicting the requirement that P is a *program*, hence $\mathcal{T}$-satisfiable.

Part 2: Ground Completeness. By ground completeness (Lemma A.1.2) a regular TTME-MSR refutation $\mathcal{D}'_{TTME-MSR}$ of M''' with length, say, n via some computation rule with start clause $\leftarrow Q\Phi'_1$ exists (this choice is arbitrary, but any of these r instances must be used). Since only the empty substitution is employed in every TTME-MSR-Ext step, $\mathcal{D}'_{TTME-MSR}$ is trivially stable wrt. minimality.

Part 3: Query usage. The stated minimality condition ensures that each of clauses $\{\leftarrow Q\Phi'_1, \ldots, \leftarrow Q\Phi'_r\}$ in M''' is used at least once in $\mathcal{D}'_{TTME-MSR}$, either as the start clause ($Q\Phi'_1$) or as an extending clause in an TTME-MSR-Ext step. Let $f(k)$ $(1 \leq k \leq r)$ be the number of usages of $\leftarrow Q\Phi'_k$ (due to regularity we can even conclude that $f(1) = 1$).

Part 4: Lifting. By application of the lifting lemma (Lemma A.1.9) we obtain a regular TTME-MSR-Ext refutation $\mathcal{D}$ of $P \cup \{\leftarrow Q\}$ with start clause $\{\leftarrow Q\}$ which is stable wrt. minimality. The length of $\mathcal{D}$ is also n, and the computed substitution can be written as $\sigma_1 \cdots \sigma_{n-1}$. From property A.9 in the lifting lemma it follows that $\leftarrow Q\sigma_1 \cdots \sigma_{n-1}$ can be instantiated by some substitution, call it δ_n here, to the start clause of $\mathcal{D}'_{TTME-MSR}$, i.e. $\leftarrow Q\sigma_1 \cdots \sigma_{n-1}\delta_n = \leftarrow Q\Phi'_1$. Further, by property A.11 in the lifting lemma, for every extending clause in $\mathcal{D}'_{TTME-MSR}$ there is a corresponding extending clause in $\mathcal{D}$. In particular, for each of the $f(k)$ occurrences of $\leftarrow Q\Phi'_k$ in $\mathcal{D}'_{TTME-MSR}$ there is an occurrence of a variant of $\leftarrow Q$, say $\leftarrow Q\rho_{k,m}$, where $1 \leq m \leq f(k)$, in some set of extending clauses, say $E_{j_{k,m}}$ $(1 \leq j_{k,m} \leq n-1)$ such that

$$\leftarrow Q\rho_{k,m}\sigma_{j_{k,m}} \cdots \sigma_{n-1}\delta_n = \leftarrow Q\Phi'_k \ .$$

In sum, the computed answer of $\mathcal{D}$ thus is by definition 3.2.5

$$\begin{aligned} Answer(\mathcal{D}) = \{ & \underbrace{Q\rho_{1,1}\sigma_{j_{1,1}} \cdots \sigma_{n-1}}_{=:Q_1}, \ldots, Q\rho_{1,f(1)}\sigma_{j_{1,f(1)}} \cdots \sigma_{n-1}, \\ & \vdots \\ & Q\rho_{r,1}\sigma_{j_{r,1}} \cdots \sigma_{n-1}, \ldots, \underbrace{Q\rho_{r,f(r)}\sigma_{j_{r,f(r)}} \cdots \sigma_{n-1}}_{=:Q_l} \}, \end{aligned}$$

and it holds

$$\begin{aligned} Answer(\mathcal{D})\delta_n &= \{Q\Phi'_1, \ldots, Q\Phi'_r\} \\ &= \{Q\Phi_1\tau_1, \ldots, Q\Phi_r\tau_1\} \ . \end{aligned} \tag{4.3}$$

Next apply the "independence of the computation rule" (Proposition A.1.3) to obtain a regular refutation $\mathcal{D}^c$ via the desired computation rule c which is stable wrt. minimality. $\mathcal{D}^c$ is the desired refutation to proof the theorem.

With respect to answers, Proposition A.1.3 gives us a substitution δ' such that $Answer(\mathcal{D}^c)\delta' = Answer(\mathcal{D})$. Thus, Equation 4.3 now rewrites to $Answer(\mathcal{D}^c)\delta'\delta_n = \{Q\Phi_1\tau_1, \ldots, Q\Phi_r\tau_1\}$.

However, in order to prove the theorem we have to find a substitution δ such that $Answer(\mathcal{D}^c)\delta = \{Q\Phi_1, \ldots, Q\Phi_r\}$. In order to define δ recall that τ_k is a Skolemizing substitution and hence can be written as

$$\tau_k = \{x_o \leftarrow a_o \mid o \in O\}$$

for some finite index set O and new constants a_o. In this case we can treat in the refutation $\mathcal{D}^c$ the a_o's as new variables and define the substitutions

$$\tau_k^{-1} = \{a_o \leftarrow x_o \mid x_o \leftarrow a_o \in \tau_k\} \ .$$

Every τ_k introduces new Skolem constants. Hence the domains of the τ_k^{-1}s are pairwise disjoint. But then with defining $\tau^{-1} = \tau_1^{-1} \cdots \tau_r^{-1}$ we get

$$\tau^{-1}|Dom(\tau_k^{-1}) = \tau_k^{-1} \ . \tag{4.4}$$

Finally define

$$\delta = \delta'\delta_n\tau^{-1} \ .$$

This is the desired substitution since

$$\begin{aligned} Answer(\mathcal{D}^c)\delta &= Answer(\mathcal{D}^c)\delta'\delta_n\tau^{-1} \\ &= Answer(\mathcal{D})\delta_n\tau^{-1} \\ &\overset{(4.3)}{=} \{Q\Phi_1\tau_1, \ldots, Q\Phi_r\tau_1\}\tau^{-1} \\ &= \{Q\Phi_1\tau_1\tau^{-1}, \ldots, Q\Phi_r\tau_1\tau^{-1}\} \\ &\overset{(4.4)}{=} \{Q\Phi_1, \ldots, Q\Phi_r\} \ . \end{aligned}$$

This concludes the "completeness" direction. We turn to the second part, "soundness". Hence let $\{Q_1, \ldots, Q_l\}$ be the answer for P and $\leftarrow Q$ computed by a refutation $\mathcal{D}^c$. We have to show that

$$P \models_{\mathcal{T}} \forall(Q_1 \vee \cdots \vee Q_l) \ ,$$

which is by Proposition 2.5.1 equivalent to saying that

$$P \cup \{\exists((\leftarrow Q_1) \wedge \cdots \wedge (\leftarrow Q_l))\} \tag{4.5}$$

is $\mathcal{T}$-unsatisfiable. Let X be the set of variables occurring in $\exists((\leftarrow Q_1) \wedge \cdots \wedge (\leftarrow Q_l))$, and let γ be a Skolemizing substitution for X, i.e. a substitution

mapping each variable in X to a distinct new constant. Now, (4.5) is $\mathcal{T}$-unsatisfiable if and only if

$$M = (P \cup \{\leftarrow Q_1\gamma, \ldots, \leftarrow Q_l\gamma\}) \tag{4.6}$$

is $\mathcal{T}$-unsatisfiable (cf. Theorem 2.4.1, which also holds in our theory case).

Now consider the given refutation $\mathcal{D}^c$. Clearly, every usage of a variant $\leftarrow Q\rho_i$ $(1 \leq i \leq l)$ of the query clause $\leftarrow Q$, which will be further instantiated towards $\leftarrow Q\rho_i\sigma_j \cdots \sigma_{n-1}$ $(= (\leftarrow Q_i))$, can be replaced by the usage of $Q_i\gamma$. This results not necessarily in a TTME-MSR refutation (because the employed substitution might no longer be most general), but since instantiation with γ does not affect the $\mathcal{T}$-complementarity of the employed key sets (Proposition 4.2.1.1), the resulting refutation, say $\mathcal{D}'$, is still a TTME-Sem refutation (cf. Def. 4.3.2) of M. Now, by Theorem 4.3.1, $\mathcal{D}'$ can be simulated by a TTCC-Link refutation, which trivially is also a TTCC refutation (Def. 4.2.3). Next, by soundness of TTCC (Theorem 4.2.2) we conclude that M is $\mathcal{T}$-unsatisfiable. But now we can go from (4.6) the equivalences upwards and conclude that $\{Q_1, \ldots, Q_l\}$ is a correct answer for P and $\leftarrow Q$.

4.4.4 Related Work

In Section 4.1 several calculi were mentioned which were extended towards theory reasoning. The closest relatives to TTME-MSR are clearly analytic clausal calculi, and in particular Petermann's theory connection methods [Petermann, 1992; Petermann, 1993a].

Theory reasoning was described for a connection method in Bibel [Bibel, 1982a] (without proof). The first such calculi which proceeds in a goal-oriented way (i.e., employs a link-condition) was an earlier version of TTME-MSR [Baumgartner, 1992a]. The TTME-MSR calculus, however without stability wrt. minimality, and without a lifting theorem was first described in [Baumgartner, 1993; Baumgartner, 1994].

In [Petermann, 1993a] the completeness for a goal-oriented theory connection calculus very similar to TTCC-Link is stated. The differences between the link-condition there and the one used in TTME-MSR were discussed in Note 4.2.8.

As other important differences we have that our calculus is more restrictive due to the regularity and the stability wrt. minimality restrictions. Further, TTME-MSR needs fewer ancestor literals than the connection calculus in [Petermann, 1993a] (recall that the "extending literals" are not included in the tableau after extension steps). A detailed discussion of the consequences of this can be found in Section 4.3.3, and in particular in Note 4.3.2.

Finally, we proved an *answer* completeness theorem, which includes the usual notion of refutational completeness as a special case.

4.4.5 A Sample Application: Terminological Reasoning

Terminological reasoning was mentioned above in Section 4.1.2 as an instance of total theory reasoning. We will show here how TTME-MSR can be used to reason about clauses containing both ordinary and A-Box literals; the latter will be interpreted over a background theory given by a T-Box. After some preliminary remarks we will go to a more formal treatment.

As a prerequisite we will assume a language for the T-Box which contains the usual constructs such as "atomic concept", "concept conjunction", etc., but allows also "concept definition, $\doteq$". See [Hollunder, 1990] for a brief introduction.

For the assertional formalism ("A-Box"), we allow *object descriptions* of the form $a:A$, where a is an object (i.e. a symbol taken from some alphabet), and A is a concept name. For instance, $\mathsf{Peter}:\mathsf{Male}$ is an object description. Furthermore, the A-Box may contain *relation descriptions* of the form $(a,b):R$, where a and b are objects and R is a role (e.g. $(\mathsf{Peter},\mathsf{Beate}):\mathsf{Married}$). Obviously, both object and relation descriptions can be formulated in first-order logic. For this, simply take the objects as constants symbols, let ":" be a two place predicate symbol and let "(.,.)" by a 2-ary function symbol. With this view we talk of A-Box literals, and we let an A-Box consists of a finite set A-Box literals.

Usually, terminological reasoning system provide (among others) a service to *decide* the consistency of an A-Box wrt. a T-Box (see e.g. [Hollunder, 1990; Schmidt-Schauß and Smolka, 1991]), i.e. whether there is a model of both the A-Box and the T-Box. Consistency is the most elementary problem, in the sense that many other interesting problems, such as the subsumption, instance, realization and retrieval problem are instances of it (see [Hollunder, 1990]).[18]

The algorithm in [Hollunder, 1990] presupposes that an A-Box literal is always *positive*, i.e. unnegated. In order to bring in negative information for A-Box literals one defines in the T-Box a new concept which expresses the complement of a given concept. For instance, in order to express that Peter is not a Male state the A-Box literal $\mathsf{Peter}:\overline{\mathsf{Male}}$ where $\overline{\mathsf{Male}}$ is a new concept which is defined in the T-Box by adding $\overline{\mathsf{Male}} \doteq \neg\mathsf{Male}$ to it.

As mentioned above, we are concerned with a more general proof task, in that instead of an A-Box we allow arbitrary clauses which contain both ordinary and A-Box literals. This clearly increases expressive power. Such an

[18] As shown in [Paramasivam and Plaisted, 1995] even good ordinary theorem provers can compete well with dedicated implementations for common services such as "concept subsumption" and "classification". For this, the knowledge base (i.e. T-Box + A-Box) is translated into predicate logic, as usual. In order to achieve good performance some preprocessing on the translation of the knowledge base (e.g. a relevance analysis for the clauses) is needed. Further, since the common services are quite often decidable, a theorem prover being capable of finding finite models (if they exist) is most suitable.

approach was first suggested by [Stickel, 1985] for theory resolution in conjunction with KL-1. In [Hanschke and Hinkelmann, 1992] a forward-chaining rule system was combined with a terminological logic for a mechanical engineering application.

Now we are going to define the combination within our framework more formally.

Let $\Sigma_{A-Box} = \langle \mathcal{O} \cup \{(.,.)\}, \{:\}, \emptyset \rangle$ be a signature. "$\mathcal{O}$" is the set of objects, ":" is the A-Box literal symbol, and "(.,.)" is the constructor symbol used in relations. An A-Box consists of a finite set of positive Σ_{A-Box}-literals.

There is no need for explicitly fixing a language for the T-Box here. Instead we only define that an A-Box $\mathcal{A}$ is *consistent* wrt. a T-Box $\mathcal{T}$ iff there is a Σ_{A-Box}-interpretation $\mathcal{I}$ which is both a model for $\mathcal{A}$ and $\mathcal{T}$. If such a Σ_{A-Box}-interpretation does not exist then $\mathcal{A}$ is said to be *inconsistent* wrt. $\mathcal{T}$. Again, a basic algorithm such as the one in [Hollunder, 1990] can be used to decide any $\mathcal{T}$-consistency problem.

Now let $\Sigma_F = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature whose elements are disjoint from Σ_{A-Box}. The signature Σ consists of the union of Σ_F and Σ_{A-Box}. We consider Σ-clauses whose literals are either Σ_F-literals or positive Σ_{A-Box}-literals.

Theorem 4.4.2 (Completeness with T-Box Reasoning). *TTME-MSR instantiated with terminological reasoning is sound and complete, where every key set $\mathcal{K}$ used in a* TTME-MSR-Ext *extension step is either a set of A-Box literals $\{A_1, \ldots, A_n\}$, and in this case*

$$MSR_{\mathcal{T}}(\{\!|A_1, \ldots, A_n|\!\}) = \begin{cases} \{\varepsilon\} & \text{if } \{\!|A_1, \ldots, A_n|\!\} \text{ is inconsistent wrt. } \mathcal{T} \\ \emptyset & \text{otherwise.} \end{cases}$$

or else $\mathcal{K}$ consists of two Σ_F literals $\{\!|P(s_1, \ldots, s_n),\ \neg P(t_1, \ldots, t_n)|\!\}$ and

$$MSR_{\mathcal{T}}(\{\!|P(s_1, \ldots, s_n), \neg P(t_1, \ldots, t_n)|\!\}) = \begin{cases} \{\sigma\} & \text{if there exists a MGU } \sigma \text{ for} \\ & P(s_1, \ldots, s_n) \text{ and } P(t_1, \ldots, t_n), \\ \emptyset & \text{otherwise.} \end{cases}$$

Proof. (Sketch, completeness direction). We instantiate the general TTME-MSR completeness Theorem 4.5.3. The only nontrivial part is to show that any minimal $\mathcal{T}$-refuter for a given key set set $\mathcal{K}$ belongs to one of two two cases as suggested. This follows from the fact that the signatures Σ_{A-Box} and Σ_F are disjoint, which allows us to consider Σ-interpretations as conservative extensions of $\mathcal{T}$-interpretations over Σ_{A-Box}.

More precisely: suppose, on the contrary, there is a key set $\mathcal{K}$ and a minimal $\mathcal{T}$-MGR σ for $\mathcal{K}$, and $\mathcal{K}$ is of the form $\mathcal{K} = \mathcal{K}_{A-Box} \cup \mathcal{K}_F$ with $\mathcal{K}_{A-Box} \neq \emptyset$ being a set of Σ_{A-Box}-literals and $\mathcal{K}_F \neq \emptyset$ being a set of Σ_F-literals.

From the minimal $\mathcal{T}$-complementarity of $\mathcal{K}\sigma$ we know that $\mathcal{I}_F \models \mathcal{K}_F\sigma$ and that $\mathcal{I}_{A-Box} \models \mathcal{K}_{A-Box}\sigma$ for some $\mathcal{T}$-Σ_F-interpretation $\mathcal{I}_F$ and $\mathcal{T}$-Σ_{A-Box}-interpretation $\mathcal{I}_{A-Box}$. Due to disjointness of signatures, we can define a $\mathcal{T}$-Σ-Interpretation $\mathcal{I}$ as $\mathcal{I}(L) = \mathcal{I}_F(L)$ if L is a Σ_F literal and $\mathcal{I}(L) = \mathcal{I}_{A-Box}(L)$ if L is an Σ_{A-Box} literal, and it will still hold $\mathcal{I} \models \mathcal{K}_F\sigma$ and $\mathcal{I} \models \mathcal{K}_{A-Box}\sigma$. That is, $\mathcal{K}\sigma$ is $\mathcal{T}$-satisfiable. Contradiction.

Thus, either $\mathcal{K}_{A-Box} = \emptyset$ or $\mathcal{K}_F = \emptyset$. In the case that $\mathcal{K}_F \neq \emptyset$ it remains to be proved that $\mathcal{K}_F$ is of the claimed form, i.e. that $\mathcal{K}_F$ is a pair of literals which can be made complementary by σ. Suppose, on the contrary, that $\mathcal{K}_F\sigma$ does not contain literals L and $\overline{L}$. Then, however, we can find a Σ_F-model for $\mathcal{K}_F\sigma$ which can be extended by any $\mathcal{T}$-Σ_{A-Box}-interpretation to a $\mathcal{T}$-Σ-model for $\mathcal{K}_F\sigma$. Contradiction. Thus, $\mathcal{K}\sigma$ contains two complementary literals. Clearly, any excess literals can be deleted which gives us that $\mathcal{K}\sigma$ contains exactly two literals.

Recall that our approach presupposes a *ground* A-Box. If one would like to deal with variables inside A-Box literals, several ways of extensions are conceivable: first, by an appropriate computation rule one would have to guarantee that only instantiated A-Box literals are passed to the decision procedure[19]. A second, more general approach would be to use more sophisticated decision algorithms to compute non-trivial $\mathcal{T}$-MGRs for A-Boxes with existentially quantified variables. This was basically the approach taken in [Baumgartner and Stolzenburg, 1995] within a more general constraint-reasoning approach.

4.5 Partial Theory Model Elimination — Inference Systems Version

The calculus which will be developed now is a refined *partial* version of theory model elimination (TME-Sem, Section 4.3), much like TTME-MSR is a refined *total* version of TME-Sem[20]. We recall that for TTME-MSR we introduced the concept of a most general set of $\mathcal{T}$-refuters (Definition 4.4.2). It is a semantical concept, and it serves as a specification for background reasoners for total theory reasoning. It seems natural to elaborate a respective framework for partial theory reasoning. That is, find criteria for the design of background reasoners such that their combination with model elimination yields a complete calculus.

[19] As with the "floundering" problem in logic programming, one would seek for syntactical conditions which guarantee that always at least one appropriately instantiated A-Box can be found.

[20] For a general discussion of partial vs. total theory reasoning and the particular interest in partial reasoning the reader is referred back to Section 4.1.1, page 70.

A first step in this direction is the description of partial extension steps by means of $\mathcal{T}$-residues (Definition 4.3.1). The next step then would be to develop completeness criteria for key set – $\mathcal{T}$-residue pairs.

The only work[21] in this direction comes from Stickel [Stickel, 1985]. Stickel introduced a *key selection criterion* within the ground case of theory resolution. In our terminology, it says that for any minimal $\mathcal{T}$-unsatisfiable (ground) literal set M there must exist a key set $\mathcal{K} \subseteq M$ and a minimal (ground) $\mathcal{T}$-residue $\mathcal{R}$ such that the *clause* set $(M \setminus \mathcal{K}) \cup \{\!|\mathcal{R}|\!\}$ is minimal $\mathcal{T}$-complementary. Further, it is required that $\mathcal{K}$ consists of at least two literals if $|M| \geq 2$ (or one literal if $|M| = 1$). This latter condition then serves as the base for the completeness proof. The idea of Stickel's proof (Theorem 9 in [Stickel, 1985]) is to split a given clause set into sets of $\mathcal{T}$-unsatisfiable unit clauses, each of them being $\mathcal{T}$-unsatisfiable. Each of these sets contains a *minimal* $\mathcal{T}$-complementary subset M, which is made subject to the key selection criterion. In particular, if $M > 2$ then also $|\mathcal{K}| \geq 2$ and thus $(M \setminus \mathcal{K}) \cup \{\!|\mathcal{R}|\!\}$ contains *strictly less* clauses M. Hence, the induction hypothesis can be applied. If the induction start with $|M| = 1$ or $|M| = 2$ is reached eventually, then it must hold that $\mathcal{R} = \Box$, because otherwise $\mathcal{R}$ would not be a *minimal* $\mathcal{T}$-residue (because $\mathcal{K} = M$ alone is minimal $\mathcal{T}$-complementary).

The key selection criterion is applicable to, for instance, the theory $\mathcal{SO}$ of strict orderings: any $\mathcal{SO}$-unsatisfiable literal set M contains either $\neg(t < t)$ or two literals $s < t$ and $t < u$. In the latter case, these literals can be *replaced* by $s < u$ without affecting the $\mathcal{T}$-unsatisfiability of M.

Unfortunately, the criterion is not applicable for covering the important paramodulation inference rule. Take, for instance, $M = \{\!|P(a,a), a = b, \neg P(b,b)|\!\}$. Since *two* paramodulation steps with $a = b$ are necessary to refute M, we cannot remove $a = b$ after one step from M. For instance, paramodulating $P(a,a)$ to $P(b,a)$, thus taking $\mathcal{K} = \{\!|P(a,a),\ a = b|\!\}$ and $\mathcal{R} = P(b,a)$, yields $(M \setminus \mathcal{K}) \cup \{\!|\mathcal{R}|\!\} = \{\!|P(b,a),\ \neg P(b,b)|\!\}$, which is not $\mathcal{E}$-unsatisfiable.

One way to improve this situation is to give up the just considered "local" method. By this, I mean that instead of looking at pairs key set – $\mathcal{T}$-residue one considers *background derivations*: recall from the introductory section on theory reasoning, that partial theory reasoning can be thought of as breaking "big" total steps into smaller, more manageable pieces. This idea can serve as a base for a completeness criterion which also includes the paramodulation case. For PTME-Sem such a criterion would roughly consist of the following two components (it will be made precise in Def. 4.5.6):

Ground completeness: For every minimal $\mathcal{T}$-unsatisfiable ground literal set $\mathcal{K}$, and any literal $L_1 \in \mathcal{K}$ there are sequences $L_1, L_2, \ldots, L_n, \Box$ ($i = 1, \ldots, n$, the L_is being ground unit clauses[22]), and $M_1, M_2, \ldots, M_n$, with $M_i \subseteq \mathcal{K}$, usually written more suggestively as

[21] Known to me.

[22] We use *unit* clauses in order to be compatible with linearizing completion.

$$L_1 \stackrel{M_1}{\Longrightarrow} L_2 \stackrel{M_2}{\Longrightarrow} \cdots L_n \stackrel{M_n}{\Longrightarrow} (\Box = L_{n+1}) \ ,$$

such that $\langle L_{i+1}, \varepsilon \rangle$ is a $\mathcal{T}$-residue for the key set $\mathcal{K}_i = (\{\!|L_i|\!\} \cup M_i)$.

Lifting: For any $\mathcal{T}$-residue $\langle L_{i+1}, \varepsilon \rangle$ for $\mathcal{K}_i$ on the ground level, there is a "first-order" $\mathcal{T}$-residue $\langle L'_{i+1}, \sigma_i \rangle$ for $\mathcal{K}'_i$, where $\mathcal{K}_i$ is an instance of $\mathcal{K}'_i$, such that L_{i+1} is an instance of $L'_{i+1}\sigma_i$. Further, some variable conditions have to be obeyed.

This splitting in two conditions is motivated by the usual "ground-proof plus lifting" proof technique. The *ground completeness* allows us to break a total extension step extending at leaf L_1 and with key set $\mathcal{K}$ at the ground level into a sequence of partial extension steps with residues $L_2, \ldots, L_n$, terminated by a total extension step with residue $L_{n+1} = \Box$. In particular, the requirements that $\mathcal{K}_i = (\{\!|L_i|\!\} \cup M_i)$ and $M_i \subseteq \mathcal{K}$ will guarantee that the sequence of partial extension steps uses not more resources than the total extension step. The *lifting* property would then allow us to lift the whole PTME refutation to the first order level.

Thus it is basically this setup we are interested in, and which will be pursued in what follows. However, it will be more concrete, and a specific framework for computing $\mathcal{T}$-residues will be proposed. This framework is called *theory inference systems*. They provide a *syntactic* characterization of how to map key sets into residues. This framework is attractive for the following reasons:

- Lifting comes for free. That is, the lifting property is a property of the framework. So it remains to prove ground completeness "only" for a given theory.
- Since the completeness criterion is "global", it covers a broader spectrum than a "local" one. For instance, the "paramodulation" inference rule is easily covered (see below for details).
- The proof of the ground completeness can be automated in many cases by the technique of *linearizing completion* (Chapter 5). Even stronger, by this method a complete theory inference system can be computed automatically from the theories' axioms.
- Due to the syntactical nature, the language of theory inference systems can be implemented once and for all in a theorem prover. Instantiation with a specific theory does not need any implementation work.
- The TME-Sem-Ext inference step is defined purely "semantically"; it is the characterizing property in Definition 4.3.2 which only guarantees the soundness of the calculus, but does not give us any restrictions on the possible inferences. In fact, since theory reasoning with total extension steps alone is complete, partial steps are in a strict sense not necessary at all.

  Thus, in order to make the total/partial theory reasoning framework more meaningful, we need some way to restrict the total/partial inferences. This

can be achieved in a completely declarative way using theory inference systems.

The plan of this section is as follows: next, theory inference systems are introduced. Building on this, we will define a specialized version of PTME-Sem. Then, the completeness proof follows. Finally, the applicability of the approach is demonstrated using some well-known instances of theory reasoning.

4.5.1 Theory Inference Systems

In the following, let $\Sigma = \langle \mathcal{F}, \mathcal{P}, \mathcal{X} \rangle$ be a signature, and let $\Sigma^{\mathsf{F}} = \langle \mathcal{F}, \mathcal{P} \cup \{\mathsf{F}\}, \mathcal{X} \rangle$ be its extension by the new predicate symbol F, meaning "*false*".

Definition 4.5.1 (Theory Inference System). *A* theory inference rule *over a given signature Σ^{F} (the short forms* inference rule *or simply* rule *will be used as well) is a pair $P \to C$, where P is a nonempty multiset of Σ-literals and C is either a nonempty set of Σ-literals or the singleton $\{\!|\mathsf{F}|\!\}$. P is called the* premise *and C is called the* conclusion *of the inference rule.*

Some notational conveniences: in inference rules we will often write $L_1, \ldots, L_n \to L_{n+1}, \ldots, L_{n+k}$ instead of $\{\!|L_1, \ldots, L_n|\!\} \to \{\!|L_{n+1}, \ldots, L_{n+k}|\!\}$, and $Q, P \to C$ instead of $Q \cup P \to C$, and $L, P \to C$ instead of $\{\!|L|\!\} \cup P \to C$, etc.

An instance *of an inference rule is obtained by application of a substitution to both the premise and the conclusion.* Ground *inference rules do not contain variables.*

A (theory) inference system (over Σ^{F}) *consists of a possible infinite set of inference rules over Σ^{F}. The letter "$\mathcal{I}$" will be used to denote inference systems.*

A ground inference system *consists of ground inference rules only. If $\mathcal{I}$ is an inference system then $\mathcal{I}^g$ is defined as the inference system consisting of all ground instances of all rules from $\mathcal{I}$.*

The operational meaning of inference is roughly the same as that of hyper resolution: from given literals $\{\!|K_1, \ldots, K_n|\!\}$ derive the literals $\{\!|L_{n+1}, \ldots, L_{n+k}|\!\}\sigma$, where σ is a simultaneous unifier for the K_i's and L_i's. More precisely:

Definition 4.5.2 (Minimal First-Order Theory Inference).
We say that a clause $\mathcal{R}\sigma$ is inferred *from a literal multiset $\mathcal{K}$ by means of a* minimal first-order theory inference step with inference rule $P \to C$ and substitution σ *(the short forms* inference step *or simply* inference *will be used as well), written as*

$$\mathcal{K} \Rightarrow_{P \to C,\, \sigma} \mathcal{R}\sigma$$

iff

1. *σ is a multiset-MGU $\mathcal{K}'\sigma = P\sigma$ for some amplification*[23] *$\mathcal{K}' \sqsupseteq \mathcal{K}$, and*
2. *$\mathcal{K}\sigma$ contains no duplicates* (Premise Minimality)*, and*
3. $\mathcal{R} = \begin{cases} \square & \text{if } C = \mathsf{F} \\ C & \text{else.} \end{cases}$

Occasionally, we will relax a bit and require that σ is only a multiset unifier, but not necessarily a multiset-MGU. This usage then, however, will always be announced explicitly.

By overloading of notation, $\mathcal{K}$ is also called the premise*, $\mathcal{R}\sigma$ is called the* conclusion *of the inference and $P \to C$ is called the* used *inference rule. We will often abbreviate $\mathcal{K} \Rightarrow_{P\to C,\,\sigma} \mathcal{R}\sigma$ to $\mathcal{K} \Rightarrow_{P\to C} \mathcal{R}\sigma$ or even $\mathcal{K} \Rightarrow \mathcal{R}\sigma$ if context allows.*

Notice that if $\mathcal{K}$ and $P \to C$ are ground then $\sigma = \varepsilon$ and we can safely write inferences as $\mathcal{K} \Rightarrow_{P\to C} \mathcal{R}$ without omitting any information. Such inferences are also called ground inferences*.*

In inferences we amplify $\mathcal{K}$ towards $\mathcal{K}'$ because from $\mathcal{K} = \{\!|A|\!\}$ and inference rule $A, A \to B$ we would like to infer B, which would not be possible if using multiset unification applied to $\mathcal{K}$ instead of $\mathcal{K}'$. Instead of allowing such redundancies in inference rules, an alternative would be to require that for each inference rule also its "factors" (in the sense of resolution) are present.

The *premise minimality* in theory inferences is motivated by their intended application within partial theory ME, where we prefer to have the amplification as part of the inference, but not as being carried out beforehand.

Example 4.5.1. Consider the theory of strict orderings in Example 4.1.1 again. This is a corresponding theory inference system:

$$\begin{aligned}
\mathcal{I}_{\mathcal{SO}}: \quad & x < y,\ y < z \to x < z && \text{(Trans-1)}\\
& \neg(x < z),\ x < y \to \neg(y < z) && \text{(Trans-2)}\\
& x < x \to \mathsf{F} && \text{(Irref)}\\
& \neg(x < y),\ x < y \to \mathsf{F} && \text{(Syn)}
\end{aligned}$$

The first three rules stem from the theory immediately, while the additional (Syn)-rule is used to treat syntactical inconsistencies. Note that only *two* rules are given to treat transitivity, although three of such contrapositives exist. Further, all residues are unit clauses or the empty clause. This prevents us in particular from deriving "useless" residues such as[24] $\neg(b < c) \vee \neg(d < a)$ from the key set $\{\!|a < b, c < d|\!\}$. We remark that this key set would be suspicious due to its "unconnected" constituents. Notice that such key sets cannot occur in inferences based on $\mathcal{I}_{\mathcal{SO}}$.

Here are some inferences (except the last one):

[23] "Amplification" is defined in Section 2.1.

[24] This example is taken from [Stickel, 1985].

$\mathcal{K} \Rightarrow_{R,\sigma} \mathcal{R}\sigma$	$\mathcal{K}'$	σ
$f(u) < v \Rightarrow_{(\text{Irref}),\sigma} \square$	$f(u) < v$	$x \leftarrow f(u),\ v \leftarrow f(u)$
$\neg(g(u) < f(u)),\ g(a) < h(a)$ $\Rightarrow_{(\text{Trans}-2),\sigma} \neg(h(a) < f(a))$	$\neg(g(u) < f(u)),$ $g(a) < h(a)$	$x \leftarrow g(a),\ y \leftarrow h(a),$ $z \leftarrow f(a),\ u \leftarrow a$
$f(u,v) < f(v,u)$ $\Rightarrow_{(\text{Trans}-1),\sigma} f(v,v) < f(v,v)$	$f(u,v) < f(v,u),$ $f(u,v) < f(v,u)$	$x,y,z \leftarrow f(v,v),$ $u \leftarrow v$
$f(u,v) < f(v,u),\ f(u,v) < f(v,u)$ $\Rightarrow_{(\text{Trans}-1),\sigma} f(v,v) < f(v,v)$	$f(u,v) < f(v,u),$ $f(u,v) < f(v,u)$	$x,y,z \leftarrow f(v,v),$ $u \leftarrow v$

The last line contains no inference because the premise minimality is violated. Also, the expression $a < a,\ a < a \Rightarrow_{(\text{Irref})} \square$ is *not* an inference, because no amplification of $\{\!|a < a,\ a < a|\!\}$ unifies with $\{\!|x < x|\!\}$. This example shows that using multiset-unification where one multiset is amplified is different from using *sets* (because $\{a < a, a < a\}$ is the same as $\{a < a\}$).

It should be noted that $\mathcal{I}_{\mathcal{SO}}$ is indeed a "complete" set of theory inference rules (in a sense which will be made precise in Def. 4.5.6 below). This result can be proven completely automatically by "linearizing completion" (Chapter 5).

Theory inferences should at least be sound wrt. a given theory. Hence we define:

Definition 4.5.3 (Soundness of Inference Systems). *A $\mathcal{T}$-Σ^{F}-interpretation is a $\mathcal{T}$-Σ-interpretation in which* F *is interpreted by false. Henceforth, we will consider only $\mathcal{T}$-Σ^{F}-interpretations.*

A theory inference system is called sound *wrt. theory $\mathcal{T}$ iff for every of its inference rules $(P \to C) \in \mathcal{T}$ it holds $\models_{\mathcal{T}} \forall(P \to C)$, where P (resp. C) is read as the conjunction (resp. disjunction) of its literals.*

That is, in a sound theory inference system, the universal closure of each of its rules $P \to C$ has to be $\mathcal{T}$-valid. Soundness is such a fundamental property that we shall assume it implicitly from now on. Notice that by Lemma 2.5.3 we can always assume that $\models_{\mathcal{T}} \forall(P \to C)\sigma$ for any instance of a sound rule $P \to C$.

4.5.2 Definition of PTME-$\mathcal{I}$

Now we are ready to apply the framework of inference systems within partial theory model elimination. It needs only to restrict the computation of residues in TME-Sem (Def. 4.3.2) to theory inferences:

Definition 4.5.4 (Partial TME, $\mathcal{I}$-Version (PTME-$\mathcal{I}$)). *Let $\mathcal{I}$ be an inference system over Σ^{F} which is sound wrt. a given theory $\mathcal{T}$. The inference rule* partial theory model elimination $\mathcal{I}$-extension step PTME-I-Ext *is the same as* TTME-Sem-Ext *(Definition 4.3.2), except that the characterizing property is changed in the following way:*

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{(\mathcal{P},\ \mathcal{Q})\sigma} \quad \text{PTME-I-Ext}$$

where

$$\mathcal{P} = \begin{cases} [p \cdot K] \times & \text{if } \mathcal{R} \vee R_1 \vee \cdots \vee R_n = \square, \\ [p \cdot K] \circ (\mathcal{R} \vee R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

where

$$\mathcal{K} \Rightarrow_{P \to C, \sigma} \mathcal{R}\sigma$$

and $\mathcal{K} = \mathcal{B} \cup \{K\} \cup \{L_1, \ldots, L_n\}$ *for some* $\mathcal{B} \subseteq p$, *and* $P \to C$ *is a new variant of some inference rule from* $\mathcal{I}$.

(characterizing property for PTME-I-Ext)

We will refer to the thus changed inference rule partial theory extension step, PTME-I-Ext. *Extending Definition 3.2.4, we will write* PME-I-Ext *inferences as* $([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},P\to C,\langle\mathcal{R},\sigma\rangle,\{L_1\vee R_1,\ldots,L_n\vee R_n\}} (\mathcal{Q}',\ \mathcal{Q})\sigma$.

The calculus of partial theory model elimination, I-version (PTME-I) *is obtained from TTME-Sem (Definition 4.3.2) by replacing its inference rule* TME-Sem-Ext *by* PTME-I-Ext. *Occasionally, we will also speak of a calculus* PTME-$\mathcal{I}$ *in order to indicate that PTME-I is instantiated with a particular inference system* $\mathcal{I}$; *respective* PTME-I-Ext *inferences will be referred to as* PTME-$\mathcal{I}$-Ext *inferences.*

The need for using sound theory inference systems should be obvious: it guarantees the soundness of the overall calculus.

In order to get an intuition for the new characterizing property we will describe its operational behavior: in order to carry out a theory extension step one has to find a key set $\mathcal{K}$, made up from all the extending literals $L_1, \ldots, L_n$ from input clauses and from a subset of the ancestor literals p. Notice the demand that the leaf literal K is part of the key set $\mathcal{K}$. As in previous versions, this can be seen as the adaption of the "link condition" towards PTME-I (cf. Definition 4.2.4 and the subsequent discussion for the link condition in total theory reasoning).

Then, a theory inference using some new variant is carried out on $\mathcal{K}$. We need a *new* variant for two purposes: first, as usual, it avoids failure of unification due to name conflicts; second, it guarantees that extra variables in the (instantiated) conclusion of the inference rule will be different from any variable in the context. The underlying problem is much the same as in TTME-MSR (Def. 4.4.3), where we had to protect extra variables introduced by $\mathcal{T}$-MGRs against unwanted bindings in the context (cf. Note 4.4.1).

Finally, the conclusion of the theory inference comprises the residue $\langle \mathcal{R}, \sigma \rangle$, which is used together with the rest literals of the extending clauses as in TME-Sem-Ext (Def. 4.3.2) for carrying out the extension step. Notice that the case with $\mathcal{R} = \langle \Box, \sigma \rangle$ describes a *total* extension step, while otherwise a *partial* step results. Further notice that the premise of a theory inference rule is never empty. This means, operationally, that a residue cannot be guessed without any "justification".

We will briefly rephrase non-theory model elimination (ME, Def. 3.2.3) within PTME-I. For this, take for every n-ary predicate symbol P of the given signature Σ an inference rule

$$P(x_1, \ldots, x_n),\ \neg P(x_1, \ldots, x_n) \rightarrow \mathsf{F}\ .$$

This inference system is referred to by $\mathcal{I}_{SYN}$. It is easy to check that $\mathcal{I}_{SYN}$ is ground complete.

In PTME-I-Ext inferences, by the premise minimality of theory inferences, the key sets are forced to contain no duplicates Hence, in PTME-$\mathcal{I}_{SYN}$ every key set consists of exactly two literals, as is the case in ME. Further, the link condition of ME is realized through the characterizing property of PTME-I-Ext, which requires the leaf to be part of the key set. All this together gives us the following trivial theorem:

Theorem 4.5.1 (Theory Model Elimination simulates ME).
Every ME derivation is also a PTME-$\mathcal{I}_{SYN}$ derivation, and vice versa.

Note 4.5.1 (PTME-I and Minimality of Residues). Since we require that the inference system $\mathcal{I}$ is sound wrt. the given theory $\mathcal{T}$, it follows immediately from the respective definitions that $\langle \mathcal{R}, \sigma \rangle$ is a $\mathcal{T}$-residue of $\mathcal{K}$ (Def. 4.3.1). Hence, as desired, PTME-I is an instance of TME-Sem (Def. 4.3.2). More precisely, since our theory inferences are not forced to compute *minimal* $\mathcal{T}$-residues, PTME-I is an instance of *weak* TME-Sem. Thus, in order to turn PTME-I into a proper instance of TME-Sem one would have to impose a respective minimality condition on the theory inferences. Fortunately, unlike in TTME-MSR (cf. Note 4.4.3) such a "local" minimality condition is not required for PTME-I and can be replaced by a more "global" one.

Further, *insisting* on minimal $\mathcal{T}$-residues would not fit nicely into our syntactical-oriented framework due to its semantic nature. However, we can safely insist on the weaker concept of premise minimality in theory inferences, because this property can easily be checked syntactically (recall that we demand that the instantiated key set contain no duplications).

A typical example to demonstrate the benefit of the premise minimality is as follows: assume an inference rule $P(x,y),\ P(y,x) \rightarrow Q(x,y)$, a branch $p = [P(a,u)]$ and an input clause $C = P(a,v) \vee R$ (think of R as a long disjunction). Then, the PTME-I-Ext inference based on the theory inference $P(a,a) \Rightarrow Q(a,a)$ exists, which lengthens p towards $[P(a,a) \cdot Q(a,a)]$. Notice that within the theory inference the premise is amplified towards $\{\!|P(a,u),\ P(a,u)|\!\}$, but the input clause C is not used.

On the other hand, if the premise minimality restriction in theory inferences were dropped, a PTME-I-Ext inference based on the the key set $\{\!|P(a,u),\ P(a,v)|\!\}$ (which is obtained from *two* copies of C), amplification $\{\!|P(a,u),\ P(a,v)|\!\}$ and resulting theory inference $P(a,a), P(a,a) \Rightarrow Q(a,a)$ would be possible, which in turn would result in the branch set $[P(a,a) \cdot Q(a,a)]$, $[P(a,a)] \circ R\{v \leftarrow a\}$. Notice that in this case unnecessary open branches are introduced. Clearly, this should be avoided.

To sum up, the premise minimality restriction in theory inferences forbids multiple occurrences of (instantiated) key set literals, which is never necessary, as required copies can be drawn within the theory inference by suitable amplification.

One more example will illustrate the new definition.

Example 4.5.2 (Strict Orderings). Consider the theory of strict orderings and the respective inference system $\mathcal{I}_{\mathcal{SO}}$ of Example 4.5.1 again. None of the extension steps in Figure 4.3.2 is a PTME-I extension step, because $\mathcal{I}_{\mathcal{SO}}$ does not contain suitable theory inference rules. A comparable derivation is depicted in Figure 4.9.

4.5.3 Soundness and Answer Completeness

Theorem 4.5.2 (Soundness of PTME-I). *Let $\mathcal{T}$ be a universal theory, $\mathcal{I}$ be an inference system which is sound wrt. $\mathcal{T}$ and let M be a clause set. If a PTME-$\mathcal{I}$ refutation of M exists then M is $\mathcal{T}$-unsatisfiable.*

This property in essence follows from the soundness of $\mathcal{I}$ wrt. $\mathcal{T}$: every instance of an inference rule which will be used in a theory extension step is $\mathcal{T}$-valid. This implies that any residue is a logical consequence of its preceding branch literals. This observation is at the heart of the soundness proof.

A formal proof would be carried out much like the soundness proof for TTME-MSR (Theorem 4.4.1), namely by reducing it to the soundness of a stronger partial theory connection calculus. This is straightforward and hence omitted. As always, the other direction — completeness — is much more complicated. The basic setup was already described in the introduction to this section, and it remains to rephrase it in the current terminology. However, we will be more general than needed in that we formulate first-order "background" derivations:

Definition 4.5.5 (Background Derivation). *A* (first-order) background derivation *of a literal L_{n+1} from a given literal set M and top literal T wrt. an inference system $\mathcal{I}$ consists of sequences*

$$\begin{array}{llll} (L_1 = T), & \ldots, & L_n, & L_{n+1} \\ M_1, & \ldots, & M_n & \\ \sigma_1, & \ldots, & \sigma_n & \\ P_1 \to C_1, & \ldots, & P_n \to C_n & \end{array}$$

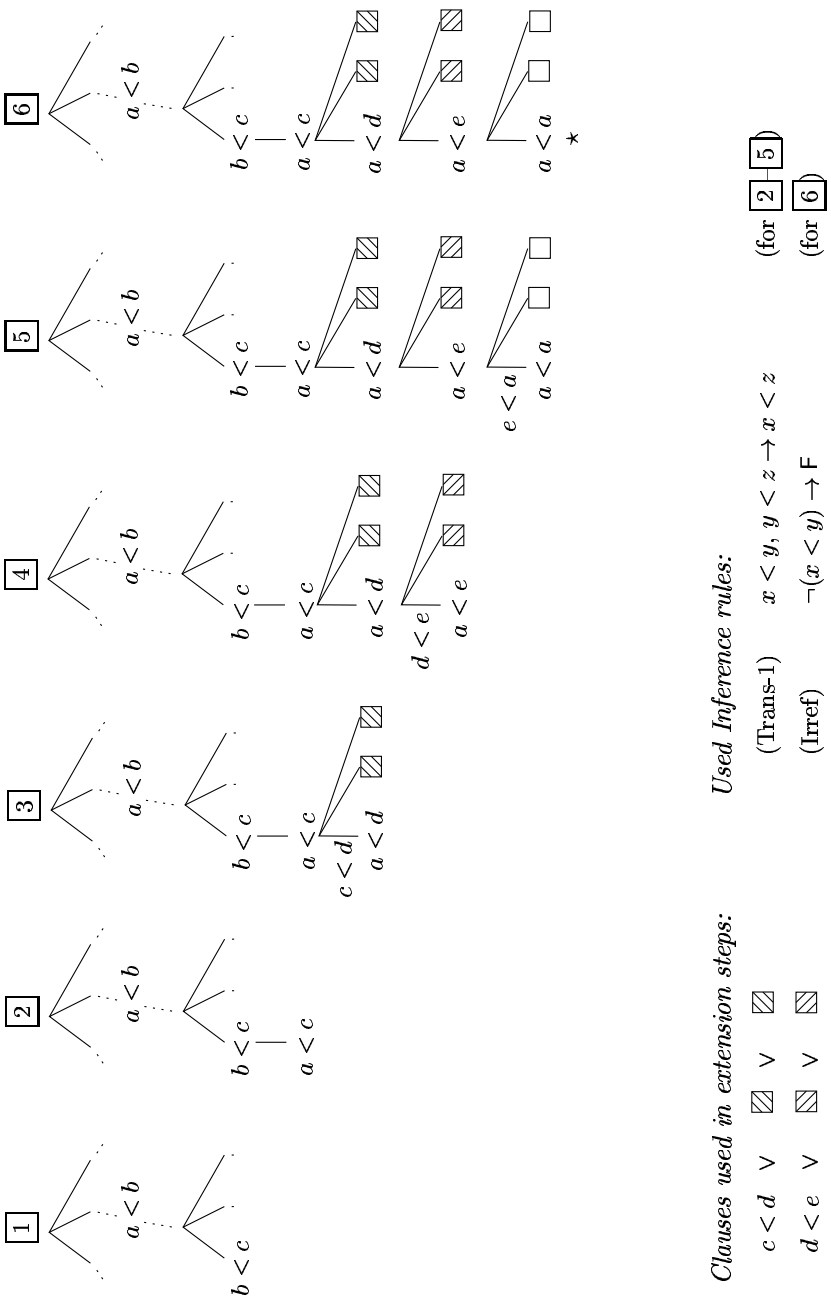


Figure 4.9. A PTME-I derivation. Missing key set literals are annotated at the edges.

such that for $i = 1, \ldots, n$, $M_i \subseteq M\sigma_1 \cdots \sigma_{i-1}$, $P_i \to C_i$ *is a new variant of an inference rule from* $\mathcal{I}$ *and*

$$L_i, M_i \Rightarrow_{P_i \to C_i,\, \sigma_i} L_{i+1}$$

Background derivations usually are written more suggestive as

$$(L_1 = T) \overset{M_1}{\Longrightarrow}_{P_1 \to C_1,\, \sigma_1} L_2 \cdots L_n \overset{M_n}{\Longrightarrow}_{P_n \to C_n,\, \sigma_n} L_{n+1}$$

We will allow dropping of indices if context allows, and we will write

$$D = (T \Rightarrow^*_{\mathcal{I}, M, \sigma} L_{n+1}), \quad \text{where } \sigma = \sigma_1 \cdots \sigma_n | \, Var(M)$$

to denote the fact that such a derivation D *exists. If* $L_{n+1} = \Box$ *then* D *is also called a* refutation. *In this case the combined substitution* σ *is called the* computed answer.

If $\mathcal{I}$, M *and* T *are ground then necessarily* $\sigma_i = \varepsilon$ *and hence* $\sigma = \varepsilon$. *Thus, in this case the indices can be dropped without losing information, and we speak of* ground *derivations.*

Notice that in background derivations *linearity* is demanded in the sense that the conclusion of one theory inference is fed into the premise part of the subsequent theory inference. Further notice that in the i-th step the side literals M_i are drawn from $M\sigma_1 \cdots \sigma_{i-1}$. In other words, in every step the substitution computed so far $\sigma_1 \cdots \sigma_{i-1}$ is applied to the input set M. This realizes a "rigid" treatment of variables (cf. Note 4.2.4).

These first-order derivations will be used below in a stand-alone completeness result for "linearizing completion" (Theorem 5.6.4). It will allow us to use theory inference systems as complete $\mathcal{T}$-unification procedures. But for the current purpose, to formulate a sufficient completeness criterion, only the ground version of "derivation" is needed:

Definition 4.5.6 (Completeness Criterion). *An inference system* $\mathcal{I}$ *is called* ground complete *wrt. a theory* $\mathcal{T}$ *iff for every minimal* $\mathcal{T}$*-unsatisfiable ground literal set* M *and every literal* $T \in M$ *there is a background refutation*

$$R = (T \Rightarrow^*_{\mathcal{I}^g, M} \Box) \ .$$

We say that $\mathcal{I}$ *is ground complete* for negative top literals *if such a refutation* R *exists for every negative literal* $T \in M$ *(but not necessarily for every positive literal).*

In words, for an inference system $\mathcal{I}$ to be ground complete we require that any minimal $\mathcal{T}$-unsatisfiable literal set M must admit the derivation of the empty clause by application of theory inferences from ground instances $\mathcal{I}^g$ of $\mathcal{I}$ to M. Notice that we demand the *independence of the top literal*: any literal may be used as a starting point (but only for negative literals in the restricted case).

Why we need a completeness criterion of the stated form to achieve completeness of PTME-I needs clarification. For this, we return to Example 4.5.1 on page 119. The inference system $\mathcal{I}_{\mathcal{SO}}$ shown there was used in Example 4.5.2 for developing a sequence of partial extension steps. These steps led to the closing of the branch with literals $a < b$ and $b < c$ in tableau $\boxed{1}$ in Figure 4.9 (page 124). This can be seen as a proof of the $\mathcal{T}$-complementarity of the involved literals, i.e. of $M = \{a < b,\ b < c,\ c < d,\ d < e,\ e < a\}$. Alternatively, one can think of replacing a given total theory extension step (cf. Figure 4.2 on page 69) with key set M by the stated sequence of partial extension steps. Since M is given only conceptually, but not factually, one can think of M as being computed by search using theory inferences.

For this computation of M, it is totally irrelevant whether in the tableau context the involved literals are ancestor literals or literals from input clauses; further, the rest literals of input clauses are also irrelevant. Instead, what matters is the following: (1) the start literal — $b < c$ — cannot be chosen arbitrarily, but is given "from outside" as the leaf literal to be extended; (2) the theory inferences are chained in a linear way, which means that the conclusion literal of one theory inference — here $a < c$ – is to be part of the premise of the subsequent theory inference; this stems from the design decision of model elimination (the "link condition"); (3) the involved literals must be taken either from ancestor context or from input literals immediately, but must not be derived.

Altogether, the properties (1), (2) and (3) together with the unit-resulting property of theory inferences states nothing but a completeness requirement with respect to *linear*, *unit-resulting* refutations for *arbitrary* goal literal.

Example 4.5.3 (Ground Complete Inference System). The inference system $\mathcal{I}_{\mathcal{SO}}$ in Example 4.5.1 is ground complete wrt. the theory $\mathcal{SO}$ of strict orderings. This will be shown in Chapter 5 below for the extended theory with equality.

For instance, a background refutation for

$$M = \{a < b,\ b < c,\ c < d,\ d < e,\ e < a\}$$

with top literal $b < c$ is as follows:

$$b < c \overset{a<b}{\Longrightarrow} a < c \overset{c<d}{\Longrightarrow} a < d \overset{d<e}{\Longrightarrow} a < e \overset{e<a}{\Longrightarrow} a < a \overset{\emptyset}{\Longrightarrow} \square$$

The last theory inference uses a ground instance of the (Irref) inference rule, all other inferences use ground instances of the $(\mathrm{Trans}-1)$ inference rule. Notice that this background refutation corresponds to the computation in Example 4.5.2.

Before turning towards completeness one more improvement will be introduced now.

In Section 4.4.2 we defined the restriction "stability wrt. minimality" for TTME-MSR. It said that a once chosen key set *remains* minimal $\mathcal{T}$-complementary, even after further instantiation. We are going to define the analogous restriction for PTME-I:

Definition 4.5.7 (Stability wrt. Minimality for PTME-I). *Let*

$$\mathcal{D} = (\mathcal{P}_1 \vdash_{[p_1],\mathcal{K}_1,P_1\to C_1,\langle\mathcal{R}_1,\sigma_1\rangle,E_1} \mathcal{P}_2 \cdots \mathcal{P}_{n-1} \vdash_{[p_{n-1}],\mathcal{K}_{n-1},\langle\mathcal{R}_{n-1},\sigma_{n-1}\rangle,E_{n-1}} \mathcal{P}_n)$$

be a PTME-I derivation, where E_i denotes the sequence of extending clauses in the i-th extension step. We say that $\mathcal{D}$ is stable wrt. minimality *iff*

for $i = 1, \ldots, n-1$: $\mathcal{K}_i\sigma_i \cdots \sigma_{n-1}$ contains no duplicates.

It is clear by definition of theory inferences (Def. 4.5.2) that $\mathcal{K}_i\sigma_i$ contains no duplicates. However, as for the respective minimality property in TTME-MSR (see the discussion following Definition 4.4.4), there is no guarantee that $\mathcal{K}_i\sigma_i \cdots \sigma_{n-1}$ contains no duplicates. Hence, this property must be demanded explicitly.

Theorem 4.5.3 (Answer Completeness of PTME-I). *Let $\mathcal{T}$ be a universal theory, $\mathcal{I}_\mathcal{T}$ be an inference system which is ground complete wrt. $\mathcal{T}$, c be a computation rule, P be a program and $\leftarrow Q$ be a query (cf. Def. 3.2.5).*

Then, for every correct answer $\{Q\Phi_1, \ldots, Q\Phi_m\}$ for P and $\leftarrow Q$ there exists a PTME-$\mathcal{I}_\mathcal{T}$ refutation $\mathcal{D}^c$ of P and $\leftarrow Q$ via c, which is stable wrt. minimality, with computed answer $Answer(\mathcal{D}^c) = \{Q_1, \ldots, Q_l\}$, and such that

$$\{Q_1, \ldots, Q_l\}\delta \subseteq \{Q\Phi_1, \ldots, Q\Phi_m\} \quad \textit{for some substitution } \delta.$$

In order to get an idea of answer computation within theory reasoning, consider the toy application of Figure 5.1 in Section 5.1.1.

Proof. The first part of the proof is taken literally from Part 1 in the respective proof of Theorem 4.4.1 on page 109. Hence let

$$M''' = P'' \cup \{\leftarrow Q\Phi'_1, \ldots, \leftarrow Q\Phi'_r\}$$

be a finite minimal $\mathcal{T}$-unsatisfiable ground clause set, consisting of instances of clauses from P and of instances of $\leftarrow Q$. Furthermore, from Part 2 (ground completeness) in the proof of Theorem 4.4.1 we take the TTME-MSR refutation $\mathcal{D}'_{TTME-MSR}$ of M''' with start clause $\leftarrow Q\Phi'_1$.

Now the PTME-I specific part comes. By Lemma A.1.13 the refutation $\mathcal{D}'_{TTME-MSR}$ can be transformed into a PTME-$\mathcal{I}^g_\mathcal{T}$ refutation $\mathcal{D}'_{PTME-I}$ of M''' with start clause $\leftarrow Q\Phi'_1$, which is stable wrt. minimality.

The further argumentation is taken literally from Part 3 (query usage) and Part 4 (lifting) of the proof of Theorem 4.4.1, however, using the refutation $\mathcal{D}'_{PTME-I}$ instead of $\mathcal{D}'_{TTME-MSR}$.

Note 4.5.2 (Regularity in PTME-I). As with the other calculi treated so far, we are interested in regularity restrictions (Def. 3.3.2) also for PTME-I.

One simple regularity refinement for background refutations is to insist (on the ground level) that $L_i \neq L_j$, for $i, j \in \{1, \ldots, n+1\}$, $i \neq j$ in Definition 4.5.5. This obviously preserves completeness, because, if violated, one simply has to delete the inferences carried out between identical literals, yielding a shorter derivation whose computed answer is the same or more general. The lifting to the first-order level would be straightforward; also, as the proof of the previous theorem shows, a background refutation is still contained as a chain of residues within a PTME-I refutation. Hence, the suggested regularity restriction can be applied within these chains.

Also, the regularity restriction possible in the TTME-MSR version (which is the base for PTME-I) carries over for such structural reasons to PTME-I *except for residue literals*. Residue literals are usually "new" with respect to the replaced total extension step, and there is no simple argument to guarantee that these are not identical to other literals in the refutation (stemming from side literals in extension steps or stemming from other residue chains). I suspect that such a strong form of regularity can be demanded, but unfortunately I did not find a proof. The difficulty is that in order to modify a regularity violating refutation into a regular one, non-local transformations on the tableau seem to be necessary which I could not prove to terminate.

4.5.4 An Application: Generalized Model Elimination

The purpose of this example is to demonstrate the usefulness of the language of theory inference systems. Other sample inference systems, namely ones which are produced by linearizing completion, will be described in Chapter 5 (modal logic in Section 5.7.3 and the combined theory of strict orderings and equality in Section 5.7.2).

In *generalized model elimination (GME)* [Bollinger, 1991] (finite) conjunctions of literals take the place where a literal is required in ordinary model elimination. Such a conjunction of literals is called a *generalized literal*.

Consequently, in a GME extension step a connection is searched for among the literals of two generalized clauses. That is, a connection between two generalized literals $L_1 \wedge \ldots \wedge L_n$ and $K_1 \wedge \ldots \wedge K_m$ is a pair (L_i, K_j) ($i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$) which can be made complementary by some substitution. Since no residues are involved, GME can be seen as an instance of total theory reasoning. Nevertheless, the underlying theory reasoning can also be "programmed" using theory inference rules. This shall be demonstrated here.

In order to express GME within our framework we adopt the convention that a conjunction $L_1 \wedge \cdots \wedge L_n$ of literals shall be represented as the list $[L_1, \ldots, L_n]$. It requires only two steps to express GME within PTME-I:

– A GME input clause

$$(L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$$

becomes the TME clause

$$g\_lit([L_1^1, \ldots, L_{n_1}^1]) \vee \cdots \vee g\_lit([L_1^m, \ldots, L_{n_m}^m]) \ .$$

Of course, *g_lit* is just a new literal symbol. Note that every *predicate* symbol in GME becomes a *function* symbol in TME. Let M_{GME} refer to the thus translated clause set.
It should further be noted that by introducing lists à la Prolog we do not leave first order logic; the notation $[L_1, L_2, \ldots, L_n]$ is nothing but syntactic sugar for the term $cons(L_1, cons(L_2, \ldots cons(L_n, nil)))$.

– The procedure for searching for complementary literals in the generalized literals can be coded immediately into the inference rules. For this we take the following inference system[25] $\mathcal{I}_{GME}$:

$$\begin{aligned}
\mathcal{I}_{GME}: \quad & g_lit(GL_1),\ g_lit(GL_2) \rightarrow connection(GL_1, GL_2) \\
& connection([\neg A|R_1], [\ A|R_2]) \rightarrow \mathsf{F} \\
& connection([\ A|R_1], [\neg A|R_2]) \rightarrow \mathsf{F} \\
& connection([L|R], GL) \rightarrow connection(R, GL) \\
& connection(GL, [L|R]) \rightarrow connection(GL, R)
\end{aligned}$$

The first rule sets up a "call" to the *connection* predicate, which in turn scans through the lists and finds complementary literals. Of course, this is a Prolog program in disguise. In other words, we used the language of theory inference systems as a programming language to describe theory inferences[26]

We conclude with the relevant theorem:

Theorem 4.5.4 (Theory Model Elimination simulates GME). *Let M be a set of generalized clauses. Then a GME-refutation of M exists if and only if a PTME-$\mathcal{I}_{GME}$ refutation of M_{GME} exists.*

4.6 Restart Theory Model Elimination

In this section it will be demonstrated that model elimination — and hence PTTP — can be defined such that it is complete without the use of contrapositives. As argued for in [Baumgartner and Furbach, 1994a] we believe that this result is interesting in at least two respects: it makes model elimination available as a calculus for non-Horn logic programming and it enables model elimination to perform proofs in a natural style by case analysis.

[25] The notation $[L|R]$ means $cons(L, R)$.

[26] A totally different application in this spirit is to express the *Hyper tableau* calculus in [Baumgartner *et al.*, 1996] within theory model elimination. For this, the input clause set is transformed into a set of theory inference rules in a tricky way, and PTME-I will "behave" like the Hyper tableau calculus.

Let us explain what we mean by the term "without the use of contrapositives". In implementations of theorem proving systems usually n procedural counterparts L_i:-$\overline{L}_1 \wedge \cdots \wedge \overline{L}_{i-1} \wedge \overline{L}_{i+1} \wedge \cdots \wedge \overline{L}_n$ for a clause $L_1 \vee \cdots \vee L_n$ have to be considered. Each of these is referred to as a *contrapositive* of the given clause and represents a different entry point during the proof search into the clause. It is well-known that for Prolog's SLD-resolution one single contrapositive suffices, namely the "natural" one, selecting the head of the clause as entry point. For full first-order systems the usually required n contrapositives are either given more *explicitly*, as in the SETHEO prover [Letz *et al.*, 1992], or more *implicitly*, as in the connection calculus of [Eder, 1992] by allowing the setting up of a connection with *every* literal in a clause (Cf. Note 3.1.1 in Chapter 3). The distinction is merely a matter of presentation and is not essential for our purposes. Now, by a system "without contrapositives" we mean more precisely a system which does not need all n contrapositives for a given n-literal clause. The modifications will be such that for any clause, only one single head literal is selected for establishing the connection in extension steps.

This calculus is called *restart model elimination* (RME) and was introduced in [Baumgartner and Furbach, 1994b]; in [Baumgartner and Furbach, 1994a] an extended version also dealing with implementational issues can be found. Further, we discovered that the connection calculus [Eder, 1992] is complete without contrapositives and *without any change to the calculus*. This surprising result is due to its relaxed complementary-literal condition which includes the RME inference rules. The *logic programming* aspects of RME were further investigated in [Baumgartner *et al.*, 1995] by further modifying the calculus and deriving answer completeness results.

Here, we are concerned with lifting this latter answer completeness result to the theory reasoning level, which I think is of particular interest for automated reasoning and problem solving purposes[27] As mentioned above, RME is attractive in this context for its "case analysis" properties. For instance, in proving theorems such as "if $x \neq 0$ then $x^2 > 0$" a human typically uses case analysis according to the axiom $X < 0 \vee X = 0 \vee 0 < X$. This seems a very natural way of proving the theorem and leads to an easily understandable proof. The RME calculus carries out precisely such a proof by case analysis.

As was carried out for theory model elimination, there are several alternatives for the restart versions. We will restrict ourselves to our primary framework of theory inference rules. A restart version corresponding to TTME-MSR would be defined in a straightforward way then.

4.6.1 Definition of Restart Theory Model Elimination

As a preliminary, we have to presuppose *definite* theories (cf. Def. 2.4.1), represented by theory inference rules of a certain form:

[27] An earlier *ground* version of *total* restart theory model elimination with a refutational completeness result was presented in [Baumgartner, 1994].

Definition 4.6.1 (Contra-Definite Inference System). *A theory inference rule $P \to C$ (over a signature Σ^{F}) is called* contra-definite *iff P is of the form $\{\!|\neg A, A_1, \ldots, A_n|\!\}$ where $n \geq 0$ and either $C = \mathsf{F}$ or C is of the form $\{\!|\neg B_1, \ldots, \neg B_m|\!\}$, where $m \geq 1$. A* contra-definite theory inference system *consists of contra-definite rules only. A* contra-definite theory inference *is a theory inference carried out with a contra-definite rule.*

That is, a contra-definite rule contains exactly one negative literal in the premise and allows for a contrapositive of the form $A_1, \ldots, A_n \to A$ or $A_1, \ldots, A_n, B_1, \ldots, B_m \to A$, which is nothing but the usual notation for a definite clause. Notice that, for instance, $A \to B$ is *not* a contra-definite rule, but $\neg B \to \neg A$ is one. Our interest in contra-definite inference system comes from the fact that these are sufficient in the calculus below, as rules of the form $A \to B$ are never applicable.

The restriction to contra-definite inference systems covers practical important cases such as equality and partial orderings. Although modifications towards more general theories are conceivable, these will be not considered. A motivation for this restriction follows below.

As said, restart model elimination is motivated by logic programming purposes. As is common in the literature, clauses shall also be written in the "logic programming style" using an arrow "$\leftarrow$". Notice that "$\leftarrow C$" is *no* well formed-formula according to Definition 2.2.3. This fact will, however, simply be neglected. The translation of $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_n$ into a clause is $A_1 \vee \cdots \vee A_m \vee \neg B_1 \vee \cdots \vee \neg B_n$.

As mentioned, only one positive literal per clause as entry point for extension steps shall be allowed. This is formalized at the calculus level by using *selection functions*:

Definition 4.6.2 (Selection Function). *A* selection function *f maps a clause $C = (A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m)$ with $n \geq 1$ (i.e. a nonnegative clause) to a nonempty subset $S_C \subseteq \{\!|A_1, \ldots, A_n|\!\}$. S is called the set of* selected literals *of C by f. The selection function f is required to be* stable under lifting, *which means that if f selects $S_C\gamma$ in the instance of the clause $C\gamma$ then f selects S_C in C.*

Notice that for definite clauses $A \leftarrow B_1, \ldots, B_m$ any selection function necessarily selects $\{\!|A|\!\}$.

Definition 4.6.3 (Partial Restart TME, $\mathcal{I}$-Version). *Let $\mathcal{I}_\mathcal{T}$ be a contra-definite inference system which is sound wrt. a given definite theory $\mathcal{T}$ and let f be a selection function. The inference rule* partial definite[28] theory model elimination $\mathcal{I}_\mathcal{T}$-extension step, PDTME-I-Ext *is the same as* PTME-I-Ext *(Definition 4.5.4), except for the following changes:*

[28] The term "contra-definite" would be too long here.

$$\frac{[p \cdot K],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{(\mathcal{P},\ \mathcal{Q})\sigma} \quad \text{PDTME-I-Ext}$$

where

$$\mathcal{P} = \begin{cases} [p \cdot K] \times & \text{if } \mathcal{R} \vee R_1 \vee \cdots \vee R_n = \square, \\ [p \cdot K] \circ (\mathcal{R} \vee R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

where K is negative literal, and

$$\mathcal{K} \Rightarrow_{P \to C, \sigma} \mathcal{R}\sigma$$

and $\mathcal{K} = \mathcal{B} \cup \{K\} \cup \{L_1, \ldots, L_n\}$ for some $\mathcal{B} \subseteq p$, and $P \to C$ is a new variant of some inference rule from $\mathcal{I}_\mathcal{T}$. Furthermore,

1. *$\mathcal{B}$ consists of positive literals only, and*
2. *L_i is a positive literal and $L_i \in f(L_i \vee R_i)$ (for $i = 1, \ldots, n$), and*
3. *$\mathcal{R} = \square$ or $\mathcal{R}$ consists of negative literals only.*

(characterizing property for PDTME-I-Ext)

The inference rule restart step, Restart, *is defined as follows:*

$$\frac{[p \cdot K],\ \mathcal{Q}}{[p \cdot K] \circ (\neg A_1 \vee \cdots \vee \neg A_n),\ \mathcal{Q}} \quad \text{Restart}$$

where K is a positive literal, and $\neg A_1 \vee \cdots \vee \neg A_n$ is a new variant of a negative clause from the given input clause set, also called restart clause *in this context.*

The calculus of partial restart theory model elimination, I-version (PRTME-I) *consists of the inference rules* PDTME-I-Ext *and* Restart. *We insist that any PRTME-I derivation starts with a negative start clause $\neg A_1 \vee \cdots \vee \neg A_n$ from the input clause set M (cf. Def. 3.2.4).*

Notice that the inference rules PDTME-I-Ext and Restart are disjoint: PDTME-I-Ext is applicable only to negative leaves and Restart is applicable only to positive leaves. The selection function achieves the possibility of restricting the set of contrapositives and of "entering" clauses only via head literals. The selection function can be restricted to select one literal only, without sacrificing completeness.

Note 4.6.1 (Redundancy of Conditions 1, 2 and 3). In a strict sense, the conditions 1, 2 and 3 in the definition of PDTME-I-Ext are not necessary, as they

are a consequence of the requirement that the leaf K is negative and the precondition that the used inference system $\mathcal{I}_\mathcal{T}$ is contra-definite: the contra-definite rules enforce that from a negative literal we either close the branch or append a residue consisting of negative literals only (giving condition 3). Further, this must be done by means of some positive literals, stemming either from the ancestor context of K (giving condition 1) or from extending clauses (giving 2). Nevertheless, although redundant, the conditions 1, 2 and 3 are stated in order to make the structure of inferences of PDTME-I-Ext inferences explicit.

In the case that the input clause set is a *Horn* set, then, with respect to derivations, the result and argumentation of Proposition 4.3.2 established for PTME-Sem-Ext inferences in PTME-Sem derivations still holds if PTME-I-Ext inferences are used instead, provided that the underlying theory inference system is *contra-definite*. In this case, Restart can never be carried out, and PRTME-I coincides with PTME-I.

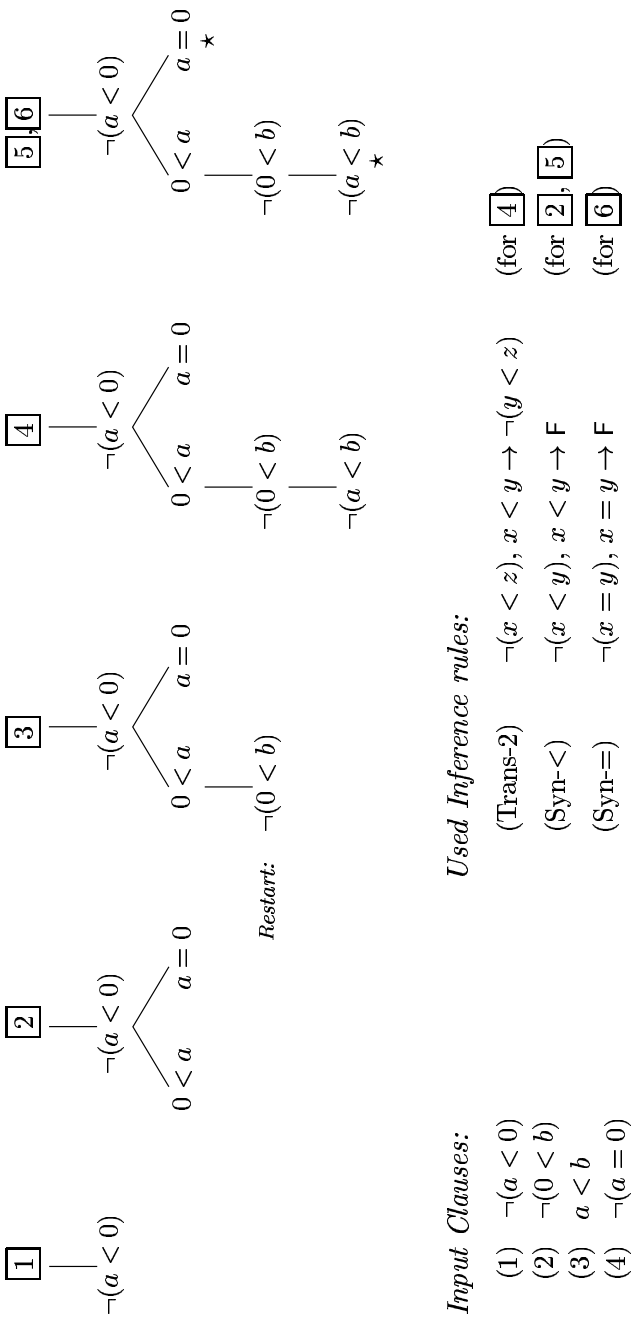
Example 4.6.1 (Strict Orderings). Consider the theory of strict orderings and the respective inference system $\mathcal{I}_{\mathcal{SO}}$ of Example 4.5.1 again. Figure 4.10 contains a sample refutation, where the depicted inference system consists of the contra-definite rules (Trans $-$ 2) and (Syn$-$ $<$) from $\mathcal{I}_{\mathcal{SO}}$ and the additional rule (Syn$-$ $=$).

Tableau $\boxed{1}$ contains the start clause clause (1). Tableau $\boxed{2}$ is obtained by a total extension step with input clause (5), using the inference rule Syn-$<$. Here, we assumed that the leftmost literal in clause (5) is selected. Since the upcoming leaf $0 < a$ is positive, only a Restart step can be carried out (Tableau $\boxed{3}$). The new leaf $\neg(a < b)$ in tableau $\boxed{4}$ is obtained by application of a PDTME-I-Ext step applied to $0 < a$ and $\neg(0 < b)$. Tableau $\boxed{5}$ and $\boxed{6}$ are obtained by extension steps with clauses (5) and (6).

The price of the absence of contrapositives is that whenever a branch ends with a positive literal, the search has to be "restarted" with a new negative clause. Now we can also motivate the restriction to definite theories.

Note 4.6.2 (Restriction to Definite Theories). If more general theories than definite theories were allowed, for instance Horn theories, then the restriction to negative clauses as queries would no longer be complete, even when more general than contra-definite inference systems were allowed. For instance, taking the inference system $\mathcal{I}_{\mathcal{SO}}$ again and the $\mathcal{SO}$-unsatisfiable clause set $\{a < a\}$ would not even allow a setup as initial branch set. In general, *any* input clause would have to be considered as a start clause and for restart steps. A respective calculus would be defined easily, although one might doubt that it is in the goal-oriented spirit of model elimination.

On the other hand, for definite theories one can allways find a contra-definite inference system by means of linearizing completion, which is ground complete for negative top literals (Corollary 5.6.2). As will be shown below



Input Clauses:

(1) $\neg(a < 0)$
(2) $\neg(0 < b)$
(3) $a < b$
(4) $\neg(a = 0)$
(5) $x < 0 \vee 0 < x \vee x = 0$

Used Inference rules:

(Trans-2) $\neg(x < z), x < y \rightarrow \neg(y < z)$ (for 4)
(Syn-<) $\neg(x < y), x < y \rightarrow \mathsf{F}$ (for 2, 5)
(Syn-=) $\neg(x = y), x = y \rightarrow \mathsf{F}$ (for 6)

Figure 4.10. A PRTME-I derivation.

in Theorem 4.6.1, this suffices to obtain a complete calculus. We take this as the motivation to restrict to contra-definite inference systems.

Note 4.6.3 (Related Work.). The non-theory version of the PRTME-I calculus was described in detail in [Baumgartner and Furbach, 1994a]. There remain some minor differences, however. In [Baumgartner and Furbach, 1994a], we used what we called the *goal-normal form*, a syntactic transformation on the input clause set which replaces every *negative* input clause $\neg L_1 \vee \cdots \vee \neg L_n$ with $Goal \leftarrow L_1 \wedge \cdots \wedge L_n$. The start clause to be used then is $\leftarrow Goal$, and a restart step there consists of copying the topmost literal, which is always the literal $\neg Goal$. In the present version, we make the purpose of this transformation explicit, which is to allow any negative clause as a start clause or for restarts.

As another difference, expressed in the present terminology, we allowed either the selection of *every* positive literal in a clause, or precisely *one*. Thus, the present definition allows for more flexibility in this respect by the possibility of selection "in between".

In some cases it helps to find a shorter refutation if additionally PTME-I-Ext steps are allowed at *positive* leaves. This is in particular the case if no extending clauses are involved. In this case the inference could be described as a "reduction step" with a positive leaf literal. In [Baumgartner and Furbach, 1994a] this version was called *non-strict* RME.

Note 4.6.4 (Non-Theory Restart ME). Short of these minor differences, the non-theory model version in [Baumgartner and Furbach, 1994a] is obtained by instantiating PRTME-I with a theory inference system consisting of syntactical rules only (cf. the Syn-$<$ and Syn-$=$ rules in Example 4.6.1). This paper also contains a comparison to related methods, such as near-Horn Prolog [Loveland and Reed, 1989] and Plaisted's problem reduction formats [Plaisted, 1988].

The only work related to *theory* restart model elimination is the calculus described in [Petermann, 1993b]. In fact, Petermann rediscovers the total version of our PRTME-I calculus, which was first described in [Baumgartner, 1993]. His total version of our PDTME-I-Ext inference rule is called "T-connection inference with positive refinement" there, and our Restart rule has the same name.

4.6.2 Soundness and Answer Completeness

The soundness statement for PRTME, and also its proof would be much the same as that of PTME-I, and hence is omitted. We only remark that the soundness of new inference rule Restart (i.e. the fanning of a new clause below some branch) is already covered by Lemma 4.2.1.1

Concerning completeness, we start with the ground version of PRTME-I. Following that, an analysis of the completeness proof will allow us to incorporate a specific regularity restriction. Then we treat the first-order version.

Lemma 4.6.1 (Ground Completeness of PRTME-I). *Let $\mathcal{T}$ be a definite theory and $\mathcal{I}_\mathcal{T}$ be a contra-definite theory inference system which is ground complete wrt. $\mathcal{T}$ for negative top literals (cf. Def. 4.5.6). Then for every minimal $\mathcal{T}$-unsatisfiable ground clause set M and every negative clause $G \in M$ and selection function f a PRTME-$\mathcal{I}^g_\mathcal{T}$ refutation of M wrt. f and start clause G exists.*

Proof. Informally, the proof is by splitting the non-Horn clause set M into Horn sets, assuming by completeness of PTME-$\mathcal{I}_\mathcal{T}$ on the ground level (Lemma A.1.13) respective refutations, and then assembling these refutations (without reduction steps) into the desired restart model elimination refutation. There, reduction steps come in by replacing extension steps with split positive unit clauses by reduction steps to the literals where the restart occurred.

For the formal proof some terminology is introduced: we say that a branch set $\mathcal{P}$ "contains (an occurrence of) a clause $A_1, \cdots, A_n \leftarrow B_1, \cdots, B_m$" iff $\mathcal{P}$ is of the form

$$\mathcal{P} = ([p \cdot A_1], \ldots, [p \cdot A_n], [p \cdot \neg B_1], \cdots, [p \cdot \neg B_m], \mathcal{P}')$$

for some branch p. If we speak of "replacing a clause C in a derivation by a clause $C \vee D$" we mean the derivation that results when using the clause $C \vee D$ in place of C in extension steps. Also, the same literal $L \in C$ must be selected as extending literal.

By a "derivation of a clause C" we mean a derivation that ends in a branch set which contains at least one occurrence of the clause C.

Let $k(M)$ denote the number of occurrences of positive literals in M minus the number of definite clauses in M ($k(M)$ is related to the *excess literal parameter* in [Anderson and Bledsoe, 1970]). Now we prove the claim by induction on $k(M)$.

Induction start ($k(M) = 0$): M must be a set of Horn clauses. Clearly, M must contain a negative clause G as claimed. By the ground completeness of TTME-MSR (Lemma A.1.2) and Lemma A.1.13 we can obtain a PTME-$\mathcal{I}^g_\mathcal{T}$ refutation $\mathcal{D}$ of M with start clause G. Since $\mathcal{I}_\mathcal{T}$, and hence also $\mathcal{I}^g_\mathcal{T}$ is given as a contra-definite theory inference system every PTME-I-Ext step must follow the structure given in Proposition 4.3.2. This was argued for in Note 4.6.1 above. Thus, any PTME-I-Ext step is also a PDTME-I-Ext step. Hence, $\mathcal{D}$ is a PRTME-$\mathcal{I}^g_\mathcal{T}$ refutation as desired.

Induction step ($k(M) > 0$): As the induction hypothesis suppose the result to hold for unsatisfiable ground clause sets M' with $k(M') < k(M)$.

Since $k(M) > 0$, M must contain a non-Horn clause

$$C = A_1, A_2, \ldots, A_n \leftarrow B_1, \ldots, B_m$$

with $n \geq 2$. Without loss of generality assume that $A_1 \in f(C)$, i.e. that A_1 is selected by f in C. Now define n sets

$$M_1 = (M \setminus \{C\}) \cup \{A_1 \leftarrow B_1, \ldots, B_m\}$$
$$M_2 = (M \setminus \{C\}) \cup \{A_2\}$$
$$\vdots$$
$$M_n = (M \setminus \{C\}) \cup \{A_n\}$$

Every set M_i $(i = 1 \ldots n)$ is $\mathcal{T}$-unsatisfiable (because otherwise, a model for one of them would be a $\mathcal{T}$-model for M). Furthermore, it holds $k(M_i) = k(M) - n + 1 < k(M)$. Thus, by the induction hypothesis respective PRTME-$\mathcal{I}^g_\mathcal{T}$ refutations $\mathcal{D}_i$ of M_i exist.

Now consider $\mathcal{D}_1$ and replace in $\mathcal{D}_1$ every PDTME-I-Ext step with the clause $A_1 \leftarrow B_1, \cdots, B_m$ by a PDTME-I-Ext step with C using the selected literal A_1. Call this derivation $\mathcal{D}'_1$.

$\mathcal{D}'_1$ is a derivation of, say, k_j occurrences of the positive unit clauses A_j $(j = 2, \ldots, n)$ from the input set M. Now every occurrence of A_j can be eliminated from the derived branch set according to the following procedure: for $j = 2, \ldots, n$ and for every of the k_j branches $[p_1], \ldots, [p_{k_j}]$ ending in A_j, cumulatively extend $\mathcal{D}'_{j-1}$ in the following way: first apply a restart step to $[p_i]$ $(1 \leq i \leq k_j)$ using as the restart clause the start clause Q_j of $\mathcal{D}_j$ (the induction hypothesis gives us that Q_j is a negative clause and hence is a legal restart clause). Now that Q_j occurs in the resulting derivation, we can append the refutation $\mathcal{D}_j$, however using $[p_i] \circ Q_j$ instead of Q_j. Note that A_j occurs now in every branch of R_j.

Let $\mathcal{D}''_j$ be the resulting derivation. $\mathcal{D}''_j$ is a PRTME-$\mathcal{I}^g_\mathcal{T}$ derivation of clauses $A_{j+1}, \ldots, A_n$ from $M \cup \{A_j\}$. In order to turn $\mathcal{D}''_j$ into a derivation $\mathcal{D}'_j$ of clauses $A_{j+1}, \ldots, A_n$ from M alone, every use of A_j as extending clause in a PDTME-I-Ext step in the appended refutations $\mathcal{D}_j$ can be replaced by the ancestor literal $A_j \in [p_i]$ occurring on the extended branch.

Notice that this construction shortens the list of positive unit clauses by one to $A_{j+1}, \ldots, A_n$. Hence, repeated application will terminate and, at the end $\mathcal{D}'_n$ is the PRTME-$\mathcal{I}^g_\mathcal{T}$ derivation of the empty list, i.e. $\mathcal{D}'_n$ is a refutation.

4.6.3 Regularity and First-Order Completeness

The regularity restriction as it is usually defined for the non-restart versions (Section 3.3, "no literal occurs more than once in a branch") no longer holds for restart theory model elimination. This is rather easy to see since after a restart step it might be necessary to repeat – in parts – a refutation derived so far up to the restart step.

However the following observations allow the definition of a somewhat weaker notion of regularity. The proof of the ground completeness lemma (Lemma 4.6.1) proceeds by recursively splitting the input clause set M into Horn sets $M_1, \ldots, M_l$ and then assembles the respective PTME-I refutations $R_1, \ldots R_l$ into PRTME-I refutation R of M. Every branch in R is of the form

$[p_1 \cdot A_1 \cdots p_{n-1} \cdot A_{n-1} \cdot p_n]$ where p_i (for $i = 1, \ldots, n$) consists of negative literals and stems from some R_k ($k \in \{1, \ldots, l\}$) and $A_1, \ldots, A_{n-1}$ are the positive literals causing the restart steps. Let us call each p_i a *block*.

Since the assembling of the R_ks does not disrupt their structure, some properties of the R_ks carry over to their respective occurrence in R. In particular, the *regularity* of the R_ks (which can be demanded due to Theorem 4.5.3) carries over in this way. Hence we define a branch as *weakly blockwise regular* iff every pair of (different occurrences of) identical negative literals is separated by at least one positive literal. In adaption of Definition 3.3.2, a derivation is is called *weakly blockwise regular* iff every branch in every of its branch sets is weakly blockwise regular.

However, since the weak blockwise regularity restriction applies only to the negative literals within a block, we might still derive a branch of the form $p = \cdots A \cdots A \cdots$ which is weakly blockwise regular. We wish to extend weak blockwise regularity to forbid such duplicate occurrences of positive literals, and say that a branch is *positive regular* iff all the (different occurrences of) positive literals occurring in it are pairwise not identical. Extending the preceding definition, we define a branch to be *blockwise regular* iff it is both weakly blockwise regular and positive regular.

The completeness for the blockwise regularity restriction is preserved by the following line of reasoning. Again, we analyze the proof of Lemma 4.6.1: while for the induction start positive regularity is trivial, for the induction step the following observation can be used: consider the split sets $M_2, \ldots, M_n$ and the associated positive literals $A_2 \in M_2, \ldots, A_n \in M_n$. Clearly we can delete any clause of the form

$$C = A_i, B_1, \ldots, B_k \leftarrow C_1, \ldots, C_m \qquad \text{(where } k \geq 0\text{)}$$

from M_i ($i = 2, \ldots, n$), and use the resulting $\mathcal{T}$-unsatisfiable set M_i' instead of M_i. For this syntactical reason, a branch containing A_i cannot come up in the refutation R_i of M_i' (which can be supposed to be positive regular by the induction hypothesis). Thus, in the assembling of the R_is into the final refutation R, the occurrence of A_i which caused the restart step remains the *only* occurrence of A_i. In other words, positive regularity holds for this branch, and hence more generally also for the whole refutation R. We conclude with the first-order completeness. Again, as was exercised for TTME-MSR and PTME-I, it is formulated as an answer completeness result (cf. again Figure 5.1 in Section 5.1.1 for a toy example on answer computation.)

Theorem 4.6.1 (Answer Completeness of PRTME-I). *Let $\mathcal{T}$ be a universal theory, $\mathcal{I}_\mathcal{T}$ be a contra-definite inference system which is ground complete wrt. $\mathcal{T}$ for negative top literals (cf. Def. 4.5.6), c be a computation rule, f be a selection function, P be a program and $\leftarrow Q$ be a query (cf. Def. 3.2.5) such that Q consists of positive literals (i.e. $\leftarrow Q$ is a negative clause).*

Then, for every correct answer $\{Q\Phi_1, \ldots, Q\Phi_m\}$ for P and $\leftarrow Q$ a PRTME-$\mathcal{I}_\mathcal{T}$ refutation $\mathcal{D}^c$ of P and $\leftarrow Q$ via c and selection function f

exists, which is stable wrt. minimality, with computed answer $Answer(\mathcal{D}^c) = \{Q_1, \ldots, Q_l\}$, *and such that*

$$\{Q_1, \ldots, Q_l\}\delta \subseteq \{Q\Phi_1, \ldots, Q\Phi_m\} \quad \textit{for some substitution } \delta.$$

In [Baumgartner *et al.*, 1995] we obtained an answer-completeness result for the (non-theory) "ancestry restart" variant of model elimination. This variant has the nice property that for correct definite answers, i.e. the answer is a singleton, the computed answer is also a singleton. I conjecture that a respective variant with theory reasoning could also be defined and proven complete.

Proof. The proof is the same as for the answer completeness of PTME-I (Theorem 4.5.3) except that the ground completeness of PTME-I (Lemma A.1.13) used there is replaced by the ground completeness of PRTME-I (Lemma 4.6.1 above). Here, also the restriction to contra-definite inference systems and negative query clauses $\leftarrow Q$ comes in.

It should further be noted that the lifting lemma, Lemma A.1.9, has to be updated with the case of the new Restart inference rule. Since this is straightforward, it is omitted. Further, since the selection function is required to be stable under instantiation (Def. 4.6.2), we are guaranteed that the lifted derivation conforms to the choice of selected literals on the ground level. Also, the blockwise regularity of the ground level, which was argued for above, carries over to the first-order level (because any violation on the first-order level necessarily results in a violation at the ground level). Concerning the independence of the computation rule, the switching lemma, Lemma A.1.11, has to be updated by the Restart inference rule. Again, this is straightforward and hence omitted.

Recall that for PRTME-I we insist on negative query clauses $\leftarrow Q$. By definition of PRTME-I, negative clauses can only be used for restart steps. Hence, the computed answer in PRTME-I refutations is obtained by looking at the restart clauses alone. But notice that this does *not* imply that every (negative) clause used at a restart step contributes to the computed answer, because negative clauses might also be present in the program P.

Other Refinements.. It is obvious that for reasons of efficiency a proof procedure based on this calculus must provide some more refinements. Compared to PTME-I, the factorization inference rule (Section 3.3) is even more significant. This is due to the relaxed regularity restriction. In ordinary model elimination it is impossible due to regularity that along one branch the same instance of a clause is used more than once. This does not hold for restart model elimination, because only blockwise regularity can be demanded. However, the new subgoals introduced by the more leafmost occurrence of such a replicated clause can be closed immediately by factoring.

Another refinement is unique to restart model elimination. Restart model elimination is "partially proof confluent": call a tableau all of whose open

branches end in *positive* leaves as a *positive disjunctive tableau*. Let $D = \mathcal{P}_0 \vdash \cdots \vdash \mathcal{P}_n$ be a PRTME-I derivation of M which ends in a positive disjunctive tableau $\mathcal{P}_n$. Then, if M admits a refutation at all (i.e. if M is $\mathcal{T}$-unsatisfiable), then also D can be continued to a refutation. In other words, PRTME-I is proof confluent in such a situation. This is fairly easy to see, as the existing refutation, say R with start clause Q, can always be appended sufficiently often to D by restarting with the clause Q at the branches of $\mathcal{P}_n$.

For a backtracking-oriented proof procedure this means that no backtracking over positive disjunctive tableaux is necessary. Such a strategy is best supported by a computation rule which always selects a branch with a *negative* leaf literal, if present.

5. Linearizing Completion

Automated reasoning with Horn clause logic (or even definite clause logic) has been widely recognised as a practically important special case. Typically, logic programming and deductive databases languages are based on Horn clause logic. Consequently, much effort has been spent into the development of calculi and proof procedures for Horn clause logic.

Like e.g. SLD-Resolution [Apt and van Emden, 1982], the method presented in this chapter — *linearizing completion* — is intended as an efficient computation technique for Horn theories. The idea of linearizing completion is to compile a Horn theory into another Horn theory which allows more efficient proof search than the original Horn theory. More technically, with the resulting Horn theory, resolution derivations can be carried out which are both linear (in the sense of Prolog's SLD-resolution) *and* unit-resulting (i.e. the resolvents are unit clauses). This is not trivial since although both strategies alone are complete, their naïve combination is not. Completeness is recovered by the method through a certain "completion" of the Horn theory in the spirit of Knuth-Bendix completion, however with different ordering criteria. A powerful redundancy criterion quite often helps to find a finite system.

Linearizing completion can be used as a stand-alone computation technique for Horn theories. However, it was conceived and motivated within the context of theory reasoning. More precisely, the transformed theory can be used in combination with linear, goal-oriented calculi such as partial theory model elimination (Sections 4.5 and 4.6) to yield sound, complete and efficient calculi for full first order clause logic over the given Horn theory. For this, the clauses of the transformed Horn theory are read as inference rules for the computation of residues from key sets.

As an example application, the method discovers a generalization of the well-known linear paramodulation calculus for the combined theory of equality and strict orderings (Section 5.7.2). Another application is an efficient handling of properties of the reachability relation for modal logics (Section 5.7.3).

5.1 Introduction

Linearizing completion works by saturating a Horn clause set $\mathcal{T}$ under several deduction operations until only redundant consequences can be added. The resulting possibly infinite inference system (cf. Def. 4.5.1) $\mathcal{I}_\infty(\mathcal{T})$ enjoys the following completeness property: for every minimal $\mathcal{T}$-unsatisfiable literal set $\mathcal{L}$ and every literal $G \in \mathcal{L}$ there is a *linear* resolution refutation of $\mathcal{I}_\infty(\mathcal{T}) \cup \mathcal{L}$ with goal literal G, i.e. G is processed stepwise until the empty clause is derived, *and*, all inferences are done in a *unit-resulting* way, i.e. in an n-literal parent clause at least $n-1$ literals have to be simultaneously resolved against $n-1$ complementary unit clauses in order to carry out an inference.

Thus linearizing completion is a device for combining the unit-resulting strategy of resolution [McCharen *et al.*, 1976] with a linear strategy à la Prolog in a refutationally complete way (See e.g. [Stickel, 1986] for an overview of theorem proving strategies; it covers the linearity and the unit-resulting restriction). This is not trivial, since although each strategy *alone* is complete for Horn theories, their naive combination is not. Furthermore we insist on completeness for an *arbitrary* goal literals taken from the input set.

5.1.1 Linearizing Completion and Theory Reasoning

As mentioned above, the development of linearizing completion was initiated by the desire to automatically construct inference systems for *theory reasoning* calculi. Further, *answer completeness* is an issue. Currently, linearizing completion is tailored towards the use with *linear*, *goal-sensitive* calculi (in the sense of [Plaisted, 1994]) such as linear resolution [Loveland, 1970] or model elimination (cf. Chapters 3 and 4). The interest in linear calculi comes from their successful application in automated theorem proving (see e.g. [Baumgartner and Furbach, 1994c; Stickel, 1989; Stickel, 1990a; Letz *et al.*, 1992; Astrachan and Stickel, 1992] and Chapter 6 for descriptions of running systems.

In Section 4.5 (4.6), the answer-complete calculus *partial* (restart) theory model elimination was introduced, and the device of *inference rules* (Def. 4.5.1) was suggested as a means of describing the computation of residues. In Definition 4.5.6 we stated a sufficient completeness criterion for inference systems. However, how to come from a given theory to a complete inference system was left open. The purpose of the present chapter is to fill this gap.

As a motivating example take the diagnosis scenario in Figure 5.1. *Answer computation* is an issue here. Intentionally, the task is to find out which treatment T can be applied if symptom s is detected. The theory declares general (i.e. non-problem specific) knowledge about the possibility of substituting one treatment by another. The tableau at the right side indicates one PTME-I step, where the involved clauses and the returned residue are high-

lighted. Although the program is non-Horn, definite answers exist (namely, $t0$ and $t1$).

The question arises as to what extent linearizing completion is tailored towards application within linear, goal-sensitive calculi, such as theory model elimination. We have presently not carried out the combination with a completely different calculi, such as resolution or its ordered version [Baumgartner, 1992b]. However, the completeness result in Section 5.6 below allows us to compute most general $\mathcal{T}$-refuters (Def. 4.4.2) by completed inference systems, and this result can be the base for the combination with resolution. In this case the completeness demand for an *arbitrary* goal literal can be dropped (although it might be advisable for efficiency reasons to keep it) and the completion requirements can be relaxed a bit. This, however, is not touched by the present work.

5.1.2 Related Work

This section covers related work concerning linearizing completion. Related work on theory reasoning calculi in general was discussed in Sections 4.1 and 4.4.4.

One exception is the *link deletion* approach proposed by Klaus Mayr in [Mayr, 1995]. It relates to the *combination* "partial restart theory model elimination plus linearizing completion" and not just linearizing completion, proper.

As linearizing completion, link deletion is conceptually to be carried out in a preprocessing phase, and works for Horn theories. Link deletion does not take the viewpoint of theory reasoning. The idea is to avoid the exploration of certain connections in the proof space of model elimination. That is, although a connection between the leaf of the branch to be extended and an input clause exists, this extension step is *not* carried out. In order to be complete, it has to be shown that instead of carrying out such an extension step to, say, T and producing T' one can alway extend T towards a tableau T'' (by a different extension step) which can "substitute" T'. More precisely, it has to be shown that whenever T' could be extended to a closed tableau, then also T'' could be extended to a closed tableau. For this, restructuring of T' is considered such that T'' results. Further, restructuring must be well-founded (that is, T'' must be strictly smaller in some well-founded ordering).

There seems to be some overlap between link deletion and the approach presented here. For instance, link deletion when applied to transitive relations behaves much the same as if the axiom of transitivity would be processed by linearizing completion. An important difference to linearizing completion is that the creation of new inference rules is not an issue in link deletion. Link deletion behaves much like a restricted form of linearizing completion, where only the generation of contrapositives of inference rules is at our disposal (a precise meaning of this should become clear in Section 5.1.4).

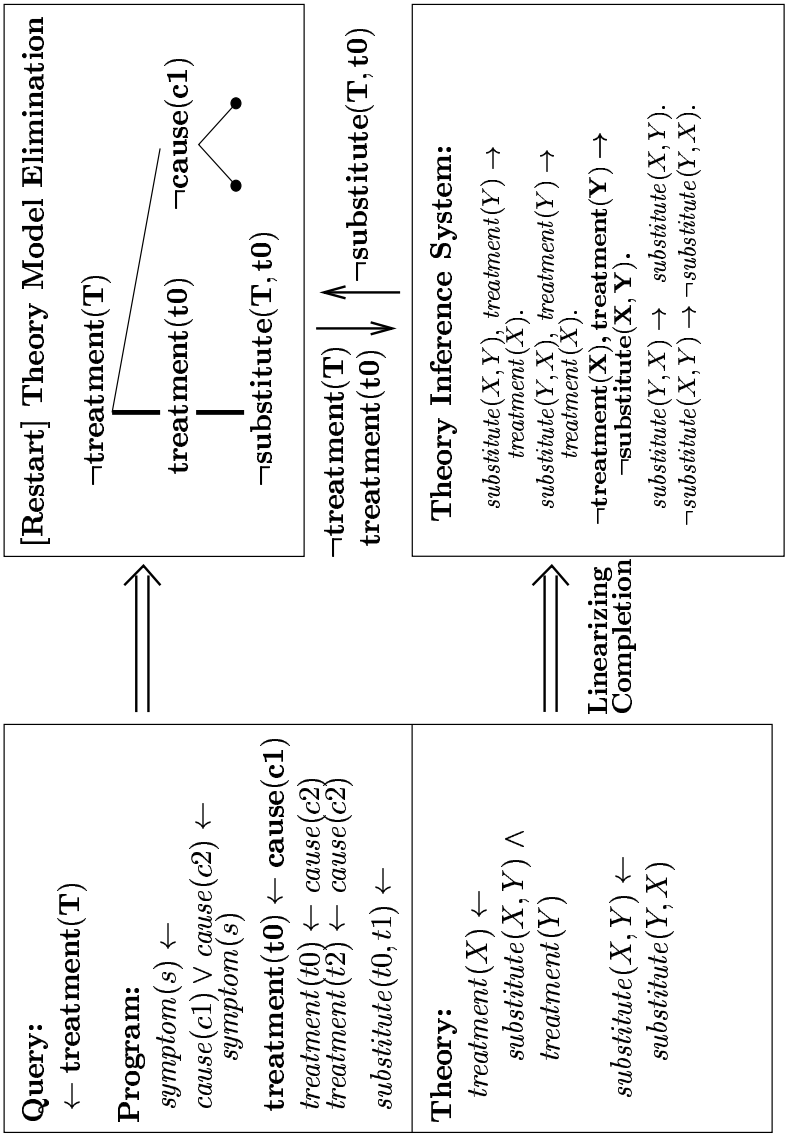


Figure 5.1. A problem-solving application of linearizing completion. The inference rules of the form $L, \neg L \rightarrow \mathsf{F}$ are not shown.

Next we turn to related work on theory reasoning, proper. In Section 4.1.2 we reviewed several systems for *dedicated* classes of theories. Linearizing completion applies to Horn theories and thus relates closest to the more general approaches mentioned under "compiled theories" there. These shall be commented on now.

Dixon's [Dixon, 1973] and Ohlbach's [Ohlbach, 1990] method is less general than linearizing completion, as it is restricted to two-literal clauses.

The method of Murray and Rosenthal [Murray and Rosenthal, 1987] is even more general than linearizing completion as the restriction to Horn theories is not necessary. As mentioned, they propose closing $\mathcal{T}$ under application of binary resolution modulo subsumption, yielding a possibly infinite set of clauses $\mathcal{T}^*$; $\mathcal{T}^*$ is used for total theory extension steps by simultaneously resolving away all literals from a clause from $\mathcal{T}^*$ against given literals in input clauses. The idea of closing the theory under resolution is similar to the linearizing completion process. As major differences we have that, first, linearity is not relevant in their context, and, second, the redundancy criterion of linearizing completion is stronger than subsumption. For instance, they would have to unfold a transitivity rule infinitely often by self-resolution, because none of the unfolded versions is subsumed. See also Section 5.1.5 below for a more detailed discussion of linearizing completion and resolution.

It seems that the related work closest to linearizing completion is the approach of the *special relation rules* [Manna and Waldinger, 1986; Manna *et al.*, 1991]. For a brief description see Section 4.1.2. The "special relation (SR) inference rules" derived by this method are embedded into a resolution calculus, much like our inference rules are embedded into model elimination (see Section 5.1.1).

The SR inference rules and the inference rules generated from linearizing completion (LC) compare on the common domain as follows: both are "unit-resulting" (i.e. all premise literals have to be resolved away simultaneously). The SR rules are fewer than the respective LC rules. For instance, no extra contrapositive is required for "transitivity" (cf. the inference system $\mathcal{I}_{\mathcal{SO}}$ in Example 4.5.1). This is not surprising as no completion takes place. Furthermore, the SR rules are more restrictive than the LC rules, since no replacement below the variable level occurs (cf. Section 5.7.1 for a discussion on that).

However, in this comparison it is important to note that the LC rules work in conjunction with a linear, and hence more restricted, calculus than the SR rules. This restriction has a large impact on the completion procedure.

Unlike the LC rules, the SR rules are incomplete [Manna *et al.*, 1991]. In the canonical counterexample it would help to replace subterms *below* the variable level, which is forbidden.

The authors of [Manna *et al.*, 1991] first speculated that the situation can be repaired by adding more inference rules, called *relation matching (RM) rules*. These rules generalize the well-known RUE-resolution inference rule

[Digricoli and Harrison, 1986] towards general monotonicity properties. Unlike the SR rules (and the LC rules), the RM rules are no longer unit-resulting, i.e. the conclusion might consist of more than one literal. Furthermore, it is now necessary to deductively close the RM rules (similar to those in linearizing completion). In order to arrive at a finite system, *variable elimination* rules are used to simplify resolvents. However, these rules are sound only when making very strong assumptions about the underlying relations; this restricts the method's applicability. Finally, as said in [Manna *et al.*, 1991], completeness is still open.

As an overall evaluation, it seems that the linearizing approach marks some progress towards solving open issues in the Manna, Stickel and Waldinger paper.

Other sources for related work are the completion techniques developed within the term-rewriting paradigm. In fact, linearizing completion was inspired by Knuth-Bendix completion [Knuth and Bendix, 1970] (cf. also Section 5.1.3 below) and its successors (e.g. [Hsiang and Rusinowitch, 1987; Bachmair *et al.*, 1986; Bachmair *et al.*, 1989; Bachmair, 1991]).

Knuth-Bendix completion has been generalized to conditional equational theories (e.g. [Kaplan, 1987; Dershowitz, 1990; Bachmair, 1991; Dershowitz, 1991a; Dershowitz, 1991b; Ganzinger, 1991]) i.e. definite clauses with built-in equality, and even to full first-order equational theories, e.g. [Bachmair and Ganzinger, 1990; Zhang and Kapur, 1988; Nieuwenhuis and Orejas, 1990; Nieuwenhuis and Rubio, 1992; Bronsard and Reddy, 1992].

These approaches allow for *equational* specifications, whereas we do not have a dedicated treatment for equations. There are several ways to translate Horn logic into equational logic. As a first method, at least for propositional logic, a *formula* F over usual logical connectives can be translated into an equation $F = \mathit{true}$ which then is processed by a term rewriting system for Boolean algebra [Hsiang, 1985; Paul, 1985; Paul, 1986]. It is even possible to model linear input strategies for Horn theories within specialized versions of extended (by associative-commutative operators) Knuth-Bendix completion (see e.g. [Dershowitz, 1985]). However, the unit-resulting restriction and the independence of the goal literal are not considered in those settings.

Another, straightforward, translation of Horn logic into equational clause logic results from simply reading a literal A as the equation $A = \mathit{true}$ (over a different signature). Note that since we allow purely negative clauses such as $\neg(x < x)$ in the specification to be completed, this translation requires full first-order clauses and not just definite clauses. It is common to methods operating on such equational clauses that they rely heavily on *term-orderings* for certain purposes: first, term-orderings are used to select — usually only maximal — literals inside clauses for inferences; second, restricted versions of paramodulation are directed in an order-decreasing way; finally, redundancy is typically defined employing term-orderings.

Here a major difference between these techniques and linearizing completed can be identified, as the *linearity* and the *unit-resulting* restrictions are not considered as a restriction for refutations in the completed theory. While linearizing completion insists on linearity of *derivations*, they consider, more "locally", derivations built from term-ordered *inferences*. Consequently, the notion of a "goal" literal is not a topic of interest.

An exception is [Bertling, 1990] which describes a procedure to complete towards a combination of term-ordering restrictions and the linear restriction. However, the unit-resulting restriction is not considered there.

As a further difference, for linearizing completion the above–mentioned independence of the selection of the goal literal requires the presence of *contrapositives* of the same clause, such as $(\text{Trans}-1)$ and $(\text{Trans}-2)$ in Example 4.5.1. In the term-rewriting paradigm this is not necessary.

All these observations indicate to us substantial differences between linearizing completion and the completion techniques described in the term-rewriting literature. It seems that none of these approaches can be instantiated in such a way that linearizing completion results. However, many standard notions and techniques developed in term-rewriting can be taken advantage of.

5.1.3 Relation to Knuth-Bendix completion

Like Knuth-Bendix completion (see [Bachmair, 1991] for a "modern" presentation), linearizing completion can be understood as a sort of compiler, which compiles a specification once and for all into an efficient algorithm. There are also some analogies in processing: Knuth-Bendix completion relates to equational theories and ordered derivations as linearizing completion relates to general Horn theories and linear derivations. Thus a different ordering criteria is used ("linearity" rather than "orderedness"), and, second, a more general viewpoint is proposed by treating arbitrary Horn theories, not just equality.

Technically, a Horn clause $\{\neg L_1, \ldots, \neg L_n, L_{n+1}\}$ is viewed as an *inference rule* $L_1, \ldots, L_n \to L_{n+1}$ which stands operationally for a unit-resulting inference "from $L_1, \ldots, L_n$ infer L_{n+1}". Linearizing completion proceeds by identifying sources for non-linearity in unit-resulting proofs[1] carried out with such inference rules. Non-linearities correspond to "peaks" in the term-rewriting paradigm. In analogy with these peaks, linearizing completion has to invent a new inference rule that repairs the situation. This is done by overlapping inference rules by the so-called Deduce transformation operation. Possible nontermination comes in by saturating the rule set under this (and similar) operations. However by the use of redundancy criteria termination is achieved quite often for practically relevant theories. If the completion does not terminate, we arrive at an "unfailing" procedure, i.e. for every provable

[1] More precisely: hyper resolution proofs.

goal eventually enough inference rules will be generated that are sufficient to prove the goal for such a system. Figure 5.2 summarizes the relationships between Knuth-Bendix completion and linearizing completion. Some of the concepts listed there will become clear as the text proceeds.

Concept	*Knuth-Bendix Completion*	*Linearizing Completion*
Underlying Theory	Equality	Arbitrary Horn Theory $\mathcal{T}$
Link Syntax-Semantics	Birkhoff's Theorem	Soundness and Completeness of Unit-Resulting Resolution
Proof Task	$s \stackrel{?}{=} t$	Is $\{K_1, \ldots, K_m\}$ $\mathcal{T}$-unsatisfiable?
Object-Level Inferences	by rewrite rules $u \rightarrow v$	by Inference Rules $L_1, \ldots, L_n \rightarrow L_{n+1}$
Non-normal Form Proof	$s \leftrightarrow^\star t$	Non-linear Unit-Resulting Proof of $\{K_1, \ldots, K_m\}$
Ordering Criteria	Orderedness	Linearity and Unit-Resulting Property
Removal of Peaks in Proofs	By critical pairs turned into rewrite rules	By Deduced inference rules
Normal form Proof	Valley Proof $s \rightarrow^\star u \ ^\star\leftarrow t$	Linear and Unit-Resulting Proof of $\{K_1, \ldots, K_m\}$
First-Order case	Narrowing Proofs	First-Order Derivations
Completeness	Yes: unfailing	Yes: unfailing

Figure 5.2. Summary of Relationships between Knuth-Bendix completion and linearizing completion.

5.1.4 Informal Description of the Method

Since the main part of this chapter is lengthy and quite technical, it might be advisable to supply a brief and informal description of the method beforehand. Readers familiar with term-rewriting will note that in order to express linearizing completion several notions from the term-rewriting paradigm have been adapted, like *completion*, *fairness*, *redundancy* and others (see e.g. [Bachmair *et al.*, 1986]).

As a first step in linearizing completion, a set of inference rules (cf. Definition 4.5.1) is obtained by re-writing a given Horn theory in an obvious way. Consider again the theory $\mathcal{SO}$ of strict orderings:

$$\mathcal{SO}: \quad \forall x, y, z \; x < y \wedge y < z \rightarrow x < z \qquad \text{(Trans)}$$
$$\forall x \qquad \neg(x < x) \; . \qquad \text{(Irref)}$$

From this theory we construct an *initial* inference system $\mathcal{I}_0(\mathcal{SO})$ that contains the following inference rules:

$$\mathcal{I}_0(\mathcal{SO}): \qquad x < y, \; y < z \rightarrow x < z \qquad \text{(Trans)}$$
$$x < x \rightarrow \mathsf{F} \qquad \text{(Irref)}$$
$$\neg(x < y), \; x < y \rightarrow \mathsf{F} \; . \qquad \text{(Syn)}$$

The (Trans) is obvious, the (Irref) shows how negative clauses are treated, namely by turning them into rules with conclusion "F", and the additional (Syn)-rule is used to treat syntactical inconsistencies. The formal definition is given in Section 5.2.1 (Def. 5.2.4).

This inference system is sufficient to refute the literal set $M = \{a < b, \; b < c, \; c < d, \; d < e, \; e < a\}$ of Example 4.5.3 on page 126. In fact, the background refutation there uses only the rules already present in $\mathcal{I}_0(\mathcal{SO})$. This background refutation is *linear* in the sense that the conclusion of an inference step is used in the premise part of the subsequent inference and that the rest of the premise literals is drawn from the input set M; further, it is *unit-resulting* by definition of "theory-inference". Later on we will also temporarily consider *non-linear* derivations, for which the rest of the premise literals – such as $b < c$ in the first inference – may also be computed in derivations themselves.

Unfortunately, the strategy of simply rewriting the theory as an initial inference system does not result in a system for which the combined unit-resulting and linear strategy is complete. To demonstrate this a different example is needed. Assume the theory consists of the clauses

$$S = \{\neg A \vee B, \; \neg C \vee D, \; \neg B \vee \neg D\}$$

The derived initial inference system then is as follows:

$$\mathcal{I}_0(S): \qquad A \rightarrow B \qquad\qquad A, \neg A \rightarrow \mathsf{F}$$
$$C \rightarrow D \qquad\qquad B, \neg B \rightarrow \mathsf{F}$$
$$B, D \rightarrow \mathsf{F} \qquad\qquad C, \neg C \rightarrow \mathsf{F}$$
$$D, \neg D \rightarrow \mathsf{F} \; .$$

Now let $M_1 = \{A, C\}$. There is, for instance, a (non-linear) Unit-Resulting refutation of $M_1 \cup S$ (Figure 5.1.4) where the Unit-Resulting inferences are carried out as suggested by the arrow-notation of the rules in $\mathcal{I}_0(S)$.

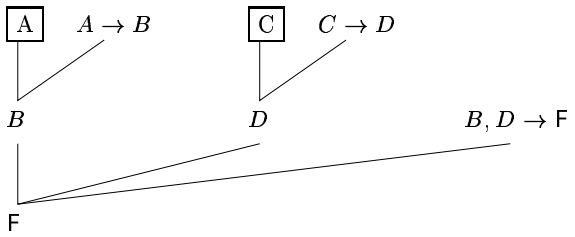


Figure 5.3. A non-linear unit-resulting refutation of $\{A, C\} \cup S$.

The inferences are carried out according to $\mathcal{I}_0(S)$. The given input literals — A and C — are boxed.

However, no *linear* refutation of M_1 with top literal A exists (neither does one exist with top literal C for reasons of symmetry); if A is given as the top literal then only the rules $A \to B$ and $A, \neg A \to \mathsf{F}$ contain the literal A in the premise and thus are the only candidates to be applied. However, the latter is not applicable as $\neg A$ is not given in M_1, and application of the former yields B which is also a dead end (because D is not contained in M_1 and thus $B, D \to \mathsf{F}$ is not applicable). However D could be derived from the input literal C by an application of the rule $C \to D$. But then, however, due to this auxiliary derivation the refutation would no longer be linear. The same argument holds for the case of C being chosen as top literal.

This problem is solved by linearizing completion by generating a new inference rule that implicitly contains the auxiliary derivation. This generation of new inference rules is the central operation in linearizing completion; it allows a new inference rule to be obtained from two present inference rules by unifying a rule's conclusion with a premise literal of another rule and forming a new rule from the collected premises and the conclusion of the other rule. The new rule is then joined to the present ones. Operations such as this one (and others) on inference systems are described by the device of *transformation rules*. Returning to the last example one can generate a new inference rule by application of the Deduce transformation rule in the following way:

$$\frac{\begin{array}{lll} C & \to & D \\ & & D, B \to \mathsf{F} \end{array}}{C, B \to \mathsf{F}}\ .$$

Using this new rule $C, B \to \mathsf{F}$ a linear refutation of M_1 can be found. Figure 5.4 depicts this refutation in the same notation as in Figure 5.1.4; in our preferred notation it reads as follows:

$$A \Rightarrow_{A \to B} B \overset{C}{\Rightarrow}_{B, C \to \mathsf{F}} \mathsf{F}$$

For a first order example of an application of Deduce consider again the system $\mathcal{I}_0\mathcal{SO}$ from above. Two copies of the (Trans) rule can be combined in

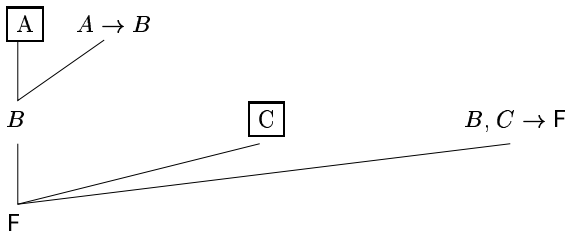


Figure 5.4. A linear unit-resulting refutation, using the new rule $B, C \to \mathsf{F}$.

the following way:

$$\frac{\begin{array}{llll} x < y, & y < z & \to & x < z \\ & & & x' < y' \end{array} \quad , \quad \begin{array}{lll} y' < z' & \to & x' < z' \end{array}}{\begin{array}{llll} x' < y, & y < y' & , & y' < z' \to x' < z' \end{array}}$$

where the unifier for $x < z$ and $x' < y'$ is $\{x \leftarrow x', z \leftarrow y'\}$. In words, the transitivity rule is unfolded once. Repeated application would yield infinitely many unfolded versions of the transitivity rule, but fortunately a redundancy criterion helps to find a finite system here.

This concludes the informal presentation of the Deduce transformation rule. Unfortunately, Deduce alone does not suffice to obtain completeness as desired. In order to demonstrate the problem here consider the slightly modified example from above:

$$\begin{aligned} S' &= S \cup \{D\} \\ M_2 &= M_1 \cup \{B\} \ . \end{aligned}$$

Of course the old clause $\neg C \vee D$ is not needed for the unsatisfiability of $S' \cup M_2$; but this is not important here. According to the transformation to initial inference systems defined above, the unit clause D becomes the rule $\neg D \to \mathsf{F}$. The initial inference system $\mathcal{I}_0(S')$ looks as follows:

$$\begin{array}{lrl} \mathcal{I}_0(S'): & A \to B & A, \neg A \to \mathsf{F} \\ & C \to D & B, \neg B \to \mathsf{F} \\ & \neg D \to \mathsf{F} & C, \neg C \to \mathsf{F} \\ & B, D \to \mathsf{F} & D, \neg D \to \mathsf{F} \ . \end{array}$$

Clearly, $S' \cup M_2$ is unsatisfiable, but there is no $\mathcal{I}_0(S')$-refutation of M_2. In particular, the rule $B, D \to \mathsf{F}$ cannot be applied since there is no *input* literal D. On the other hand the literal D is "hidden" in the theory and has been turned into a rule $\neg D \to \mathsf{F}$. In order to obtain completeness it is necessary to consider all such rules of the form $\neg D \to \mathsf{F}$, where D is a positive literal, as an operational substitute for the side literal D in an inference.

This could be done either dynamically, i.e. during the proof search, or else in the compilation phase. In order not to spend extra time during the proof search we have decided on the latter alternative. Similar to Deduce above, the necessary operations are carried out by transformation rules. In order to solve the example the Unit2 transformation rule is used, which works in this example as follows:

$$\frac{\begin{array}{rrcl} & \neg D & \to & \mathsf{F} \\ B, & D & \to & \mathsf{F} \end{array}}{\begin{array}{rrcl} & B & \to & \mathsf{F}\ . \end{array}}$$

Thus, one could say that, when the rules are read as their equivalent clauses, a *unit resolution step* has been carried out. This is what Unit2 does. There is a second form, Unit1, which is like Unit2 but for the case when the second involved inference rule contains only one literal in the premise. In both cases the use of a rule $\neg D \to \mathsf{F}$ instead of a side literal D in an inference is simulated. It is easy to see that $\{B\}$ now has a one-step refutation with the new rule $B \to \mathsf{F}$.

All these transformation rules — Deduce, Unit1 and Unit2 — yield, when applied properly, complete inference systems wrt. the desired linear and unit-resulting restrictions. This is the central result. If Deduce is omitted, then completeness wrt. the unit-resulting restriction alone results.

These results hold if the transformation rules are carried out in a *fair* way. Fairness means that no possible application is deferred infinitely long. But then the rule generation can be iterated and would result in infinite inference system quite often. For example, the presence of a rule for transitivity alone suffices for infiniteness (the transitivity rule will be unfolded without bound). In order to avoid this, additional transformation rules are needed for the deletion of inference rules. A powerful deletion rule is based on the concept of *redundancy*. Informally, an inference rule is to be redundant if its application in a derivation can be simulated by the other inference rules. As a sufficient and reasonably implementable condition we say that a rule $L_1, \ldots, L_n \to L_{n+1}$ is redundant in an inference system if a linear derivation of L_{n+1} exists from input set $L_1, \ldots, L_n$ with any top literal from $\{L_1, \ldots, L_n\}$. For example, the once unfolded version $x' < y$, $y < y'$, $y' < z' \to x' < z'$ of the transitivity rule can easily be shown as redundant with this criterion.

Linearizing completion proceeds by repeated fair application of generating and deleting transformation rules to the initial system. Generation can be further restricted: it is fair to generate only new inference rules from persisting rules, i.e. rules that are generated eventually and never deleted afterwards. However, *each* of the transformation rules Deduce, Unit1 and Unit2 have to be considered for the generation of new inference rules. Consequently, these transformation rules are labeled as *mandatory*.

The result of a (mandatory) generating transformation rule need not be added if it can be shown to be redundant. The result of this process is a

(possibly infinite) system that is closed under derivation of non-redundant inference rules. Such systems are called "completed", as they can be shown to be (refutationally) complete. Once a completed system is discovered, the notion of redundancy can be extended to cover more cases ("redundancy for derivations"). This is done formally by introducing the *NonRed* operator (Def. 5.3.4) on inference systems. Although "completedness" is destroyed then, refutational completeness is preserved.

There is one thing more to say about deletion: deletion of a redundant inference rule must enable a refutation which is *strictly smaller* wrt. some well-founded ordering on refutations than the refutation which uses the redundant inference rule (this does not mean that the substituted refutations are necessarily *shorter*). This property is crucial for the completion process because it guarantees that an inference system capable of proving a given proof task (provided it is provable at all, of course) will be reached after *finitely* many steps[2]. Concerning implementation, this means that we may approximate stepwise infinite inference systems by ever increasing finite systems until a system "large" enough for a concrete proof task is obtained.

Besides the mentioned *mandatory* transformation rules, there is also an *optional* one which allows the addition of a contrapositive of a given rule. It is called Contra. The Contra transformation rule allows, for instance, to derive from the (Trans) inference rule $x < y,\ y < z \rightarrow x < z$ the new rule $\neg(x < z),\ x < y \rightarrow \neg(y < z)$ by "swapping" the conclusion and some premise literal.

The Contra rule is most useful, because it allows to obtain a finite set of inference rules more often than it would be the case if it were not present. See Note 5.7.1 for an example.

5.1.5 Linearizing Completion and Resolution Variants

We conclude this informal presentation with a note on the relation to resolution variants.

Unit-Resulting Resolution. In traditional unit-resulting resolution [McCharen *et al.*, 1976] *every* literal from a $n+1$ literal clause $L_1 \vee \ldots \vee L_{n+1}$ can serve as the unit resolvent. If this is to be modeled as inference rules, all $n+1$ contrapositives ($i = 1 \ldots n+1$)

$$\overline{L}_1, \ldots, \overline{L}_{i-1}, \overline{L}_{i+1}, \ldots, \overline{L}_{n+1} \rightarrow L_i$$

have to be used. For example, the transitivity axiom for strict orderings results in the following three contrapositives:

[2] Thus the approach here of proving termination is similar to the approach of *proof orderings* [Bachmair *et al.*, 1986; Bachmair, 1987; Bachmair, 1991] in equational logic. Indeed we use a similar complexity measure, based on multiset orderings. However, we found it advantageous to extend the comparison of refutations by additionally considering (optional) *weights* assigned to the used inference rules.

$$\begin{array}{lrcl} 1. & x < y, \quad y < x & \rightarrow & x < z \\ 2. & \neg x < z, \quad y < x & \rightarrow & \neg x < y \\ 3. & x < y, \quad \neg x < z & \rightarrow & \neg y < x \ . \end{array}$$

However, there are cases where not all contrapositives are needed. For example, one of contrapositive 2 or 3 can safely be deleted without affecting completeness. Such restrictions are expressible in our more fine-grained framework, while they are not in traditional unit-resulting resolution framework. This motivated us not to use the traditional formalism but to define a new one.

Hyper Resolution. Hyper resolution [Robinson, 1965a] is a complete calculus for Horn clause logic. It implements a bottom-up evaluation by starting from the given positive unit clauses and deriving new unit clauses in a unit-resulting way. In our terminology, only the "natural" contrapositives, such as 1, in the last example are needed for this. However, hyper resolution alone does not suggest any completion procedure and yields inherently non-linear refutations. But hyper resolution refutations will serve as a starting point for the completeness proof of linearizing completion.

Binary Resolution. As mentioned above, the Unit1 and Unit2 rules are instances of *unit resolution*. Similarly, the Deduce rule works much like the traditional binary resolution inference rule (see e.g. [Chang and Lee, 1973]). In fact, short of implicit factoring due to set notation, it is merely a suggestive notation for it which seems appropriate for the purposes here. So the question might come up where linearizing completion is different from ordinary resolution.

First, ordinary resolution does not have a restriction to certain contrapositives, as just explained.

Second, every refutation in the linearizing completion paradigm can be simulated stepwise by an ordinary resolution refutation in the following way, but the other direction does not hold. Let $\mathcal{T}_0$ be the Horn clause theory subject to linearizing completion, M be a $\mathcal{T}_0$-unsatisfiable literal set, and $G_0 \in M$ be the desired top literal of the refutation.

Then a refutation in the linearizing completion framework can be written as a resolution refutation

$$\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n, G_0, G_1, \ldots, G_n, \mathsf{F} \ ,$$

where (1) the $\mathcal{T}_i$'s are obtained from the $\mathcal{T}_{i-1}$'s by application of binary resolution, corresponding to the transformation rules of linearizing completion, and (2) the G_j's are obtained from the G_{j-1}'s and literals from M by unit-resulting resolution, using a nucleus clause from $\mathcal{T}_n$. These unit-resulting steps could be simulated by sequences of ordinary resolution steps, of course.

It is apparent that this refutation is a highly structured one; evidently not every ordinary resolution refutation is structured in that way. Thus, stepwise simulation in the other direction does not hold.

Third, linearizing completion uses powerful redundancy criteria which are usually not applied in ordinary resolution.

Fourth (connected with three), linearizing completion functions as a compiler, which allows for careful analysis of the input clause set independent of the proof task to be given later. This means that $\mathcal{T}_n$ can be computed once and for all. This is not done to that extent in ordinary resolution.

Linked Inference Principle. The *linked inference principle* [Wos *et al.*, 1984; Veroff and Wos, 1992], or *linked resolution*, is related to linearizing completion. According to [Kunen, 1991], linked resolution is a general technique for doing several binary resolution steps at once, keeping the final clause, and discarding the intermediate clauses. This technique diminishes the number of clauses stored in the database. The most unlimited version of linked resolution is trivially complete, but is also not feasible to implement because of the large number of sequences of binary resolutions which must be examined at each stage of the deduction.

In [Kunen, 1991] a rather general completeness result was obtained. In this setup, *linking clauses* are written as $\phi \parallel \psi$ where ϕ and ψ are disjunctions of literals. Semantically, $\parallel$ stands for "or". Thus, by $\parallel$ a clause is devided in two parts. In *pure linking clauses* $\psi = \Box$ (the empty clause), and in *pure standard clauses* $\phi = \Box$. The following inference rules are needed:

1. linking (ground case): from *center clause* $\alpha_1 \vee \ldots \vee \alpha_n \parallel \psi$ and n pure standard clauses $\overline{\alpha_n} \vee \psi_i$ $(i = 1, \ldots, n)$ derive $\Box \parallel \psi \vee \psi_1 \vee \ldots \vee \psi_n$. This rule lifts as usual to the first order case by taking new variants.
2. left resolution: "standard" resolution among linking clauses using complementary literals both to the left of $\parallel$.
3. left factoring: "standard" factoring, analogously to 2.
4. right resolution: "standard" resolution among linking clauses using complementary literals both to the right of $\parallel$.
5. right factoring: "standard" factoring, analogously to 4.

A refutation is found if $\Box \parallel \Box$ is derived. As a pruning technique, tautologies can be deleted except "right tautologies" which contain A and $\neg A$ on the right of $\parallel$ (for some literal A).

The question arises whether linked resolution can simulate linearizing completion, or vice versa. In order to make linked resolution simulate linearizing completion for given Horn theory $\mathcal{T}$ one can write every clause $C \in \mathcal{T}$ as $C \parallel \Box$. For instance, the theory S and literals set $M_1 = \{A, C\}$ from above become

$$S^{\parallel} = \{\neg A \vee B \parallel \Box,\ \neg C \vee D \parallel \Box,\ \neg B \vee \neg D \parallel \Box\}$$

Now, application of the transformation rules of linearizing completion can be simulated by linked resolution in such a way that for every new derived inference rule there is an equivalent clause derivable by linked resolution: the Unit1, Unit2 and Deduce are nothing but standard resolution inferences,

and hence can be simulated by left resolution. The Contra transformation rule only reorders an inference rule, but as a clause it remains the same. The same holds when applying transformation steps to the inference rules $A, \neg A \to \mathsf{F}$ from $\mathcal{I}_0(\mathcal{T})$.

For instance, the new inference rule $C, B \to \mathsf{F}$ would be obtained as the clause $\neg C \vee \neg B \parallel \square$ by left resolution of $\neg C \vee D \parallel \square$ and $\neg B \vee \neg D \parallel \square$.

Now let a $\mathcal{T}$-unsatisfiable literal set $\mathcal{L}$ as given. It seems natural to write every literal $L \in \mathcal{L}$ as $\square \parallel L$ (writing it as $L \parallel \square$ would reduce linked resolution to ordinary binary resolution).

Now, the linear and unit-resulting refutation of $\mathcal{L}$ and the completed theory (cf. Def 4.5.6) would *not* be exactly simulated by linked resolution because *only one* link inference is necessary to obtain the empty clause $\square \parallel \square$. The price to be paid for this is a possible much larger completed theory (e.g. a transitivity clause $\neg(X < Y) \vee \neg(Y < Z) \vee (X < Z) \parallel \square$ would lead to nontermination). Thus, in this setting linked resolution is much like Murray and Rosenthal's method [Murray and Rosenthal, 1987], which was discussed above.

On the other side, since linked resolution is not restricted to Horn theories, linked resolution could possibly be used as a generalisation of linearizing completion towards the non-Horn case (this is future work).

An alternative approach: linked resolution has hyper resolution as an instance. To obtain this, clauses are divided such that the negative literals come the left of $\parallel$ and positive literals come to the right of $\parallel$. Writing a theory $\mathcal{T}$ in this way has a similar effect as using the initial inference system $\mathcal{I}_0(\mathcal{T})$. It was argued above that $\mathcal{I}_0\mathcal{T}$ does not suffice for the purpose of theory reasoning. Application of the transformation rules can in general not be simulated by linked resolution. For instance, from $A \to B$ and $B \to C$ one can Deduce $A \to C$, but there is no inference possible beween $\neg A \parallel B$ and $\neg B \parallel C$. This is not intended to argue that one approach is better than the other. Instead it shows us that the calculi are *different*.

In sum, although linked resolution and linearising completion have similar motivation and inference rules, there seems to be no perfect simulation of one approach using the other.

The rest of this chapter is organized as follows: after recalling some preliminaries in Section 5.1.6, we extend in Section 5.2 the definition of inference systems (Def. 4.5.1) towards non-linear derivations; it also contains the completeness of initial inference systems for non-linear refutations. As mentioned, deletion of redundant inference rules is tightly coupled with associated orderings of derivations. Section 5.3 describes *orderings and redundancy*. Then we are prepared for the *transformation systems* of Section 5.4. This section introduces the related important notions of *limit* and *fairness* of a deduction, and also that of a *completed* inference system. There it will be shown that the defined transformation rules and redundancy criterion preserve refutational completeness. In Section 5.5 we build on Section 5.4 and show that the trans-

formation systems have the *complexity-reducing* property, which means that eventually a normal-form refutation will be reached. In Section 5.7 we will apply the material developed so far to two non-trivial examples (one is equality extended by strict orderings, and the other is concerned with modal logic). Then, in Section 5.6 the material developed so far will be assembled into various *completeness results*; notably, first-order results are also contained.

5.1.6 Preliminaries

Concerning *multisets* we refer to Section 2.1. If a *set* is used below where a *multi*set is required, then the type conversion is done in the obvious way.

Furthermore we make heavy use of the data structure 'sequence'. If in the computations below a sequence appears where a multiset is required, the transformation from sequences to multisets is done in the obvious way.

We are dealing with clause logic and Herbrand interpretations. See Section 2.4 for the definition of clause logic; concerning the semantics of clause logic we recall Lemma 2.4.1. In particular, Lemma 2.4.1.4 justifies the interest in Herbrand-interpretations.

A *Horn theory* $\mathcal{T}$ is a satisfiable set of Horn clauses[3]. For the model-theoretic notions (satisfiability etc.) see Definition 2.5.6.

5.2 Inference Systems

Inference systems play the same role as *sets of rewrite rules* in Knuth-Bendix completion: they are used for inferences on the object-level. Inference systems are the objects of computation by *transformation systems* which are introduced in the next section.

We will slightly generalize the previous definition of inference rules (Def. 4.5.1) in that inference rules may be labeled by *weights*. Further, the notion of a *background refutation* (Def. 4.5.5) will be generalized towards non-linear refutations.

Definition 5.2.1 (Theory Inference System). *A* theory inference rule with weight *over a given signature* Σ^{F} *(Σ^{F} is defined in Section 4.5.1) is a triple* $P \to_w C$*, where* P *is a nonempty multiset of* Σ*-literals,* w *is a non-negative integer (called* weight*) and* C *is either a nonempty set of* Σ*-literals or the singleton* $\{\mathsf{F}\}$*. Quite often, we will drop the weight index if not relevant in the context. The short forms* inference rule *or simply* rule *will be used as well.*

The notions premise, conclusion, (ground) instance, (theory) inference system, *as well as the notational conventions are adapted from Def. 4.5.1*

[3] To be precise, we would have to speak of "theories which are axiomatized by a finite set of Horn clauses"; however, for simplicity we will prefer the shorter expression. See also Section 2.5.

accordingly. Unless otherwise noted, in this chapter always theory inference rules with weight are considered.

Obviously, an inference rule in the old sense is obtained by forgetting about the weight. Weights are motivated by extended redundancy checks (cf. Example 5.3.2 below).

Definition 5.2.2 (Matching Theory Inference Step). *We say that a literal C' is* inferred *from a literal multiset P' by means of a* matching theory inference step with inference rule $P \to_w C$ and substitution δ, *written as*

$$P' \Rightarrow_{P \to_w C,\, \delta} C'$$

iff $P' = P\delta$ and $C' = C\delta$.

The notions of premise, conclusion, ground inference, used inference rule, *as well as the abbreviated notation are taken from Definition 4.5.2.*

Unless otherwise noted, in this chapter the term "inference step" always refers to matching theory inference steps, but not to the minimal first-order theory inferences of Definition 4.5.2.

This definition differs from the minimal first-order theory inferences of Definition 4.5.2 in two respects: firstly, instead of a unifier now matching substitution is used. Matching based inferences will be used for the major part of this chapter. Secondly, the premise minimality and the attached notion of amplification is dropped. This makes the proof in this chapter technically simpler. The lifting to the first-order level, as well as the premise minimality will be recovered below in Section 5.6.

Next we are going to inductively define *$\mathcal{I}$-derivations* based on this notion of inference:

Definition 5.2.3 ($\mathcal{I}$-derivation). *Let M be a finite set of literals, also called* input literals *and $\mathcal{I}$ be a not necessarily finite inference system.*

1. *The literal L_1 (not necessarily from M) is an $\mathcal{I}$-derivation of L_1 from M with top literal L_1 and length* 1. *Such a derivation is also called* trivial.
2. *If*
   a) *D_n is an $\mathcal{I}$-derivation of L_n from input literals M with top literal L_1 and length n, and*
   b) *$D_n^1, \ldots, D_n^{m_n}$ ($m_n \geq 0$) are $\mathcal{I}$-derivations (called* side derivations *in this context) of* side literals *$L_n^1, \ldots, L_n^{m_n}$, respectively, from M, all respective top literals are contained in M, and*
   c) *$P_n \to_{w_n} C_n$ is an inference rule from $\mathcal{I}$, and*
   d) *δ_n is a substitution, such that*

$$L_n, L_n^1, \ldots, L_n^{m_n} \Rightarrow_{P_n \to_{w_n} C_n, \delta_n} L_{n+1}$$

then

$$D_{n+1} = (D_n \xRightarrow{D_n^1 \cdots D_n^{m_n}} {}_{P_n \to_{w_n} C_n, \delta_n} L_{n+1})$$

is an $\mathcal{I}$-derivation of L_{n+1} from input literals M with top literal L_1 and length $n+1$.
If context allows, the involved inference rule and/or the substitution δ_n shall be omitted.

The symbol ε denotes the empty sequence of $\mathcal{I}$-derivations. Often we use the term derivation *instead of* $\mathcal{I}$-derivation.

Note that *derivations* are nothing but syntactically sugared terms over a respective signature, and thus can be subject to structural induction. In non-trivial derivations the principal (4-ary) construct symbol is "$\Rightarrow$", which takes as arguments the derivation D_n derived so far, the sequence $D_n^1 \cdots D_n^{m_n}$ of side derivations, the inference rule and the substitution involved. We will often omit parenthesis and write derivations like

$$D_{n+1} = (L_1 \xRightarrow{D_1^1 \cdots D_1^{m_1}} {}_{P_1 \to_{w_1} C_1, \delta_1} L_2 \cdots L_n \xRightarrow{D_n^1 \cdots D_n^{m_n}} {}_{P_n \to_{w_n} C_n, \delta_n} L_{n+1})$$

Some more terminology is convenient: if D is a derivation of L we also say that L is the *derived* literal of D; the derivation D is called *ground* iff every of its inferences is ground; D is called a *refutation* iff D is a derivation of F. An inference rule is said to be *used* in D iff (some instance of) it is used in some inference in D. D is called *linear* iff every side derivation occurring in it is a trivial derivation. Otherwise D is called *non-linear*. We will write

$$L_1 \Rightarrow^*_{\mathcal{I},M} L_n$$

to denote the fact that a $\mathcal{I}$-derivation of L_n from M with top literal L_1 exists; similarly the notation

$$D = (L_1 \Rightarrow^*_{\mathcal{I},M} L_n)$$

means that D is such a derivation.

Note that a derivation does not instantiate the input literals. This is the same as in a "rewriting" proof in the term rewriting paradigm. Carrying on this analogy, a derivation relates to a first-order derivation (to be defined in Section 5.6) much like "rewriting" relates to "narrowing" [Hullot, 1980].

The top literal plays the rôle of a goal to be proved. We are interested in arbitrary goal literals, not just negative literals as usually defined for SLD-resolution. Positive goal literals arise naturally: think e.g. of an inference system for strict orderings, containing a rule $X < X \to \mathsf{F}$ for irreflexivity. A provable query is then for example (in Prolog notation) $?- \neg(a < a)$.

5.2.1 Initial Inference Systems

As the first step of linearizing completion a given Horn theory $\mathcal{T}$ is re-written in a straightforward way as a set of inference rules:

Definition 5.2.4 (Initial Inference Systems). *Let $\mathcal{T}$ be Horn theory and let $W \subset \mathbb{N}$ be a finite set of weights. The* initial inference system of $\mathcal{T}$, $\mathcal{I}_0(\mathcal{T})$, *is the inference system consisting of the rules*

1. $\neg A \to \mathsf{F}$ *for every positive unit clause $A \in \mathcal{T}$, and*
2. $A_1, \ldots, A_k \to \mathsf{F}$ *for every purely negative Horn clause $\neg A_1 \vee \ldots \vee \neg A_k \in \mathcal{T}$ $(k \geq 1)$, and*
3. $A_1, \ldots, A_k \to B$ *for every definite clause $\neg A_1 \vee \ldots \vee \neg A_k \vee B \in \mathcal{T}$ $(k \geq 1)$, and*
4. $Q(x_1, \ldots, x_n), \neg Q(x_1, \ldots, x_n) \to \mathsf{F}$ *for every n-ary predicate symbol Q in $\mathcal{T}$.*

Furthermore, some arbitrary chosen weight $w \in W$ is attached to every rule in $\mathcal{I}_0(\mathcal{T})$. Note that with the theory being satisfiable, the empty clause is not contained in $\mathcal{T}$. Thus for every clause in $\mathcal{T}$ exactly one case applies.

Example 5.2.1. Let $\mathcal{T} = \{\neg A \vee B,\ \neg C \vee D,\ \neg B \vee \neg D, D\}$ be a ground Horn theory (it was also given as S' in the introduction). Then an initial inference $\mathcal{I}_0(\mathcal{T})$ system is

$$
\begin{array}{lrl}
\mathcal{I}_0(\mathcal{T}) & A \to_5 B & A, \neg A \to_1 \mathsf{F} \\
& C \to_4 D & B, \neg B \to_1 \mathsf{F} \\
& B, D \to_3 \mathsf{F} & C, \neg C \to_1 \mathsf{F} \\
& \neg D \to_6 \mathsf{F} & D, \neg D \to_1 \mathsf{F}
\end{array}
$$

Next let $M_1 = \{A, C\}$. This is a derivation of F from M_1 with top literal A (weights are omitted, as not needed here):

$$D_1 = (A \Rightarrow_{A \to B} B \overset{C \Rightarrow_{C \to D} D}{\Longrightarrow}_{B, D \to \mathsf{F}} \mathsf{F})$$

D_1 is non-linear, since the second inference step violates linearity. Figure 5.1.4 in the introduction depicts the same derivation in an alternative notation which emphasizes the tree character of derivations.

Example 5.2.2. As a non-trivial example consider the joint theory $\mathcal{ES}$ of equality and strict orderings (Figure 5.5). The predicate symbol $=$ is interpreted as an equivalence relation and the predicate symbol $<$ is interpreted as a strict ordering (i.e. as a transitive and irreflexive relation). Furthermore we have included a single function symbol f of arity 1, which gives rise to the substitution axiom $\forall x\, x = x' \to f(x) = f(x')$. The extension to a richer signature is straightforward.

The corresponding initial system is given in Figure 5.6.

Equivalence:		*Strict Order:*	
$\rightarrow x = x$	(Ref=)	$x < x \rightarrow$	(IRef<)
$x = y \rightarrow y = x$	(Sym=)	$x < y,\ y < z \rightarrow x < z$	(Trans<)
$x = y,\ y = z \rightarrow x = z$	(Trans=)		

f-Substitution:		*<-Substitution:*	
$x = x' \rightarrow f(x) = f(x')$	(Sub$f$)	$x = x',\ x < y \rightarrow x' < y$	(Sub<-1)
		$y = y',\ x < y \rightarrow x < y'$	(Sub<-2)

Figure 5.5. The theory $\mathcal{ES}$ of equality and strict orderings with unary function symbol f. Here, and below the usual "$\rightarrow$" notation is used for clauses.

Equivalence:		*Strict Order:*	
$x = y,\ \neg(x = y) \rightarrow \mathsf{F}$	(Syn=)	$x < y,\ \neg(x < y) \rightarrow \mathsf{F}$	(Syn<)
$\neg(x = x) \rightarrow \mathsf{F}$	(Ref=)	$x < x \rightarrow$	(IRef<)
$x = y \rightarrow y = x$	(Sym=-1)	$x < y,\ y < z \rightarrow x < z$	(Trans<)
$x = y,\ y = z \rightarrow x = z$	(Trans=)		

f-Substitution:		*<-Substitution:*	
$x = x' \rightarrow f(x) = f(x')$	(Sub$f$)	$x = x',\ x < y \rightarrow x' < y$	(Sub<-1)
		$y = y',\ x < y \rightarrow x < y'$	(Sub<-2)

Figure 5.6. The inference system $\mathcal{I}_0(\mathcal{ES})$.

5.2.2 Completeness of Initial Inference Systems

The completeness result (wrt. unit-resulting refutations) of initial inference systems is needed as a starting point for the proof of the completeness result (wrt. unit-resulting and linear refutations) of the generated inference systems. More precisely, the following line of reasoning applies: given a $\mathcal{T}$-unsatisfiable literal set, by ground completeness a — possible nonlinear — refutation exists in the initial inference system. By repeated generation of new inference rules in a fair way (see the introduction) eventually an inference system is generated that admits a linear refutation. Furthermore, deletion of redundant inference rules does not destroy linearity.

Thus, ground completeness is the initial link between semantics and syntax and plays much the same role as Birkhoff's completeness theorem (see [Huet and Oppen, 1980]) in Knuth-Bendix completion.

As explained in the informal presentation in the introduction, initial inference systems constitute an "almost complete" calculus for the underlying theory (cf. the example of the clause set S' there again). As was previously stated, in order to obtain completeness it is necessary to access all positive literals, no matter whether they are contained in the input set or contained in the theory. Regarding this, we will slightly differ from the introduction in that we will not immediately compile away the "hidden" positive literals A into a rule $\neg A \to \mathsf{F}$. Instead it is more convenient for us to define for an inference system $\mathcal{I}$ the set

$$\mathit{Punit}(\mathcal{I}) = \{A \mid \neg A \to \mathsf{F} \in \mathcal{I}\}$$

as the set of *positive unit clauses from* $\mathcal{I}$. For example, $\mathit{Punit}(\mathcal{I}_0(\mathcal{ES})) = \{x = x\}$, and in Example 5.2.1 we find $\mathit{Punit}(\mathcal{I}_0(\mathcal{T})) = \{D\}$.

The set $\mathit{Punit}(\mathcal{I})$ shall temporarily be accessible for derivations as additional input literals (later their usage will be compiled away by respective transformation rules). But then it is clear that the present formalism is just a reformulation of traditional Unit-Resulting resolution (which is biased towards our completion application). Taking the soundness and completeness of traditional Unit-Resulting resolution for granted, it is thus not surprising that respective results can be obtained for our formulation. Nevertheless we will state the precise ground completeness result here, since it will be needed below.

Lemma 5.2.1 (Ground completeness of $\mathcal{I}_0(\mathcal{T})$). *Let $\mathcal{T}$ be a ground theory (i.e. a theory consisting of ground clauses only) and M be a set of ground literals. If M is $\mathcal{T}$-unsatisfiable then $L \Rightarrow^*_{\mathcal{I}_0(\mathcal{T}), M \cup \mathit{Punit}(\mathcal{I}_0(\mathcal{T}))} \mathsf{F}$ for some $L \in (M \cup \mathit{Punit}(\mathcal{I}_0(\mathcal{T})))$.*

For instance, it is easily verified that in Example 5.2.1 a $\mathcal{I}_0(\mathcal{T})$-refutation of $\{A\} \cup \mathit{Punit}(\mathcal{I}_0(\mathcal{T})) = \{A, D\}$ exists, but a $\mathcal{I}_0(\mathcal{T})$-refutation of $\{A\}$ alone does not exist.

Proof. By definition of $\mathcal{T}$-unsatisfiability, M is $\mathcal{T}$-unsatisfiable iff $M \cup \mathcal{T}$ is unsatisfiable, where M is considered as a set of unit clauses. Since the unit-clauses of $\mathcal{T}$ can be used as input literals (via $Punit(\mathcal{I}_0(\mathcal{T}))$) an existing Hyper-resolution refutation of $M \cup \mathcal{T}$ can be reflected in our framework. For an explicit proof from scratch see Appendix A, SectionA.2.1[4].

We cannot be content with this completeness result because (1) the literals from $Punit(\mathcal{I}_0(\mathcal{T}))$ are needed, and (2) a linear derivation may not always exist. For instance, there is no linear refutation of $M_1 = \{A, C\}$ in Example 5.2.1, either with top literal A or with top literal C.

5.2.3 Subderivations

We will have to access and replace *subderivations* within derivations. For this let D be a $\mathcal{I}$-derivation[5]

$$D = (L_1 \overset{D_1}{\Longrightarrow} L_2 \cdots L_n \overset{D_n}{\Longrightarrow} L_{n+1})$$

It is apparent that

$$L_i \overset{D_i}{\Longrightarrow} L_{i+1} \cdots L_{j-1} \overset{D_{j-1}}{\Longrightarrow} L_j$$

for $1 \leq i \leq j \leq n+1$ is a $\mathcal{I}$-derivation of L_j with top literal L_i, called an *immediate subderivation of* D; it is denoted by $D|_{i,j}$. Immediate subderivations with $j = i + 1$ are also called *derivation steps*.

Derivations may be concatenated. If E is a $\mathcal{J}$-derivation of the form

$$E = (K_1 \overset{E_1}{\Longrightarrow} K_2 \cdots K_m \overset{E_m}{\Longrightarrow} K_{m+1})$$

and $L_{n+1} = K_1$ then the *concatenation of* D *and* E, denoted by $D \cdot E$, is defined as

$$D \cdot E = (L_1 \overset{D_1}{\Longrightarrow} L_2 \cdots L_n \overset{D_n}{\Longrightarrow} K_1 \overset{E_1}{\Longrightarrow} K_2 \cdots K_m \overset{E_m}{\Longrightarrow} K_{m+1})$$

Evidently, $D \cdot E$ exists as $\mathcal{I} \cup \mathcal{J}$-derivation from $M \cup N$, where D (resp. E) is a derivation from M (resp. N). It is easily verified that this concatenation is associative. Hence we can omit parenthesis when writing expressions as $D \cdot E \cdot F$.

In order to recursively access subderivations we need the *positions* of derivations. A *position* is a finite string over nonnegative integers and is written in Dewey decimal notation. The empty string is denoted by λ. The *set of positions of* D, $P(D)$ is the smallest set satisfying:

1. $\lambda \in P(D)$.

[4] In the sequel the proofs can often be found in the appendix.

[5] Inference rules and substitutions being omitted for brevity; the D_is stand for *sequences* of derivations.

2. $i.j.r \in P(D)$ if
   a) D is of the form $L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots L_i \overset{D_i}{\Longrightarrow} L_{i+1} \ldots L_n \overset{D_n}{\Longrightarrow} L_{n+1}$ where $1 \leq i \leq n$,
   b) and D_i is a sequence of the form $D_i^1 \cdots D_i^{j-1} D_i^j D_i^{j+1} \cdots D_i^{k_i}$ where $1 \leq j \leq k_i$,
   c) and $r \in P(D_i^j)$

Building on positions, subderivations can be introduced: let D be a derivation and $p \in P(D)$. The *(occurrence of a) subderivation of D at position p,* $D|_p$ is defined recursively as follows:

$$D|_p = \begin{cases} D & \text{if } p = \lambda \\ D_i^j|_r & \text{if } p = i.j.r \text{ and } D \text{ is of the form} \\ & \quad L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots L_i \overset{D_i}{\Longrightarrow} L_{i+1} \ldots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} L_n \\ & \quad \text{where } D_i = D_i^1 \cdots D_i^j \cdots D_i^{k_i} \end{cases}$$

For instance, take

$$X = (A \overset{B \overset{C}{\Longrightarrow} D \overset{E}{\Longrightarrow} F \quad G \Longrightarrow H}{\Longrightarrow} I)$$

then $X|_{1.1} = (B \overset{C}{\Longrightarrow} D \overset{E}{\Longrightarrow} F)$, $X|_{1.2} = (G \Longrightarrow H)$ and $X|_{1.1.2.1} = E$.

For ease of notation we abbreviate the selection of an immediate subderivation $(D|_p)|_{a,b}$ of a subderivation $D|_p$ to $D|_{p,a,b}$. In words, first p is used to locate a subderivation in D and then this subderivation is returned in between the indices a and b. For instance $X|_{1.1,2,3} = (D \overset{E}{\Longrightarrow} F)$ and $X|_{1.2,1,1} = G$.

Next we turn to replacement of derivations. Suppose D is an $\mathcal{I}$-derivation from M, and suppose E is an $\mathcal{J}$-derivation from N. Furthermore suppose that $D|_{p,a,b}$ and E have the same top literal and derived literal, respectively. Then the sequence which results from replacing $D|_{p,a,b}$ in D by E, written as $D[E]_{p,a,b}$, evidently is an $\mathcal{I} \cup \mathcal{J}$-derivation from $M \cup N$ with same top and derived literal as in D.

For instance,

$$X[D \overset{L}{\Longrightarrow} M \overset{N}{\Longrightarrow} F]_{1.1,2,3} = (A \overset{B \overset{C}{\Longrightarrow} D \overset{L}{\Longrightarrow} M \overset{N}{\Longrightarrow} F \quad G \Longrightarrow H}{\Longrightarrow} I)$$

More formally, replacement is defined as follows:

$$D[E]_{p,a,b} = \begin{cases} D|_{1,a} \cdot E \cdot D|_{b,n}, & \text{if } p = \lambda \\ D|_{1,i} \cdot (L_i \overset{D'}{\Longrightarrow}_{P_i \to C_i, \delta_i} L_{i+1}) \cdot D|_{i+1,n} & \text{if } p = i.j.r \\ \text{where } D' = D_i^1 \cdots D_i^{j-1} D_i^j[E]_r D_i^{j+1} \cdots D_i^{k_i} \end{cases}$$

5.3 Orderings and Redundancy

The transformation systems defined below allow for the deletion of *redundant* inference rules. Redundancy in turn is based on orderings for derivations. Hence these notions have to be introduced first.

5.3.1 Orderings on Nested Multisets with Weight

Section 2.1 contains standard definitions concerning orderings. Here we are concerned with a certain kind of multisets and orderings on them.

A *multiset with weight over a set* X is a tuple $\langle N, w\rangle$, also written as N_w (or $\{\!|\ldots|\!\}_w$) where N is a multiset over X and $w \in \mathbb{N}$. Thus, a multiset with weight is obtained from an ordinary multiset by attaching some integer to it. As a recursive generalization, define a *nested multiset with weight over a set* X as either an element from X or else as a multiset with weight over a nested multiset with weights. For instance, $\{\!|a, \{\!|b, c, \{\!|d|\!\}_3|\!\}_4, e|\!\}_2$ is such a set over $\{a, b, c, d, e\}$. They will be used as a complexity measure for proofs below.

Since multisets with weight are tuples, they can be compared lexicographically. The resulting ordering $\gg_{MW,\succ}$ over multisets with weight over a set X of terms, base ordering $\succ$ on terms and the usual ordering $>$ on integers then reads as follows:

$$M_v \gg_{MW,\succ} N_w \quad = \quad (M \gg_{\succ} N \text{ or else } (M = N \text{ and } v > w))$$

where M_v, N_w are multisets with weights over X. Thus we first compare the set-components; if these are equal then the weight gives the decision. It is clear that with the well-founded orderings $\gg_{\succ}$ and $>$ the lexicographic extension $\gg_{MW,\succ}$ also is well-founded.

Orderings on multisets can be generalized to *nested multisets*[Dershowitz and Manna, 1979]. For our purposes we have to go a little further and to compare *nested multisets with weight over a set of constants* C. This ordering is nothing but a recursive path ordering with status (see e.g. [Steinbach, 1990]) where the multiset constructor "$\{\!|.|\!\}$" is given a multiset status, and the tuple-constructor $\langle N, w\rangle$ (to attach weights to multisets) is given a left-right status. The proof of proposition 5.3.1 below will make this more precise.

Definition 5.3.1 (Nested Multisets with Weight). *Let C be a finite set of constants, ordered by the well-founded ordering $\succ$, and let $W \subset \mathbb{N}$ be a finite set of weights. Define the* nested multiset ordering with weights, *$\gg_{NMW,\succ}$, recursively as follows:*

1. $X = \{\!\{x_1, \ldots, x_m\}\!\}_v \succ_{NMW,\succ} \{\!\{y_1, \ldots, y_n\}\!\}_w = Y$
   if (1.1) $x_i \succeq_{NMW,\succ} Y$ *for some* $i = 1 \ldots m$,
   or (1.2) $X \succ_{MW,\succ_{NMW,\succ}} Y$ *and* $v, w \in W$,
   (i.e. $\succ_{MW,\succ_{NMW,\succ}}$ *is the recursive extension of* $\succ_{NMW,\succ}$ *to multisets with weight).*
2. $X = \{\!\{x_1, \ldots, x_m\}\!\}_v \succ_{NMW,\succ} y$
   if $y \in C$ *and* $x_i \succeq_{NMW,\succ} y$ *for some* $i = 1 \ldots m$

3. $x \succ_{NMW,\succ} y$
   if $x, y \in C$ *and* $x \succ y$.

Most often we will write $\succ_{NMW}$ *instead of* $\succ_{NMW,\succ}$ *when* $\succ$ *is clear from the context.*

Example 5.3.1. Let $C = \{a, b\}$ with $a \succ b$, and $W = \{0, \ldots, 10\}$. It holds

$$\begin{array}{ll} & \{\!\{a, \{\!\{a, \{\!\{a\}\!\}_4\}\!\}_3\}\!\}_3 \\ \succ_{NMW,\succ} & \{\!\{b, \{\!\{a, \ a \ \}\!\}_4\}\!\}_6 \ . \end{array}$$

This is, because with $a \succ_{NMW,\succ} b$ (by 3.) and

$$\{\!\{a, \{\!\{a\}\!\}_4\}\!\}_3 \succ_{NMW,\succ} \{\!\{a, a\}\!\}_4 \tag{5.1}$$

case 1.2. applies (we replaced in a multiset with weight two elements by smaller ones, hence increasing the weight from 3 to 6 does not matter). We still have to show Equation 5.1. It holds $\{\!\{a\}\!\}_4 \succ_{NMW,\succ} a$ by 2., and hence for the multiset extension,

$$\{\!\{a, \{\!\{a\}\!\}_4\}\!\} \succ_{\succ_{NMW,\succ}} \{\!\{a, a\}\!\}$$

and thus, after attaching the weights, also

$$\{\!\{a, \{\!\{a\}\!\}_4\}\!\}_3 \succ_{MW,\succ_{NMW,\succ}} \{\!\{a, a\}\!\}_6 \ ,$$

which is by 1.2. the same as Equation 5.1.

Fortunately, the ordering $\succ_{NMW}$ is monotonic and well-founded; this holds by Theorem 2.1.1 and the following property:

Proposition 5.3.1. *The ordering* $\succ_{NMW}$ *is a simplification ordering.*

Proof. Nested multisets with weights to be compared by $\succ_{NMW}$ are terms built from the variadic constructor symbol $\{\!\{.\}\!\}$, the 2-ary sequence constructor $\langle ., . \rangle$ and a set C of constants. Furthermore a finite set $W \subset \mathbb{N}$ is given. Now consider the recursive path ordering with status $>_{RPOS}$ (see e.g. [Steinbach, 1990]), where $\{\!\{.\}\!\}$ is assigned a multiset status and $\langle ., . \rangle$ is assigned a lexicographical left-to-right status. The precedence $>_{RPOS}$ on function symbols uses the given well-founded ordering $\succ$ on C; the set W is mapped

isomorphically (say, by ϕ) to a set of new constants, and $>_{RPOS}$ is extended order-isomorphically wrt. the ordering $>$ on naturals. Finally, the constructor symbols $\{\!|.|\!\}$ and $\langle .,. \rangle$ are given in $>_{RPOS}$ more weight than $\phi(max(W))$. This implies $\{\!|.|\!\} >_{RPOS} \phi(w)$ for every multiset.

With this definition it can be verified by unfolding $>_{RPOS}$ according to the cases of term structure, that $\succ\!\!\!\succ_{NMW}$ satisfies $>_{RPOS}$. In particular, the condition that the constructor symbols $\{\!|.|\!\}$ and $\langle .,. \rangle$ is given in $>_{RPOS}$ more weight than $\phi(max(W))$ implies that we can obtain $M_v >_{RPOS} N_w$ in case N is a true subset of M (because then $M_v >_{RPOS} w$ holds, as is required by $>_{RPOS}$).[6] Thus, with $>_{RPOS}$ being a simplification ordering (see [Steinbach, 1990]), $\succ_{NMW}$ is also.

5.3.2 Derivation Orderings

In order to be as general as possible we introduce the following notion:

Definition 5.3.2 (Derivation Ordering). *A binary relation $\succ$ on ground derivations is called a* derivation ordering *iff $\succ$ is a well-founded and strict ordering. Now let D be a derivation. A derivation ordering $\succ$ is called* monotonic *iff $D|_{p,i,j} = G$ and $G \succ F$ implies $D \succ D[F]_{p,i,j}$, where F is a derivation which agrees with G on top literal and derived literal.*

Next we will design an appropriate derivation ordering for linearizing completion. For this let D be a derivation

$$D = (L_1 \overset{D_1}{\Longrightarrow} L_2 \cdots L_n \overset{D_n}{\Longrightarrow} L_{n+1}),$$

and define the *complexity of D, $compl(D)$* as

$$compl(D) = \{\!|0, \langle compl(D_1), w_1 \rangle, \ldots, \langle compl(D_n), w_n \rangle|\!\} \ ,$$

where for a sequence $D^1 D^2 \cdots D^n$ of derivations we define

$$compl(D^1 D^2 \cdots D^m) = \bigcup_{i=1 \ldots m} compl(D^i) \ .$$

Thus, $compl(D)$ is a multiset whose elements are nested multisets with weights over the set $\{0\}$. $compl(D)$ contains structural information about the derivation: it expresses the shape (when read as a tree) of the derivation encoded as multisets, and occurrences of input literals are mapped to the dummy element 0 at the leaves. Furthermore, the weight of a used inference rule comes into the complexity measure as the weight of the multiset corresponding to the rule application. For instance, the derivation D_1 in example 5.2.1 has the complexity $\{\!|0, \{\!|\}_5, \{\!|0, \{\!|\}_4|\!\}_3|\!\}$.

[6] We conjecture that even without the restriction to finite W a simplification ordering results. This proof, however, does not go through in that case.

In order to compare two derivations D_1 and D_2 we attach the artificial weight 0 to them and use the nested multiset ordering with weights. More formally define

$$D_1 \succ_{Lin} D_2 \text{ iff } \langle compl(D_1), 0\rangle \gg_{NMW,\perp} \langle compl(D_2), 0\rangle \ ,$$

where the base set X is $\{0\}$ and its base ordering $\perp$ is the empty ordering $\emptyset$ (this is trivially an ordering). Evidently, a derivation D is *linear* (cf. Section 5.2) iff its complexity $compl(D)$ is of the form

$$\{\!\{0, \{\!\{0, \ldots, 0\}\!\}_{w_1}, \ldots, \{\!\{0, \ldots, 0\}\!\}_{w_n}\}\!\} \ ,$$

where the w_is are are the weights of the used inference rules. The ordering $\succ_{Lin}$ is defined in such a way that when weights are neglected smaller derivations are "more linear". Even more, if D is linear and $D \succ_{Lin} D'$ then D' is also linear (Lemma 5.5.1).

It holds:

Proposition 5.3.2. *The relation $\succ_{Lin}$ is a monotonic derivation ordering.*

5.3.3 Redundancy

Building on orderings we come to redundancy. In order to be as flexible as possibly the following definition is rather general and includes the notion of redundancy presently used for the linearizing completion as a special case.

Definition 5.3.3 ($\succ$-Redundancy). *Let $\succ$ be a derivation ordering, and let $\mathcal{I}$ be an inference system. An inference rule $P \to C$ is called* ($\succ$-)redundant in $\mathcal{I}$ for derivations *($\mathcal{I}$ need not necessarily contain $P \to C$) iff for every inference system $\mathcal{J}$ with $\mathcal{I} \subseteq \mathcal{J}$ and every ground derivation*

$$D = (L_1 \Rightarrow^*_{(\mathcal{J} \cup \{P \to C\})^g, M} L_n)$$

which uses $P \to C$ there is also a ground derivation $D' \prec D$ of the form

$$D' = (L_1 \Rightarrow^*_{(\mathcal{J} \setminus \{P \to C\})^g, M} L_n) \ .$$

The rule $P \to C$ is called ($\succ$-)redundant in $\mathcal{I}$ for completion *iff (1) $P \to C$ is not of the form $\neg A \to \mathsf{F}$, where A is an atom, and (2) $P \to C$ is $\succ$-redundant in $\mathcal{I}$ for derivations.*

The short forms ($\succ$-)c-redundant in $\mathcal{I}$ *and* ($\succ$-)d-redundant in $\mathcal{I}$ *used in the sequel mean* for completion *and* for derivations, *respectively*

An inference rule of the form $\neg A \to \mathsf{F}$ is never $\succ$-redundant due to (1). The motivation for this comes from the use of such rules in the rôle of input literals during completion. It turns out that the deletion of a rule $\neg A \to \mathsf{F}$ in general does not provide a substitute for this purpose, even if the rule would be redundant according to $\succ$-redundancy for derivations.

$\succ$-redundancy for derivations means that any ground derivation in a possibly extended inference system $\mathcal{J}$ using the redundant inference rule can be replaced by a smaller derivation wrt. $\succ$, which uses at most the input literals as given, and this derivation does not use the redundant rule. The strict decreasing property is required since otherwise the redundancy of a rule in a certain inference system would not carry over to the inference system obtained as the limit of the completion process.

Notice that $\succ$-redundancy for derivations quantifies over derivations and hence does not suggest a respective procedure for testing $\succ$-redundancy. We would thus like to have a more "local", constructive way of characterizing $\succ$-redundancy for derivations. The following sufficient criterion achieves this for the case of linearizing completion:

Proposition 5.3.3 (Sufficient $\succ_{Lin}$-Redundancy Criterion). *Let $\mathcal{I}$ be an inference system and $P \to_w C$ be an inference rule. Suppose that for every $L \in P$ there is a linear $\mathcal{I} \setminus \{P \to_w C\}$-derivation from P*

$$(L = L_1) \overset{D_1}{\Longrightarrow}_{P_1 \to_{w_1} C_1} L_2 \overset{D_2}{\Longrightarrow}_{P_2 \to_{w_2} C_2} L_3 \cdots L_{n-1} \overset{D_{n-1}}{\Longrightarrow}_{P_{n-1} \to_{w_{n-1}} C_{n-1}} (L_n = C)$$

with $n \geq 1$ and such that for $i = 1, \ldots, n-1$ it holds $\langle P \setminus \{\!\{L\}\!\}, w\rangle \gg_{MW} \langle D_i, w_i\rangle$ In this comparison the sequence D_i of literals is to be read as a multiset. Then $P \to_w C$ is $\succ_{Lin}$-redundant in $\mathcal{I}$ for derivations.

The proof is by induction on the structure of a derivation, thereby making use of the monotonicity property of $\succ_{Lin}$ in the induction step.

Informally, the condition $\langle P \setminus \{\!\{L\}\!\}, w\rangle \gg_{MW} \langle D_i, w_i\rangle$ means that the i-th inference step either uses strictly fewer side literals (the set D_i) than the side literals $P \setminus \{\!\{L\}\!\}$ of an inference step carried out with $P \to_w C$, or else, the side literals are the same, but an inference rule of less weight is used. This check has to be done for every $L \in P$ according to the potential applications of the inference rule in a derivation.

Example 5.3.2. Consider the following ground inference system:

(1) $A, B, C \to_4 D$
(2) $A, B, C \to_3 E$
(3) $E, C \to_5 D$
(4) $C \to_5 D$.

The rule (1) is $\succ_{Lin}$-redundant for completion , because it is not of the form $\neg A \to \mathsf{F}$ and it is redundant for derivations. Using Proposition 5.3.3 this is checked as follows: for A as top literal consider the derivation $A \overset{BC}{\Longrightarrow}_{(1)} D$. It can be replaced by the derivation $A \overset{BC}{\Longrightarrow}_{(2)} E \overset{C}{\Longrightarrow}_{(3)} D$; for the first inference step we have

$$\{\!|B, C|\!\} = (\{\!|A, B, C|\!\} \setminus \{\!|A|\!\}) \ .$$

Hence the weights must be considered, which yields $3 < 4$. Thus the first inference step satisfies the condition stated in the proposition. For the second step we have $\{\!|C|\!\} \subset \{\!|B, C|\!\}$, hence the weights do not matter. The case for B as top literal is similar, and for the top literal C rule (4) can be used.

Example 5.3.3. Consider the rule $R = (x < x', x' < y', y' < z' \to x < z')$ which expresses a once unfolded transitivity rule $Trans = (x < y, y < z \to x < z)$. We claim that R is $\succ_{Lin}$-d-redundant in an inference system which contains *Trans*. Using Proposition 5.3.3 this is checked as follows: for $x < x'$ as top literal consider the derivation

$$x < x' \xRightarrow{x'<y' \quad y'<z'}_{R} x < z'$$

whose side literals are $\{\!|x' < y', y' < z'|\!\}$. It can be replaced by the derivation

$$x < x' \xRightarrow{x'<y'}_{(Trans)} x < y' \xRightarrow{y'<z'}_{(Trans)} x < z' \ .$$

Furthermore for the first inference step it holds $\{\!|x' < y'|\!\} \subset \{\!|x' < y', y' < z'|\!\}$ and for the second step $\{\!|y' < z'|\!\} \subset \{\!|x' < y', y' < z'|\!\}$. Similarly derivations with top literals $x' < y'$ and $y' < z'$ exist. Weights are not needed for these derivations.

The practical value of the stronger $\succ$-redundancy for *derivations* alone is the possibility to delete rules from an inference system $\mathcal{I}$ which are redundant for derivations without losing completeness. In the search for derivations (and in particular refutations) we will thus never have to consider rules of the form $\neg A \to \mathsf{F}$, provided they are redundant for derivations. Section 5.7.3 below contains an example of a successful application of this idea.

The formal account is as follows:

Definition 5.3.4 (*NonRed*($\mathcal{I}$))**.** *Let $\mathcal{I}$ be a possibly infinite inference system. Consider any possibly infinite sequence*

$$(\mathcal{I}_0 = \mathcal{I}) \supseteq \mathcal{I}_1 \supseteq \cdots \supseteq \mathcal{I}_n \supseteq \cdots$$

of inference systems, where $\mathcal{I}_{i+1}$ is obtained from $\mathcal{I}_i$ (for $i \geq 0$) by deletion of an inference rule which is redundant in $\mathcal{I}_i$ for derivations. Define the non-redundant inference rules of this sequence as

$$NonRed(\mathcal{I}) = \bigcap_{i \geq 0} \mathcal{I}_i \ .$$

That is, we stepwisely remove inference rules which are redundant for derivations. The limit $NonRed(\mathcal{I})$ just consists of the rules which are persistent, i.e. which are not deleted. Of course, $NonRed(\mathcal{I})$ is not determined uniquely. The next proposition justifies this concept:

Proposition 5.3.4 (Completeness of $NonRed(\mathcal{I})$). *For every ground derivation $D = (L_1 \Rightarrow^*_{\mathcal{I}^g,M} L_n)$ a ground derivation $D' = (L_1 \Rightarrow^*_{NonRed(\mathcal{I})^g,M} L_n)$ with $D' \preceq D$ exists.*

Proof. It is clear from the definition of redundancy for derivations that $L_1 \Rightarrow^*_{\mathcal{I}_i^g,M} L_n$ for all i. Further, for some $\mathcal{I}_i$ it must be the case that all inference rules $\mathcal{I}' \subseteq \mathcal{I}_i$ ground instances of which are used in $D_i = L_1 \Rightarrow^*_{\mathcal{I}_i^g,M} L_n$ are never deleted afterwards, i.e. $\mathcal{I}' \subseteq \mathcal{I}_j$ for all $j \geq i$ (*). Reason: otherwise there would be an infinite sequence $(D_{k_0} = D) \succ D_{k_1} \succ \cdots \succ D_{k_n} \succ \cdots$ of derivations $D_{k_n} = L_1 \Rightarrow^*_{\mathcal{I}_{k_j}^g,M} L_n$, which however, is impossible by well-foundedness of the derivation ordering $\succ$. Thus, from (*) it follows immediately that $\mathcal{I}' \subseteq NonRed(\mathcal{I})$, and D' exist as claimed.

5.4 Transformation Systems

Transformation systems are the formal device for transformations of the *inference systems* of the previous section. A transformation system transforms an initial inference system in a *fair* way by application of certain *transformation rules* into a *completed state*. Completed inference systems in turn are refutationally complete wrt. the desired restrictions (in our case "linearity" and "unit-resulting").

Definition 5.4.1 (Transformation system). *A* transformation rule *with* premise P *and* conclusion C *is an expression of the form* $\frac{P}{C}$ *where P is a multiset of inference rules and C is an inference rule. By* applying *a transformation rule to a multiset of inference rules P' we mean the matching of P to P' by some substitution μ. The* result *of such an application then is $C\mu$. A transformation rule can be labeled as* mandatory *or* optional. *A* transformation system *consists of a set of transformation rules and a well-founded ordering $\succ$ on derivations.*

*Let $\mathcal{S}$ be a transformation system. The relation $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I}'$ (*deduction step*) on inference systems means that $\mathcal{I}'$ is obtained from $\mathcal{I}$ by either (1) adding the result of an application of a transformation rule from $\mathcal{S}$ to variable disjoint variants of rules in $\mathcal{I}$, or else (2) by deleting a $\succ$-c-redundant inference rule from $\mathcal{I}$. In case (1) the weight of the added inference rule can be chosen arbitrarily. A $\mathcal{S}$-*deduction from an inference system $\mathcal{I}_0$ *is a sequence $\mathcal{I}_0 \vdash_{\mathcal{S}} \mathcal{I}_1 \vdash_{\mathcal{S}} \cdots \vdash_{\mathcal{S}} \mathcal{I}_n \vdash_{\mathcal{S}} \cdots$. Deductions may be of finite or infinite length.*

In this book we concentrate on the instance "linearizing completion". Therefore we define the transformation system Lin to consists of transformation rules given in Figure 5.7. The transformation rules Unit1, Unit2 *and* Deduce *are labeled as mandatory, and* Contra *is labeled as optional. As the derivation ordering we use $\succ_{Lin}$ as defined in Section 5.3.*

Unit1:

$$\frac{A_1 \to C \qquad \neg A_2 \to \mathsf{F}}{\overline{C}\sigma \to \mathsf{F}} \quad \left\{ \text{If } A_1\sigma = A_2\sigma \text{ by MGU } \sigma. \right.$$

Unit2:

$$\frac{A_1, P \to C \qquad \neg A_2 \to \mathsf{F}}{(P \to C)\sigma} \quad \left\{ \begin{array}{ll} \text{If} & (1)\ P \neq \emptyset, \text{ and} \\ & (2)\ A_1\sigma = A_2\sigma \text{ by MGU } \sigma \end{array} \right.$$

Deduce:

$$\frac{P_1 \to C_1 \qquad L_2, P_2 \to C_2}{(P_1, P_2 \to C_2)\sigma} \quad \left\{ \begin{array}{ll} \text{If} & (1)\ P_2 \neq \emptyset, \text{ and} \\ & (2)\ C_1\sigma = L_2\sigma \text{ by MGU } \sigma \end{array} \right.$$

Contra (optional):

$$\frac{L, P \to C}{\overline{C}, P \to \overline{L}} \quad \left\{ \text{If } C \neq \mathsf{F} \right.$$

For atoms A_1,A_2, literals C, L, L_2 and literal multisets P,P_1 and P_2.

Figure 5.7. The transformation rules of the transformation system *Lin*.

Example 5.4.1. From the ground rules $A \to B$ and $\neg A \to \mathsf{F}$ the rule $\neg B \to \mathsf{F}$ can be obtained by Unit1[7]. Consider the inference system $\mathcal{I}_0(\mathcal{T})$ of Example 5.2.1 again. From $B, D \to \mathsf{F}$ and $\neg D \to \mathsf{F}$ we can obtain $B \to \mathsf{F}$ by Unit2. Unit1 and Unit2 share the same purpose: to eliminate in derivations applications of literals from $Punit(\mathcal{I})$ (these are initially needed, as stated in Lemma 5.2.1). For instance, the set $M_1 = \{A, C\}$ in Example 5.2.1 now admits the $\mathcal{I}_0(\mathcal{T}) \cup \{B \to \mathsf{F}\}$-refutation

$$D_2 = (A \Rightarrow_{A \to B} B \Rightarrow_{B \to \mathsf{F}} \mathsf{F}) \ .$$

From the rules $C \to D$ and $B, D \to \mathsf{F}$ in Example 5.2.1 the rule $B, C \to \mathsf{F}$ can be obtained by Deduce. Deduced rules are used to turn a refutation stepwise into a "more linear" refutation. Again, using $\mathcal{I}_0(\mathcal{T}) \cup \{B, C \to \mathsf{F}\}$ the refutation D_1 from Example 5.2.1 of $M_1 = \{A, C\}$ can be linearized with the new rule. We have:

$$D_1' = (A \Rightarrow_{A \to B} B \overset{C}{\Rightarrow}_{B,C \to \mathsf{F}} \mathsf{F}) \ .$$

If the new rule $B, C \to \mathsf{F}$ is given the weight, say, 10, then the complexity of D_1' is $\{\!|0, \{\!|\}_5, \{\!|0|\!\}_{10}|\!\}$ and it can be verified that $D_1 \succ_{Lin} D_1'$.

From the transitivity rule $x < y, y < z \to x < z$ and a copy $x' < y', y' < z' \to x' < z'$ by Deduce with $\sigma = \{y \leftarrow x', z \leftarrow z'\}$ one obtains $x < x', x' < y', y' < z' \to x < z'$.

From $A, B \to C$ by Contra $\neg C, B \to \neg A$ can be obtained.

This Contra transformation rule is an optional rule and thus is not labeled as *mandatory*. It is sometimes valuable in order to come to a finite system (see Section 5.1.4 in the introduction for an example). However, the Contra rule should be applied carefully since it increases the search space of the generated inference systems; applying Contra exhaustively will produce every possible contrapositive of a theory clause. This is clearly not intended. Surprisingly, an application of the Contra rule may increase the deductive power of an inference system: consider e.g. $\mathcal{I} = \{A \to B\}$. The only non-trivial $\mathcal{I}$-derivation is $A \Rightarrow B$. However, when $\mathcal{I}$ is enriched with the contrapositive $\neg B \to \neg A$, the derivation $\neg B \Rightarrow \neg A$ also exists. However its application does not increase the refutational power of inference systems.

We collect for later use the following lemma, which states that c-redundancy persists along transformation steps:

Lemma 5.4.1. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Suppose $P \to C$ is $\succ$-c-redundant in some $\mathcal{I}$, and suppose that $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{J}$. Then $P \to C$ is also $\succ$-c-redundant in $\mathcal{J}$.*

Proof. If $\mathcal{J}$ is obtained by adding a new rule to $\mathcal{I}$ the lemma is trivial by the property $\mathcal{J} \supseteq \mathcal{I}$ in the definition of c-redundancy (Def. 5.3.3). If $\mathcal{J}$ is

[7] Weights are not considered in this example.

obtained from $\mathcal{I}$ by deletion of a redundant rule then well-founded induction is used to eliminate every use of $P \to C$ and the deleted rule in derivations. See the appendix for the full proof.

It should be remarked that the proof goes through even if d-redundancy would be used instead of c-redundancy. This holds in other (but not all) situations as well. But we do not need this here.

Note 5.4.1 (Soundness). We conclude this section with a note on soundness. In order to achieve the soundness of the overall approach, one has first to guarantee that all derivations obtainable from initial inference systems are sound. This, however, is clear since they are nothing but Unit-Resulting refutations. Secondly, one has to guarantee that derivations obtainable from transformed inference systems are sound. The key to this result is the observation that newly generated inference rules are just resolvents of present rules, and hence are *logical consequences* of these.

5.4.1 Limit Inference Systems

The process of applying the transformation rules of a transformation system may terminate or not. In order to treat both cases in a uniform way it is useful to define the *limit* inference system $\mathcal{I}_\infty$, which is finite if the transformation system eventually does not produce new inference rules any more, and infinite otherwise.

Definition 5.4.2 (Limit, [Bachmair, 1991]). *The* limit of a deduction $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots \vdash \mathcal{I}_n \cdots$ *is defined as*

$$\mathcal{I}_\infty = \bigcup_i \bigcap_{j \geq i} \mathcal{I}_j \ .$$

The elements of $\mathcal{I}_\infty$ *are also called the* persisting *inference rules.*

The limit $\mathcal{I}_\infty$ of a deduction is the set of inference rules generated eventually and never deleted afterwards:

Lemma 5.4.2. *Let* $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \ldots$ *be a deduction and suppose* $(P \to C) \in \mathcal{I}_\infty$. *Then for some* k, *and for all* $i \geq k$: $(P \to C) \in \mathcal{I}_i$

Proof. By contradiction. Assume $P \to C \in \mathcal{I}_\infty$ and suppose that for all k an $i \geq k$ exists such that $P \to C \notin \mathcal{I}_i$. Hence whenever $P \to C \in \mathcal{I}_k$ then $P \to C \notin \bigcap_{i \geq k} \mathcal{I}_i$. Thus $P \to C \notin \bigcup_k \bigcap_{i \geq k} \mathcal{I}_i$ and hence by definition of $\mathcal{I}_\infty$, $P \to C \notin \mathcal{I}_\infty$. This however contradicts the assumption of the lemma.

This result can be extended for derivations:

Proposition 5.4.1. *Let* $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \ldots$ *be a deduction and let* D *be a* $\mathcal{I}_\infty^g$*-derivation (resp.* $\mathcal{I}_\infty$ *derivation). Then for some* k, D *is also a* $\mathcal{I}_k^g$*-derivation (resp.* $\mathcal{I}_k$ *derivation).*

Proof. Let $\{r_1, \ldots, r_n\} \subseteq \mathcal{I}_\infty$ be the (finite) inference system used in the given derivation. For every r_j $(j = 1 \ldots n)$ by Lemma 5.4.2 it holds that for some k_j, all $i \geq k_j$: $r_j \in \mathcal{I}_i$. Now take $k = max(\{k_1, \ldots, k_n\})$ and observe that $\{r_1, \ldots, r_n\} \subseteq \mathcal{I}_k$.

The next lemma extends Lemma 5.4.1 to the limit $\mathcal{I}_\infty$, i.e. c-redundant inference rules remain redundant in the limit:

Lemma 5.4.3. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be a $\mathcal{S}$-deduction. If for some k, $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$ then $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_\infty$.*

Also the *Punit*-literals persist:

Lemma 5.4.4. *Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be a deduction. If $A \in Punit(\mathcal{I}_k)^g$ then also $A \in Punit(\mathcal{I}_\infty)^g$.*

Proof. A is a ground instance of some atom A', where $\neg A' \to \mathsf{F} \in \mathcal{I}_k$. By definition of $\succ$-c-redundancy, a rule $\neg A \to \mathsf{F} \in \mathcal{I}_k$ is never $\succ$-c-redundant and thus is never deleted. Hence $\neg A \to \mathsf{F} \in \mathcal{I}_i$ for every $i \geq k$. Thus

$$(\neg A \to \mathsf{F}) \in \bigcap_{i \geq k} \mathcal{I}_i \subseteq \bigcup_{k \geq 0} \bigcap_{i \geq k} \mathcal{I}_i = \mathcal{I}_\infty \ .$$

By this the claim follows.

Lemma 5.4.3 and Lemma 5.4.4 imply the following central property:

Lemma 5.4.5. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be an $\mathcal{S}$-deduction. If there is a derivation*

$$D = (L \Rightarrow^*_{\mathcal{I}_k^g, M \cup Punit(\mathcal{I}_k^g)} L')$$

then there is also a derivation

$$D' = (L \Rightarrow^*_{\mathcal{I}_\infty^g, M \cup Punit(\mathcal{I}_\infty^g)} L')$$

with $D' \preceq D$.

5.4.2 Fairness and Completion

Deductions must be *fair*, which roughly means that no application of a mandatory transformation rule is deferred infinitely long. Fairness is important since it entails that "enough" inference rules to obtain normal derivations are generated. Our definition of fairness is an adaption of standard definitions in the term-rewriting literature (see e.g. [Bachmair, 1991]).

Definition 5.4.3 (Fairness). *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. An $\mathcal{S}$-deduction $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots \vdash \mathcal{I}_n \cdots$ is called* fair *iff whenever $\mathcal{I}_\infty \vdash \mathcal{I}_\infty \cup \{P \to C\}$ for some application of a mandatory transformation rule from $\mathcal{S}$, then for some k, $P \to C \in \mathcal{I}_k$ up to renaming, or $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$.*

Fairness states that it is sufficient either to generate an inference rule or to prove it c-redundant from persisting inference rules only. This notion of fairness enables the use of a "delete as many inference rules as possible" strategy in implementations, since a rule once shown to be c-redundant is redundant in all subsequent stages (Lemma 5.4.3) and thus need not persist.

The next central concept is that of a *completed* inference system, which means that only c-redundant new inference rules can be generated. Completion is a useful concept since it allows us to characterize refutationally complete inference systems, which is a semantic concept, in a more syntactical way[8].

Definition 5.4.4 (Completed Inference System). *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. An inference system $\mathcal{I}$ is* completed (wrt. $\mathcal{S}$) *iff whenever $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I} \cup \{P \to C\}$ by application of a mandatory transformation rule from $\mathcal{S}$ then $P \to C \in \mathcal{I}$ up to renaming or $P \to C$ is $\succ$-c-redundant in $\mathcal{I}$.*

Fairness, deductions and completion relate as follows:

Theorem 5.4.1. *Let $\mathcal{S}$ be a transformation system and $\mathcal{I}_0$ be an inference system. The limit $\mathcal{I}_\infty$ of a fair $\mathcal{S}$-deduction $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ is completed wrt. $\mathcal{S}$.*

Proof. By contradiction. Suppose $\mathcal{I}_\infty$ is not completed. Then, for some application of a mandatory transformation rule from $\mathcal{L}$ we have $\mathcal{I}_\infty \vdash \mathcal{I}_\infty \cup \{P \to C\}$ such that $P \to C \notin \mathcal{I}_\infty$ and $P \to C$ is not $\succ$-c-redundant in $\mathcal{I}_\infty$ (*). However, by fairness, for some k, $P \to C \in \mathcal{I}_k$ or $P \to C$ is c-redundant in $\mathcal{I}_k$. This suggests the following case analysis:

Case 1: Suppose $P \to C \in \mathcal{I}_k$. If $P \to C$ is not deleted afterwards, i.e. for all $i \geq k$, $P \to C \in \mathcal{I}_i$, then $P \to C \in \bigcap_{i \geq k} \mathcal{I}_i$ and thus also $P \to C \in \mathcal{I}_\infty$. Contradiction to (*).
Otherwise, if $P \to C$ is deleted in some $\mathcal{I}_j$, $P \to C$ must have been $\succ$-c-redundant in $\mathcal{I}_{j-1}$. But then by Lemma 5.4.3 $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_\infty$. Contradiction to (*).

Case 2: If $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$ then by Lemma 5.4.3, $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_\infty$. Contradiction to (*).

[8] Note that the term "complete" has two distinct technical meanings here, and the refutational completeness of a completed system depends additionally on how the inference rules are used (unit-resulting linear hyper resolution, in this case), and what deduction rules are used to complete the system.

5.5 Complexity-Reducing Transformation Systems

Up to now we have seen that the inference systems generated along a deduction never increase the complexity of a once obtained derivation. However, in order to obtain completeness wrt. normal-form derivations (linear derivations) more is required: the transformation system has to eventually generate inference rules that will strictly *de*crease the complexity of a non-normal derivation.

Definition 5.5.1 (Normal Derivations, Order-Normalizing System). *A set $\mathcal{N}$ of ground derivations is called* normal *iff $\mathcal{N}$ is downward closed wrt. a given derivation ordering $\succ$, i.e. if $D \in \mathcal{N}$ and $D' \preceq D$ for some ground derivation D' then $D' \in \mathcal{N}$. In the sequel $\mathcal{N}$ always denotes a set of normal derivations.*

Now let $\mathcal{I}$ be an inference system and let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Then $\mathcal{S}$ is called order-normalizing wrt. $\mathcal{N}$ *iff whenever there is a ground derivation $D = (L \Rightarrow^*_{\mathcal{I}^g,M} L')$ such that $D \notin \mathcal{N}$ then there is a derivation $D' = (L \Rightarrow^*_{(\mathcal{I}')^g,M} L')$ with $D' \prec D$, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I}'$ by one single application of some mandatory transformation rule.*

The normal derivations are the ones to be achieved by the completion process. In the case of linearizing completion we thus target the set $LinG$ of all linear ground derivations.

From downward closure we can always conclude that if ground derivation D is normal and $D' \preceq D$ then also D' is normal. This is important because if D' replaces a normal derivation D according to the definition of $\succ$-c-redundancy (Def. 5.3.3), then also D' will be normal.

For the case of linearizing completion it is therefore important to have:

Lemma 5.5.1. *The set $LinG$ of all linear ground derivations is normal wrt. the derivation ordering $\succ_{Lin}$. That is, if ground derivation D is linear and $D \succ_{Lin} D'$ then D' is also linear.*

Proof. Suppose, to the contrary, there is a linear ground derivation D and a ground derivation D' such that $D \succ_{Lin} D'$ but D' is not linear. Hence $compl(D')$ is of the form (weights omitted)

$$compl(D') = \{\!\{0, c', \ldots\}\!\}, \quad \text{where } c' = \{\!\{0, \{\!\{0, \ldots\}\!\}, \ldots\}\!\},$$

whereas

$$compl(D) = \{\!\{0, \{\!\{0, \ldots, 0\}\!\}, \ldots \{\!\{0, \ldots, 0\}\!\}\}\!\} \ .$$

Suppose $compl(D)$ contains n multisets as indicated. Let c be a maximal of those. Clearly, by property of nested multiset ordering it holds $c' \succcurlyeq_{NMW} c$. Next let c'' be obtained from $compl(D')$ by replacing c' by $n+1$ occurrences of c. It follows $compl(D') \succcurlyeq_{NMW} c''$. On the other hand also $c'' \succcurlyeq_{NMW} compl(D)$. Thus, together $compl(D') \succcurlyeq_{NMW} compl(D)$, which is a contradiction to the assumption $D \succ_{Lin} D'$ (i.e. $compl(D) \succcurlyeq_{NMW} compl(D')$).

An order-normalizing transformation system enables the generation of a new inference rule (if not present already) that allows a decrease in complexity of a given non-normal ground derivation. Note that not necessarily $D' \in \mathcal{N}$ in Definition 5.5.1. Such a property is not required, because the derivation ordering $\succ$ is well-founded. So we will eventually end up with a normal derivation $D' \in \mathcal{N}$ for any given derivation D (see Proposition 5.5.2 below).

As an example for an order-normalizing transformation system take "linearizing completion", which is order-normalizing wrt. linear ground derivations:

Proposition 5.5.1. *The transformation system Lin is order-normalizing wrt. LinG.*

In essence, the proof uses the facts that the Deduce transformation rule is labeled as mandatory and furthermore works towards strictly decreasing derivations wrt. the well-founded ordering $\succ_{Lin}$.

Returning to the general level, we find that a completed inference system in conjunction with order-normalizing transformation systems yields normal derivations:

Proposition 5.5.2. *Let $\mathcal{S}$ be an order-normalizing transformation system wrt. $\mathcal{N}$ and let $\mathcal{I}$ be a completed inference system wrt. $\mathcal{S}$ with derivation ordering $\succ$. Whenever there a ground derivation $D = (L \Rightarrow^*_{\mathcal{I}^g,M} L')$ exists then a ground derivation $D' = (L \Rightarrow^*_{\mathcal{I}^g,M} L')$ with $D' \in \mathcal{N}$ and $D' \preceq D$ also exists.*

Proof. By well-founded induction on the derivation ordering $\succ$: either $D \in \mathcal{N}$ and we are done by taking $D' = D$, or else by definition of order-normalizing transformation systems there is a derivation $D'' = (L \Rightarrow^*_{(\mathcal{I}')^g,M} L')$ with $D'' \prec D$, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I}'$ by application of some mandatory transformation rule. If $\mathcal{I}' = \mathcal{I}$ we can immediately apply the induction hypothesis to D'' to obtain the desired derivation D'. Otherwise, $\mathcal{I}' = \mathcal{I} \cup \{P \to C\}$. Since $\mathcal{I}$ is completed either (a) a variant of $P \to C$ is contained in $\mathcal{I}$, or (b) $P \to C$ is $\succ$-c-redundant in $\mathcal{I}$. In case (a) we can replace in D'' every use of $P \to C$ by its variant from $\mathcal{I}$ and obtain a $\mathcal{I}$-derivation alone (to which the induction hypothesis can be applied). Now for case (b): either $P \to C$ is not used and thus D'' is a $\mathcal{I}$-derivation alone (to which the induction hypothesis can be applied), or else, by definition of c-redundancy (Def. 5.3.3) there is a $\mathcal{I}^g$-derivation $D''' \prec D''$ of the same kind as D''. By applying the induction hypothesis to D''' this concluding case is done.

Chaining ground completeness (Lemma 5.2.1), the completion property of limits (Theorem 5.4.1) with Proposition 5.5.2 we can obtain normal refutations of $M \cup Punit(\mathcal{I})$, where M is some $\mathcal{T}$-unsatisfiable literal set. Thus, a refutation might still use unit inference rules being turned via $Punit(\mathcal{I})$ into input literals. However, every use of a literal from $Punit(\mathcal{I})$ represents a computation among the inference rules and should be avoided. In order to admit

normal refutations with input literals M alone, we will compile the literals $Punit(\mathcal{I})$ into the inference system. The situation is much like normalizing above, but now "normal" means "free of usages of elements of $Punit(\mathcal{I})$". In order to obtain termination when eliminating these cases we require the respective mandatory transformation rules to work strictly decreasing wrt. $\succ$. This is captured in the next definition.

Definition 5.5.2 (*Punit*-Normalizing Transformation System). *The multiset of* used input literals of a derivation D *is defined as*

$$Used(D) = \{\!|L \mid L \text{ is the top literal of } D|_p, \text{ where } p \in P(D)|\!\} \ .$$

The function used is extended homomorphically to sequences and multisets of derivations as expected.

Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$ and let $\mathcal{N}$ be a set of normal derivations. Then $\mathcal{S}$ is called Punit-normalizing wrt. $\mathcal{N}$ *iff whenever there is a non-trivial ground derivation*

$$D = (L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} L')$$

with $D \in \mathcal{N}$ such that $Used(D) \cap Punit(\mathcal{I}^g) \neq \emptyset$ then there is a derivation

$$D' = (K \Rightarrow^*_{(\mathcal{I}')^g, M \cup Punit((\mathcal{I}')^g)} L')$$

with $D' \prec D$ where $K \in M \cup Punit((\mathcal{I}')^g) \cup \{L\}$ and $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I}'$ by application of some mandatory transformation rule.

"*Punit*-normalization" means that whenever an inference rule $\neg A \rightarrow \mathsf{F}$ is used as an input literal A then a strictly smaller derivation exists. That derivation, however, need not start with the same top literal L, but may as well start with a top literal from $M \cup Punit((\mathcal{I}')^g)$. Note that according to this definition it suffices for a *Punit*-normalizing transformation system to work on normal-form derivations only.

Fortunately it holds:

Proposition 5.5.3. *The transformation system Lin is Punit-normalizing wrt. LinG.*

This is essentially due to the transformation rules Unit1 and Unit2.

Example 5.5.1. Consider the inference system $\mathcal{I}_0(\mathcal{T})$ in Example 5.2.1 again. It holds $Used(D_1) = \{A, C\}$. Since $\mathcal{I}_0(\mathcal{T})$ includes a rule $\neg D \rightarrow \mathsf{F}$, we have

$$M_3 = M_1 \cup Punit(\mathcal{I}_0(\mathcal{T})) = \{A, C, D\} \ ,$$

and we can find a $\mathcal{I}_0(\mathcal{T})$-derivation

$$D_3 = (A \Rightarrow_{A \rightarrow B} B \overset{D}{\Rightarrow}_{B, D \rightarrow \mathsf{F}} \mathsf{F})$$

from M_3. It holds that

$$Used(D_3) \cap Punit(\mathcal{I}_0(\mathcal{T})) = \{A, D\} \cap \{D\} \neq \emptyset \ .$$

By the previous proposition we should be able to find a smaller derivation wrt. $\succ_{Lin}$ after application of some mandatory inference rule from Lin. Indeed, the Unit2 rule applied to $B, D \to \mathsf{F}$ and $\neg D \to \mathsf{F}$, gives the new rule $B \to \mathsf{F}$ and allows for a smaller derivation. This derivation is just D_2 in Example 5.4.1.

Completed inference systems in conjunction with *Punit*-normalizing transformation systems yield derivations that are free of applications of inference rules $\neg A \to \mathsf{F}$ as input literals A. This result is similar to Proposition 5.5.2 for order-normal derivations:

Proposition 5.5.4. *Let $\mathcal{S}$ be a Punit-normalizing transformation system wrt. $\mathcal{N}$, and let $\mathcal{I}$ be a completed inference system wrt. $\mathcal{S}$. Whenever there is a ground derivation $D = (L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} L')$ with $D \in \mathcal{N}$ then there is also a ground derivation $D' = (K \Rightarrow^*_{\mathcal{I}^g, M} L')$ with $D' \preceq D$, $D' \in \mathcal{N}$ and some $K \in M \cup \{L\}$.*

Since we are interested in both properties — *Punit*-normalization and order-normalization — we define:

Definition 5.5.3 (Normalizing Transformation System). *A transformation system $\mathcal{S}$ is called* normalizing wrt. $\mathcal{N}$ *iff $\mathcal{S}$ is both order-normalizing wrt. $\mathcal{N}$ and Punit-normalizing wrt. $\mathcal{N}$.*

Combining Propositions 5.5.1 and 5.5.3 we then obtain:

Theorem 5.5.1. *The Transformation system Lin is normalizing wrt. the set LinG of linear ground derivations.*

5.6 Completeness

The goal of this section is to assemble the material of the previous sections into several completeness results. A *soundness* result for linearizing completion shall not be formally established. Soundness can easily be shown by observing that all derivations and deductions can be simulated by resolution. Hence, if a refutation of a $\mathcal{T}$-*satisfiable* set M would exist, it could also be found by resolution, which is impossible by soundness of resolution.

We first establish a general ground completeness result which does not refer to a special derivation ordering. This result will then be instantiated for the case of linearizing completion, and it will be shown that linearizing completion yields suitable background reasoners for the intended application within both total and partial theory reasoning.

5.6.1 Ground Completeness

In purely *equational* logic and Knuth-Bendix completion, Birkhoff's theorem links model theory and proof theory: two ground terms are equal in an equational theory $\mathcal{T}$ — i.e. a set of equations — iff they can be made identical by replacement operations using the equations from $\mathcal{T}$. Since we deal with more general theories we proved in Section 5.2 a corresponding result (ground completeness, Lemma 5.2.1). In order to apply it we find it helpful to introduce the following notion:

Definition 5.6.1 (Relative Completeness). *Inference system $\mathcal{I}$ is called* relatively complete wrt. an inference system $\mathcal{J}$ *iff whenever* $L \Rightarrow^*_{\mathcal{J}^g, M \cup Punit(\mathcal{J}^g)} L'$ *then also* $L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} L'$

Now we can turn towards completeness. Notice that quite often we will use an inference system $NonRed(\mathcal{I})$ and not just $\mathcal{I}$. That is, during "refutation time" there is no need to keep inference rules of the form $\neg A \rightarrow \mathsf{F}$ which are redundant for derivations.

We start with a general result:

Theorem 5.6.1 (General Ground Completeness Theorem). *Let $\mathcal{T}$ be a theory and let $\mathcal{S}$ be a normalizing transformation system wrt. some set $\mathcal{N}$ of normal derivations. Suppose an inference system $\mathcal{I}$ is completed wrt. $\mathcal{S}$, and also suppose that $\mathcal{I}$ is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$. Then for every $\mathcal{T}$-unsatisfiable ground literal set M there is a $NonRed(\mathcal{I})^g$-refutation $D \in \mathcal{N}$ of M with some top literal from M.*

Proof. It holds that M is $\mathcal{T}$-unsatisfiable iff $M' \cup \mathcal{T}'$ is unsatisfiable for some finite subsets $M' \subseteq M$ and $\mathcal{T}' \subseteq \mathcal{T}^g$ (apply Proposition 2.5.1.3, equivalence (a)-(e), and the Skolem, Herbrand, Löwenheim Theorem, Theorem 2.4.3). By ground completeness, Lemma 5.2.1, then

$$L \Rightarrow^*_{\mathcal{I}_0(\mathcal{T}'), M' \cup Punit(\mathcal{I}_0(\mathcal{T}'))} \mathsf{F} \ ,$$

for some $L \in M' \cup Punit(\mathcal{I}_0(\mathcal{T}'))$. With $\mathcal{I}_0(\mathcal{T}') \subseteq \mathcal{I}_0(\mathcal{T})^g$ and $M' \subseteq M$ it follows

$$L \Rightarrow^*_{\mathcal{I}_0(\mathcal{T})^g, M \cup Punit(\mathcal{I}_0(\mathcal{T})^g)} \mathsf{F} \ .$$

Since $\mathcal{I}$ is given as relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$ we find by definition

$$L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} \mathsf{F} \ .$$

With $\mathcal{S}$ being given as normalizing wrt. $\mathcal{N}$ we can first find (by Proposition 5.5.2) an order-normal refutation

$$(L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} \mathsf{F}) \in \mathcal{N} \ ,$$

and then we can find (by Proposition 5.5.4) a *Punit*-normal refutation

$$(K \Rightarrow^*_{\mathcal{I}^g,M} \mathsf{F}) \in \mathcal{N} \ ,$$

for some $K \in M$. Finally apply Proposition 5.3.4.

This theorem requires the existence of a completed and relatively complete inference system wrt. $\mathcal{I}_0(\mathcal{T})$. Such a system can be obtained in a constructive way as the limit of a fair deduction:

Theorem 5.6.2. *The limit $\mathcal{I}_\infty$ of an $\mathcal{S}$-deduction $\mathcal{I}_0(T) \vdash_\mathcal{S} \mathcal{I}_1 \vdash_\mathcal{S} \mathcal{I}_2 \ldots$ is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$, where $\mathcal{T}$ is a theory.*

Proof. Use Lemma 5.4.5, setting there $\mathcal{I}_0 = \mathcal{I}_0(\mathcal{T})$ and $k = 0$.

Thus we can instantiate the general ground completeness theorem:

Corollary 5.6.1. *(to Theorem 5.6.1) Let $\mathcal{T}$, $\mathcal{S}$ and $\mathcal{N}$ as in Theorem 5.6.1, and let $\mathcal{I}_\infty$ be the limit of a fair $\mathcal{S}$-deduction $\mathcal{I}_0(\mathcal{T}) \vdash_\mathcal{S} \mathcal{I}_1 \vdash_\mathcal{S} \ldots$. Then for every $\mathcal{T}$-unsatisfiable ground literal set M there is a $NonRed(\mathcal{I}_\infty)^g$-refutation $D \in \mathcal{N}$ of M with some top literal from M.*

Proof. By Theorem 5.6.2 $\mathcal{I}_\infty$ is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$, and by Theorem 5.4.1 $\mathcal{I}_\infty$ is completed wrt. $\mathcal{S}$. Hence the corollary follows from the theorem.

Next we are going to instantiate this corollary towards "linearizing completion". Before we do so, observe that the corollary (as well as the theorem) states completeness for *some* top literal chosen from the input literal set. However, as motivated in the discussion following Definition 4.5.6 (page 125), the intended application of completed inference systems as background reasoners within theory reasoning calculi demands that we insist on a completeness result with respect to *every* literal as top literal. This "independence of the goal" property corresponds to e.g. linear resolution for Horn theories where it suffices to start derivations with a negative clause. In practice, this means that the top literal need not be guessed in a don't-know nondeterministic way, but can be chosen a priori.

Fortunately, the transformation system *Lin* is powerful enough to generate inference rules that allow rearrangement of a given linear derivation such that any literal used inside a derivation can be switched to the top position. This is expressed in the following lemma:

Lemma 5.6.1 (Top Literal Lemma). *Let $\mathcal{I}$ be a completed inference system wrt. the transformation system Lin. Suppose there is a linear ground derivation $D = (L_1 \Rightarrow^*_{\mathcal{I}^g,M} L_n)$ with $L_1 \in M$. Let $T \in M$ such that $T \in Used(D)$. Then there is a linear ground derivation $D' = (T \Rightarrow^*_{\mathcal{I}^g,M} L_n)$.*

Example 5.6.1. The inference system $\mathcal{I}_{\mathcal{SO}}$ in Example 4.5.3 is completed wrt. *Lin*. Let D be the refutation stated there (the current notation is a bit different, but that does not matter). Since $c < d \in Used(D)$ a refutation with top literal $c < d$ exists, which is as follows:

$$c < d \overset{d<e}{\Longrightarrow} c < e \overset{e<a}{\Longrightarrow} c < a \overset{a<b}{\Longrightarrow} c < b \overset{b<c}{\Longrightarrow} c < c \Longrightarrow \mathsf{F} \ .$$

Note that the top literal lemma is not formulated within the general uninstantiated completion theory, since in general it may be not useful to demand the completeness for arbitrary chosen top literals for arbitrary transformation systems.

Now with the top literal lemma we can obtain the desired completeness result for linearizing completion.

Theorem 5.6.3 (Ground Completeness of Linearizing Completion). *Let $\mathcal{I}$ be an inference system completed wrt. the transformation system Lin. Let $\mathcal{T}$ be a theory and suppose $\mathcal{I}$ is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$. Then for every minimal $\mathcal{T}$-unsatisfiable ground literal set M and every literal $L \in M$ there is a linear refutation*

$$L \Rightarrow^*_{NonRed(\mathcal{I})^g, M} \mathsf{F} \ .$$

Note that a relatively complete inference system as assumed in the theorem can be obtained by application of Theorem 5.6.2.

Proof. By the general ground completeness theorem (Theorem 5.6.1) there is a linear $\mathcal{I}^g$-refutation $D'' \in LinG$ of M. Since M is given as minimal $\mathcal{T}$-unsatisfiable, the intended top literal L must be used at least once in the refutation, because otherwise the refutation of $M \setminus \{L\}$ implies by soundness a contradiction to the minimality assumption about M. Next apply the top literal lemma (Lemma 5.6.1) to D'' and obtain a linear refutation D' with top literal L. Finally, from Proposition 5.3.4 we learn that a refutation $D \prec_{Lin} D'$ using $NonRed(\mathcal{I})$ exists. Further, by Lemma 5.5.1 we know that D is thus a linear refutation as desired.

5.6.2 The Link to Theory Model Elimination

Now the link between linearizing completion and its intended application within PTME-I and PRTME-I is made:

Corollary 5.6.2 (Ground Complete Background Reasoners). *Let $\mathcal{T}$ and $\mathcal{I}$ as in Theorem 5.6.3. Then $NonRed(\mathcal{I})^-$ is ground complete wrt. $\mathcal{T}$ (Def. 4.5.6), where $NonRed(\mathcal{I})^-$ is obtained from $NonRed(\mathcal{I})$ by removing the weight from each inference rule.*

Further, if $\mathcal{T}$ is a definite theory, then the inference system obtained from $NonRed(\mathcal{I})^-$ by deleting all inference rules which are not contra-definite is still ground complete for negative top literals (cf. Def. 4.5.6 again).

Proof. For the first part, let M be a minimal ground $\mathcal{T}$-complementary literal set and let $T \in M$. We have to show that a background refutation as defined in Def. 4.5.6 exists.

By Theorem 5.6.3 there is a ground, linear $NonRed(\mathcal{I})$-refutation

$$D = (T \Rightarrow^*_{NonRed(\mathcal{I})^g, M} \mathsf{F}) \ .$$

It is rather trivial to map D into a background refutation according to Definition 4.5.6 of the same length: simply replace each matching theory inference (Def. 5.2.2) in D, say, $P' \Rightarrow_{P \to_w C,\, \delta} C'$, where $P \to_w C \in \mathcal{I}^g$ by a minimal first-order theory inference (Def. 4.5.2) $P'' \Rightarrow_{P \to C,\, \delta} C'$, where $P \to C \in \mathcal{I}^g$ and $P'' \subseteq P'$ such that P'' contains no duplicates and P' is an amplification of P'' (cf. Section 2.1). Of course, the last step in D yielding "F" yields "$\Box$" instead in the background refutation (this is only a matter of notation).

For the second part — ground completeness for negative top literals when restricting to contra-definite rules — we use a simply syntactic argument. Suppose that T is a negative literal, and R is the background refutation just constructed in the first part of the proof. We show that in R only contra-definite inference rules can have been used. This will prove the claim immediately.

Suppose, to the contrary, a non contra-definite rule is used in R. Locate the leftmost inference step in R using a non contra-definite rule, say $P \to C$ (i.e. if $P \to C$ is used in the i-th inference step in R, then all rules applied in inference steps $j < i$ are contra-definite).

Since in all inference rules treated by linearizing completion the conclusion C is a single literal or F, $P \to C$ must take one of the following forms:

$$A_1, \ldots, A_n \to \mathsf{F} \tag{a}$$
$$A_1, \ldots, A_n \to B \tag{b}$$
$$A_1, \ldots, A_n \to \neg B \tag{c}$$
$$\neg A, A_1, \ldots, A_m \to B \tag{d}$$
$$\neg B_1, \ldots, \neg B_k, A_1, \ldots, A_m \to \mathsf{F} \tag{e}$$
$$\neg B_1, \ldots, \neg B_k, A_1, \ldots, A_m \to B \tag{f}$$
$$\neg B_1, \ldots, \neg B_k, A_1, \ldots, A_m \to \neg B \ , \tag{g}$$

where $n \geq 1$, $m \geq 0$ and $k \geq 2$. We can cancel some of these possibilities immediately: let the *clause form of a rule* $L_1, \ldots, L_n \to L_{n+1}$ be the clause $\overline{L_1} \vee \cdots \vee \overline{L_n} \vee X$ where $X = \Box$ if $L_{n+1} = \mathsf{F}$ and $X = L_{n+1}$ otherwise. It is clear from the construction in Definition 5.2.4 that the rules in the initial inference system for the (definite!) theory $\mathcal{T}$ all have clause forms which are definite. Inspection of the transformation system *Lin* (Figure 5.7) reveals that this remains invariant under application of any of its transformation rules. Hence, the possibilities (a) and (d) – (g) for $P \to C$ cannot apply, because their respective clause forms all are not definite ((d) – (g) are even not Horn).

It remains to consider cases (b) and (c). In case (b) the considered inference step in R must be of the form $A_j \xRightarrow{A_1 \cdots A_{j-1} A_{j+1} \cdots A_n} {}_{P \to C} B$, for some j $(1 \leq j \leq n)$. Now, contra-definite rules never have a negative conclusion literal. Hence, A_j must either (1) be the top literal of R or (2) be obtained by an inference step with a non contra-definite rule. Case (1) is impossible,

since the top literal is given as a negative literal, and case (2) leads to a contradiction to the selection of $P \to C$ as the *leftmost* non contra-definite rule in R. Thus, case (b) cannot apply.

In case (c) the same argumentation applies and hence is omitted. Now since all cases lead to a contradiction we conclude that the assumption must be wrong, and that R can use contra-definite inference rules only.

5.6.3 First-Order Completeness

Besides the combination result with theory model elimination (Theorem 4.5.3), the next theorem is the main result for linearizing completion, and its proof employs most of the material presented in this chapter.

Theorem 5.6.4 (Completeness wrt. Minimal $\mathcal{T}$-MGRs). *Let $\mathcal{T}$ be a theory, and suppose that $\mathcal{I}$ is an inference system which is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$ and that $\mathcal{I}$ is completed wrt. the transformation system Lin (or, somewhat weaker, $\mathcal{I}$ is the limit $\mathcal{I} = \mathcal{I}_\infty$ of a fair Lin-deduction*

$$\mathcal{I}_0(\mathcal{T}) \vdash_{Lin} \mathcal{I}_1 \vdash_{Lin} \mathcal{I}_2 \cdots) \ .$$

Let M be a minimal $\mathcal{T}$-complementary literal set, $T \in M$, and suppose γ is a $\mathcal{T}$-refuter for M. Then there is a first-order background refutation

$$D = (T \Rightarrow^*_{NonRed(\mathcal{I})^-,M,\sigma} \Box)$$

with computed answer substitution $\sigma \leq \gamma\,[Var(M)]$.

Stated differently, this theorem guarantees that linearizing completion can be used as a complete $\mathcal{T}$-unification procedure (Def. 4.4.1). The relevance of this result is stated in Note 4.4.3, where it was argued for the possibility of using complete $\mathcal{T}$-unification procedures as background reasoners within total theory model elimination (TTME-MSR) or theory resolution [Stickel, 1985; Baumgartner, 1992b].

Proof. Since γ is a minimal $\mathcal{T}$-refuter, $M\gamma$ is, by definition, minimal $\mathcal{T}$-complementary. Hence also the instance $L\gamma$ of $L \in M$ is contained in $M\gamma$.

By definition, $M\gamma$ is $\mathcal{T}$-complementary iff $\exists M\gamma$ is $\mathcal{T}$-unsatisfiable. Clearly, all existentially quantified variables can now be Skolemized away, which preserves $\mathcal{T}$-unsatisfiability (See Note 4.2.3 for a proof of this fact). Let γ' be such a Skolemizing substitution which replaces all variables in $M\gamma$ by constants not occurring in $M\gamma$. Hence $M\gamma\gamma'$ is ground and is $\mathcal{T}$-unsatisfiable. By Corollary 5.6.2 and the definition of ground completeness, Def. 4.5.6, there is a background refutation of $M\gamma\gamma'$ with top literal $L\gamma\gamma'$ with respect to the inference system $NonRed(\mathcal{I})^-$. Finally, we apply the lifting lemma (Lemma A.2.9) to obtain a background refutation on the first-order level. We are not quite done, because for the computed answer σ it holds (for some δ) that $\sigma\delta = \gamma\gamma'\ [Var(M)]$, but not $\sigma\delta = \gamma\ [Var(M)]$ as required.

However, recall that γ' is a Skolemizing substitution, i.e. it is of the form $\gamma = \{x_1 \leftarrow a_1, \ldots x_n \leftarrow a_n\}$, where the a_i's are new and pairwise different constants. Therefore, the inverse $\rho = \{a_1 \leftarrow x_1, \ldots a_n \leftarrow x_n\}$ exists, and it holds $\sigma\delta\rho = \gamma\gamma'\rho = \gamma\ [Var(M)]$. In other words, $\sigma \leq \gamma\ [Var(M)]$ as desired. A more explicit proof can be carried out analogously to the answer completeness of TTME-MSR (Theorem 4.4.3, "Part 4: Lifting").

For the weaker variant, observe that by Theorem 5.6.2 $\mathcal{I}_\infty$ is relatively complete wrt. $\mathcal{I}_0(\mathcal{T})$, and by Theorem 5.4.1 $\mathcal{I}_\infty$ is completed wrt. *Lin*. Hence the stronger variant follows from the weaker one.

5.7 Sample Theories

The purpose of this section is to demonstrate the application of linearizing completion to non-trivial examples.

5.7.1 Equality and Paramodulation

We start with the theory of equality alone and then extend it with strict orderings.

Consider the theory of equality in a language without function symbols and a 2-ary predicate symbol P (Figure 5.8).

Equivalence:		*P-Substitution:*	
$\rightarrow x = x$	(Ref=)	$P(x,y),\ x = x' \rightarrow P(x',y)$	(Sub$P$-1)
$x = y \rightarrow y = x$	(Sym=)	$P(x,y),\ y = y' \rightarrow P(x,y')$	(Sub$P$-2)
$x = y,\ y = z \rightarrow x = z$	(Trans=)		

Figure 5.8. The theory of equality in a language with a 2-ary predicate symbol P.

The inference system in Figure 5.9 corresponds to that theory and it is completed wrt. the transformation system *Lin* (Figure 5.7). The notation $x \doteq y$ is a nondeterministic notation for $x = y$ or $y = x$. If part of an inference rule, the rule has to be expanded to both cases. This system was obtained as the result of a fair deduction, starting from the initial inference system (Definition 5.2.4) associated with the theory. There, we gave a weight of 0 to every rule. The system in Figure 5.9 was then obtained semi-automatically with the assistance of an implementation (called *LC*, see Section 6.0.5) in the following way: first we added the following contrapositives manually by application of the Contra transformation rule:

$\neg(x = z),\ y = z \rightarrow \neg(x = y)$	Contrapositive of (Trans=)
$\neg P(x',y),\ x = x' \rightarrow \neg P(x,y)$	Contrapositive of (Sub$P$-1)
$\neg P(x,y'),\ y = y' \rightarrow \neg P(x,y)$	Contrapositive of (Sub$P$-1) .

Equivalence:		*P-Substitution:*
$x = y,\ \neg(x = y) \to \mathsf{F}$	(Syn=)	$P(x,y),\ \neg P(x,y) \to \mathsf{F}$
$\neg(x = x) \to \mathsf{F}$	(Ref=)	$P(x,y),\ x \doteq x' \to P(x',y)$
$x = y \to y = x$	(Sym=-1)	$P(x,y),\ y \doteq y' \to P(x,y')$
$x = y,\ y = z \to x = z$	(Trans=-1)	$\neg P(x,y),\ x \doteq x' \to \neg P(x',y)$
$\neg(x = z),\ y \doteq z \to \neg(x = y)$	(Trans=-2)	$\neg P(x,y),\ y \doteq y' \to \neg P(x,y')$

Figure 5.9. A completed inference system for equality without function symbols.

All these rules were given the weight 0. Then the mandatory transformation rules of *Lin*, i.e. Deduce, Unit1 and Unit2 were applied repeatedly in an exhaustive way, until no more non c-redundant inference rules could be generated. This part of the construction was carried out automatically. The generated rules were given weights of somewhat above 0.

Alternatively to this semi-automatic processing, the LC tool can be run in a fully automatic setting. Then the same system results as in Figure 5.9 except that additionally the rule $\neg(x = y) \to \neg(y = x)$ is generated as a contrapositive of the (Sym=-1) rule. This happens due to a particular heuristic built into LC, which builds a contrapositive rule if it can successfully be applied in the redundancy proofs of two or more other inference rules.

Note 5.7.1 (Usefulness of the Contra *Transformation Rule).* The completed system in Figure 5.9 also serves as an example for a useful application of the Contra rule, which lies in the proof of redundancy: applying Deduce to $x = y, y = z \to x = z$ and $x = z, \neg(x = z) \to \mathsf{F}$ results in $x = y, y = z, \neg(x = z) \to \mathsf{F}$. In order to avoid an *infinite* chain of applications of Deduce we would like to show that $x = y,\ y = z,\ \neg(x = z) \to \mathsf{F}$ is redundant. This is possible due to the contrapositive (Trans=-2). For the crucial case with $\neg(x = z)$ as top literal the redundancy proof is by the derivation

$$\neg(x = z) \overset{y=z}{\Longrightarrow}_{\text{(Trans=-2)}} \neg(x = y) \overset{x=y}{\Longrightarrow}_{\text{(Syn)}} \mathsf{F} \ .$$

Hence, the inference rule $x = y,\ y = z,\ \neg(x = z) \to \mathsf{F}$ is redundant and can be deleted.

How to efficiently mechanize the equality relation has been widely studied in the literature. It may thus be interesting how the inference system in Figure 5.9 relates to well-known approaches.

The most prominent approach to dealing with equality is certainly *paramodulation* [Robinson and Wos, 1969] and its refinements (see Section 4.1.2). Briefly, the inference system in Figure 5.9 reflects the linear paramodulation calculus for equational theories (see e.g. [Furbach *et al.*, 1989c]) without function symbols.

Linear paramodulation proceeds by repeated subterm replacement in a given goal equation $\neg(s = t)$ until a trivial goal of the form $\neg(s = s)$ has been reached. In order to obtain a corresponding refutation in our system, we have to start with the top literal $\neg(s = t)$. Subterm replacement in paramodulation is mirrored in our system by the (Trans=-2) inference rule, while the derivation of the trivial goal in paramodulation is mirrored by an application of the (Ref=) inference rule. Note that according to the (Trans=-2) inference rule it is sufficient to paramodulate into the right hand side of a negative equation. Technically this is realized by the absence of certain contrapositives of the (Trans-=) inference rule.

Indeed, for *negative* goal equations, (Trans=-2), (Syn=) and (Ref=) are sufficient. The formal account for this is the second part of Corollary 5.6.2. Further, it is easy to see that (Syn=) is redundant for derivations, because its applications can be replaced by a two step derivation using (Trans=-2) and (Ref=).

When positive used as a rule of inference within a linear calculus such as model elimination, paramodulation must also be defined for *positive* goal literals (see [Loveland, 1978]). This is already achieved in our system by means of the (Trans=-) and (Sym=-1) rules.

Things become more complicated when function symbols are involved. For the sake of simplicity assume a single 2-ary function symbol f as given. This implies for the theory of equality additional substitution axioms:

$$x = x' \rightarrow f(x, y) = f(x', y)$$
$$y = y' \rightarrow f(x, y) = f(x, y') \ .$$

The thus enhanced theory can then be completed in a way similar to above. The resulting system is *in*finite and contains rules like

$$x = x', \ \neg(f(f(x, y), z) = w) \rightarrow \neg(f(f(x', y), z) = w) \qquad \text{(Inst-Par)} \ .$$

In general, such rules for subterm replacement are generated for arbitrary depth. These rules have also a counterpart in linear paramodulation: it is well-known (see e.g. [Hölldobler, 1989]) that in linear paramodulation the *functional reflexive axioms, i.e. axioms of the form* $f(x, y) = f(x, y)$ are necessary for completeness. Equivalently, the additional inference rule *instantiation* can be used instead (see again [Hölldobler, 1989]). It is straightforward to show that an application of an instantiation inference rule, followed by a paramodulation step, has the same result as carrying out a first-order inference with a respective inference rule such as (Inst-Par).

These considerations about paramodulation lead us to the following conclusion: linearizing completion did not discover an essentially new calculus for equality treatment. However, it succeeded in re-inventing a well-known and fairly efficient calculus, namely linear paramodulation. Furthermore, this was done in an automatic way, and the completeness of linear paramodulation

is obtained as an instance of the general completeness for PTME-I (Theorem 4.5.3).

We conclude with a note on RUE-resolution [Digricoli and Harrison, 1986]. Since our approach is successful on rediscovering paramodulation, the question arises whether the inference rules of RUE-resolution could be derived as well. It seems that this is not possible, mainly because in RUE resolution the conclusion of the inference rules are *clauses* but not single literals. However, this unit-resulting property is central to our approach.

5.7.2 Equality plus Strict Orderings

As a generalization of the above system for equality (Figure 5.9) consider example 5.2.2 again. Applying a fair *Lin*-deduction to the initial system $\mathcal{I}_0(\mathcal{ES})$ results in the (infinite) completed inference system depicted in Figure 5.10. The strategy for deriving this system was the same as in Section 5.7.1 above. Note that not all contrapositives of the (Trans<) axiom have to be generated.

Equivalence:

$$x = y,\ \neg(x = y) \to \mathsf{F}$$
$$\neg(x = x) \to \mathsf{F}$$
$$x = y \to y = x$$
$$x = y,\ y = z \to x = z$$
$$\neg(x = z),\ y \doteq z \to \neg(x = y)$$

f-Substitution:

$$x \doteq x',\ f^i(x) = y \to f^i(x') = y$$
$$y \doteq y',\ x = f^i(y) \to x = f^i(y')$$
$$x \doteq x',\ \neg(f^i(x) = y) \to \neg(f^i(x') = y)$$

Strict Order:

$$x < y,\ \neg(x < y) \to \mathsf{F}$$
$$x < x \to \mathsf{F}$$
$$x = y \to \neg(x < y)$$
$$x < y \to \neg(y < x)$$
$$x' < x,\ x < y \to x' < y$$
$$x < x',\ \neg(x < y) \to \neg(x' < y)$$

≤-Substitution:

$$x \doteq x', x < y \to x' < y$$
$$x \doteq x', f^i(x) < y \to f^i(x') < y$$
$$y \doteq y', x < y \to x < y'$$
$$y \doteq y', x < f^i(y) \to x < f^i(y')$$
$$x \doteq x', \neg(x < y) \to \neg(x' < y)$$
$$x \doteq x', \neg(f^i(x) < y) \to \neg(f^i(x') < y)$$
$$y \doteq y', \neg(x < y) \to \neg(x < y')$$
$$y \doteq y', \neg(x < f^i(y)) \to \neg(x < f^i(y'))$$

Figure 5.10. A completed inference system for the theory $\mathcal{ES}$ of equality with function symbols and strict orderings. Inference rules containing $f^i(x)$ have to be replaced by all i-fold instances $\underbrace{f(f(\cdots f}_{i}(x)))$, for $i > 0$.

5.7.3 Modal Logics

In this section I will argue that linearizing completion can be used to support reasoning in modal logics. The theories to be completed will be given by a particular translation approach of (propositional) modal logics into predicate logics. I will therefore briefly recall this *semi-functional translation*, and then turn towards linearizing completion.

The Semi-Functional Translation. I will presuppose basic knowledge of modal theory. An introduction can be found in [Fitting, 1993]. The semi-functional translation is due to A. Nonnengart. His thesis [Nonnengart, 1995] provides a much more elaborate presentation (of course) than the brief summary of relevant issues for our case given here.

The semi-functional translation is best explained by starting with the *relational* translation of modal logic formulae, and then identifying the differences. The idea of the relational translation is to relativise the modal operators "$\Box$" and "$\Diamond$" by a "reachability" relation R according to the operators' possible worlds semantics: a formula $\Box\Phi$ ($\Diamond\Phi$) is translated wrt. a world context u in the following way:

$$\begin{aligned} \lceil \Box\Phi \rceil_u &= \forall v\,(R(u,v) \rightarrow \lceil \Phi \rceil_v) \\ \lceil \Diamond\Phi \rceil_u &= \exists v\,(R(u,v) \wedge \lceil \Phi \rceil_v)\ . \end{aligned}$$

This transformation distributes over the quantifiers and junctors, and atoms are to be extended as $\lceil P(t_1,\ldots,t_n)\rceil_u = P(t_1,\ldots,t_n,u)$ (see [Nonnengart, 1995]). The translation for a given input formula Γ starts with $\lceil \Gamma \rceil_\iota$, where ι stands for the initial world.

The disadvantage of this translation is that the clauses are blown up considerably. Even with simple examples theorem provers will have severe difficulties. Thus the purpose of translation into predicate logic — gaining efficiency over, say, Hilbert-style calculi — will be totally negated.

As a solution, H.J. Ohlbach (and others, see [Ohlbach, 1993] for an overview) suggested converting the relational translation into a *functional* one. He observed that whenever a world v is reachable from a world u, i.e. $R(u,v)$ holds, this is equivalent to saying that a function f exists such that $f(u) = v$. This holds under the proviso that the reachability relation is serial, i.e., for every world a reachable world exists[9]. Further, it can be shown[10] that instead of the second-order predicate $\exists f\ f(u) = v$ the first-order predicate $\exists x\ u : x = v$ can be used equivalently (read ":" as the "apply" function). Applying this idea to the result of the relational translation gives the translations

[9] The semantics of first-order logics allows only total functions, which means here that any accessible world is defined, i.e. seriality holds.

[10] It requires only countably many functions to "simulate" a reachability relation over a countable domain.

$$\begin{array}{lcll} \lceil \Box \Phi \rceil_u & = & \forall v\,((\exists x\, u : x = v) \rightarrow \lceil \Phi \rceil_v) & = & \forall x\, \lceil \Phi \rceil_{u:x} \\ \lceil \Diamond \Phi \rceil_u & = & \exists v\,((\exists x\, u : x = v) \wedge \lceil \Phi \rceil_v) & = & \exists x\, \lceil \Phi \rceil_{u:x} \ . \end{array}$$

There is another source for relativized formulas, which are the properties of the modal logic under consideration. It is natural to consider these as a background theory within a theory reasoning calculi. For instance, the reachability relations of $K4$ is just transitivity, $\forall x, y, z\ R(x, y) \wedge R(y, z) \rightarrow R(x, z)$, and its functional translation results in the equation $\forall x, u, v\ (v : u : x = f(x) : x)$ (f is a Skolem function and says that the composition of u and v applied to world x exists). In this case, the background theory is an equational theory[11] and can be treated by a suitable E-unification algorithm. It is important to notice that the nested function symbols in the equational theory stem from the translation of *negative* R-literals.

It would be nice to avoid non-trivial equational theories but at the same time to avoid the drawback of a blown-up clause set as in the relational translation. Here is the point where the semi-functional translation comes in: it treats only the $\Diamond$ operator functionally (because it introduces a *positive* R-literal) and treats the $\Box$ operator relationally (because it introduces a *negative* R-literal). Here is a trivial example:

$$\begin{array}{lcl} \lceil \neg(\Box P \rightarrow \Box P) \rceil_\iota & = & \lceil \Box P \wedge \Diamond \neg P \rceil_\iota \\ & = & \lceil \Box P \rceil_\iota \wedge \lceil \Diamond \neg P) \rceil_\iota \\ & = & (\forall x\, R(\iota, x) \rightarrow P(x)) \wedge \exists y\, \neg P(\iota : y) \end{array}$$

This gives the two clauses $P(x) \leftarrow R(\iota, x)$ and $\neg P(\iota : c)$, where c is a Skolem constant. Obviously, there is no refutation of this trivial theorem possible. The reason is that the link between the reachability relation R and the reachability functions is missing; it is given by the unit clause $R(u, u : x)$, which says that the world $u : x$, for any world u and reachability *function* x is in the reachability *relation* R. Now, these three clauses together allow for a refutation, as expected.

Application of Linearizing Completion. As mentioned previously, different modal logics result in different properties of the reachability relations, which can naturally be taken as background theories, i.e. for linearizing completion. Further, since the $R(u, u : x)$ clause is part of every translation, it seems natural to take it into the theory. Thus, the (serial) modal logic $S4$ with a transitive and reflexive reachability relation yields the background theory

$$\begin{array}{lll} R(u, w) \leftarrow R(u, v) \wedge R(v, w) & (1) & \qquad R(u, u : x) \quad (3) \\ R(u, u) & (2) & \end{array}$$

When fed into the LC tool (see Section 6.0.5), and using the automatic setting, the following completed inference system results (syntactic rules $P, \neg P \rightarrow \mathsf{F}$ omitted):

[11] It can be shown that any Horn theory results in an equational theory.

$$
\begin{array}{rll}
R(u,v), R(v,w) \to R(u,w). & & 1/\mathrm{L}\colon\ 1/\mathrm{W}\colon\ 0/\mathrm{initial(input)} \\
\neg R(u,u) \to \mathsf{F}. & & 2/\mathrm{L}\colon\ 1/\mathrm{W}\colon 499/\mathrm{initial(input)} \\
\neg R(u, u:x) \to \mathsf{F}. & & 3/\mathrm{L}\colon\ 1/\mathrm{W}\colon 499/\mathrm{initial(input)} \\
R(u:x, w) \to R(u,w). & & 5/\mathrm{L}\colon\ 2/\mathrm{W}\colon 499/\mathrm{unit2}(1,\ 3) \\
R(u,v) \to R(u, v:x). & & 6/\mathrm{L}\colon\ 2/\mathrm{W}\colon 498/\mathrm{unit2}(1,\ 3) \\
\neg R(u,w), R(v,w) \to \neg R(u,v). & & 8/\mathrm{L}\colon\ 2/\mathrm{W}\colon 496/\mathrm{contra}(1\ /\ 1) \\
\neg R(u, (u:x_1):x_2) \to \mathsf{F}. & & 9/\mathrm{L}\colon\ 3/\mathrm{W}\colon 500/\mathrm{unit1}(5,\ 3) \\
\neg R(u, v:x) \to \neg R(u,v). & & 10/\mathrm{L}\colon\ 3/\mathrm{W}\colon 494/\mathrm{unit2}(8,\ 3) \\
\neg R(u, ((u:x_1):x_2):x_3) \to \mathsf{F}. & & 12/\mathrm{L}\colon\ 4/\mathrm{W}\colon 500/\mathrm{unit1}(5,\ 9) \\
\neg R(u, (((u:x_1):x_2):x_3):x_4) \to \mathsf{F}. & & 14/\mathrm{L}\colon\ 5/\mathrm{W}\colon 500/\mathrm{unit1}(5,\ 12) \\
\vdots & &
\end{array}
$$

The comments in the right column are output by LC and show a trace how each particular rule is generated. In the term "N/L: l/w: w/*rule*(*params*)", N is the rule number, l is the level where it was generated (LC implements a level saturation strategy), w is the weight, *rule* is the applied deduction rule, and *params* specifies in which way the deduction rule was applied. For instance, "unit1(5,3)" in rule 9 means that rule 9 was obtained by a Unit1 deduction step applied to rules 5 and 3.

This completed system, call it $\mathcal{I}_{S4}$, demonstrates the usefulness of the concept "redundancy for derivations" (Section 5.3.3). All the rules 9, 12, 14, 16,... are of the form $\neg A \to \mathsf{F}$ and hence are not redundant *for completion*, but they are redundant *for derivations*; for instance, for rule 9 the following refutation according to the sufficient completeness criterion (Proposition 5.3.3) exists:

$$\neg R(u, (u:x_1):x_2) \Rightarrow_{(10)} \neg R(u, u:x_1) \Rightarrow_{(10)} \neg R(u,u) \Rightarrow_{(2)} \mathsf{F}\ .$$

Similar refutations exist for the other rules 12, 14, 16,.... Hence, the final system $NonRed(\mathcal{I}_{S4})$ (cf. Def. 5.3.4), which is sufficient for refutational completeness (cf. the completeness results in Section 5.6) can be built by deleting these rules from $\mathcal{I}_{S4}$, one after the other. Notice that although $\mathcal{I}_{S4}$ is *in*finite, $NonRed(\mathcal{I}_{S4})$ is *finite* and contains the rules 1, 2, 3, 5, 6, 8 and 10 only. Of course, this insight requires induction which currently is not implemented. In practice, the completion process is stopped after some levels and then rules redundant for derivations are deleted.

Practical experiments using $NonRed(\mathcal{I})$ are documented in Chapter 6 on page 205. Similar inference systems exist for a great variety of other modal logics.

6. Implementation

In the last few years, several implementations related to this book have been developed at our research group. These are

LC: A tool for linearizing completion. It transforms a given Horn clause set into a completed state, according to the calculus of linearizing completion (Chapter 5). LC was used in establishing the library used by the SCAN-IT:

SCAN-IT: Library handling for completed theories. The SCAN-IT tries to recognize in a given input file a Horn subset for which a respective completed version in a library exists. SCAN-IT recognizes the *structure* of axioms, and thus the naming of the predicate symbols and the order in writing the clauses is irrelevant. If successful, the theory axioms are replaced by the completed version. The resulting file is ready to be fed into PROTEIN:

PROTEIN: A *PRO*ver with a *T*heory *E*xtension *IN*terface. Among others, PROTEIN implements the partial theory model elimination calculus, PTME-I (Section 4.5) and its restart variant PRTME-I (Section 4.6). For search space pruning, *regularity* and *factorization* (Section 3.3) are built in; the *independence of the computation rule* result (also Section 3.3) is needed to guarantee the completeness of the depth-first, left-to-right iterative deepening strategy.

These tools are implemented in ECRC's ECL^{i}PS^e Prolog. The implementational work was mostly carried out by students[1].

An in-depth description of all three of these tools is beyond the scope of this text. I will therefore briefly describe only some key features of LC and SCAN-IT, and then turn towards PROTEIN in greater depth. Finally, I describe the practical experiments we have carried out with our systems.

6.0.4 SCAN-IT

The motivation for the development of SCAN-IT was the observation that inexperienced users often have difficulties in selecting appropriate Horn subsets of the problem in question for completion. It turned out in practical

[1] The whole system is available in the World Wide Web, using the URL `http://www.uni-koblenz.de/ag-ki/Systems/PROTEIN/`.

experience that in most cases it is *not* appropriate to select the whole Horn subset for this. The reason is that in this case the resulting inference systems are often large (> 100 inference rules), even if LC is stopped after two levels (running only one level yields too weak inference systems in practice). Large systems, however, cause a large local search space, with the result that the model elimination tableaux cannot be explored to a significant depth in reasonable time.

Thus, we prepared a library of theories and respective completed versions thereof which turned out to work well in practice. It is this "know-how" which is thus made available to uninformed users by the SCAN-IT utility. As a further advantage, runs of LC — which are quite often time-consuming — can be saved if a completed version of a theory was established beforehand. Currently, the library for SCAN-IT is moderately complete. It includes equality according to Section 5.7.1, several kinds of orderings (strict orders, preorders, see Section 5.7.2), associative theories and group theory. It identifies theories by the *structure* of their axioms. Thus, orderings of literals and the names given to the function and predicate symbols do not matter. This flexibility turned out to be quite useful and necessary in practice. If desired, SCAN-IT invokes LC on the Horn subset of the input file if no matching theory has been found. In the future it is planned to make this selection more intelligent.

6.0.5 LC

A student has implemented the linearizing completion calculus as described in Chapter 5. It runs either interactively as a shell or in a fully automatic setting. Several flags control features concerning termination (i.e. aborting), resources allocated for redundancy tests, profiling, etc.

Fairness is guaranteed by a level-saturation strategy as employed in resolution calculi (see [Chang and Lee, 1973]). The optional application of the Contra transformation rule is controlled by heuristics. The most valuable heuristics is to apply the Contra rule[2] only if it enables proof of *two* or more already present inference rules as redundant for completion.

The attachments of weights to inference rules was also done heuristically as follows: the weight of every inference rule in the initial system is set at 0. This value was chosen heuristically. In general, lower values for weights imply that it is less likely that the rule will be detected as redundant for completion, while it is more likely that the rule is used in the redundancy proof of a different rule. Higher values imply the opposite behavior. The weights of the generated inference rules were determined heuristically: when generated, an inference rule becomes the highest admissible weight such that any redundancy proof employing it still succeeds. Using this highest possible weight increases the chances that it will become redundant later, as the generation of new rules proceeds. An exception to this heuristics are inference rules of

[2] This rule adds a contrapositive of a rule to the database.

the form $\neg A \rightarrow \mathsf{F}$, which are always given a high weight. This facilitates proofs of redundancy for derivations for these rules, which are carried out at the end of a run.

It should be noted that these heuristics are implemented, and no user assignment of weights is necessary. The comments in the completed inference system in the modal logic example of Section 5.7.3 give an impression of the trace output of LC.

6.1 PROTEIN

Efficiency is not such a concern for LC, because it is run at "compile time", or even before. The SCAN-IT is uncritical in this respect because the deductive tasks are not very deep and can be controlled quite well. Clearly the most critical component in this chain is PROTEIN.

6.1.1 High Inference Rate Based Theorem Proving

As exemplified by the METEOR [Astrachan and Stickel, 1992] and SETHEO [Letz *et al.*, 1992] systems, high inference rates still seem to be important for model elimination-based theorem provers. The "cheapest" way to achieve high inference rates is to use the *PTTP* (*P*rolog *T*echnology *T*heorem *P*roving) implementation technique [Stickel, 1988; Stickel, 1989]. In the PTTP approach Prolog is viewed as an "almost complete" theorem prover, which has to be extended by only a few ingredients in order to handle the non-Horn case. By this technique, the benefits of optimizing Prolog compilers are accessible to theorem proving.

However, arguing for high inference rates alone easily results in a too optimistic assessment of the power of PTTP provers. Stickel points out [Stickel, 1990a]: "the high inference rate can be overwhelmed by its exponential search space". As a consequence, PTTP, at least in its original formulation, is often better suited for problems having a moderate search space, e.g. those with a tree-shaped dependency graph.

This observation was taken advantage of in [Tarver, 1990]. There, a heuristic decomposition of the problem in question is proposed. Decomposition is pattern-driven and employs tuples of the form $\langle method, goal, context \rangle$. If the *goal* expression matches the current goal to be proven in context *context*, then the *method* is invoked. For example, in set theory context, the goal $X = Y$ could be rewritten towards the two proof obligations $X \subseteq Y$ and $Y \subseteq X$. Now, these rewritten goals hopefully enjoy a moderate search space and can be proved by a PTTP prover. The potential of this approach was demonstrated in [Tarver, 1990] using set theory domain.

Other approaches are in a sense complementary in that they improve on the calculi/proof procedures proper. For instance, it was suggested using *Caching* and *Lemmaizing* [Astrachan and Stickel, 1992], *Anti-lemmata*

combined with *Folding-up/Folding-Down* [Christoph Goller and Schumann, 1994], several *ancestor refinements* [Plaisted, 1990], addition of unit lemmas derived by unit-resulting resolution [Schumann, 1994], *database unification* [Bibel *et al.*, 1994], *Link Deletion* [Mayr, 1995] and *subsumption* techniques [Baumgartner and Brüning, 1997]. Although most of these refinements slow down the inference rates to a certain degree, it could be shown experimentally that in total they pay off quite well. Nevertheless, all these provers are still based on high inference rates inference engines.

In sum, PTTP should thus be seen as a kernel technique which needs some improvements to reduce the search space. Accepting this theorem-proving philosophy, the challenge for us therefore was to generalize the PTTP technique towards the theory case. More specifically, we had to deal with the question how to implement the comparatively complex PTME-I-Ext and PRTME-I-Ext inference rules for partial (restart) theory model elimination, where the theory is given by a completed theory inference system. Further, in order not to lose the search space advantages achievable by theory reasoning due to a poor implementation, PTTP's high inference had to be preserved as much as possible. In order to achieve this, the theory inference systems are compiled to Prolog code much like the foreground clauses. However, we were faced with the difficulty that Prolog's depth-first computation rule had to be overcome in order to achieve execution according to the definition of the theory extension step. In the sequel, I will therefore first briefly review the standard PTTP approach, and then describe the tricks necessary for the theory extension.

6.1.2 The PTTP Implementation Technique

As mentioned, the PTTP-approach transforms a given clause set into a Prolog program. The transformed Prolog program must execute the clauses according to some complete proof procedure. *Model elimination* turns out to be particularly useful for this, since it is, like Prolog, an input proof procedure. In particular, the transformation from the input clauses to Prolog works as follows (more details can be found in [Stickel, 1988], except for the "restart" and "theory reasoning" items, which are original):

Contrapositives. An input clause such as

$$C \vee D \leftarrow A \wedge B$$

is transformed into a Prolog clause

(1) `c :- not_d, a, b.`

This example also shows how negation is treated, namely by making it part of the predicate name. The order of the body literals is determined during compile time. The underlying justification for completeness is nothing but the "independence of the computation rule" (cf. Section 3.3).

In restart model elimination with selection function (Section 4.6.1) it is sufficient to generate one such *contrapositive*. In this example we would have a selection function f which selects only $\{C\}$. If f selects $\{C, D\}$ then the contrapositive

(2) `d :- not_c, a, b.`

would have to be added. In the non-restart variants of model elimination, every literal in a clause can serve as an entry point into the clause. Thus, all contrapositives are needed. In this case these are additionally

(3) `not_a :- not_c, not_d, b.`

(4) `not_b :- not_c, not_d, a.`

In PROTEIN it is up to the user to declare some clauses as *query* clauses which are used as top clauses for the tableau construction. A query clause, say $\leftarrow E \wedge \neg F$, is transformed into

(Q) `query :- e, not_f.`

The new `query` literal is the same for all query clauses. In order to start the search, the Prolog goal

`?- query.`

is invoked.

Sound unification. Prolog's unsound unification has to be replaced by a sound unification algorithm. This can either be done by directly building-in sound unification into the Prolog implementation (as is available in ECL^{i}PS^e) , or by reprogramming sound unification in Prolog and calling this code instead of Prolog's unsound unification.

Search strategy. A complete search strategy is needed. Usually depth-bounded iterative deepening is used. The strategy can be compiled into the prolog program by additional parameters, being used as "current depth" and "limit depth". The cost of an extension step can be uniformly 1 (depth-bounded search), or can be proportional to the length of the input clause (inference-bounded search). In PROTEIN, the depth-bounded search proved to be superior in most cases.

Reduction steps. The model elimination reduction operation has to be implemented. This can be realized by memorizing the subgoals solved so far (the A-literals) as a list in an additional argument, and by Prolog code that checks a goal for a complementary member of that list. Of course, this check has to be carried out with sound unification.

The Prolog clause (1) from above then looks like

(1') `c(Anc) :- not_d([d|Anc]), a([-a|Anc]), b([-b|Anc]).`

where `Anc` is a Prolog list which contains the ancestor literals (called A-literals in Loveland's model elimination (cf. Note 3.2.1) [Loveland, 1968]); the query clauses from above becomes

(Q) `query(Anc) :- e(Anc), not_f(Anc).`

and the Prolog goal becomes `?- query([]).`. The code for reduction steps then looks like[3]

(Red-c) `c(Anc) :- member(c, Anc).`

(Red-not_c) `not_c(Anc) :- member(-c, Anc).`

Thus, the reduction step code has to be generated for each predicate symbol.

Restart Model Elimination. The modification to obtain (strict) restart model elimination (Section 4.6) is minimal: one only has to replace the code for reduction steps at positive literals, i.e. (Red-not_c), by the following call to the query clauses:

(Restart-not_c) `not_c(Anc) :- query(Anc).`

Recall that the `query` procedure accesses all clauses declared as "query". Hence, in order to obtain a complete calculus (Theorem 4.5.3), every negative clause has to be declared this way.

Theory Reasoning. Theory reasoning has to be incorporated. We are primarily interested in partial theory model elimination calculi, where the theory inferences are described by theory inference systems (Sections 4.5 and 4.6). The necessary adaptions for the PTTP approach are described in the following.

We concentrate on the translation of a typical inference rule. Hence let

$$\neg E,\ C,\ F \rightarrow \neg G.$$

be given. We recall from Definition 4.5.4 and the subsequent discussion the operational semantics of inference rules: for the stated rule, a PTME-I-Ext step consists of extending a leaf literal $\neg E$ by $\neg G$ in presence of the extending literals C and F, which in turn are taken from the ancestor context of the leaf $\neg E$, or from extending clauses. Of course, since the PTME-I-Ext rule is symmetrical wrt. the premise literals of the used inference rules, two more possibilities exist. In the sequel we will describe the first possibility only.

A first idea would be to transform the given inference rule into the prolog clause (ancestor lists left away, for simplicity):

(R) `e :- not_c, not_f, g.`

However, this approach would not work for two reasons: first, the solution of `not_c` includes the possibility to call Prolog procedures stemming from other *inference rules*, for instance $C \rightarrow \mathsf{F}$. This, however, is not in accordance with the semantics of inference rules, which requires the literals C and F to be resolved away against ancestor literals or extending literals from input *clauses*.

A second problem is the order of execution of the subgoals of (R). In the current translation the body of (R) is solved in the following order:

[3] The membership predicate `member` is defined "as usual":

```
member(X,[X|Rest]).
member(X,[Y|Rest]) :- member(X,Rest).
```

1. solve the goal `not_c`.
2. solve the goal `not_f`.
3. solve the goal `g`.

The problem is the recursive solving in step 1 before step 2. If, for instance, `not_c` is solved by the Prolog procedure corresponding to input clause $C \leftarrow G$, the body G would be solved *before* the `not_f` goal is solved. Thus, the PTME-I-Ext inference rule would not be implemented correctly (while this might not be considered an issue for ground problems, but it certainly is when variables are present). In other words, the usual depth-first left-to-right Prolog strategy has to be circumvented.

A correct respective translation of the investigated inference rule is as follows (this time ancestor lists included):

(R') `e(Anc) :- theory([c,f], Anc), g([-e|Anc]).`

The `theory` procedure has to collect the passed literals from either (1) ancestor literals or (2) from input clauses. Notice that due to the result on the "order of extending clauses" on page 91 only one single permutation of the argument list to the `theory` call suffices.

To realize case (1) one single Prolog clause suffices:

(Th-Anc)
```
theory([Lit|RestLits], Anc) :-
        member(Lit, Anc),
        theory(RestLits, Anc).
```

For case (2) the condition that the rest literals of the extending clauses are not solved during theory extension has to be obeyed. This is achieved by additionally transforming every input clause, say $C \vee D \leftarrow A \wedge B$ from above, into the form:

(Th-1')
```
theory([c|RestLits], Anc) :-
        theory(RestLits, Anc),
        not_d([d|Anc]), a([-a|Anc]), b([-b|Anc]).
```

Notice that the call to the rest literals of the input clause is postponed until all theory literals are resolved away.

Finally, the search for theory literals has to be terminated:

(Th-End) `theory([], _Anc).`

In sum, the modified transformation lets (R') behave as follows:

1. get a literal `c` either from the ancestor list or an input clause. In the latter case let $R_{\mathtt{c}}$ be the subgoals stemming from the rest clause.
2. get a literal `f` either from the ancestor list or an input clause. In the latter case let $R_{\mathtt{f}}$ be the subgoals stemming from the rest clause.
3. Solve the subgoals $R_{\mathtt{f}}$.
4. Solve the subgoals $R_{\mathtt{c}}$.
5. solve the goal `g`.

This behavior is in accordance with the operational semantics of the PTME-I-Ext inference rule.

This concludes the description of the PTTP transformation as is realized in our PROTEIN prover. Further modifications, such as the extraction

of answers (Def. 3.2.5), regularity (Def. 3.3.2), factorization (Section 3.3.3), ground reduction steps (Section 3.3.4) and the combined connection calculus - model elimination calculus (Note 4.3.2) are straightforward, and hence omitted.

Finally, I want to refer to the work of [Neugebauer and Petermann, 1995] where a language is proposed to specify inference rules (e.g. extension, reduction, factorization, equality handling etc.) for model elimination-based theorem proving. By this, the translation process just explained can be described in a more declarative way, which facilitates the construction of respective provers.

6.2 Practical Experiments

Running practical experiments and comparing runtime results is a widely used technique to evaluate the power of theorem proving systems. Our investigations are biased towards assessing the potential of theory reasoning according to the linearizing completion approach. I will first describe the general setup taken for all experiments, and then comment on the results.

As the theory reasoning prover we used PROTEIN as described above. In order to see the relative advantages of theory reasoning vs. non-theory reasoning PROTEIN was run in the following calculi settings: model elimination (ME, Def. 3.2.3), restart model elimination (RME, Def. 4.6.3 and Note 4.6.4), partial theory model elimination (PTME-I, Def. 4.5.4) and partial restart theory model elimination (PRTME-I, Def. 4.6.3). Further, to get an impression of the difficulty of the investigated problems we also run SETHEO (version 3.2.5) and OTTER (version 3.0.4). All provers were run in the default mode.

PROTEIN in its default mode employs regularity (cf. Def. 3.3.2 for PTME-I and Section 4.6.3 PRTME-I) and the ground version of factorization (Section 3.3.3).

SETHEO is a highly developed model elimination prover, featuring in its default mode subgoal reordering, purity deletion, anti-lemmas, folding-up, regularity, tautology and subsumption constraints (in [Letz *et al.*, 1994; Christoph Goller and Schumann, 1994] some of these are described).

OTTER is a state-of-the-art resolution prover. Its numerous flags are set automatically in the "autonomous mode" according to built-in heuristics. In the experiments below OTTER chooses hyper resolution as the primary inference rule, possibly augmented by special inference rules for equality handling such as paramodulation and demodulation.

The columns in Tables 6.2, 6.2 and 6.3 below are labeled with these provers respectively. The entries are to be read as follows: the timing results are given in seconds and are obtained on a SUN SPARCstation 20/712 (2 SuperSparc processores, 70 Mhz, with 1 MB cache, Solaris 2.5.).

The entries "#*Inf*." give the total number of inferences carried out in the proof search of the model elimination based provers. Since both PROTEIN

and SETHEO enumerate derivations via an iterative deepening backtracking regime these numbers can easily get quite high. When comparing the values of PROTEIN to the values of SETHEO, one should keep in mind that the *theory* versions of PROTEIN employ more complex inference steps due to the underlying multiset unification problems (recall from Def. 4.5.4 that the task in each PTME-I-Ext step is to simultaneously resolve away all premise literals of the chosen theory inference rule). Our implementation of this search problem is straightforward and is of complexity $O((|b|+|L|)^{n-1})$, where $|b|$ is the length of the branch to be extended, L is the number of literal occurrences in the input clause set, and n is the number of premise literals of the inference rule in question.

In the examples below, this value for n is one or two in most cases, three in few cases, and four and greater very rarely.

The entries "$\#E + R + F$" for ME denote the number of extension, reduction and factorisation steps, respectively, in the refutation. Similarly, for the other calculi, r and T mean restart steps and theory steps different to those which are also ordinary ME extension and reduction steps.

The TPTP library (version 1.2.0) [Sutcliffe *et al.*, 1994] contains thousands of problems for automated theory reasoning from various problem domains. In our selection we concentrate on some of them which are moderately difficult for at least one of PROTEIN in non-theory version, OTTER or SETHEO, and which can be solved better using theory reasoning. Of course, there are numerous problems not mentioned here which are very easy for all provers, or where theory reasoning does not help very much, or which are easy for resolution provers but hard for model elimination based provers (or vice versa).

In the TPTP library the input clauses are classified as being either an *axiom*, a *hypothesis* or a *theorem* clause. This suggests the assumption (which is in fact wrong occasionally) that the axiom plus hypothesis together constitute a satisfiable set, i.e. a program. But if so, it suffices according to our completeness result for PTME-I (Theorem 4.5.3) to consider only the theorem clauses as queries (Def. 3.2.5) for the proof search (for the restart variant we addionally need to have that the theorem clauses are negative, which is the case for most examples). This restriction is attractive as it cuts down the search space. However, in the experiments we did *not* do so and allowed any negative clause as query. We did this to ensure completeness, and to make our results comparable to SETHEO which uses the same policy.

For the theory versions of PROTEIN (PTME-I and PRTME-I) we relied on the SCAN-IT program (see above) to select an appropriate Horn-subset of the input clause set and to replace it by a previously completed theory. Typically, the completed inference system consisted of 3-50 rules.

The example referred to as *Non-obvious*[4] in Table 6.2 is taken from the October 1986 Newsletter of the *Association of Automated Reasoning*. The selected theory here consists of a transitive and symmetric relation p and a transitive relation q, and the completed system is finite. All other examples in Figure 6.2 employ equality. For the SYN examples the completion is finite, as no function symbols are present. For the other PUZ and GEO examples a finite approximation of the infinite completed system was used (in Section 5.7.1 it was explained how equality is treated by linearizing completion).

The Wos examples (GRP) in Table 6.2 are from group theory. Notably, it suffices to use the *same* theory to prove all examples. Here we took equality and the associativity of the group operation. The BOO examples are from the boolean algebra domain. Here, equality again is the selected theory.

In Section 5.7.3 the application of linearizing completion to a background theory stemming from the semi-functional translation of S4 is described. Figure 6.3 reports about results.

We interpret the results in Tables 6.2, 6.2 and 6.3 as follows: when comparing ME to RME there are examples where either one of these is better suited. However, ME finds quite often a refutation where RME cannot. In sum, ME seems to be the better "default" calculus. This observation carries over to the theory case, i.e. when comparing PTME-I to PRTME-I: PTME-I wins in allmost all cases significantly.

On the other hand, PROTEIN currently does not take full advantage of the potential of *restart* ME, namely the partial proof confluence (cf. Section 4.6.3). It is conceivable that PROTEIN can substantially be improved in this way.

When comparing ME to its theory version PTME-I it is apparent that in almost all examples theory reasoning yields substantial improvements, and it is counterproductive only occasionally. Quite often, theory reasoning enables PROTEIN to find a proof at all. The same holds when relating RME to its theory version PRTME-I. I regard these observations as a strong support for the usefulness of the chosen approach.

Although a prototypical implementation, PROTEIN can even compete with SETHEO and OTTER. On some examples, PROTEIN equipped with theories performs significantly better than SETHEO and/or OTTER. The good results for OTTER (e.g. for the Wos examples) are partially explained by its demodulation based equality handling, which is not developed that far for model elimination.

We note again that SETHEO features numerous calculus improvements (mentioned above) which are not built into PROTEIN, but which are the source for SETHEO's power. For instance, for the modal logics examples SETHEO has no problems at all, quite unlike PROTEIN in ME setting. It can be expected that many of these improvements, such as anti-lemmata and

[4] Entries such as MSC006-1 refer to the respective TPTP-names [Sutcliffe *et al.*, 1994].

	PROTEIN				SETHEO	OTTER
	ME	**RME**	**PTME-I**	**PRTME-I**	**ME**	**auto**
Example	*Time*(*sec.*) #*Inf.* $\#E+R+F$	*Time*(*sec.*) #*Inf.* $\#E+r+R+F$	*Time*(*sec.*) #*Inf.* $\#E+R+F+T$	*Time*(*sec.*) #*Inf.* $\#E+r+R+F+T$	*Time*(*sec.*) #*Inf.* $\#E+R+F$	*Time*(*sec.*)
Non-obvious MSC006-1	< 1 1495 $24+6+0$	< 1 2885 $29+8+10+0$	< 1 1947 $3+5+0+11$	< 1 2589 $5+7+5+0+8$	< 1 4349 $16+2+1$	1.2
Pelletier 48 SYN071-1	< 1 2678 $23+5+3$	< 1 8309 $18+3+5+3$	< 1 2233 $3+2+0+8$	< 1 2172 $2+4+6+0+15$	< 1 5102 $12+2+1$	< 1
Pelletier 49 SYN072-1	$> 1h$	$> 1h$	1.5 12185 $3+3+0+12$	< 1 2219 $5+6+2+0+11$	3.1 152499 $32+6+2$	4.2
Pelletier 55 PUZ001-2	25 39290 $35+2+3$	337 4286026 $23+2+0+0$	10 67060 $11+0+0+5$	167 1127314 $8+6+3+2+4$	15 631703 $23+2+0$	< 1
MarsVenus1-1 PUZ006-1	$> 1h$	$> 1h$	48 468940 $31+3+0+3$	$> 1h$	4.3 138037 $23+1+4$	< 1
MarsVenus2-1* PUZ007-1	$> 1h$ ($> 1h$)	$> 1h$ ($> 1h$)	7.2 (26) 58001 $32+4+2+3$	$> 1h$ ($> 1h$)	(3.0) 151517 $26+2+4$	< 1
BtwnSymm-1* GEO001-1	44 ($> 1h$) 606340 $13+0+0$	478 ($> 1h$) 6396470 $13+0+0+0$	< 1 (98) 3174 $10+0+0+1$	< 1 (1.1) 1165 $10+0+0+0+1$	($> 1h$)	14
BtwnSymm-3* GEO001-3	$> 1h$ ($> 1h$)	$> 1h$ ($> 1h$)	17 ($> 1h$) 228531 $6+0+0+1$	3.5 ($> 1h$) 38587 $6+0+0+0+1$	($> 1h$)	21

Figure 6.1. Runtime results for various provers on selected TPTP problems. In problems marked with a * only clauses marked as "theorem" in the TPTP library are used as queries. Standard policy results are in parenthesis.

	PROTEIN				SETHEO	OTTER
	ME	**RME**	**PTME-I**	**PRTME-I**	**ME**	**auto**
Example	*Time*(*sec.*) #*Inf.* $\#E+R+F$	*Time*(*sec.*) #*Inf.* $\#E+r+R+F$	*Time*(*sec.*) #*Inf.* $\#E+R+F+T$	*Time*(*sec.*) #*Inf.* $\#E+r+R+F+T$	*Time*(*sec.*) #*Inf.* $\#E+R+F$	*Time*(*sec.*)
Wos 4 GRP008-1	5.2 83866 $9+4+0$	3.9 69626 $8+0+2+1$	<1 400 $2+2+1+2$	<1 214 $3+0+4+0+2$	<1 350 $9+4+0$	$>1h$
Wos 10 GRP001-1	$\Longrightarrow$	11 12331 $19+0+0$	$\Longrightarrow$	1.9 10741 $2+0+0+0+7$	273 1355089 $14+0+0$	<1
Wos 11 GRP013-1	$\Longrightarrow$	3.2 40594 $11+0+0+0$	$\Longrightarrow$	<1 2268 $2+0+0+0+2$	<1 18040 $11+0+0$	<1
Wos 15 GRP035-3	$\Longrightarrow$	131 1644878 $16+0+0+0$	$\Longrightarrow$	18 120053 $10+0+0+0+2$	3.0 104817 $16+0+0$	<1
Wos 16 GRP036-3	$\Longrightarrow$	116 1554761 $7+0+0$	$\Longrightarrow$	<1 31 $5+0+0+0+1$	2.4 113524 $7+0+0$	<1
Wos 17 GRP037-3	$\Longrightarrow$	$>1h$	$\Longrightarrow$	<1 557 $6+0+0+0+1$	5.6 154570 $9+0+0$	<1
MultIdem-4 BOO003-4	$\Longrightarrow$	73 932647 $17+0+0$	$\Longrightarrow$	<1 2999 $1+0+0+0+5$	17 1109319 $13+0+0$	80
AddIdem-2 BOO004-2	$\Longrightarrow$	$>1h$	$\Longrightarrow$	5.7 12858 $1+0+0+0+5$	271 19194615 $13+0+0$	$>1h$
AddIdem-4 BOO004-4	$\Longrightarrow$	81 1186132 $17+0+0$	$\Longrightarrow$	<1 2445 $1+0+0+0+5$	24 1597936 $13+0+0$	$>1h$

Figure 6.2. Runtime results for various provers on TPTP problems on group theory and boolean algebra. The entry $\Longrightarrow$ means that these examples are Horn, and hence the results are the same as in the next column.

	PROTEIN				SETHEO	OTTER
	ME	**RME**	**PTME-I**	**PRTME-I**	**ME**	**auto**
Example	$Time(sec.)$ $\#Inf.$ $\#E+R+F$	$Time(sec.)$ $\#Inf.$ $\#E+r+R+F$	$Time(sec.)$ $\#Inf.$ $\#E+R+F+T$	$Time(sec.)$ $\#Inf.$ $\#E+r+R+F+T$	$Time(sec.)$ $\#Inf.$ $\#E+R+F$	$Time(sec.)$
$\Box\neg p \leftrightarrow (\Box(p \rightarrow q)$ $\wedge\Box(p \rightarrow \neg q))$	< 1 187 $35+18+6$	< 1 332 $29+6+24+6$	< 1 268 $15+18+6+20$	< 1 265 $9+6+24+6+20$	< 1 183 $16+6+3$	< 1
$\Diamond\Box p \leftrightarrow \Diamond\Box\Diamond\Box p$	6.7 14435 $31+2+0$	428 896969 $33+2+4+0$	17 27121 $6+2+0+21$	35 62695 $6+2+4+0+27$	< 1 615 $31+2+2$	2.3
$(\Box p \vee \Box q) \leftrightarrow$ $\Box(\Box p \vee \Box q)$	< 1 816 $61+10+2$	4 9503 $62+4+12+2$	< 1 218 $10+10+2+25$	< 1 906 $8+4+12+12+38$	< 1 1117 $28+7+6$	4.3
$\Diamond\Box(\Diamond p \leftrightarrow \Box\Diamond p)$	< 1 410 $28+2+0$	1.2 2184 $26+2+4+0$	< 1 71 $4+2+0+13$	< 1 166 $2+2+4+0+13$	< 1 1148 $21+1+3$	1
$\Diamond\Box(\Box p \leftrightarrow \Diamond\Box p)$	< 1 770 $24+2+0$	< 1 229 $24+0+2+0$	< 1 31 $2+2+0+8$	< 1 31 $2+0+2+2+8$	< 1 1410 $29+2+5$	< 1
$\Diamond\Box(\Box(p \vee \Box q)$ $\leftrightarrow (\Box p \vee \Box q))$	418 762705 $78+13+2$	$> 1h$	< 1 1203 $9+11+2+30$	6.3 10709 $6+4+18+0+36$	< 1 12153 $29+4+14$	355
$\Diamond\Box(\Box(p \vee \Diamond q)$ $\leftrightarrow (\Box p \vee \Diamond q))$	18 32536 $57+6+0$	2942 5721417 $95+6+16+4$	22 34446 $19+14+0+54$	74 124411 $6+6+16+4+38$	6.5 35947 $31+4+7$	13
$\Diamond\Box(\Diamond(p \vee \Diamond q)$ $\leftrightarrow (\Diamond p \vee \Diamond q))$	2.5 5630 $29+6+7$	2.0 3667 $23+6+12+7$	< 1 126 $8+6+7+20$	< 1 308 $2+6+12+7+20$	< 1 543 $25+4+4$	1.2
$\Diamond\Box(\Diamond(p \vee \Box q)$ $\leftrightarrow (\Diamond p \vee \Box q))$	398 824206 $36+6+4$	5.5 9420 $32+4+10+4$	87 147965 $6+6+4+16$	471 764535 $2+4+10+4+16$	< 1 601 $28+3+8$	30

Figure 6.3. Runtime results for various provers for S4 theorems.

folding-up, can be adapted towards PTME-I without great problems. It will be exciting[5] to investigate into such combinations!

[5] At least for me.

7. Conclusions

Summary

In Chapter 4 various theory connection calculi were described and related to each other. This comparison was concluded with the partial theory model elimination version and its restart variant as the most restricted versions. These calculi then were combined with the framework of theory inference rules. An abstract completeness criterion for theory inference rules was formulated, such that the combination with the foreground calculi is complete. Moreover, *answer* completeness result were obtained. Answer completeness generalises refutational completeness and is relevant for problem solving applications and, in the case of the restart variant, for disjunctive logic programming purposes. I consider these completeness results as the main results for Chapter 4.

In order to make the chosen framework applicable in practice, the background theory has to be instantiated with a set of theory inference rules satisfying the completenes criterion. However, it is by no means clear or trivial how this can be achieved for a given theory. Therefore, in Chapter 5 a rather general technique — linearizing completion — was presented which fills this gap. As a sample application the combined theory of equality and strict orderings was used. The resulting inference system acts in conjunction with theory model elimination much like a generalization of the well-known paramodulation inference rule towards strict orderings.

Technically, linearising completion allows for the combination of the linear and unit-resulting restrictions. The central operation is to add new inference rules that detour violations of the linearity restrictions in unit-resulting refutations. A redundancy criterion allows deletion of many of the thus added inference rules, which makes the resulting inference systems quite compact.

Two completeness results were given. The one guarantees the intended application as partial background reasoners within theory model elimination; the other is a "stand-alone" first-order completeness result and shows that linearizing completion alone can be used as a complete calculus for Horn clause logic. This result also allows one to use linearizing completion within other calculi for theory reasoning, say theory resolution.

Linearizing completion is fully implemented and runs in cooperation with the theory model elimination theorem prover PROTEIN. On numerous ex-

amples drastic speedups were obtained. Notably, the method works well not only for single selected examples, but also for whole classes of examples. We demonstrated this by completion of a subset of group theory, which resulted in a significant speedup for the Wos examples, and by a successful application in the context of modal logic.

Further work

As always, a calculus can be further improved and fine-tuned. One promising way for our theory ME calculi is to adapt the subsumption techniques in [Baumgartner and Brüning, 1997] towards theory reasoning. As was argued for in [Baumgartner and Brüning, 1997], this is in particular promising for equational problems. In [Baumgartner and Brüning, 1997], however, we used the trivial treatment of equality by using the equality axioms. It can be expected that the more refined equality treatment by linearizing completion (Section 5.7.1) can be improved significantly if subsumption is employed.

Another improvement concerns *regularity*. Currently, regularity (Def. 3.3.2) is defined purely syntactically. It should be possible to define a theory-version of regularity instead. That is, a branch would violate the "$\mathcal{T}$-regularity" restriction if one of its literals is a $\mathcal{T}$-consequence of its ancestor literals.

Now we turn to linearizing completion. Currently, linearising completion is limited to Horn theories. It might be worthwhile designing an extension towards general, non-Horn theories. This would probably result in inference rules with non-unit conclusions. From the technical point of view, the problem is that unit-resulting resolution is not complete for this case. Hence we get a gap in the completeness proof. It is conceivable that splitting into Horn theories helps here. As an alternative, a modification of the linked inference principle might also work (cf. Section 5.1.5).

Even for the Horn case it might be interesting to allow non-unit conclusions in inference rules. I expect that this gives the possibility to obtain finite systems more often, at the the cost of additional branching in the constructed tableaux.

The presentation of linearizing completion was general enough to allow for instantiation with an ordering criteria different from the chosen one; for example one might think of term-ordering restrictions as applied in the term-rewriting paradigm, or the combination of term-ordering restrictions and linearity restrictions. Of course, different restrictions require different transformation systems, but many of the concepts and claims not related to a specific restrictions can be kept.

The availability of unification algorithms for dedicated theories motivates us to extend the method towards "completion modulo a built-in theory" E. By this, the combined theory consisting of the Horn theory and E would be subject to theory reasoning. In order to do so, one has to use E-unification instead of syntactic unification during the completion phase *and* in inferences using the completed system.

The open problem stated in Section 4.3.1 concerning the compatibility of the total theory connection calculus with link condition, TTCC-Link, should be solved.

A higher priority task, however, is to go to practice. PROTEIN is stable enough now, and supporting tools to ease theory handling by the linearizing completion technique are available as well. It would be most interesting (at least for me) to embed PROTEIN into an interactive software verification system like KIV [Reif, 1992] and let it relieve the software engineer from boring proofs.

A. Appendix: Proofs

This appendix contains the missing proofs from the main part of this paper. Also, if an additional lemma is needed solely for a proof and has no further relevance outside the scope of the proof, it is given here.

A.1 Proofs for Chapter 4 — Theory Reasoning

Lemma 4.2.1. *Let M be a clause set and $([p], \mathcal{Q})$ be a branch set.*

1. *If $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ then $M \models_{\mathcal{T}} \forall([p] \circ C, \mathcal{Q})$ for every variant C of a clause $C' \in M$.*
2. *If $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ then $M \models_{\mathcal{T}} \forall(([p], \mathcal{Q})\sigma)$, for any substitution σ.*
3. *If $M \models_{\mathcal{T}} \forall([p]\times, \mathcal{Q})$ then $M \models_{\mathcal{T}} \forall(\mathcal{Q})$, where $[p]\times$ is a closed branch.*

Proof. **1.** Suppose, to the contrary, that $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ but $M \not\models_{\mathcal{T}} \forall(p \circ C, \mathcal{Q})$. Thus, there is a $\mathcal{T}$-model $\mathcal{I}$ for M such that $\mathcal{I} \not\models_{\mathcal{T}} \forall([p] \circ C, \mathcal{Q})$. That is, for some assignment v we have $\mathcal{I}_v([p] \circ C, \mathcal{Q}) = \mathit{false}$. According to the definition of "*Sem*", then $\mathcal{I}_v([p] \circ C) = \mathit{false}$ (*) and $\mathcal{I}_v(\mathcal{Q}) = \mathit{false}$ (**).

On the other hand, from given $M \models_{\mathcal{T}} \forall([p], \mathcal{Q})$ we conclude that $\mathcal{I}_v([p]) = \mathit{true}$ or $\mathcal{I}_v(\mathcal{Q}) = \mathit{true}$. Thus, together with (**), $\mathcal{I}_v([p]) = \mathit{true}$ (***).

Now, since $\mathcal{I}$ is given as a $\mathcal{T}$-model for M, we know in particular $\mathcal{I} \models_{\mathcal{T}} \forall C'$. We must further have $\mathcal{I}_v(C) = \mathit{true}$, because otherwise we would find an assignment falsifying C, which would be a witness such that $\mathcal{I} \models_{\mathcal{T}} \forall C'$ would not hold. But then $\mathcal{I}_v(L) = \mathit{true}$ for some $L \in C$. With (***) thus $\mathcal{I}_v(p \cdot L) = \mathit{true}$, which implies $\mathcal{I}_v([p] \circ C) = \mathit{true}$. This, however, plainly contradicts (*). Hence, the assumption must be wrong and the lemma holds.

2. Apply Lemma 2.5.3.

3. As in 1, suppose, to the contrary, that $M \models_{\mathcal{T}} \forall([p]\times, \mathcal{Q})$ but $M \not\models_{\mathcal{T}} \forall(\mathcal{Q})$. Thus, there is a $\mathcal{T}$-model $\mathcal{I}$ for M such that $\mathcal{I} \not\models_{\mathcal{T}} \forall(\mathcal{Q})$. That is, for some assignment v we have $\mathcal{I}_v(\mathcal{Q}) = \mathit{false}$ (*).

Now, by definition of the TTCC-Ext inference rule, the branch $[p] = [L_1, \ldots, L_n]$ is marked as closed only if it is $\mathcal{T}$-complementary. By definition of $\mathcal{T}$-complementarity, $\exists(L_1 \wedge \cdots \wedge L_n)$ is $\mathcal{T}$-unsatisfiable. That, is, for

every $\mathcal{T}$-interpretation $\mathcal{J}$ and every assignment w for the variables of $[p]$, $\mathcal{J}_w(L_1 \wedge \cdots \wedge L_n) = \mathit{false}$ (**).

Now let v' be the assignment for the variables of $[p]$ which is obtained from v by extending v with arbitrary assignments to the variables of $[p]$ which do not occur in $\mathcal{Q}$. By (**) thus $\mathcal{I}_{v'}(L_1 \wedge \cdots \wedge L_n) = \mathit{false}$, but still (by (*)) $\mathcal{I}_{v'}(\mathcal{Q}) = \mathit{false}$. In sum, $\mathcal{I}_{v'}([p]\times,\ \mathcal{Q}) = \mathit{false}$, which contradicts the given assumption $M \models_{\mathcal{T}} \forall([p]\times,\ \mathcal{Q})$.

Proposition 4.3.1. *Let $\mathcal{T}$ be definite theory. Then any minimal $\mathcal{T}$-complementary literal set (or multiset) contains exactly one negative literal.*

Proof. Let $M = \{L_1, \ldots, L_n\}$ be a minimal $\mathcal{T}$-complementary literal set. Following the reasoning in Note 4.2.3, this is equivalent to saying that $\{L'_1 \wedge \cdots \wedge L'_n\} \cup Ax$ is unsatisfiable, where Ax is a set of axioms (definite clauses) for $\mathcal{T}$, and $L'_1 \wedge \cdots \wedge L'_n$ is a Skolem form for $\exists(L_1 \wedge \cdots \wedge L_n)$. By definition of satisfiability, we have equivalently that $M' \cup Ax$ is unsatisfiable, where $M' = \{L'_1, \ldots, L'_n\}$. Now suppose, to the contrary, that the proposition does not hold. This gives rise to the following case analysis:

Case 1: M contains no negative literals. Thus M' contains no negative clauses, either. But then, with Ax being a set of definite clauses we find that every element from $M' \cup Ax$ contains (exactly) one positive literal. But then $M' \cup Ax$ is satisfiable (take the interpretation that assigns *true* to every positive literal), which is a contradiction.

Case 2: M contains more than one negative literal. M' will contain the same number of negative literals. It is well-known that for Horn sets, as $M' \cup Ax$ is one, one single negative clause suffices for unsatisfiability. More precisely, if a Horn set M is unsatisfiable then all but one negative clauses can be deleted and the resulting set is still unsatisfiable. This follows e.g. from inspection of the Hyper resolution proof of M which uses precisely one negative clause at the very last step.

Hence by soundness of Hyper resolution a subset $N' \subset M'$ exists such that $N' \cup Ax$ is unsatisfiable. But then by finding a corresponding subset $N \subset M$ such that N' is the Skolem form of N we learn that N alone is $\mathcal{T}$-complementary. This plainly contradicts the minimality assumption about M.

Proposition 4.4.1 (Context Extension of $\mathcal{T}$-MGRs). *Let σ' be a $\mathcal{T}$-refuter for M, and let W be a set of variables. Then substitutions $\sigma \in MSR_{\mathcal{T}}(M)$ with $\sigma \leq \sigma'\ [Var(M)]$ and δ with $\sigma' = \sigma\delta\ [W]$ exist.*

Proof. The proof is quite technical, and so we will illustrate the idea before. Consider e.g. $\sigma' = \{x \leftarrow f(u),\ u \leftarrow g(x)\}$, $Var(M) = \{x\}$, $W = \{x, u\}$. Note that σ' is *not* idempotent. The idea is to first split σ' into two parts, $\sigma'_M = \{x \leftarrow f(u)\}$ and $\sigma'_{\overline{M}} = \{u \leftarrow g(x)\}$. Then, we will find a most

general substitution σ and a δ such $\sigma'_M = \sigma\delta'$ $[Var(M)]$. Next, we would like to replace σ'_M by $\sigma\delta'$ and define $\delta = \sigma'_{\overline{M}}\delta'$. However, σ' is possibly not idempotent, and so neither $\sigma' = \sigma'_M\sigma'_{\overline{M}}$ $[Dom(\sigma')]$ nor (as a consequence) $\sigma' = \sigma\sigma_{\overline{M}}\delta'$ $[Dom(\sigma')]$ holds. One problem is that δ' might act on the variables of $VCod(\sigma'_{\overline{M}})$ (composing in any other way results in similar problems). Hence we will change $\sigma'_{\overline{M}}$ such that $VCod(\sigma'_{\overline{M}})$ contains only new variables. Let $\sigma''_{\overline{M}}$ be the thus obtained substitution. In order to simulate $\sigma'_{\overline{M}}$, we have to find a (renaming) substitution ρ such that $\sigma'_{\overline{M}} = \sigma''_{\overline{M}}\rho$ $[Dom(\sigma'_{\overline{M}})]$. In the example we will have $\sigma''_{\overline{M}} = \{u \leftarrow g(x')\}$ and $\rho = \{x' \leftarrow x\}$. If additionally possible conflicts between the newly introduced variables are avoided, then finally defining $\delta = \sigma''_{\overline{M}}\delta'\rho$ works.

In the example we will have e.g. $\sigma = \{x \leftarrow u'\}$, $\delta' = \{u' \leftarrow f(u)\}$. This gives us

$$\begin{aligned}\sigma''_{\overline{M}}\delta'\rho &= \{u \leftarrow g(x')\}\{u' \leftarrow f(u)\}\{x' \leftarrow x\} \\ &= \{u \leftarrow g(x),\ u' \leftarrow f(u),\ x' \leftarrow x\}\ .\end{aligned}$$

It is easy to see that the claim of the proposition holds in this example.

Now for the proof proper. Split σ' as follows:

$$V_M = Var(M) \qquad \sigma'_M = \sigma'|V_M \tag{A.1}$$

$$V_{\overline{M}} := Dom(\sigma') \setminus Var(M) \qquad \sigma'_{\overline{M}} = \sigma'|V_{\overline{M}}\ . \tag{A.2}$$

Hence, σ'_M is the part of σ' acting on $Var(M)$ and $\sigma'_{\overline{M}}$ is the part not acting on $Var(M)$. Note that trivially σ'_M is still a $\mathcal{T}$-refuter for M.

As mentioned above, we would ideally like $\sigma_{\overline{M}}$ to be away from $Var(M)$, i.e. $VCod(\sigma'_{\overline{M}}) \cap Var(M) = \emptyset$. However, since this cannot be assumed, we have to find an "equivalent" substitution satisfying this property. We will even have to go further and demand that this substitution is away from $X = W \cup Var(M) \cup VCod(\sigma')$; we claim that there is a substitution $\sigma''_{\overline{M}}$ with $VCod(\sigma''_{\overline{M}}) \cap X = \emptyset$ and a renaming substitution ρ such that

$$\sigma'_{\overline{M}} = \sigma''_{\overline{M}}\rho\ [Dom(\sigma'_{\overline{M}})]\ . \tag{A.3}$$

The construction is as follows: initially let $\sigma''_{\overline{M}} = \sigma'_{\overline{M}}$, and let $C = VCod(\sigma''_{\overline{M}}) \cap X$ be the set of common variables. Starting with $\rho = \emptyset$, Equation A.3 holds trivially. Now, every variable $y_1, \ldots, y_n \in C$ can be eliminated from $VCod(\sigma''_{\overline{M}})$ by repeated application of the the following procedure ($i = 1, \ldots, n$): let z_i be a new variable such that $z_i \notin X$ and $z_i \neq z_j$ for $i \neq j$. Replace every binding $x \leftarrow t \in \sigma''_{\overline{M}}$ by $x \leftarrow t'$, where t' is obtained from t by replacing every occurrence of y_i by z_i, and extend ρ to $\rho \cup \{z_i \leftarrow y_i\}$. Clearly, this preserves the property (A.3), and also ρ remains a renaming substitution. Thus, at the end we will further have $VCod(\sigma''_{\overline{M}}) \cap X = \emptyset$ as desired. Note further that $Dom(\sigma'_{\overline{M}}) = Dom(\sigma''_{\overline{M}})$. Let $\{z_1, \ldots, z_n\} = Dom(\rho) \subseteq VCod(\sigma''_{\overline{M}})$ be the "new" introduced variables.

Next we will find a suitable most general substitution σ to "replace" σ'_M. Care has to be taken that σ does not introduce variables used elsewhere. Hence let

$$V = W \setminus Var(M) \cup V_{\overline{M}} \cup VCod(\sigma''_{\overline{M}}) \setminus Var(M) \tag{A.4}$$

be the set of variables to be protected. It is easily verified that $V \cap Var(M) = \emptyset$. Thus, by this, and the fact that σ'_M is a $\mathcal{T}$-refuter for M, and by Definitions 4.4.2 and 4.4.1 (in particular, completeness) a most general $\sigma \in MSR_{\mathcal{T}}(M)[V]$ exists and there is a substitution δ' with

$$\sigma\delta' = \sigma'_M \ [Var(M)] \ . \tag{A.5}$$

Using the definition of σ'_M above we have $\sigma\delta' = \sigma' \ [Var(M)]$, or, equivalently, $\sigma \leq \sigma' \ [Var(M)]$ as desired. Following Convention 4.4.1 it holds that even $\sigma \in MSR_{\mathcal{T}}(M)$; it remains to show the existence of a substitution δ with $\sigma' = \sigma\delta \ [W]$.

Below we will have to assume that $Dom(\delta') \subseteq Var(M) \cup VCod(\sigma)$. This can safely be assumed, since for any $x \in Var(M)$ we have either $x\sigma = x \in Var(M)$ or otherwise $Var(x\sigma) \subseteq VCod(\sigma)$.

In order to obtain the desired substitution δ we define

$$\delta = \sigma''_{\overline{M}}\delta'\rho \ . \tag{A.6}$$

The claim of the proposition, namely $\sigma' = \sigma\delta \ [W]$, is obtained by proving the equivalent statement

$$x\sigma' = x\sigma\sigma''_{\overline{M}}\delta'\rho \qquad (\text{for } x \in W) \ . \tag{A.7}$$

Hence let $x \in W$. We carry out a case analysis:

1. $x \in Var(M)$: We need first that $x\sigma' = x\sigma'\rho$. Proof: either $x = x\sigma'$, and so $x\sigma' \in Var(M)$. However, $Dom(\rho) \cap Var(M) = \emptyset$ by construction (because $Dom(\rho) \subseteq VCod(\sigma''_{\overline{M}})$ and $\sigma''_{\overline{M}}$ is away from $X \supseteq Var(M)$). Otherwise $x \neq x\sigma'$ and thus $Var(x\sigma') \subseteq VCod(\sigma')$. However, also $Dom(\rho) \cap VCod(\sigma') = \emptyset$ by construction. Hence we continue

$$x\sigma' = x\sigma'\rho \overset{x \in Var(M)}{=} x\sigma'_M\rho \overset{(A.5)}{=} x\sigma\delta'\rho \overset{(*)}{=} x\sigma\sigma''_{\overline{M}}\delta'\rho \ .$$

It remains to prove $(*)$. For this is suffices to show $x\sigma = x\sigma\sigma''_{\overline{M}}$:

1.1. $x \in Dom(\sigma)$: Hence, $Var(x\sigma) \subseteq VCod(\sigma)$. By construction, σ is away from $V_{\overline{M}}$, i.e. $VCod(\sigma) \cap V_{\overline{M}} = \emptyset$. On the other hand, $Dom(\sigma''_{\overline{M}}) = Dom(\sigma'_{\overline{M}}) \subseteq V_{\overline{M}}$. Thus, $VCod(\sigma) \cap Dom(\sigma''_{\overline{M}}) = \emptyset$, and with $Var(x\sigma) \subseteq VCod(\sigma)$ even $Var(x\sigma) \cap Dom(\sigma''_{\overline{M}}) = \emptyset$, and so $x\sigma = x\sigma\sigma''_{\overline{M}}$

1.2. $x \notin Dom(\sigma)$: So $x = x\sigma$ and $x\sigma \in Var(M)$. By definition of $V_{\overline{M}}$, $Var(M) \cap V_{\overline{M}} = \emptyset$. This can be rephrased as $Var(x\sigma) \cap V_{\overline{M}} = \emptyset$. On the other

hand, as in case 1.1 we know that $Dom(\sigma''_{\overline{M}}) \subseteq V_{\overline{M}}$. Thus even $Var(x\sigma) \cap Dom(\sigma''_{\overline{M}}) = \emptyset$. Hence, also in this case $x\sigma = x\sigma\sigma''_{\overline{M}}$.

2. $x \notin Var(M)$: Again two cases apply.

2.1. $x \in Dom(\sigma')$: With $Dom(\sigma') \subseteq V_{\overline{M}}$ it follows $x \in V_{\overline{M}}$ and thus $x\sigma' = x\sigma'_{\overline{M}}$. Thus with $x \in Dom(\sigma')$ also $x \in Dom(\sigma'_{\overline{M}})$. We carry on

$$x\sigma' = x\sigma'_{\overline{M}} \stackrel{(A.3)}{=} x\sigma''_{\overline{M}}\rho \stackrel{(**)}{=} x\sigma''_{\overline{M}}\delta'\rho \ .$$

It remains to show $(**)$: For this it suffices to prove

$$x\sigma''_{\overline{M}} = x\sigma''_{\overline{M}}\delta' \ . \tag{A.8}$$

From $x \notin Var(M)$ and $x \in Dom(\sigma')$ we conclude $x \in Dom(\sigma') \setminus Var(M)$. Hence $x \in V_{\overline{M}}$ and again with $x \in Dom(\sigma')$ we find $x \in Dom(\sigma'_{\overline{M}})$. From the construction of $\sigma''_{\overline{M}}$ we know $Dom(\sigma''_{\overline{M}}) = Dom(\sigma'_{\overline{M}})$. Hence also $x \in Dom(\sigma''_{\overline{M}})$. In other words, $\sigma''_{\overline{M}}$ acts on x, and so (A.8) is equivalent to show that $y = y\delta'$ for any $y \in VCod(\sigma''_{\overline{M}})$. This shall be done next.

Recall from above that we can assume $Dom(\delta') \subseteq Var(M) \cup VCod(\sigma)$. Hence in order to show $y = y\delta'$ it suffices to show that $y \notin Var(M)$ and $y \notin VCod(\sigma)$: by construction, $\sigma''_{\overline{M}}$ is away from $Var(M)$, i.e. $VCod(\sigma''_{\overline{M}}) \cap Var(M) = \emptyset$, which shows $y \notin Var(M)$. We know that σ is away from $VCod(\sigma''_{\overline{M}}) \setminus Var(M)$, i.e. $VCod(\sigma) \cap VCod(\sigma''_{\overline{M}}) \setminus Var(M) = \emptyset$. Since we know already $VCod(\sigma''_{\overline{M}}) \cap Var(M) = \emptyset$ we must even have $VCod(\sigma''_{\overline{M}}) \setminus Var(M) = VCod(\sigma''_{\overline{M}})$. Thus also $VCod(\sigma) \cap VCod(\sigma''_{\overline{M}}) = \emptyset$ which gives us $y \notin VCod(\sigma)$ as desired. Thus the proof for $y = y\delta'$ and so for $(**)$ is completed.

2.2. $x \notin Dom(\sigma')$: hence $x\sigma' = x$. We show several identities:

1. $x = x\sigma$, because $Dom(\sigma) \subseteq Var(M)$ by property of $\mathcal{T}$-MGRs and $x \notin Var(M)$.
2. $x = x\sigma''_{\overline{M}}$ because $Dom(\sigma''_{\overline{M}}) = Dom(\sigma'_{\overline{M}}) \subseteq Dom(\sigma')$ and $x \notin Dom(\sigma')$.
3. $x = x\delta'$ because $Dom(\delta') \subseteq Var(M) \cup VCod(\sigma)$ as previously obtained, and $x \notin Var(M)$ and $x \notin VCod(\sigma)$ by the following line of reasoning: $x \notin Var(M)$ is given. This implies $x \in W \setminus Var(M)$. On the other side, we know that σ is away from $W \setminus Var(M)$, i.e. $VCod(\sigma) \cap W \setminus Var(M) = \emptyset$. Hence $x \notin VCod(\sigma)$. Thus, in sum, $x = x\delta$.
4. $x = x\rho$ because $Dom(\rho) \subseteq VCod(\sigma''_{\overline{M}})$ and $VCod(\sigma''_{\overline{M}}) \cap W = \emptyset$ by construction (recall that $x \in W$).

Putting these identities together, we easily obtain $x\sigma' = x\sigma\sigma''_{\overline{M}}\delta'\rho$ which is just Equation A.7.

A.1.1 Completeness of TTME-MSR

The proof of the completeness result for TTME-MSR requires considerable efforts. Here, we will prove the open issues in the proof in Section 4.4.3. This

concerns *ground completeness*, *lifting* and the *independence of the computation rule*.

We will heavily use the "branch set" view of tableaux (Section 3.2.2) and thus consider a tableau as a multiset of branches (including closed branches), which in turn are sequences of literals.

Ground Completeness Notice that in the ground case the notions "$\mathcal{T}$-complementary" and "$\mathcal{T}$-unsatisfiable" coincide (cf. Def. 4.2.1) and hence can be used interchangeable here.

As was announced in Section 3.2.2 we will identify the tableaux of our calculi with multisets of branches, where a branch is a sequence of literals.

Lemma A.1.2 (Ground completeness of Regular Total TME). *Let $\mathcal{T}$ be a theory and let M be a $\mathcal{T}$-unsatisfiable ground clause set. Let $G \in M$ be such that G is contained in some minimal $\mathcal{T}$-unsatisfiable subset of M. Then a regular TTME-MSR refutation of M with start clause G exists.*

The invariant in the ground proof is to show that any open tableau derived so far can be transformed by an application of a TTME-MSR-Ext step into a tableau which is strictly smaller wrt. some well-founded ordering (to be defined). Another idea would be to directly apply the well-known k-literal parameter induction ([Anderson and Bledsoe, 1970], but see also [Weidenbach, 1994] for a more recent exposition of the power of this technique). This approach allows for a more compact proof and was used in [Baumgartner, 1992a]. Unfortunately, it seems that completeness with regularity needs the further refinements developed in the proof below.

Definition A.1.1 (Definitions Related to Completeness Proof).
A path through a clause set $\{C_1, \ldots, C_n\}$ *is a multiset of literals* $\{\!|L_1, \ldots, L_n|\!\}$ *such that $L_i \in C_i$ for $i = 1 \ldots n$. A literal multiset q is called a* partial path through M *iff $q \subseteq p$ for some path p through M.*

A $\mathcal{T}$-unsatisfiable clause set M is called minimal $\mathcal{T}$-unsatisfiable wrt. a clause $C \in M$ *iff $M \setminus \{C\}$ is $\mathcal{T}$-satisfiable.*

A continuation of a branch $[L_1 \cdots L_n]$ $(n \geq 1)$ wrt. a ground clause set M *is a possibly empty clause set $N \subseteq M$ such that $Lit([L_1 \cdots L_n]) \cup N$ is minimal $\mathcal{T}$-unsatisfiable wrt. L_n. We usually write $N_{[p]}$ to indicate that $N_{[p]}$ is a continuation of branch $[p]$ wrt. some clause set M given from the context.*

Recall that $[L_1 \cdots L_n]$ denotes an open *branch. Intentionally, continuations need to be defined for this case only. Extending to branch sets, a* continuation $\mathcal{C}_\mathcal{P}$ of a branch set $\mathcal{P}$ wrt. a ground clause set M *is a multiset*

$$\mathcal{C}_\mathcal{P} = \{\!|N_{[p]} \mid N_{[p]} \text{ is a continuation of (open) branch } [p] \text{ wrt. } M, [p] \in \mathcal{P}|\!\} \ .$$

Notice that $\{\!|\}$ is a continuation of a closed tableau, whereas $\{\!|\emptyset|\!\}$ is a continuation of some tableau with one single branch, which is $\mathcal{T}$unsatisfiable by itself, e.g. "$[a < a]$".

As a measure for the complexity *of clause sets, let $k(M)$ denote the number of occurrences of literals in M minus the number of clauses in M ($k(M)$ is called the* excess literal parameter *in [Anderson and Bledsoe, 1970]). Below we will make use of the fact that $N \subseteq M$ implies $k(N) \leq k(M)$.*

The complexity measure k is homomorphically extended to multisets of continuations, i.e. if $\mathcal{C}_\mathcal{P}$ is a continuation of a given branch set then

$$k(\mathcal{C}_\mathcal{P}) = \{\!|k(c) \mid c \in \mathcal{C}_\mathcal{P}|\!\} \ .$$

Note that complexity of continuations of branch sets are multisets over non-negative integers. Hence we can use the multiset-extension $\gg_>$ of the usual "$>$" ordering (denoted by "$\gg$" in the sequel for simplicity). It is well-known that $\gg$ is a well-founded ordering (cf. Section 2.1).

Intuitively, a continuation of a branch $[L_1 \cdots L_n]$ can be seen as the "ressources" to complete the proof of $[L_1 \cdots L_n]$. Notice that if $Lit([L_1 \cdots L_{n-1}])$ alone is $\mathcal{T}$-unsatisfiable, then, by definition, a continuation does not exist. Below we will show at the heart of the completeness argument that extension steps go along with strictly decreasing k-values of accompanying continuations.

Since we will have that no literal from a branch $[p]$ need occur in its continuation $N_{[p]}$, and that it suffices to take clauses for extension steps from $N_{[p]}$, the regularity restriction follows easily. A proof without regularity would be considerably simpler; it was given in [Baumgartner, 1992a]. The proof below first appeared in [Baumgartner, 1993]. A proof which employs a similar idea to ours was described for the non-theory case in [Letz, 1993].

Lemma A.1.3. *Suppose a ground clause set M is minimal $\mathcal{T}$-unsatisfiable wrt. $C = L_1 \vee \ldots \vee L_n \in M$. Then for every $i = 1 \ldots n$, the clause set*

$$M_i = (M \setminus \{C\}) \cup \{L_i\}$$

is minimal $\mathcal{T}$-unsatisfiable wrt. L_i.

Proof. First, M_i is unsatisfiable because otherwise a model for M_i would be a model for M. Second, M_i is minimal unsatisfiable wrt. L_i because otherwise $M_i \setminus \{L_i\} = M \setminus \{C\}$ would be unsatisfiable, which contradicts the minimality assumption about C.

Lemma A.1.4. *Let $M = \{L_1, \ldots, L_n\} \cup N$ be a ground clause set containing unit clauses $L_1, \ldots, L_n$, and suppose M is minimal $\mathcal{T}$-unsatisfiable wrt. L_n. Then a set $N' \subseteq N$ exists such that (1) $M' = \{L_1, \ldots, L_n\} \cup N'$ is minimal $\mathcal{T}$-unsatisfiable wrt. L_n, and M' is minimal $\mathcal{T}$-unsatisfiable wrt. every clause in N', and (2) no literal in $\{L_1, \ldots, L_n\}$ occurs in N'.*

Proof. First, obtain N'' from N by deleting every clause containing a literal in $\{L_1, \ldots, L_n\}$. This preserves $\mathcal{T}$-unsatisfiability, i.e. $M'' = \{L_1, \ldots, L_n\} \cup N''$ is $\mathcal{T}$-unsatisfiable (because otherwise a model for M'' would be a model for M). Next minimize on N'', i.e. find a minimal set $N' \subseteq N''$ such that $M' =$

$\{L_1, \ldots, L_n\} \cup N'$ is $\mathcal{T}$-unsatisfiable. M' is trivially minimal $\mathcal{T}$-unsatisfiable wrt. every clause in N'. M' is also minimal $\mathcal{T}$-unsatisfiable wrt. L_n. Proof: Suppose, to the contrary, that

$$(\{L_1, \ldots, L_n\} \cup N') \setminus \{L_n\} = \{L_1, \ldots, L_{n-1}\} \cup N'$$

is $\mathcal{T}$-unsatisfiable. But then with $N' \subseteq N$ it follows that

$$\{L_1, \ldots, L_{n-1}\} \cup N = M \setminus \{L_n\}$$

is $\mathcal{T}$-unsatisfiable. This contradicts the given assumption that M is minimal $\mathcal{T}$-unsatisfiable wrt. L_n.

Thus claim (1) is proven. With claim (2) holding for N'' it holds with $N' \subseteq N''$ also for N'.

The following lemma is central for all connection methods:

Lemma A.1.5 ([Bibel, 1987]). *A (finite) set M of ground clauses is $\mathcal{T}$-unsatisfiable if and only if every path through M is $\mathcal{T}$-unsatisfiable.*

Proof. For the easy proof recall that a clause set is a conjunction of disjunction of literals. The lemma then is an immediate consequence of properties of boolean algebra.

More precisely ("if"-direction): Assume that every path through M is $\mathcal{T}$-unsatisfiable, but, to the contrary, that M is $\mathcal{T}$-satisfiable. Hence let $\mathcal{I}$ be a $\mathcal{T}$-model for M, i.e. $\mathcal{I} \models_{\mathcal{T}} C_i$, for every $C_i \in M$ ($i = 1, \ldots, n$, with $|M| = n$), i.e. $\mathcal{I} \models_{\mathcal{T}} L_{f(i)}$ for some $L_{f(i)} \in C_i$. Thus, $\mathcal{I} \models_{\mathcal{T}} \{\!|L_{f(1)}, \ldots, L_{f(n)}|\!\}$ which means that a $\mathcal{T}$-satisfiable path through M exists. Contradiction.

For the "only-if" direction notice that for given $\mathcal{T}$-unsatisfiable clause set M the existence of a $\mathcal{T}$-satisfiable path would immediately give us a $\mathcal{T}$-model for M, which is a contradiction.

Lemma A.1.6. *Suppose a ground clause set M is minimal $\mathcal{T}$-unsatisfiable wrt. some unit clause $L \in M$. Then a partial path p through M exists such that p is minimal $\mathcal{T}$-unsatisfiable and $L \in p$.*

Proof. First, $\mathcal{T}$-satisfiable path q through $M \setminus \{L\}$ exists. Proof: suppose, to the contrary, that every path q through $M \setminus \{L\}$ is $\mathcal{T}$-unsatisfiable. Then by lemma A.1.5 $M \setminus \{L\}$ is also $\mathcal{T}$-unsatisfiable. Hence M is not minimal $\mathcal{T}$-unsatisfiable wrt. L. Contradiction.

However $q \cup \{\!|L|\!\}$ is $\mathcal{T}$-unsatisfiable, because otherwise a model for $q \cup \{\!|L|\!\}$ would be a model for M. Next minimize on $q \cup \{\!|L|\!\}$, i.e. find a minimal $\mathcal{T}$-unsatisfiable subset $p \subseteq q \cup \{\!|L|\!\}$. This is the claimed path. Furthermore, with $p \setminus \{\!|L|\!\} \subseteq q$ and q being $\mathcal{T}$-satisfiable (as derived above) it holds $L \in p$.

The following lemma is crucial for the inductive argument of the completeness proof:

Lemma A.1.7 (Extension Decreases Complexity). *Let $\mathcal{C}_{([q],\mathcal{Q})} \neq \{\!|\}$ be a continuation of a regular[1] branch set $([q], \mathcal{Q})$ wrt. a given ground clause*

[1] Cf. Def. 3.3.2.

set M. Then there is a TTME-MSR-Ext *inference (Def. 4.4.3)*

$$\frac{[q],\ \mathcal{Q} \qquad C_1 \quad \cdots \quad C_n}{\mathcal{Q}',\ \mathcal{Q}}\ \text{TTME-MSR-Ext}\ ,$$

such that

1. $C_1, \ldots, C_n \in M$, *and*
2. $(\mathcal{Q}', \mathcal{Q})$ *is a regular branch set, and*
3. *there is a continuation* $\mathcal{C}_{(\mathcal{Q}',\mathcal{Q})}$ *of* $(\mathcal{Q}', \mathcal{Q})$ *wrt.* M *with* $k(\mathcal{C}_{([q],\mathcal{Q})}) \not\succ k(\mathcal{C}_{(\mathcal{Q}',\mathcal{Q})})$.

Proof. The given branch $[q]$ can be written as $[q] = [K_1 \cdots K_m]$. Let $N_{[q]} \in \mathcal{C}_{([q],\mathcal{Q})}$ be a continuation of $[q]$. By definition of continuation, $\mathcal{C}_{([q],\mathcal{Q})}$ is minimal $\mathcal{T}$-unsatisfiable wrt. K_m. By Lemma A.1.4 there is a set $N'_{[q]} \subseteq N_{[q]}$ such that (1) $Lit([q]) \cup N'_{[q]}$ is minimal $\mathcal{T}$-unsatisfiable wrt. K_m and every clause from $N'_{[q]}$ (*), and (2) no literal from $Lit([q])$ occurs in $N'_{[q]}$ (**).

By lemma A.1.6 there is a minimal $\mathcal{T}$-unsatisfiable partial path p through $Lit([q]) \cup N'_{[q]}$ with $K_m \in p$. p consists of literals from $Lit([q])$ and of literals from clauses in N'_q. In other words p is of the form

$$p = \{\!|K_{i_1}, \ldots, K_{i_k}, K_m, L_1, \ldots, L_n|\!\}\ ,$$

for some literals $K_{i_j} \in \{\!|K_1, \ldots, K_{m-1}|\!\}$, $1 \leq i_j \leq m-1$ and some clauses $L_i \vee R_i \in N'_q$, $1 \leq i \leq n$.

Since p is minimal $\mathcal{T}$-unsatisfiable and p is ground this is trivially the same as saying that p is minimal $\mathcal{T}$-complementary. Thus the following TTME-MSR-Ext inference exists, using the empty substitution and key set p (In terms of Definition 4.4.3 set $\mathcal{B} = \{\!|K_{i_1}, \ldots, K_{i_k}|\!\}$ and $K = K_m$):

$$\frac{[q],\ \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{\mathcal{Q}',\ \mathcal{Q}}$$

where

$$\mathcal{Q}' = \begin{cases} [q]\times & \text{if } R_1 \vee \cdots \vee R_n = \Box, \\ [q] \circ (R_1 \vee \cdots \vee R_n) & \text{else,} \end{cases}$$

Next we turn to the claims 1–3 of the lemma, showing that this inference has the desired properties.

With $L_i \vee R_i \in N_{[q]} \subseteq M$ (by property of continuation) the claim (1) is trivial.

Since the literals in $R_1 \vee \cdots \vee R_n$ all stem from clauses in $N'_{[q]}$, by (**) above the given regularity of $[q]$ carries over to $[q] \circ (R_1 \vee \cdots \vee R_n)$, and this suffices to show regularity of $(\mathcal{Q}', \mathcal{Q})$.

Concerning (3), the given continuation $\mathcal{C}_{[q],\mathcal{Q}}$ is of the form $\{\!|N_{[q]}|\!\} \cup \mathcal{C}_{\mathcal{Q}}$ where $\mathcal{C}_{\mathcal{Q}}$ is a continuation of $\mathcal{Q}$. We will construct the desired continuation as

$$\mathcal{C}_{(\mathcal{Q}',\mathcal{Q})} = \{\!|N_{[q']} \mid N_{[q']} \text{ is a continuation of } [q'] \text{ wrt. } M, \text{ and } k(N_{[q']}) < k(N_{[q]}), \text{ for } [q'] \in \mathcal{Q}' \text{ being an open branch}|\!\} \cup \mathcal{C}_{\mathcal{Q}} \ .$$

Since $\mathcal{C}_{(\mathcal{Q}',\mathcal{Q})}$ is obtained from $\mathcal{C}_{([q],\mathcal{Q})}$ by replacing $N_{[q]}$ by the finitely many elements $N_{[q']}$ with $k(N_{[q']}) < k(N_q)$ as indicated, it holds

$$k(\mathcal{C}_{([q],\mathcal{Q})}) \gg k(\mathcal{C}_{(\mathcal{Q}',\mathcal{Q})}) \ .$$

Now for the construction of $\mathcal{C}_{\mathcal{Q}' \cup \mathcal{Q}}$: if $\mathcal{Q}' = ([q]\times)$ (that is, all rest clauses are empty and thus $[q]$ gets closed) then trivially $\mathcal{C}_{\mathcal{Q}'} = \{\!|\}$. Hence with $N_{[q]}$ being a continuation of $[q]$ and $\{\!|N_{[q]}|\!\} \gg \{\!|\}$, condition (3) is immediately obtained by

$$\mathcal{C}_{([q],\mathcal{Q})} \gg \mathcal{C}_{\mathcal{Q}} = \mathcal{C}_{(\mathcal{Q}',\mathcal{Q})} \ .$$

Otherwise let $[q'] \in \mathcal{Q}'$. $[q']$ is of the form $[q \cdot K]$ where $K \in R_1 \vee \ldots \vee R_n$. Now let $K \in R_i$, for $i \in \{j \mid j \in \{1,\ldots,n\} \text{ and } R_j \neq \{\!|\}\!\}$. K stems from an extending clause $L_i \vee R_i \in N_{[q']}$ which contains at least 2 literals. By (*) above it holds that $Lit([q]) \cup N'_{[q]}$ is minimal $\mathcal{T}$-unsatisfiable wrt. $L_i \vee R_i$. Then by Lemma A.1.6 the set

$$((Lit([q]) \cup N'_{[q]}) \setminus \{L_i \vee R_i\}) \cup \{K\}$$

is minimal $\mathcal{T}$-unsatisfiable wrt. K. This set is the same set as

$$Lit([q \cdot K]) \cup (N'_{[q]} \setminus \{L_i \vee R_i\})$$

With $[q \cdot K] = [q']$ recognize that $N_{[q']} = N'_{[q]} \setminus \{L_i \vee R_i\}$ is thus a continuation of $[q']$.

It remains to show the ordering property $k(N_{[q']}) < k(N_{[q]})$. This follows immediately from the fact that we deleted a clause with at least 2 literals. More precisely $N_{[q']} = N'_{[q]} \setminus \{L_i \vee R_i\} \subset N'_{[q]} \subseteq N_{[q]}$ implies

$$k(N_{[q']}) = k(N'_{[q]} \setminus \{L_i \vee R_i\}) < k(N'_{[q]}) \leq k(N_{[q]})$$

This lemma can be iterated:

Lemma A.1.8. *Let M be a $\mathcal{T}$-unsatisfiable ground clause set, $\mathcal{Q}$ be a tableau and $\mathcal{C}_{\mathcal{Q}}$ be a continuation of $\mathcal{Q}$ wrt. M. Then there is a finite sequence*

$$(\mathcal{Q} = \mathcal{Q}_1) \vdash \mathcal{Q}_2 \vdash \cdots \vdash \mathcal{Q}_n$$

ending in a closed tableau $\mathcal{Q}_n$, and there is a finite chain of respective continuations wrt. M,

$$(\mathcal{C}_\mathcal{Q} = \mathcal{C}_1) \succcurlyeq \mathcal{C}_2 \succcurlyeq \cdots \succcurlyeq (\mathcal{C}_n = \{\!\{\}\!\}) \ ,$$

such that every $\mathcal{Q}_{i+1}$ is obtained from $\mathcal{Q}_i$ ($1 \leq i \leq n-1$) and some clauses from M by application of a TTME-MSR-Ext *inference step, and every $\mathcal{C}_i$ is a continuation of $\mathcal{Q}_i$.*

Proof. By well-founded induction on the ordering $\succcurlyeq$: if $\mathcal{C}_\mathcal{Q} = \{\!\{\}\!\}$ then $\mathcal{Q}$ must trivially be a closed tableau. Otherwise a branch $q \in \mathcal{Q}$ exists. Next apply lemma A.1.7 to obtain a branch set $\mathcal{Q}'$ with continuation $\mathcal{C}_{\mathcal{Q}'}$ such that $\mathcal{C}_\mathcal{Q} \succcurlyeq \mathcal{C}_{\mathcal{Q}'}$. Now apply the induction hypothesis to $\mathcal{Q}'$ and $\mathcal{C}_{\mathcal{Q}'}$.

Now we are ready for the proof of the ground completeness lemma:

Proof. (Lemma A.1.2) Without loss of generality assume that M is already a minimal $\mathcal{T}$-unsatisfiable set containing G (otherwise we are given that an appropriate subset exists). Clearly, M is minimal $\mathcal{T}$-unsatisfiable wrt. G.

The given start clause can be written as $G = L_1 \vee \cdots \vee L_n$. By Lemma A.1.3 every set $M_i = (M \setminus \{L_1 \vee \cdots \vee L_n\}) \cup \{L_i\}$ $(1 \leq i \leq n)$ is minimal $\mathcal{T}$-unsatisfiable wrt. L_i. In other words $M_i' = M_i \setminus \{L_i\}$ is a continuation for $[L_i]$. Thus, $\mathcal{C}_1 = \{\!\{M_1', \ldots, M_n'\}\!\}$ is a continuation for the branch set $\mathcal{Q}_1 = \{\!\{[L_1], \ldots, [L_n]\}\!\}$. Next apply Lemma A.1.8, which shows that a TTME-MSR refutation with start clause G exists.

Lifting Lemma for Theory Model Elimination The completeness proofs for our model elimination calculi all work by assuming the existence of a refutation on the ground level, which is then "simulated" by a refutation on the first-order technique. This widely used technique is known as the *lifting* technique. It is applicable for a wide spectrum of calculi, including resolution (see e.g. [Chang and Lee, 1973]) and first-order tableaux with free variables (see e.g. [Fitting, 1990]).

I feel the strong need for proofs at a detailed level, because, at least in my experience, statements like "lifting works as usual" can easily go wrong.

Convention A.1.1 (Conventions for Lifting Proof). Our two most advanced versions of theory model elimination are the total TTME-MSR version (Def. 4.4.3) and the partial PTME-I version (Def. 4.5.4). The proofs of the respective lifting lemmas share the same argumentation and structure. Hence we will not repeat ourselves, and supply only one lemma, with one single unified proof, which makes a case analyses wrt. the applied inference rules TTME-MSR-Ext, respectively PTME-I-Ext. As a further notational simplification we define

$$[p] \circ_\times C = \begin{cases} [p]\times & \text{if } C = \Box, \\ [p] \circ C & \text{else .} \end{cases}$$

This will keep some case analyses required for the TTME-MSR-Ext and PTME-I-Ext inference rules implicit in the "$\circ_\times$" operator.

Recall from Definition 4.5.4 that PTME-I-Ext inferences are written as (we will quite often omit the theory inference rule subscript) $\mathcal{P} \vdash_{p,\mathcal{K},\langle\mathcal{R},\sigma\rangle,E} \mathcal{P}'$. In order to avoid repetition we will allow this notation also for TTME-MSR-Ext inferences by setting $\mathcal{R} = \square$ and meaning that σ is a minimal $\mathcal{T}$-MGR for $\mathcal{K}$, as required. This is consistent with the Definition of TTME-MSR-Ext.

Lemma A.1.9 (Lifting lemma for TTME-MSR and PTME-I). *Let M be a clause set. For PTME-I, let $\mathcal{I}'$ be an inference system consisting of (not necessarily ground) instances of inference rules from the inference system $\mathcal{I}$.*

Suppose a PTME-$\mathcal{I}'$ derivation exists, respectively a TTME-MSR derivation

$$\mathcal{D}'_l = (\mathcal{Q}'_1 \vdash_{[p'_1],\mathcal{K}'_1,\langle\mathcal{R}'_1,\sigma'_1\rangle,E'_1} \mathcal{Q}_2 \cdots \mathcal{Q}'_{l-1} \vdash_{[p'_{l-1}],\mathcal{K}'_{l-1},\langle\mathcal{R}'_{l-1},\sigma'_{l-1}\rangle,E'_{l-1}} \mathcal{Q}'_l)$$

from a set M' of (not necessarily ground) instances of clauses from M.

Then there also exists a PTME-$\mathcal{I}$ derivation, respectively a TTME-MSR derivation

$$\mathcal{D}_l = (\mathcal{Q}_1 \vdash_{[p_1],\mathcal{K}_1,\langle\mathcal{R}_1,\sigma_1\rangle,E_1} \mathcal{Q}_2 \cdots \mathcal{Q}_{l-1} \vdash_{[p_{l-1}],\mathcal{K}_{l-1},\langle\mathcal{R}_{l-1},\sigma_{l-1}\rangle,E_{l-1}} \mathcal{Q}_l)$$

from M such that for some substitution δ_l

$$Q'_l = Q_l\delta_l{}^2, \tag{A.9}$$

and, for $j = 1, \ldots, l-1$,

$$\mathcal{K}'_j\sigma'_j \cdots \sigma'_{l-1} = \mathcal{K}_j\sigma_j \cdots \sigma_{l-1}\delta_l, \text{ and} \tag{A.10}$$

$$E'_j\sigma_j \cdots \sigma'_{l-1} = E_j\sigma_1 \cdots \sigma_{l-1}\delta_l \ . \tag{A.11}$$

Furthermore, (1) stability wrt. minimality is preserved (that is, if $\mathcal{D}'$ is stable wrt. minimality, then so is $\mathcal{D}$, cf. Def. 4.5.7 for PTME-I and Def. 4.4.4 for TTME-MSR), and, (2) regularity is preserved (that is, if $\mathcal{D}'$ is regular, then so is $\mathcal{D}$, cf. Def. 3.3.2).

The property A.9 is needed for *refutational* completeness: if Q'_l is closed then Q_l must be closed, too. Property A.10 is needed for the preservation of stability wrt. minimality, and property A.11 is used for the *answer* completeness. Before we can turn to the proof of this lifting lemma for *derivations*, we need a further lifting lemma for single PTME-I-Ext inferences; such a lemma is not needed for TTME-MSR-Ext inferences, because the respective property is given as a property of CSRs.

Lemma A.1.10 (Lifting Lemma for PTME-I-Ext inferences). *Let $P' \to C'$ be an instance of an inference rule $P^{base} \to C^{base}$, and let W be a set of variables. Suppose that*

$$\mathcal{K} \Rightarrow_{P'\to C',\,\sigma'} \mathcal{R}'\sigma'$$

and that $Var(P^{base} \to C^{base}) \cap Var(\mathcal{K}) = \emptyset$.

Then, for any variant $P \to C$ *of* $P^{base} \to C^{base}$ *such that* $Var(P \to C) \cap (W \cup Var(\mathcal{K})) = \emptyset$ *there is a multiset-MGU* σ *and a substitution* δ *such that the following holds:*

$$\mathcal{K} \Rightarrow_{P \to C,\, \sigma} \mathcal{R}\sigma \qquad \textit{(theory inference exists)}$$

$$\mathcal{R}'\sigma' = \mathcal{R}\sigma\delta \qquad \textit{(residue lifts)}$$

$$\sigma' = \sigma\delta\ [W] \qquad \textit{(most generality and context extension)}$$

Proof. Let γ' be the substitution such that $P' \to C' = (P^{base} \to C^{base})\gamma'$. Let ρ be the renaming substitution for $P \to C$, i.e. $P \to C = (P^{base} \to C^{base})\rho$. From this conclude

$$P' \to C' = (P \to C)\gamma \qquad \text{where } \gamma = \rho^{-1}\gamma' | Var(P \to C)\ . \tag{A.12}$$

First, we need to know that $\mathcal{K} = \mathcal{K}\gamma$: the given assumption $Var(P \to C) \cap (W \cup Var(\mathcal{K})) = \emptyset$ implies in particular $Var(P \to C) \cap Var(\mathcal{K}) = \emptyset$. By definition of γ, $Dom(\gamma) \subseteq Var(P \to C)$. Hence also $Dom(\gamma) \cap Var(\mathcal{K}) = \emptyset$. Therefore, $\mathcal{K} = \mathcal{K}\gamma$.

Using this fact and Equation A.12 we can rewrite the given theory inference as

$$\mathcal{K}\gamma \Rightarrow_{(P \to C)\gamma,\, \sigma'} \mathcal{R}'\sigma'\ .$$

By definition of theory inference, σ' thus is a multiset unifier for some amplification $(\mathcal{K}\gamma)'$ of $\mathcal{K}\gamma$ and $P\gamma$; also, by definition, $\mathcal{K}\gamma\sigma'$ contains no duplicates.

Let $\mathcal{K}' = \{\!|K \mid K\gamma \in (\mathcal{K}\gamma)'|\!\}$. Notice that $\mathcal{K}'$ is an amplification of $\mathcal{K}$; $\mathcal{K}$ will just be the premise of the theory inference in the lemma statement.

Using this reformulation we find that $\gamma\sigma'$ is a (not necessarily most general) multiset unifier for $\mathcal{K}'$ and P, that is $\mathcal{K}'\gamma\sigma' = P\gamma\sigma'$. By virtue of well-known unification algorithms[3] a most general multiset unifier σ for $\mathcal{K}'$ and P exists. Stated differently, $\gamma\sigma'$ is a $\mathcal{T}$-refuter for $\{\!|\mathcal{K}' \neq P|\!\}$, and $\sigma \in MSR_{\mathcal{T}}\{\!|\mathcal{K}' \neq P|\!\}$, where $\mathcal{T}$ is the theory of multisets. Using Proposition 4.4.1, we can assume that $\sigma \leq \gamma\sigma'\ [Var\{\!|\mathcal{K}' \neq P|\!\}]$ (*) and that there is a substitution δ such that

$$\gamma\sigma' = \sigma\delta\ [W \cup Var(C)]\ . \tag{A.13}$$

We must insist on the context $Var(C)$ here because C might contain extra variables not touched by σ.

From (*) we conclude in particular $\sigma \leq \gamma\sigma'\ [Var(\mathcal{K}')]$, and since $Var(\mathcal{K}) = Var(\mathcal{K}')$ even $\sigma \leq \gamma\sigma'\ [Var(\mathcal{K})]$. Hence, with $\mathcal{K}\gamma\sigma'$ containing no duplicates,

[3] See e.g. [Büttner, 1986] for a multiset unification algorithm. A simple algorithm for multiset unification (without tail!) is to read $\mathcal{K}'$ and P as lists, and to collect the (syntactical) MGUs of $\mathcal{K}'$ and all permutations of P.

$\mathcal{K}\sigma$ does not contain duplicates either. Next define $\mathcal{R} = \square$ if $C' = C = \mathsf{F}$ and $\mathcal{R} = C$ otherwise. In the former case, $\mathcal{R}'\sigma' = \mathcal{R}\sigma\delta$ holds trivially, and in the latter case we have

$$\mathcal{R}'\sigma' = C'\sigma' \overset{(A.12)}{=} C\gamma\sigma' \overset{(A.13)}{=} C\sigma\delta = \mathcal{R}\sigma\delta \ . \tag{A.14}$$

From this we learn that the theory inference as claimed in the lemma statement exists (it uses the amplification $\mathcal{K}'$ of $\mathcal{K}$ and multiset unifier σ). Further, Equation A.14 is also a proof of the nontrivial case of the "residue lifts" property.

It remains to show $\sigma' = \sigma\delta$ $[W]$. From (A.13) we can trivially obtain $\gamma\sigma' = \sigma\delta$ $[W]$. The given assumption $Var(P \to C) \cap (W \cup Var(\mathcal{K})) = \emptyset$ implies in particular $Var(P \to C) \cap W = \emptyset$. By definition of γ, $Dom(\gamma) \subseteq Var(P \to C)$. Hence also $Dom(\gamma) \cap W = \emptyset$. Therefore, $x\gamma = x$ for any $x \in W$, and thus more generally $\gamma\sigma' = \sigma'$ $[W]$. Thus with $\gamma\sigma' = \sigma\delta$ $[W]$ conclude $\sigma' = \sigma\delta$ $[W]$, which completes the proof.

Proof. (Lemma A.1.9) We need the following *fact*: Suppose C' is a variant of a clause from M'. Then a clause $C^{base} \in M$ exists such that for any variant C of C^{base} a substitution γ with $Dom(\gamma) = Var(C)$ exists such that $C\gamma = C'$.

Proof of *fact*: Let ρ be the renaming substitution by which C' was obtained from a clause in M'. That means $C'\rho^{-1} \in M'$. Now, every clause in M' is an instance of a clause from M. That is, for some $C^{base} \in M$ and some θ we have $C^{base}\theta = C'\rho^{-1}$. Applying ρ to both sides yields $C^{base}\theta\rho = C'$. Any variant C of C^{base} is obtained by means of a renaming substitution (cf. [Lloyd, 1987]), say ρ_C. In other words, $C = C^{base}\rho_C$. Using $C\rho_C^{-1} = C^{base}$ we have $C\rho_C^{-1}\theta\rho = C'$. Now setting $\gamma = \rho_C^{-1}\theta\rho|\,Var(C)$ proves the fact. The proof of the lemma proper is by induction on the length l of the derivation.

Induction start $(l = 1)$: Here, Q'_1 is a branch set for the start clause $C'_1 \in M'$. By the *fact* there is a clause $C_1 \in M$ (which is also a variant of itself) and a substitution δ_1 with $C_1\delta_1 = C'_1$. Thus the branch set Q_1 for the start clause C_1 constitutes a proof of (A.9). Further, since no inference steps are carried out, both stability wrt. minimality and regularity trivially hold. Also, properties A.10 and A.11 hold trivially.

Induction step: $(l \to l+1)$: As the induction hypothesis suppose that properties A.9, A.10 and A.11 hold for l, and let the given derivation $\mathcal{D}'_{l+1}$ now be

$$\mathcal{D}'_{l+1} = (\mathcal{D}'_l \vdash_{[p'_l],\mathcal{K}'_l,\langle\mathcal{R}'_l,\sigma'_l\rangle,E'_l} \mathcal{Q}'_{l+1}) \ .$$

We will show how to extend the derivation $\mathcal{D}_l$ towards

$$\mathcal{D}_{l+1} = (\mathcal{D}_l \vdash_{[p_l],\mathcal{K}_l,\langle\mathcal{R}_l,\sigma_l\rangle,E_l} \mathcal{Q}_{l+1}) \ ,$$

such that Properties A.9, A.10 and A.11 hold for $l+1$.

Suppose $E'_l = \{L'_1 \vee R'_1, \ldots, L'_n \vee R'_n\}$ are the new variants of clauses from M' used as extending clauses in the concluding step of $\mathcal{D}'_{l+1}$ with $\mathcal{L}' = \{\!|L'_1, \ldots, L'_n|\!\}$ being the extending literals. Suppose $\mathcal{K}'_l = \mathcal{B}' \cup \{\!|Leaf([p'_l])|\!\} \cup \mathcal{L}'$, for some $\mathcal{B}' \subseteq [p_l] \setminus \{\!|Leaf([p'_l])|\!\}$. This setup can be taken for both calculus variants. However, there are some differences in further argumentation.

In case of TTME-MSR, by definition, $\mathcal{K}'_l\sigma'_l$ is minimal $\mathcal{T}$-complementary (we need not insist that σ'_l is a $\mathcal{T}$-MGR).

In case of PTME-$\mathcal{I}'$, the PTME-I-Ext step is based on the theory inference

$$\mathcal{K}'_l \Rightarrow_{P'_l \to C'_l,\, \sigma'_l} \mathcal{R}'_l\sigma'_l \tag{A.15}$$

for some new variant $P'_l \to C'_l$ of a rule from $\mathcal{I}'$. In both versions the resulting branch set Q'_{l+1} can be written as

$$Q'_{l+1} = ([p'_l] \circ_\times (\mathcal{R}'_l \vee R'_1 \vee \cdots \vee R'_n) \cup (\mathcal{Q}'_l \setminus \{\!|[p'_l]|\!\}))\sigma'_l \ ,$$

where for TTME-MSR we set $\mathcal{R}'_l = \Box$.

By the induction hypothesis A.9 there is in particular (both versions) a branch $[p_l] \in Q_l$ with $[p_l]\delta_l = [p'_l]$. Thus also a subset$\mathcal{B} \subseteq [p_l] \setminus \{\!|Leaf([p_l])|\!\}$ with $\mathcal{B}\delta_l = \mathcal{B}'$ exists.

Next we are going to set up a corresponding set E_l of extending clauses. By the *fact* we can find for every clause $(L'_i \vee R'_i) \in E'_l$ $(i = 1, \ldots, n)$ a new variant $L_i \vee R_i$ of a clause from M and a substitution $\gamma_{L_i \vee R_i}$ such that $(L_i \vee R_i)\gamma_{L_i \vee R_i} = L'_i \vee R'_i$. Hence let $E_l = \{L_1 \vee R_1, \ldots, L_n \vee R_n\}$, $\mathcal{L} = \{\!|L_1, \ldots, L_n|\!\}$ and define the key set to be used in the lifted inference as $\mathcal{K}_l = \mathcal{B} \cup \{\!|Leaf([p_l])|\!\} \cup \mathcal{L}$. We have to show that E_l and $\mathcal{K}_l$ are suitably defined.

All the clauses $L_i \vee R_i$ are new variants, and hence are variable disjoint. So without loss of generality we can assume that $\gamma_{L_i \vee R_i}$ acts on the variables of $L_i \vee R_i$ only, that is $Dom(\gamma_{L_i \vee R_i}) \subseteq Var(L_i \vee R_i)$. But then we can define $\gamma_l = \gamma_{L_1 \vee R_1} \cdots \gamma_{L_n \vee R_n}$ and it holds

$$(L_i \vee R_i)\gamma_l = L'_i \vee R'_i \ , \tag{A.16}$$

which implies

$$\mathcal{L}\gamma_l = \mathcal{L}' \ . \tag{A.17}$$

Since E_l consists of new variants we can assume that δ_l does not act on the variables of $L_i \vee R_i$. Hence

$$L_i \vee R_i = (L_i \vee R_i)\delta_l, \tag{A.18}$$

which implies

$$\mathcal{L} = \mathcal{L}\delta_l \ . \tag{A.19}$$

Recall that γ_l acts on the variables of E_l only. But with E_l consisting of new variants we find that E_l is variable disjoint from $Q_l\delta_l$, and hence

$$Q_l\delta_l = Q_l\delta_l\gamma_l \ . \tag{A.20}$$

We continue

$$\begin{aligned}
&\mathcal{K}'_l\sigma'_l \\
&= (\mathcal{B}' \cup \{\!|Leaf([p'_l])|\!\} \cup \mathcal{L}')\sigma'_l \\
&= (\mathcal{B}\delta_l \cup \{\!|Leaf([p_l])\delta_l|\!\} \cup \mathcal{L}\gamma_l)\sigma'_l \\
&\qquad \text{(by (A.9) for } l \text{ and (A.17))} \\
&= (\mathcal{B}\delta_l\gamma_l \cup \{\!|Leaf([p_l])\delta_l\gamma_l|\!\} \cup \mathcal{L}\delta_l\gamma_l)\sigma'_l \\
&\qquad \text{(by (A.19) and (A.20))} \\
&= (\mathcal{B} \cup \{\!|Leaf([p_l])|\!\} \cup \mathcal{L})\delta_l\gamma_l\sigma'_l \\
&= \mathcal{K}_l\delta_l\gamma_l\sigma'_l \ .
\end{aligned} \tag{A.21}$$

This setup holds for both variants.

Now we have to distinguish. In case of TTME-sem, (A.21) tells us that $\delta_l\gamma_l\sigma'_l$ is a, although minimal, not necessarily most general $\mathcal{T}$-refuter for $\mathcal{K}_l$. We conclude with Proposition 4.4.1 that there is a $\mathcal{T}$-MGR $\sigma_l \in MSR_{\mathcal{T}}(\mathcal{K}_l)$ and a substitution δ_{l+1} such that

$$\begin{aligned}
&\delta_l\gamma_l\sigma'_l = \sigma_l\delta_{l+1}\,[W], \qquad \text{where} \\
&W = \bigcup_{j=1}^{l} (\,Var(\mathcal{Q}_j\sigma_j\cdots\sigma_{l-1}) \cup Var(\mathcal{K}_j\sigma_j\cdots\sigma_{l-1}) \cup Var(E_j\sigma_j\cdots\sigma_{l-1}))\ .
\end{aligned} \tag{A.22}$$

By construction of $\mathcal{K}_j$ it holds that $Var(\mathcal{K}_j) \subseteq Var(\mathcal{Q}_j) \cup Var(E_j)$, and hence the $Var(\mathcal{K}_j)$ term could be omitted in the definition of W.

By Proposition 4.2.1.2 we have that if $\delta_l\gamma_l\sigma'_l$ is a minimal $\mathcal{T}$-refuter, then so must be σ_l. Hence, a TTME-MSR-Ext step based on σ_l and using extending clauses E_l and key set $\mathcal{K}_l$ can be carried out to Q_l.

For PTME-$\mathcal{I}'$ we need the following reasoning: recall that we demand that the used theory inference rule (in $\mathcal{D}'_{l+1}$) $P' \to C'$ is a *new* variant. Hence neither δ_l nor γ_l will act on $P' \to C'$ and we will thus have

$$(P' \to C')\delta_l\gamma_l = P' \to C' \ . \tag{A.23}$$

But then, using this fact and Equation A.21, the theory inference A.15) is the same as

$$\mathcal{K}_l \Rightarrow_{P'_l \to C'_l,\, \delta_l\gamma_l\sigma'_l} \mathcal{R}'_l\delta_l\gamma_l\sigma'_l \ . \tag{A.24}$$

With A.23 it trivially follows that $\delta_l\gamma_l$ does not affect $\mathcal{R}'_l$. Hence

$$\mathcal{R}'_l\delta_l\gamma_l\sigma'_l = \mathcal{R}'_l\sigma'_l \ . \tag{A.25}$$

This trivially also holds for the TTME-MSR-Ext case, where $\mathcal{R}'_l = \Box$.

The inference rule $(P'_l \to C'_l) \in \mathcal{I}'$ must be a new variant of some instance of some inference rule $(P^{base} \to C^{base}) \in \mathcal{I}$. Let $P_l \to C_l$ be any new variant of $P^{base} \to C^{base}$, and consider the set W from A.22. With $P_l \to C_l$ being a *new* variant, it will clearly hold $Var(P_l \to C_l) \cap (W \cup Var(\mathcal{K}_l)) = \emptyset$. Thus we can apply Lemma A.1.10, which gives us the following theory inference step with multiset-MGU σ_l:

$$\mathcal{K}_l \Rightarrow_{P_l \to C_l,\, \sigma_l} \mathcal{R}_l\sigma_l \ . \tag{A.26}$$

Hence, a PTME-I-Ext step based on $\mathcal{K}_l$, $P_l \to C_l$, σ_l and extending clauses E_l, yielding residue $\langle \mathcal{R}_l, \sigma_l \rangle$ can be carried out to Q_l.

Further, Lemma A.1.10 also gives us a substitution δ_{l+1} such that

$$\delta_l\gamma_l\sigma'_l = \sigma_l\delta_{l+1}\ [W] \tag{A.27}$$

$$\mathcal{R}'_l\delta_l\gamma_l\sigma'_l = \mathcal{R}_l\sigma_l\delta_{l+1} \ . \tag{A.28}$$

This will be needed below.

In both cases (TTME-MSR and PTME-$\mathcal{I}$), the resulting branch set Q_{l+1} is

$$Q_{l+1} = ([p_l] \circ_\times (\mathcal{R}_l \vee R_1 \vee \cdots \vee R_n) \cup (\mathcal{Q}_l \setminus \{\!\{[p_l]\}\!\}))\sigma_l \ . \tag{A.29}$$

In order to prove Equation A.9 for $l+1$ we have to show $Q'_{l+1} = Q_{l+1}\delta_{l+1}$:

$$\begin{aligned}
Q'_{l+1} &= ([p'_l] \circ_\times (\mathcal{R}'_l \vee R'_1 \vee \cdots \vee R'_n) \cup (\mathcal{Q}'_l \setminus \{\!\{[p'_l]\}\!\}))\sigma'_l \\
&= ([p_l\delta_l] \circ_\times (\mathcal{R}'_l \vee R_1\gamma_l \vee \cdots \vee R_n\gamma_l) \cup (\mathcal{Q}_l\delta_l \setminus \{\!\{[p_l\delta_l]\}\!\}))\sigma'_l \\
&\qquad \text{(by A.9 for } l \text{ and (A.16))} \\
&= ([p_l\delta_l\gamma_l] \circ_\times (\mathcal{R}'_l \vee R_1\delta_l\gamma_l \vee \cdots \vee R_n\delta_l\gamma_l) \cup (\mathcal{Q}_l\delta_l\gamma_l \setminus \{\!\{[p_l\delta_l\gamma_l]\}\!\}))\sigma'_l \\
&\qquad \text{(by (A.18) and (A.20))} \\
&= [p_l\delta_l\gamma_l\sigma'_l] \circ_\times (\mathcal{R}'_l\delta_l\gamma_l\sigma'_l \vee R_1\delta_l\gamma_l\sigma'_l \vee \cdots \vee R_n\delta_l\gamma_l\sigma'_l) \\
&\qquad\qquad \cup (\mathcal{Q}_l\delta_l\gamma_l\sigma'_l \setminus \{\!\{[p_l\delta_l\gamma_l\sigma'_l]\}\!\}) \\
&\qquad \text{(by distributing } \sigma'_l \text{ and (A.25))} \\
&= [p_l\sigma_l\delta_{l+1}] \circ_\times (\mathcal{R}_l\sigma_l\delta_{l+1} \vee R_1\sigma_l\delta_{l+1} \vee \cdots \vee R_n\sigma_l\delta_{l+1}) \\
&\qquad\qquad \cup (\mathcal{Q}_l\sigma_l\delta_{l+1} \setminus \{\!\{[p_l\sigma_l\delta_{l+1}]\}\!\}) \\
&\qquad \text{(by (A.22) (TTME-MSR), resp.} \\
&\qquad\ \text{by (A.27) and (A.28) (PTME-}\mathcal{I}\text{))} \\
&= Q_{l+1}\delta_{l+1} \qquad \text{by (A.29)}
\end{aligned}$$

We have to prove Equation A.10 for $l+1$, i.e. for $j = 1, \ldots, l$:

$$\mathcal{K}'_j\sigma'_j\cdots\sigma'_{l-1}\sigma'_l = \mathcal{K}_j\sigma_j\cdots\sigma_{l-1}\sigma_l\delta_{l+1}. \tag{A.30}$$

We distinguish two cases:

$j = l$: $\mathcal{K}'_l\sigma'_l \overset{(A.21)}{=} \mathcal{K}_l\delta_l\gamma_l\sigma'_l \overset{(A.22,A.27)}{=} \mathcal{K}_l\sigma_l\delta_{l+1}$.

$1 \leq j \leq l-1$: By construction, γ_l acts on the new variants E_l only, and hence does not affect $\mathcal{K}'_j\sigma'_j\cdots\sigma'_{l-1}$. Thus

$$\begin{aligned}(\mathcal{K}'_j\sigma'_j\cdots\sigma'_{l-1})\sigma'_l &= (\mathcal{K}'_j\sigma'_j\cdots\sigma'_{l-1}\gamma_l)\sigma'_l \\ &\overset{(A.10)}{=} (\mathcal{K}_j\sigma_j\cdots\sigma_{l-1})\delta_l\gamma_l\sigma'_l \\ &\overset{(A.22,A.27)}{=} (\mathcal{K}_j\sigma_j\cdots\sigma_{l-1})\sigma_l\delta_{l+1} = \mathcal{K}_j\sigma_j\cdots\sigma_l\delta_{l+1}.\end{aligned}$$

Thus, in both cases, A.10 holds for $l+1$. The proof of property A.11 for $l+1$ is analogue and hence is omitted.

It remains to prove preservation of stability wrt. minimality and preservation of regularity. Concerning the stability wrt. minimality we use the just derived result, Equation A.10, and conclude with the given minimal $\mathcal{T}$-complementarity of $\mathcal{K}'_j\sigma'_j\cdots\sigma'_l$ and Proposition 4.2.1.2 that $\mathcal{K}_j\sigma_j\cdots\sigma_l$ is minimal $\mathcal{T}$-complementary as well.

Concerning regularity, we have to show that every occurrence of a branch $[p] \in \mathcal{Q}_{l+1}$ is regular. From $Q'_{l+1} = Q_{l+1}\delta_{l+1}$ it follows that there is a corresponding occurrence of branch $[p'] \in Q'_{l+1}$ such that $[p'] = [p\delta_{l+1}]$. Now, regularity of $[p]$ holds by Proposition 3.3.1 and regularity of $[p']$, which holds if $\mathcal{D}'$ is given as regular.

Independence of the Computation Rule The "next" subgoal to be processed in model elimination derivations may be chosen in a don't-care nondeterministical way. This result is known as the "independence of the computation rule" (cf. also Section 3.3). The proof works by developing a *switching lemma* which says that two subsequent selected subgoals may be extended in exchanged order. When applied repeatedly, this shows us that any refutation can be reordered until it is in accordance with a given computation rule.

There are basically two approaches: the easier approach operates on the ground-level (recall that we use a ground-proof-and-lifting technique). This, however, has the disadvantage that the computation rule must be demanded to be "stable under lifting", i.e. that if a computation rule selects a literal $L\gamma$ within a clause $C\gamma$, then it has to selected L in C.

Since such a restriction would be poorly motivated, and is not even necessary, we will develop the result directly on the first-order level. The same approach was taken in [Lloyd, 1987] for SLD-resolution. However, a non-trivial complication comes in by the "stability wrt. minimality" (cf. Definitions 4.4.4 and 4.5.7) which trivially holds in the non-theory case.

The "independence of the computation rule" holds for both, TTME-MSR and PTME-I. Since the argumentation of the proof has much in common with

the preceding lifting lemma, we will adopt the conventions A.1.1 here, too. The key to the desired result is the following lemma:

Lemma A.1.11 (Switching Lemma). *Suppose a refutation $\mathcal{D}$, stable wrt. minimality,*

$$\begin{aligned}
\mathcal{D} = \;& \mathcal{Q}_1 \vdash \cdots \vdash \mathcal{Q}_{i-1} \\
& \vdash [p_i], [p_{i+1}], \mathcal{Q} && =: \mathcal{Q}_i \\
& \vdash_{[p_i], \mathcal{K}_i, P_i \to C_i, \langle \mathcal{R}_i, \sigma_i \rangle, E_i} (\mathcal{P}_i, [p_{i+1}], \mathcal{Q})\sigma_i && =: \mathcal{Q}_{i+1} \\
& \vdash_{[p_{i+1}\sigma_i], \mathcal{K}_{i+1}, P_{i+1} \to C_{i+1}, \langle \mathcal{R}_{i+1}, \sigma_{i+1} \rangle, E_{i+1}} (\mathcal{P}_i, \mathcal{P}_{i+1}, \mathcal{Q})\sigma_i\sigma_{i+1} && =: \mathcal{Q}_{i+2} \\
& \vdash \mathcal{Q}_{i+3} \vdash \cdots \vdash \mathcal{Q}_l)
\end{aligned}$$

with answer substitution $\sigma_1 \cdots \sigma_{i-1}\sigma_i \cdots \sigma_{l-1}$ as given, where

$$\begin{aligned}
E_i &= \{\!|L_1^i \vee R_1^i, \ldots, L_{m_i}^i \vee R_{m_i}^i|\!\} \\
\mathcal{P}_i &= [p_i] \circ_\times (\mathcal{R}_i \vee R_1^i \vee \cdots \vee R_{m_i}^i) \\
E_{i+1} &= \{\!|L_1^{i+1} \vee R_1^{i+1}, \ldots, L_{m_{i+1}}^{i+1} \vee R_{m_{i+1}}^{i+1}|\!\} \\
\mathcal{P}_{i+1} &= [p_{i+1}] \circ_\times (\mathcal{R}_{i+1} \vee R_1^{i+1} \vee \cdots \vee R_{m_{i+1}}^{i+1}) \;.
\end{aligned}$$

Then also a refutation $\mathcal{D}'$ of the same clause set exists, with the same start clause, stable wrt. minimality, where the extension steps yielding $\mathcal{Q}_{i+1}$ and $\mathcal{Q}_{i+2}$ are swapped. More precisely, $\mathcal{D}'$ takes the form

$$\begin{aligned}
\mathcal{D}' = \;& \mathcal{Q}_1 \vdash \cdots \vdash \mathcal{Q}_{i-1} \\
& \vdash [p_i], [p_{i+1}], \mathcal{Q} \\
& \vdash_{[p_{i+1}], \mathcal{K}_i', P_{i+1} \to C_{i+1}, \langle \mathcal{R}_{i+1}, \sigma_i' \rangle, E_{i+1}} ([p_i], \mathcal{P}_{i+1}, \mathcal{Q})\sigma_i' && =: \mathcal{Q}_{i+1}' \\
& \vdash_{[p_i\sigma_i], \mathcal{K}_{i+1}', P_i \to C_i, \langle \mathcal{R}_i, \sigma_{i+1}' \rangle, E_i} (\mathcal{P}_i, \mathcal{P}_{i+1}, \mathcal{Q})\sigma_i'\sigma_{i+1}' && =: \mathcal{Q}_{i+2}' \\
& \vdash \mathcal{Q}_{i+3}' \vdash \cdots \vdash \mathcal{Q}_l')
\end{aligned}$$

with computed substitution $\sigma_1 \cdots \sigma_{i-1}\sigma_i' \cdots \sigma_{l-1}'$, and there is a substitution δ_l' such that

$$Q_l = Q_l'\delta_l, \tag{A.31}$$

$$\begin{aligned}
E_j\sigma_j \cdots \sigma_{i-1}\sigma_i \cdots \sigma_{l-1} &= E_j\sigma_j \cdots \sigma_{i-1}\sigma_i' \cdots \sigma_{l-1}'\delta_l' \\
&\quad \textit{(for } j = 1, \ldots, i-1\textit{, with } E_j \textit{ as extending clauses)} \\
E_i\sigma_i \cdots \sigma_{l-1} &= E_i\sigma_{i+1}' \cdots \sigma_{l-1}'\delta_l' \\
E_{i+1}\sigma_{i+1} \cdots \sigma_{l-1} &= E_{i+1}\sigma_i' \cdots \sigma_{l-1}'\delta_l' \\
E_j\sigma_j \cdots \sigma_{l-1} &= E_j\sigma_j' \cdots \sigma_{l-1}'\delta_l' \\
&\quad \textit{(for } j = i+2, \ldots, l-1\textit{, with } E_j \textit{ as extending clauses)}
\end{aligned} \tag{A.32}$$

Furthermore, regularity is preserved (that is, if $\mathcal{D}$ is regular, then so is $\mathcal{D}'$).

Both the statement and the proof of the lemma are an extension towards theory-reasoning of the respective lemma for SLD-resolution in [Lloyd, 1987], with the additional complication that stability wrt. minimality is invariant. Interestingly, the proof will not go through if stability wrt. minimality would not hold for the given derivation.

Proof. Let $\mathcal{L}_i = \{\!|L_1^i, \ldots, L_{m_i}^i|\!\}$ denote the extending literals of the "first given" extension step. That is, $\mathcal{K}_i = \mathcal{B}_i \cup \{\!|Leaf([p_i])|\!\} \cup \mathcal{L}_i$, for some $\mathcal{B}_i \subseteq p_i \setminus \{\!|Leaf([p_i])|\!\}$. Analogously, let $\mathcal{L}_{i+1} = \{\!|L_1^{i+1}, \ldots, L_{m_{i+1}}^{i+1}|\!\}$ denote the extending literals of the "second given" extension step. That is, $\mathcal{K}_{i+1} = \mathcal{B}_{i+1}\sigma_i \cup \{\!|Leaf([p_{i+1}\sigma_i])|\!\} \cup \mathcal{L}_{i+1}$, for some $\mathcal{B}_{i+1} \subseteq p_{i+1} \setminus \{\!|Leaf([p_{i+1}])|\!\}$.

We note that since $\mathcal{L}_{i+1}$ stems from new variants we can assume that σ_i does not act on $\mathcal{L}_{i+1}$, i.e.

$$\mathcal{L}_{i+1} = \mathcal{L}_{i+1}\sigma_i \ . \tag{A.33}$$

For both calculi variants this gives us

$$\begin{aligned} \mathcal{K}_{i+1}\sigma_{i+1} &= \mathcal{B}_{i+1}\sigma_i\sigma_{i+1} \cup \{\!|Leaf([p_{i+1}\sigma_i\sigma_{i+1}])|\!\} \cup \mathcal{L}_{i+1}\sigma_{i+1} \\ &\overset{(A.33)}{=} \mathcal{B}_{i+1}\sigma_i\sigma_{i+1} \cup \{\!|Leaf([p_{i+1}\sigma_i\sigma_{i+1}])|\!\} \cup \mathcal{L}_{i+1}\sigma_i\sigma_{i+1} \\ &= (\underbrace{\mathcal{B}_{i+1} \cup \{\!|Leaf([p_{i+1}])|\!\} \cup \mathcal{L}_{i+1}}_{=:\mathcal{K}_i'})\sigma_i\sigma_{i+1} \ . \end{aligned} \tag{A.34}$$

We will swap the given two extension steps now, starting with the PTME-I version. The second given extension step is based on the theory inference

$$\mathcal{K}_{i+1} \Rightarrow_{P_{i+1} \to C_{i+1}, \sigma_{i+1}} \mathcal{R}_{i+1}\sigma_{i+1} \ . \tag{A.35}$$

Since always *new* variants of inference rules are used, σ_i will not act on the new variant $P_{i+1} \to C_{i+1}$ used in the given second step (and hence trivially does not act on $\mathcal{R}_{i+1}$). In other words, $P_{i+1} \to C_{i+1} = (P_{i+1} \to C_{i+1})\sigma_i$. Together with A.34 the theory inference A.35 can be reformulated as follows:

$$\mathcal{K}_i'\sigma_i \Rightarrow_{(P_{i+1} \to C_{i+1})\sigma_i,\ \sigma_{i+1}} \mathcal{R}_{i+1}\sigma_i\sigma_{i+1} \ .$$

This, however, is the same inference as

$$\mathcal{K}_i' \Rightarrow_{P_{i+1} \to C_{i+1},\ \sigma_i\sigma_{i+1}} \mathcal{R}_{i+1}\sigma_i\sigma_{i+1} \ .$$

Now let

$$\begin{aligned} X &= \bigcup_{j=1}^{i-1} (\mathit{Var}(\mathcal{K}_j\sigma_j \cdots \sigma_{i-1}) \cup \mathit{Var}(E_j\sigma_j \cdots \sigma_{i-1})) \\ W_i &= \mathit{Var}(p_i) \cup \mathit{Var}(p_{i+1}) \cup \mathit{Var}(E_i) \cup \mathit{Var}(E_{i+1}) \cup \mathit{Var}(\mathcal{Q}) \cup \\ &\quad \mathit{Var}(\mathcal{R}_i) \cup X \end{aligned}$$

and apply Lemma A.1.10 (it is easily verified that the preconditions are met). It gives us the theory inference

$$\mathcal{K}'_i \Rightarrow_{P_{i+1}\to C_{i+1},\,\sigma'_i} \mathcal{R}_{i+1}\sigma'_i$$

with multiset-MGU σ'_i, and a substitution γ'_i such that

$$\sigma_i\sigma_{i+1} = \sigma'_i\gamma'_i\,[W_i] \tag{A.36}$$

$$\mathcal{R}_{i+1}\sigma_i\sigma_{i+1} = \mathcal{R}_{i+1}\sigma'_i\gamma'_i. \tag{A.37}$$

Hence, a PTME-I-Ext step can be carried out to $\mathcal{Q}_i$, yielding $\mathcal{Q}_{i+1}$, as suggested.

In case of TTME-MSR, by definition of TTME-MSR-Ext, σ_{i+1} is a minimal $\mathcal{T}$-MGR for $\mathcal{K}_{i+1}$. From A.34 we learn that $\sigma_i\sigma_{i+1}$ is a, although minimal, not necessarily most general $\mathcal{T}$-refuter for $\mathcal{K}'_i$. However, with Proposition 4.4.1 we conclude that there is a $\mathcal{T}$-MGR $\sigma'_i \in MSR_{\mathcal{T}}(\mathcal{K}'_i)$, and that there is a substitution γ'_i such that

$$\sigma_i\sigma_{i+1} = \sigma'_i\gamma'_i\,[W_i]\ . \tag{A.38}$$

By Proposition 4.2.1.2 we know that with $\sigma_i\sigma_{i+1}$ being a minimal $\mathcal{T}$-MGR, the more general σ'_i is a minimal $\mathcal{T}$-MGR for $\mathcal{K}'_i$, too. Hence, a TTME-MSR-Ext step can be carried to $\mathcal{Q}_i$, yielding $\mathcal{Q}_{i+1}$ as suggested.

For both variants recall that the extending clauses always are new variants; hence we clearly have $Var(E_i) \cap Var(\mathcal{K}'_i) = \emptyset$. For TTME-MSR, by definition of $MSR_{\mathcal{T}}$ we know that $Dom(\sigma'_i) \subseteq Var(\mathcal{K}'_i)$. Thus, $Dom(\sigma'_i) \cap Var(E_i) = \emptyset$, and in particular, σ'_i will not act on the variables of $\mathcal{L}_i$. For PTME-I, we can assume that $Dom(\sigma'_i) \subseteq Var(\mathcal{K}'_i) \cup Var(P_{i+1} \to C_{i+1})$. With $P_{i+1} \to C_{i+1}$ being a new variant, we can further assume that σ'_i does not act on the variables of $\mathcal{L}_i$, and that σ'_i will not act on the variables of the new variant $P_i \to C_i$. The latter fact implies in particular

$$\mathcal{R}_i = \mathcal{R}_i\sigma'_i\ . \tag{A.39}$$

For both variants, let

$$W_{i+1} = Var(p_i\sigma'_i) \cup Var(p_{i+1}\sigma'_i) \cup Var(E_i\sigma'_i) \cup Var(E_{i+1}\sigma'_i) \cup Var(\mathcal{Q}\sigma'_i) \cup Var(\mathcal{R}_{i+1}\sigma'_i) \cup Var(X\sigma'_i).$$

Next we will show how to append the given first inference step to $\mathcal{Q}'_{i+1}$. Again, we start with PTME-I: the given first PTME-I-Ext step is based on the following theory inference:

$$\mathcal{K}_i \Rightarrow_{P_i\to C_i,\sigma_i} \mathcal{R}_i\sigma_i\ .$$

Clearly, we can instantiate with σ_{i+1}, yielding

$$\mathcal{K}_i \Rightarrow_{P_i \to C_i, \sigma_i \sigma_{i+1}} \mathcal{R}_i \sigma_i \sigma_{i+1} \ . \tag{A.40}$$

We know by minimality requirement of theory inferences that $\mathcal{K}_i\sigma_i$ contains no duplicates. By the given stability wrt. minimality of $\mathcal{D}$, it holds even the stronger result that $\mathcal{K}_i\sigma_i\sigma_{i+1}$ contains no duplicates. Thus A.40 is a minimal theory inference (although $\sigma_i\sigma_{i+1}$ need not be a multiset-MGU).

Recall from above that σ_i' will not act on the variables of $\mathcal{L}_i$. Hence we have

$$\begin{aligned}\mathcal{K}_i\sigma_i\sigma_{i+1} &\overset{(A.36)}{=} \mathcal{K}_i\sigma_i'\gamma_i' \\ &= (\mathcal{B}_i \cup \{\!|Leaf([p_i])|\!\} \cup \mathcal{L}_i)\sigma_i'\gamma_i' \\ &= (\underbrace{\mathcal{B}_i\sigma_i' \cup \{\!|Leaf([p_i\sigma_i'])|\!\} \cup \mathcal{L}_i}_{=:\mathcal{K}_{i+1}'})\gamma_i' \ .\end{aligned}$$

Recall further from above that σ_i' will not act on the variables of $P_i \to C_i$. Hence we have

$$(P_i \to C_i)\sigma_i\sigma_{i+1} \overset{(A.36)}{=} (P_i \to C_i)\sigma_i'\gamma_i' = (P_i \to C_i)\gamma_i' \ .$$

Altogether, A.40 can be rewritten to the minimal theory inference

$$\mathcal{K}_{i+1}' \Rightarrow_{P_i \to C_i, \gamma_i'} \mathcal{R}_i\gamma_i' \ .$$

Now we can apply Lemma A.1.10 (again, it is easily verified that the preconditions are met), which gives us the theory inference

$$\mathcal{K}_{i+1}' \Rightarrow_{P_i \to C_i, \sigma_{i+1}'} \mathcal{R}_i\sigma_{i+1}'$$

with multiset-MGU σ_{i+1}' and a substitution γ_{i+1}' such that

$$\gamma_i' = \sigma_{i+1}'\gamma_{i+1}' \ [W_{i+1}] \tag{A.41}$$

$$\mathcal{R}_i\gamma_i' = \mathcal{R}_i\sigma_{i+1}'\gamma_{i+1}' \ . \tag{A.42}$$

Hence, a PTME-I-Ext step can be carried out to $\mathcal{Q}_{i+1}'$, yielding $\mathcal{Q}_{i+2}'$, as suggested.

Now we turn to TTME-MSR. From the given first extension step we know that $\mathcal{K}_i\sigma_i$ is minimal $\mathcal{T}$-complementary. By Proposition 4.2.1.1 we know that $\mathcal{K}_i\sigma_i\sigma_{i+1}$ also is $\mathcal{T}$-complementary. We need the stronger result that $\sigma_i\sigma_{i+1}$ is even a *minimal* $\mathcal{T}$-refuter for $\mathcal{K}_i$, which holds because $\mathcal{D}$ is given as stable wrt. minimality. Thus

$$\mathcal{K}_i\sigma_i\sigma_{i+1} \overset{(A.38)}{=} \mathcal{K}_i\sigma_i'\gamma_i'$$

is minimal $\mathcal{T}$-complementary. Recall from above that σ_i will not act on the variables of $\mathcal{L}_i$. Hence we continue

$$\mathcal{K}_i\sigma_i'\gamma_i' = (\mathcal{B}_i \cup \{\!|Leaf([p_i])|\!\} \cup \mathcal{L}_i)\sigma_i'\gamma_i' = \underbrace{(\mathcal{B}_i\sigma_i \cup \{\!|Leaf([p_i\sigma_i])|\!\} \cup \mathcal{L}_i)}_{=:\mathcal{K}'_{i+1}}\gamma_i' \ .$$

Hence, with $\sigma_i\sigma_{i+1}$ being a minimal $\mathcal{T}$-refuter for $\mathcal{K}_i$, γ_i' is also a minimal $\mathcal{T}$-refuter for $\mathcal{K}'_{i+1}$. By Proposition 4.4.1 we conclude that a $\mathcal{T}$-MGR $\sigma'_{i+1} \in MSR_{\mathcal{T}}(\mathcal{K}'_{i+1})$ exists, and that a substitution γ'_{i+1} exists such that

$$\gamma_i' = \sigma'_{i+1}\gamma'_{i+1}\ [W_{i+1}] \ . \tag{A.43}$$

By Proposition 4.2.1.2 we know that with γ_i' being a minimal $\mathcal{T}$-MGR for $\mathcal{K}'_{i+1}$, the more general σ'_{i+1} also is a minimal $\mathcal{T}$-MGR for $\mathcal{K}'_{i+1}$. Hence, a TTME-MSR-Ext step can be carried to $\mathcal{Q}'_{i+1}$, yielding $\mathcal{Q}'_{i+2}$ as suggested.

This shows us that the first and second extension step can be swapped. We are going to show now $\mathcal{Q}_{i+2} = \mathcal{Q}'_{i+2}\gamma'_{i+1}$[4]. As an abbreviation we use $R^i = R^i_1 \vee \cdots \vee R^i_{m_i}$:

$$\begin{aligned}
\mathcal{Q}_{i+2} &= (\mathcal{P}_i, \mathcal{P}_{i+1}, \mathcal{Q})\sigma_i\sigma_{i+1} \\
&= ([p_i] \circ_\times (\mathcal{R}_i \vee R^i), [p_{i+1}] \circ_\times (\mathcal{R}_{i+1} \vee R^{i+1}), \mathcal{Q})\sigma_i\sigma_{i+1} \\
&= ([p_i] \circ_\times (\mathcal{R}_i \vee R^i), [p_{i+1}] \circ_\times (\mathcal{R}_{i+1} \vee R^{i+1}), \mathcal{Q})\sigma_i'\gamma_i' \\
&\quad \text{(by A.36 and A.37 (PTME-I),} \\
&\quad\ \text{by A.38 (TTME-MSR))} \\
&= ([p_i] \circ_\times (\mathcal{R}_i \vee R^i), [p_{i+1}] \circ_\times (\mathcal{R}_{i+1} \vee R^{i+1}), \mathcal{Q})\sigma_i'\sigma'_{i+1}\gamma'_{i+1} \\
&\quad \text{(by A.41, A.42 (PTME-I),} \\
&\quad\ \text{by A.43 (TTME-MSR))} \\
&= (\mathcal{P}_i, \mathcal{P}_{i+1}, \mathcal{Q})\sigma_i'\sigma'_{i+1}\gamma'_{i+1} \\
&= \mathcal{Q}'_{i+2}\gamma'_{i+1} \ .
\end{aligned}$$

Hence we can write the rest of the given derivation $\mathcal{D}$ as

$$\mathcal{D}_R = \mathcal{Q}'_{i+2}\gamma'_{i+1} \vdash \mathcal{Q}_{i+3} \vdash \cdots \vdash \mathcal{Q}_n \ .$$

We wish to lift $\mathcal{D}_R$ to a derivation $\mathcal{D}'_R$ starting with $\mathcal{Q}'_{i+2}$, which would yield the tail of the desired refutation $\mathcal{D}'$. This requires a slight generalization of the lifting lemma (Lemma A.1.9) by allowing to start derivations with any instantiated branch set, instead of a branch set corresponding to a start clause. In this case, $\mathcal{Q}'_{i+2}\gamma'_{i+1}$ will be lifted to a refutation

$$\mathcal{D}'_R = \mathcal{Q}'_{i+2} \vdash \mathcal{Q}'_{i+3} \vdash \cdots \vdash \mathcal{Q}'_n \ .$$

Further, this lifting preserves regularity which implies the regularity of $\mathcal{D}'$, provided that $\mathcal{D}$ is regular.

[4] It can even be shown that for some γ_{i+1} it holds $\mathcal{Q}'_{i+2} = \mathcal{Q}_{i+2}\gamma_{i+1}$. That is, $\mathcal{Q}'_{i+2}$ and $\mathcal{Q}_{i+2}$ are variants. However, as the further proof shows, this argumentation is not necessary.

We still have to show that in $\mathcal{D}'$ is stable wrt. minimality. By the identities derived so far we can obtain the following ($j = 1, \ldots, i-1$):

$$\begin{aligned}
\mathcal{K}_j\sigma_j \cdots \sigma_{i-1}\sigma'_i\sigma'_{i+1}\gamma'_{i+1} &= \mathcal{K}_j\sigma_j \cdots \sigma_{i-1}\sigma_i\sigma_{i+1} \\
\mathcal{K}'_i\sigma'_i\sigma'_{i+1}\gamma'_{i+1} &= \mathcal{K}_{i+1}\sigma_{i+1} \\
&\qquad \text{(by A.34, A.36 and A.41 (PTME-I),} \\
&\qquad \text{by A.34, A.38 and A.43 (TTME-MSR))} \\
\mathcal{K}'_{i+1}\sigma'_{i+1}\gamma'_{i+1} &= \mathcal{K}_i\sigma_i\sigma_{i+1} \\
&\qquad \text{(by Def. of } \mathcal{K}'_{i+1} \text{ and A.41 (PTME-I),} \\
&\qquad \text{resp. A.43 (TTME-MSR))} \ .
\end{aligned}$$

In words, the left sides of the equations are just the key sets in $\mathcal{D}'$ up to $\mathcal{Q}'_{i+2}$, instantiated by the computed answer substitution derived so far and γ'_{i+1}.

Now consider, for instance, $\mathcal{K}'_i\sigma'_i\sigma'_{i+1}\gamma'_{i+1}$ (for the other listed key sets the following argumentation is the same). In $\mathcal{D}_R$, this set is further instantiated towards $\mathcal{K}'_i\sigma'_i\sigma'_{i+1}\gamma'_{i+1}\sigma_{i+2} \cdots \sigma_{l-1}$. Again slightly generalizing the lifting lemma, the computed answer in $\mathcal{D}'_R$, say $\sigma'_{i+2} \cdots \sigma'_{l-1}$, leaves us more general key sets. The lifting lemma also gives us a substitution δ'_l such that

$$\begin{aligned}
\mathcal{K}'_i\sigma'_i\sigma'_{i+1}\sigma'_{i+2} \cdots \sigma'_{l-1}\delta'_l &= \mathcal{K}'_i\sigma'_i\sigma'_{i+1}\gamma'_{i+1}\sigma_{i+2} \cdots \sigma_{l-1} \\
&= \mathcal{K}_{i+1}\sigma_{i+1}\sigma_{i+2} \cdots \sigma_{l-1} \ .
\end{aligned}$$

Now, the given stability wrt. minimality of $\mathcal{D}$ means in particular that $\mathcal{K}_{i+1}\sigma_{i+1}\sigma_{i+2} \cdots \sigma_{l-1}$ contains no duplicates. But then $\mathcal{K}'_i\sigma'_i\sigma'_{i+1}\sigma'_{i+2} \cdots \sigma'_{l-1}$ trivially cannot contain duplicates either. Using this argumentation for all the key sets listed above convinces us that $\mathcal{D}'$ is stable wrt. minimality, too.

For the other properties A.31 and A.32 the same argumentation as for the key sets applies and hence is omitted.

Proposition A.1.3 (Independence of the Computation Rule). *Let P be a program, $\leftarrow Q$ be a query (cf. Def. 3.2.5) and $\mathcal{D}$ be a PTME-$\mathcal{I}$ refutation (resp. TTME-MSR refutation) of P and $\leftarrow Q$ with computed answer $\{Q_1, \ldots, Q_l\}$. Assume that $\mathcal{D}$ is stable wrt. minimality (Def. 4.5.7, resp. Def. 4.4.4).*

Then, for any computation rule c, a PTME-$\mathcal{I}$ refutation (resp. TTME-MSR refutation) $\mathcal{D}'$ of P and $\leftarrow Q$ exists with computed answer $\{Q'_1, \ldots, Q'_l\}$ via c, and, for some substitution δ', it holds that $\{Q'_1, \ldots, Q'_l\}\delta' = \{Q_1, \ldots, Q_l\}$.

Furthermore, regularity is preserved. That is, if $\mathcal{D}$ is strictly regular, then so is $\mathcal{D}'$.

Proof. Let

$$\mathcal{D} = \mathcal{Q}_1 \vdash \cdots \vdash \mathcal{Q}_{i-1} \vdash \mathcal{Q}_i \vdash \cdots \vdash (\square = \mathcal{Q}_l)$$

be the given refutation, and suppose that $\mathcal{D}$ is in accordance with c up to inclusive $i-1$. That is, c selects a branch $[p_i] \in \mathcal{Q}_i$ different from the actual

selected branch in $\mathcal{D}$ in $\mathcal{Q}_i$. Since $\mathcal{D}$ is a refutation, at some stage $j > i$ the instance $[p_i\sigma_i \cdots \sigma_{j-1}]$ must be selected in $\mathcal{D}$, where $\sigma_i \cdots \sigma_{j-1}$ are the substitutions used in the intermediate inferences. By applying the Switching Lemma (Lemma A.1.11) $j - i$ times, the branch $[p_i\sigma_i \cdots \sigma_{j-1}]$ can stepwisely be moved backwards until it is selected as $[p_i]$ in $\mathcal{Q}_i$.

Concerning the stated property of the computed answer recall that the members of the computed answer consists of (1) the literals of the start clause and (2) of the literals of the query used as extending clauses. Now, the switching lemma gives us a δ' such that $\mathcal{Q}'_l\delta' = \mathcal{Q}_l$, where $\mathcal{Q}'_l$ is the concluding (closed) tableau in the switched refutation, say $\mathcal{D}'$. Hence, in particular $Q'_1\delta' = Q_1$ where Q_1 (resp. Q'_1 is (without loss of generality) the start clause in $\mathcal{D}$ (resp. $\mathcal{D}'$). Concerning the usage of $\leftarrow Q$ as extending clauses we also learn from the switching lemma that the extending clauses in $\mathcal{D}'$ are the *same* as those of $\mathcal{D}$, and their final instantiations along $\mathcal{D}'$ can be mapped by δ into the respective final instantiations in $\mathcal{D}$ (in fact, these are variants). Hence, in particular $\{Q'_1, \ldots, Q'_l\}\delta' = \{Q_1, \ldots, Q_l\}$.

Notice that $\mathcal{D}'$ is one more step in accordance with c as $\mathcal{D}$. Further, since the Switching Lemma preserves the length of derivations, repeated application of this procedure terminates and finally results in the desired refutation wrt. c.

A.1.2 Ground Completeness of PTME-I

This section is devoted to the critical lemma in the proof of Theorem 4.5.3. The crucial point is how to transform a single TTME-MSR-Ext step into a sequence of PTME-I steps.

Lemma A.1.12. *Let $\mathcal{I}$ be a ground complete inference system wrt. a theory $\mathcal{T}$ (cf. Def. 4.5.6). Suppose a ground* TTME-MSR-Ext *inference*

$$\mathcal{P} \vdash_{[q],\mathcal{K},\{\!|C_1,\ldots,C_n|\!\}} \mathcal{P}'$$

as given, where $\mathcal{P}$ and $\mathcal{P}'$ are ground branch sets and $C_1, \ldots, C_n$ are ground clauses from M. Furthermore suppose there are continuations (cf. Def. A.1.1) $\mathcal{C}_\mathcal{P}$ and $\mathcal{C}_{\mathcal{P}'}$ wrt. M of $\mathcal{P}$ and $\mathcal{P}'$, respectively, with $\mathcal{C}_\mathcal{P} \succcurlyeq \mathcal{C}_{\mathcal{P}'}$.

Then there is a sequence $(\mathcal{P}_0 = \mathcal{P}) \vdash \cdots \vdash \mathcal{P}_l$ of ground branch sets such that for $j = 1 \ldots l$, $\mathcal{P}_j$ is obtained by a PTME-$\mathcal{I}^g$ *step from $\mathcal{P}_{j-1}$, where $\mathcal{I}^g$ consists of all ground instances of all inference rules in $\mathcal{I}$.*

The extending clauses in these steps are all taken from $\{C_1, \ldots, C_n\}$, and furthermore there is a continuation $\mathcal{C}_{\mathcal{P}_l}$ of $\mathcal{P}_l$ with $\mathcal{C}_\mathcal{P} \succcurlyeq \mathcal{C}_{\mathcal{P}_l} \succcurlyeq \mathcal{C}_{\mathcal{P}'}$.

It is possible that the branch set $\mathcal{P}_l$ consists of more branches than the corresponding $\mathcal{P}'$, but every branch in $\mathcal{P}'$ is contained with possibly some extra branch literals in $\mathcal{P}_l$.

Proof. Assume that $C_i = L_i \vee R_i$ (for $i = 1, \ldots, n$) and $\mathcal{K} = \mathcal{B} \cup \{\!|\mathit{Leaf}([q])|\!\} \cup \mathcal{L}$, where $\mathcal{L} = \{\!|L_1, \ldots, L_n|\!\}$ and for some $\mathcal{B} \subseteq q \setminus \{\!|\mathit{Leaf}([q])|\!\}$. Further, $\mathcal{P}$ and $\mathcal{P}'$ are of the form

$$\begin{aligned} \mathcal{P} &= [q], \mathcal{Q} \\ \mathcal{P}' &= \mathcal{Q}', \mathcal{Q} \qquad \text{where } \mathcal{Q}' = [q] \circ_\times (R_1 \vee \cdots \vee R_n). \end{aligned}$$

By definition of TTME-MSR-Ext the key set $\mathcal{K}$ is minimal $\mathcal{T}$-unsatisfiable (recall that since everything is ground, no substitution is needed and $\mathcal{T}$-complementarity coincides with $\mathcal{T}$-unsatisfiability).

Since $\mathcal{I}$ is given as ground complete, there is a background refutation R of $\mathcal{K}$ with top literal $\mathit{Leaf}([q])$. It is of the form

$$(\mathcal{R}_0 = \mathit{Leaf}([q])) \overset{M_1}{\Longrightarrow} \mathcal{R}_1 \overset{M_2}{\Longrightarrow} \mathcal{R}_2 \cdots \mathcal{R}_{l-1} \overset{M_l}{\Longrightarrow} \square$$

where the side literals M_j $(1 \leq j \leq l)$ are taken from $\mathcal{K}$.

In order to transform this background refutation into a model elimination refutation starting from the branch $[q]$, we have to split the M_j's into two parts: let

$$\begin{aligned} \mathcal{B}_j &= M_j \cap \mathcal{B}, \\ \mathcal{L}_j &= M_j \cap \mathcal{L} \ . \end{aligned}$$

Intuitively, $\mathcal{B}_j$ is the subset of M_j which stems from the ancestor context of the key set $\mathcal{K}$ of the TTME-MSR-Ext step, and $\mathcal{L}_j$ is the subset of M_j which stems from the extending literals of the TTME-MSR-Ext step; $\mathcal{L}_j$ consists of literals from $\mathcal{L}$ and can be written as

$$\mathcal{L}_j = \{\!|L_1^j, \ldots, L_{n_j}^j|\!\}. \tag{A.44}$$

Let

$$\mathcal{E}_j = \{\!|L_1^j \vee R_1^j, \ldots, L_{n_j}^j \vee R_{n_j}^j|\!\}$$

be the corresponding clauses from $\{C_1, \ldots, C_n\}$. Using these definitions and rewriting $[q]$ as $[q] = [p \cdot \mathcal{R}_0]$, we can define the claimed sequence

$$\begin{array}{ll} [p \cdot \mathcal{R}_0], \mathcal{Q} & (=\mathcal{P}_0 = \mathcal{P}) \\ \vdash_{[p \cdot \mathcal{R}_0], \mathcal{K}_1, \mathcal{R}_1, E_1} [p \cdot \mathcal{R}_0 \cdot \mathcal{R}_1], \mathcal{Q}_1, \mathcal{Q} & =: \mathcal{P}_1 \\ \vdash_{[p \cdot \mathcal{R}_0 \cdot \mathcal{R}_1], \mathcal{K}_2, \mathcal{R}_2, E_2} [p \cdot \mathcal{R}_0 \cdot \mathcal{R}_1 \cdot \mathcal{R}_2], \mathcal{Q}_2, \mathcal{Q}_1, \mathcal{Q} & =: \mathcal{P}_2 \\ \quad \vdots & \\ \vdash_{[p \cdot \mathcal{R}_0 \cdots \mathcal{R}_{l-2}], \mathcal{K}_{l-1}, \mathcal{R}_{l-1}, E_{l-1}} [p \cdot \mathcal{R}_0 \cdots \mathcal{R}_{l-1}], \mathcal{Q}_{l-1}, \ldots, \mathcal{Q}_2, \mathcal{Q}_1, \mathcal{Q} & =: \mathcal{P}_{l-1} \\ \vdash_{[p \cdot \mathcal{R}_0 \cdots \mathcal{R}_{l-1}], \mathcal{K}_l, \mathcal{R}_l, E_l} [p \cdot \mathcal{R}_0 \cdots \mathcal{R}_{l-1}] \times, \mathcal{Q}_l, \ldots, \mathcal{Q}_2, \mathcal{Q}_1, \mathcal{Q} & =: \mathcal{P}_l \end{array}$$

where

$$\mathcal{Q}_j = [p \cdot \mathcal{R}_0 \cdot \mathcal{R}_1 \cdots \mathcal{R}_{j-1}] \circ (R_1^j \vee \cdots \vee R_{n_j}^j)$$
$$\mathcal{K}_j = \mathcal{B}_j \cup \{\mathcal{R}_{j-1}\} \cup \mathcal{L}_j \ .$$

That is, the j-th PTME-$\mathcal{I}^g$ step is based on the theory inference

$$\mathcal{B}_j \cup \{\mathcal{R}_{j-1}\} \cup \mathcal{L}_j \Rightarrow \mathcal{R}_j \ ,$$

which, using the identity $M_j = \mathcal{B}_j \cup \mathcal{L}_j$, exists according to the background refutation R.

It remains to prove the stated ordering property $\mathcal{C}_\mathcal{P} \succnsim \mathcal{C}_{\mathcal{P}_l} \succnsim \mathcal{C}_{\mathcal{P}'}$. Note that the given continuations are of the form

$$\begin{aligned} \mathcal{C}_\mathcal{P} &= \{N_{[q]}\} \cup \mathcal{C}_\mathcal{Q} \\ \mathcal{C}_{\mathcal{P}'} &= \mathcal{C}_{\mathcal{Q}'} \cup \mathcal{C}_\mathcal{Q} \end{aligned}$$

according to the structure of $\mathcal{P}$ and $\mathcal{P}'$, where $\mathcal{C}_\mathcal{Q}$ is a continuation of $\mathcal{Q}$. Since $\mathcal{C}_\mathcal{P} \succnsim \mathcal{C}_{\mathcal{P}'}$ we have by multiset ordering and the fact that $\{N_{[q]}\}$ is a singleton

$$\text{for every } N_{[q']} \in \mathcal{C}_{\mathcal{Q}'} \colon N_{[q]} \succnsim N_{[q']}. \tag{A.45}$$

Now consider a branch $[q \cdot \mathcal{R}_1 \cdots \mathcal{R}_{j-1} \cdot K] \in \mathcal{Q}_j$, where $K \in R_1^j \vee \cdots \vee R_{n_j}^j$. K stems from the Rest R_i of some given clause $L_i \vee R_i \in M$. For the given TTME-MSR-Ext step we know $[q \cdot K] \in \mathcal{Q}'$ and there is a respective continuation $N_{[q \cdot K]} \in \mathcal{C}_{\mathcal{Q}'}$. Now define

$$N_{[q \cdot \mathcal{R}_1 \cdots \mathcal{R}_{j-1} \cdot K]} = N_{[q \cdot K]}$$

as a continuation of $[q \cdot \mathcal{R}_1 \cdots \mathcal{R}_{j-1} \cdot K]$ wrt. M (the additional residue literals do not violate the continuation property). Note that with A.45 it holds that

$$N_{[q]} \succnsim N_{[q \cdot \mathcal{R}_1 \cdots \mathcal{R}_{j-1} \cdot K]} \ . \tag{A.46}$$

Since no special assumptions about $\mathcal{Q}_j$ were made we can generalize and find for every branch in $(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)$ a continuation in this way. Let $\mathcal{C}_{(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)}$ be the thus existing continuation of the branch set $(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)$ consisting of elements from $\mathcal{C}_{\mathcal{Q}'}$.

With (A.46) it follows $N_{[q]} \succnsim N$ for every $N \in \mathcal{C}_{(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)}$, and by property of multiset orderings thus ($\{N_{[q]}\}$ is a singleton)

$$\{N_{[q]}\} \succnsim \mathcal{C}_{(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)} \ .$$

Finally define $\mathcal{C}_{\mathcal{P}_l} = \mathcal{C}_{(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)} \cup \mathcal{C}_\mathcal{Q}$ as the desired continuation of $\mathcal{P}_l$. Also by property of multiset orderings it follows

$$\mathcal{C}_\mathcal{P} = \{N_{[q]}\} \cup \mathcal{C}_\mathcal{Q} \succnsim \mathcal{C}_{(\mathcal{Q}_l, \ldots, \mathcal{Q}_1)} \cup \mathcal{C}_\mathcal{Q} = \mathcal{C}_{\mathcal{P}_l} \ .$$

In order to see the other stated ordering property observe that every clause $L_i \vee R_i$ must be used at least once in the constructed partial derivation, because otherwise the key set $\mathcal{K}$ of the given TTME-MSR-Ext step would be not *minimal* $\mathcal{T}$-valid. Hence R_i is used in the construction of at least one $\mathcal{Q}_j$. Thus every element in $\mathcal{C}_{\mathcal{Q}'}$ occurs at least once in $\mathcal{C}_{(\mathcal{Q}_l,\ldots,\mathcal{Q}_1)}$. From this it follows

$$\mathcal{C}_{\mathcal{P}_l} = \mathcal{C}_{(\mathcal{Q}_l,\ldots,\mathcal{Q}_1)} \cup \mathcal{C}_{\mathcal{Q}} \succnsim \mathcal{C}_{\mathcal{Q}'} \cup \mathcal{C}_{\mathcal{Q}} = \mathcal{C}_{\mathcal{P}'} \ .$$

This lemma can be applied to whole derivations:

Lemma A.1.13 (PTME-I simulates TTME-MSR). *Let $\mathcal{I}_\mathcal{T}$ be a ground complete inference system wrt. a theory $\mathcal{T}$. Let $\mathcal{D}$ be a TTME-MSR refutation of a ground clause set M with start clause $G \in M$. Suppose $\mathcal{D}$ is constructed as figured out in the ground completeness proof for TTME-MSR (Lemma A.1.2). Thus, $\mathcal{D}$ is of the form*

$$\mathcal{D} = \mathcal{P}_1 \vdash \cdots \vdash \mathcal{P}_n$$

and there are respective continuations wrt. M

$$\mathcal{C}_1 \succ \cdots \succ (\mathcal{C}_n = \{\!\{\}\!\}).$$

Then there is a $PTME-\mathcal{I}^g_\mathcal{T}$ refutation

$$\mathcal{D}_{PTME-I} = \mathcal{P}_1 \vdash \mathcal{P}'_2 \vdash \cdots \vdash \mathcal{P}'_m,$$

of M, stable wrt. minimality and with start clause G.

Proof. By definition, every TTME-MSR-Ext step in the refutation $\mathcal{D}$ employs a minimal $\mathcal{T}$-complementary (ground) key set $\mathcal{K}$. By applying Lemma A.1.12 every such step can be replaced by a sequence of PTME-I-Ext extension steps. However, a little care must be taken for the inductive argument. Since Lemma A.1.12 possibly *in*creases the number of branches, simple induction on e.g. the number of occurrences of total extension steps does not work.

The lemma is shown by well-founded induction on the continuation $\mathcal{C}_1$ of the first tableau of derivations of the given form, which are compared by the multiset ordering "$\succ$" on complexities of continuations.

The *induction start* with $\mathcal{C}_1 = \{\!\{\}\!\}$ being trivial (take $\mathcal{P}'_m = \mathcal{P}_1$ which must be a closed tableau according to the definition of continuation) we turn immediately to the *induction step:* we have $\mathcal{C}_1 \neq \{\!\{\}\!\}$. Hence at least one TTME-MSR-Ext step is applied in $\mathcal{D}$. Next we apply Lemma A.1.12 to $\mathcal{P}_1$ and $\mathcal{P}_2$ and replace the underlying TTME-MSR-Ext step by a sequence of $\mathcal{I}^g_\mathcal{T}$-extension steps. Let $(\mathcal{P}_1 = \mathcal{P}'_0) \vdash \mathcal{P}'_1 \cdots \mathcal{P}'_l$ be the resulting sequence. Notice that Lemma A.1.12 also gives us $\mathcal{C}_1 \succ \mathcal{P}'_l$ (*). Also by that lemma there is a continuation $\mathcal{C}_{\mathcal{P}'_l}$ of $\mathcal{P}'_l$. Hence we can apply Lemma A.1.8 and obtain a TTME-MSR refutation $\mathcal{D}'$ starting with $\mathcal{P}'_l$.

$\mathcal{D}'$ is of the same form as the given $\mathcal{D}$, in that respective decreasing continuations corresponding to the tableaux along $\mathcal{D}''$ exist. Furthermore since (*) holds, the induction hypothesis can be applied to $\mathcal{D}'$, which yields a $PTME-\mathcal{I}_{\mathcal{T}}^{g}$ refutation $\mathcal{D}''$. Now prepend $\mathcal{D}''$ with $\mathcal{P}'_0 \vdash \cdots \vdash \mathcal{P}'_{l-1}$ to obtain the desired derivation $\mathcal{D}_{PTME-I}$.

Further, $\mathcal{D}_{PTME-I}$ is trivially stable wrt. minimality, because every underlying theory inference uses, by definition, key sets without duplicates, and this status cannot change along the derivation as all key sets are ground.

A.2 Proofs for Chapter 5 — Linearizing Completion

A.2.1 Section 5.2 — Inference Systems

Lemma 5.2.1 (Ground completeness of $\mathcal{I}_0(\mathcal{T})$). *Let $\mathcal{T}$ be a ground theory (i.e. a theory consisting of ground clauses only) and M be a set of ground literals. If M is $\mathcal{T}$-unsatisfiable then $L \Rightarrow^{*}_{\mathcal{I}_0(\mathcal{T}),M\cup Punit(\mathcal{I}_0(\mathcal{T}))} \mathsf{F}$ for some $L \in (M \cup Punit(\mathcal{I}_0(\mathcal{T})))$.*

Proof. By Proposition 2.5.1.3, equivalence (a)–(e), M is $\mathcal{T}$-unsatisfiable iff $M \cup \mathcal{T}$ is unsatisfiable, where M is considered as a set of unit clauses. We apply induction on the size n of the atom set of $M \cup \mathcal{T}$ (the atom set of a Horn clause set consists of all atoms occurring in clauses in it).

Induction start ($n = 1$): $M \cup \mathcal{T}$ must contain a positive literal A and a purely negative clause $\neg A^1 \vee \ldots \vee \neg A^k$ with $k \geq 1$ occurrences of $\neg A$ (the superscripts denote the distinct occurrences of the same literal $\neg A$).

The literal A and the clause $\neg A^1 \vee \ldots \vee \neg A^k$ may be contained in M or in $\mathcal{T}$, which results in the following cases:

If $k = 1$ then $\neg A \in \mathcal{T}$ or $\neg A \in M$. If $\neg A \in \mathcal{T}$ then the same refutation as in case $k > 1$ below exists. Therefore suppose now $\neg A \in M$. We have $A \in M$ or $A \in \mathcal{T}$. If $A \in M$ then a refutation

$$A \overset{\neg A}{\Longrightarrow}_{A,\neg A\to\mathsf{F}} \mathsf{F}$$

exists. If $A \in \mathcal{T}$ then by definition of $\mathcal{I}_0$, $\neg A \to \mathsf{F} \in \mathcal{I}_0(\mathcal{T})$. In this case a refutation

$$\neg A \overset{\varepsilon}{\Longrightarrow}_{\neg A\to\mathsf{F}} \mathsf{F}$$

exists.

If $k > 1$ then $\neg A^1 \vee \ldots \vee \neg A^k \in \mathcal{T}$ follows. By definition of $\mathcal{I}_0$ $A^1 \cdots A^k \to \mathsf{F} \in \mathcal{I}_0(\mathcal{T})$. We have $A \in M$ or $A \in \mathcal{T}$. However, since a theory is satisfiable by definition, $A \in \mathcal{T}$ is not possible. Hence $A \in M$. Thus a refutation

$$A^1 \overset{A^2\cdots A^k}{\Longrightarrow}_{A^1,\ldots,A^k\to\mathsf{F}} \mathsf{F}$$

from $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$ exists.

Induction step ($n > 1$): $M \cup \mathcal{T}$ must contain at least one positive unit clause. (proof: if not, then every clause contains at least one negative literal. But then an interpretation that assigns *true* to every negative literal is a model for $M \cup \mathcal{T}$. Contradiction). Thus let $A \in M \cup \mathcal{T}$ be a positive unit clause. If $A \in M$ then clearly $A \in M \cup Punit(\mathcal{I}_0(T))$. If $A \in \mathcal{T}$ then $\neg A \to \mathsf{F} \in \mathcal{I}_0(\mathcal{T})$ and thus $A \in Punit(\mathcal{I}_0(T))$. Thus always $A \in M \cup Punit(\mathcal{I}_0(T))$. This fact will be used below.

If $M \cup \mathcal{T}$ contains a clause $\neg A \vee \ldots \vee \neg A$ the same argument as for $n = 1$ applies. This check guarantees that the following processing does not yield the empty clause.

Let $\mathcal{T}'$, resp. M', be obtained from $\mathcal{T}$, resp. M, by deleting every clause of the form $A \vee R$ (R may be empty), and then by replacing every clause of the form $\neg A^1 \vee \ldots \vee \neg A^k \vee R$ (where $\neg A$ does not occur in R, and R cannot be empty) by R. In $\mathcal{T}' \cup M'$ neither A nor $\neg A$ occurs. Thus the atom size of $M' \cup \mathcal{T}'$ is $n-1$. Furthermore $M' \cup \mathcal{T}'$ must be unsatisfiable, because otherwise a model for $M' \cup \mathcal{T}'$ can be extended to a model that assigns *true* to A, which would be in turn a model for $M \cup \mathcal{T}$, and thus M would be $\mathcal{T}$-satisfiable. Hence we can apply the induction hypothesis and assume that a $\mathcal{I}_0(\mathcal{T}')$-refutation of $M' \cup Punit(\mathcal{I}_0(\mathcal{T}'))$ exists. Let D' be that refutation. We show how to transform D' into a $\mathcal{I}_0(\mathcal{T})$-refutation of $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$. For this, every inference step involving a positive literal $B \in M' \cup Punit(\mathcal{I}_0(\mathcal{T}'))$ which is not contained in $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$ has to be eliminated from D' (subcase 1), *and* every inference step involving an inference rule $P \to C \in \mathcal{I}_0(\mathcal{T}')$ which is not contained in $\mathcal{I}_0(\mathcal{T})$ has to be eliminated from D' (subcase 1)[5].

Subcase 1: We conclude from $M' \subseteq M$ that $B \notin Punit(\mathcal{I}_0(\mathcal{T}))$. From $B \in Punit(\mathcal{I}_0(\mathcal{T}'))$ it follows $B \in \mathcal{T}'$. Since $B \notin Punit(\mathcal{I}_0(\mathcal{T}))$ it follows $B \notin \mathcal{T}$. Hence $B \in \mathcal{T}'$ is obtained from $\neg A^1 \vee \ldots \vee \neg A^k \vee B \in \mathcal{T}$ by the replacement operation described above. With $A \in M \cup Punit(\mathcal{I}_0(\mathcal{T}))$ and $A_1, \ldots, A_k \to B \in \mathcal{I}_0(\mathcal{T})$ a $\mathcal{I}_0(\mathcal{T})$-derivation

$$D_B = (A^1 \overset{A^2 \ldots A^k}{\Longrightarrow}_{A_1,\ldots,A_k \to B} B)$$

of B from $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$ exists. Hence a $\mathcal{I}_0(\mathcal{T}')$-derivation B from $M' \cup Punit(\mathcal{I}_0(\mathcal{T}'))$ occurring in D' can be replaced by the $\mathcal{I}_0(\mathcal{T})$-derivation D_B from $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$.

Subcase 2: We have $P \to C \in \mathcal{I}_0(\mathcal{T}')$ but $P \to C \notin \mathcal{I}_0(\mathcal{T})$. Here we distinguish two cases: in the first case, $P \to C$ is of the form $\neg B \to \mathsf{F}$. Thus $B \in \mathcal{T}'$ was obtained from $\neg A^1 \vee \ldots \vee \neg A^k \vee B \in \mathcal{T}$ by the replacement operation above. Hence $A^1, \ldots, A^k \to B \in \mathcal{I}_0(\mathcal{T})$ and we can again find the $\mathcal{I}_0(\mathcal{T})$-derivation D_B of B from $M \cup Punit(\mathcal{I}_0(\mathcal{T}))$. D' is of the form

$$D' = (L_1 \overset{D_1}{\Longrightarrow}_{R_1} \ldots L_n \overset{D_n}{\Longrightarrow}_{R_n} \neg B \overset{\varepsilon}{\Rightarrow}_{\neg B \to \mathsf{F}} \mathsf{F}) \ .$$

[5] To be completely formal, this requires induction on the number of applications of the rule $P \to C$ and the literal B in D'

Now replace the last inference with $\neg B \to \mathsf{F}$ by an inference with $\neg B, B \to \mathsf{F} \in \mathcal{I}_0(\mathcal{T})$ to obtain the refutation

$$L_1 \overset{D_1}{\Longrightarrow}_{R_1} \ldots L_n \overset{D_n}{\Longrightarrow}_{R_n} \neg B \overset{D_B}{\Longrightarrow}_{\neg B, B \to \mathsf{F}} \mathsf{F} \ .$$

In the second case $P \to C$ is not of the form $\neg B \to \mathsf{F}$. By definition of $\mathcal{I}_0$, $P \to C$ then must be of the form $B_1, \ldots, B_l \to C$, where the B_is are positive literals and C is either a positive literal or F. Thus $\mathcal{T}'$ contains a clause $\neg B_1 \vee \ldots \vee \neg B_l \vee C$ or $\neg B_1 \vee \ldots \vee \neg B_l$, respectively. The further argumentation holds for both cases. Let us therefore consider only the first case. $\neg B_1 \vee \ldots \vee \neg B_l \vee C$ is obtained from the clause $\neg A^1 \vee \ldots \neg A^k \vee \neg B_1 \vee \ldots \vee \neg B_l \vee C \in \mathcal{T}$ by the replacement operation above. So $A^1, \ldots, A^k, B_1, \ldots, B_l \to C \in \mathcal{I}_0(\mathcal{T})$. Since $A \in M \cup \mathit{Punit}(\mathcal{I}_0(\mathcal{T}))$ every inference step

$$B_1 \overset{D_2 \cdots D_l}{\Longrightarrow}_{B_1, \ldots, B_l \to C} C$$

in D' with $B_1, \ldots, B_l \to C \in \mathcal{I}_0(\mathcal{T}')$ can be replaced by an inference with $A^1, \ldots, A^k, B_1, \ldots, B_l \to C \in \mathcal{I}_0(\mathcal{T})$ to obtain

$$B_1 \overset{A^1 \cdots A^k D_2 \cdots D_l}{\Longrightarrow}_{A^1, \ldots, A^k, B_1, \ldots, B_l \to C} C \ .$$

A.2.2 Section 5.3 — Orderings and Redundancy

Proposition 5.3.2. *The relation $\succ_{Lin}$ is a monotonic derivation ordering.*

Proof. By definition $D \succ_{Lin} E$ iff $\mathit{compl}(D) \gg_{NMW} \mathit{compl}(E)$. By Proposition 5.3.1, $\gg_{NMW}$ is a simplification ordering and hence by Theorem 2.1.1 well-founded.

Monotonicity is proven by simple structural induction on the derivation, using in the induction step the fact that $\gg_{NMW}$ is monotonic. More formally, let D be a derivation and suppose $D|_{p,i,j} = G$, F is a derivation which agrees with G on top literal and derived literal and $G \succ_{Lin} F$. We have to show $D \succ_{Lin} D[F]_{p,i,j}$.

Induction start: if $p = \lambda$ then $D = D|_p$ is of the form

$$D = (L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots \underbrace{L_i \overset{D_i}{\Longrightarrow} L_{i+1} \ldots L_{j-1} \overset{D_{j-1}}{\Longrightarrow} L_j}_{=G} \overset{D_j}{\Longrightarrow} L_{j+1} \ldots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} L_n) \ ,$$

where $1 \leq i < j \leq n$ (the case $i = j$ is impossible since then G would be a trivial derivation, whose complexity is a bottom element wrt. $\succ_{Lin}$). The complexities of G and D can be written as

$$\begin{aligned} \mathit{compl}(G) &= \{\!\{0\}\!\} \cup R_G, \text{ where} \\ &\quad R_G = \{\!\{\mathit{compl}(D_i)_{w_i}, \ldots, \mathit{compl}(D_{j-1})_{w_{j-1}}\}\!\} \qquad \text{(A.47)} \\ \mathit{compl}(D) &= \{\!\{0, \mathit{compl}(D_1)_{w_1}, \ldots, \mathit{compl}(D_{i-1})_{w_{i-1}}\}\!\} \cup R_G \\ &\quad \cup \{\!\{\mathit{compl}(D_j)_{w_j}, \ldots, \mathit{compl}(D_{n-1})_{w_{n-1}}\}\!\} \ . \qquad \text{(A.48)} \end{aligned}$$

Concerning the derivation F, $compl(F)$ can be written as

$$compl(F) = \{\!|0|\!\} \cup R_F \ . \tag{A.49}$$

By definition, $G \succ_{Lin} F$ iff $compl(G) \gg_{NMW} compl(F)$. But then it follows with (A.47) and (A.49) by monotonicity property (Theorem 2.1.1) of $\gg_{NMW}$ (deleting identical elements does not change the relationship among multisets with weights) $R_G \succ_{NMW} R_F$ (*).

Now consider the derivation $D[F]_{\lambda,i,j}$; its complexity is

$$\begin{aligned} compl(D[F]_{\lambda,i,j}) \quad = \quad & \{\!|0, compl(D_1)_{w_1}, \ldots, compl(D_{i-1})_{w_{i-1}}|\!\} \cup \\ & R_F \cup \{\!|compl(D_j)_{w_j}, \ldots, compl(D_{n-1})_{w_{n-1}}|\!\} \ . \end{aligned}$$

From (*) it follows again by monotonicity property of $\gg_{NMW}$ (replacing a subset by a smaller set makes the whole set smaller) $compl(D) \gg_{NMW} compl(D[F]_{\lambda,i,j})$ and thus also $D \succ_{Lin} D[F]_{\lambda,i,j}$.

Induction step: p is of the form $k.l.r$ and thus D is of the form[6]

$$\begin{aligned} D \quad &= \quad (L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots L_k \overset{D_k}{\Longrightarrow} L_{k+1} \ldots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} L_n), \text{ where} \\ D_k \quad &= \quad D_k^1 \cdots D_k^l \cdots D_k^{n_k} \ . \end{aligned}$$

Hence $compl(D)$ is of the form

$$\begin{aligned} compl(D) \quad = \quad & \{\!|0, compl(D_1), \ldots, compl(D_{k-1}), \\ & compl(D_k^1) \cup \cdots \cup compl(D_k^l) \cup \cdots \cup compl(D_k^{n_k}), \\ & compl(D_{k+1}), \ldots, compl(D_n)|\!\} \ , \end{aligned}$$

with $D|_{k.l} = D_k^l$. In order to build $D[F]_{k.l.r,i,j}$ we have to replace D_k^l by $D_k^l[F]_r$. This means for the complexity of the new derivation

$$\begin{aligned} compl(D[F]_{k.l.r,i,j}) \quad = \quad & \{\!|0, compl(D_1), \ldots, compl(D_{k-1}), \\ & compl(D_k^1) \cup \cdots \cup compl(D_k^l[F]_r) \cup \cdots \cup \\ & \qquad compl(D_k^{n_k}), \\ & compl(D_{k+1}), \ldots, compl(D_n)|\!\} \end{aligned}$$

By the induction hypothesis $D_k^l \succ_{Lin} D_k^l[F]_r$.
Thus by definition $compl(D_k^l) \gg_{NMW} compl(D_k^l[F]_r)$. Thus by monotonicity of $\gg_{NMW}$ $compl(D) \succ_{NMW} compl(D[F]_{k.l.r,i,j})$ and hence $D \succ_{Lin} D[F]_{k.l.r,i,j}$.

The following lemma is needed in the proof of Proposition 5.3.3:

Lemma A.2.2 (Instantiation Lemma for Linear Derivations). *Suppose a linear $\mathcal{I}$-derivation*

$$D \ = \ (L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} L_n)$$

[6] Weights omitted.

as given, and let γ be a ground substitution for L_1, M and L_n. Then a linear ground $\mathcal{I}^g$-derivation

$$L_1\gamma \overset{D_1\gamma}{\Longrightarrow} L_2\gamma_2 \ldots L_{n-1}\gamma_{n-1} \overset{D_{n-1}\gamma}{\Longrightarrow} L_n\gamma$$

exists, where each γ_i is some ground substitution for L_i.

Thus, derivations may be ground instantiated; the additional (ground) substitutions γ_i come in due to extra variables in the conclusion of inference rules, and these variables have to be grounded in order to match the ground literal in the premise of the subsequent inference step.

Proof. Induction on the length n of the derivation.

Induction start ($n = 1$) trivial, simply take $L_1\gamma$ as the desired derivation.

Induction step ($n - 1 \to n$): assume $n > 1$ and assume the result holds for derivations of length $n - 1$. The concluding inference step of the given derivation D is more precisely

$$L_{n-1} \overset{D_{n-1}}{\Longrightarrow}_{P_{n-1} \to C_{n-1}, \sigma_{n-1}} L_n \ ,$$

where $\{\!|L_{n-1}|\!\} \cup D_{n-1} = P_{n-1}\sigma_{n-1}$ (*) and $C_{n-1}\sigma_{n-1} = L_n$.

Next consider the prefix $L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots L_{n-2} \overset{D_{n-2}}{\Longrightarrow} L_{n-1}$ of D. Let γ' be a ground substitution for $L_{n-1}\gamma$. $\gamma\gamma'$ a ground substitution for L_1, M and L_{n-1} (because γ alone is a ground substitution, as given). Hence, by the induction hypothesis exists a $\mathcal{I}^g$-derivation

$$D' \quad = \quad L_1\gamma\gamma' \overset{D_1\gamma\gamma'}{\Longrightarrow} L_2\gamma_2 \ldots L_{n-2}\gamma_{n-2} \overset{D_{n-2}\gamma\gamma'}{\Longrightarrow} L_{n-1}\gamma\gamma' \ .$$

Since $L_1\gamma$ being ground it follows $L_1\gamma = L_1\gamma\gamma'$. Similarly, since $M\gamma$ is ground and all literals in the sequence D_k are taken from M it follows that $D_k\gamma$ is also ground. But then $D_k\gamma = D_k\gamma\gamma'$. By these considerations D' is a $\mathcal{I}^g$-derivation of $L_{n-1}\gamma\gamma'$ from $M\gamma$ with top literal $L_1\gamma$.

With $P_{n-1}\sigma_{n-1} = \{\!|L_{n-1}|\!\} \cup D_{n-1}$, as given, it follows that D' can be extended by means of the substitution $\gamma\gamma'$ with one inference step to a derivation

$$D'' = (D' \cdot (L_{n-1}\gamma\gamma' \overset{D_{n-1}\gamma\gamma'}{\Longrightarrow}_{(P_{n-1}\sigma_{n-1} \to C_{n-1}\sigma_{n-1})\gamma\gamma'} C_{n-1}\sigma_{n-1}\gamma\gamma')) \ .$$

By the same arguments as for D_k above it holds $D_{n-1}\gamma = D_{n-1}\gamma\gamma'$; furthermore, with $L_n\gamma$ being ground and with $L_n = C_{n-1}\sigma$ it follows that $L_n\gamma = C_{n-1}\sigma\gamma = C_{n-1}\sigma\gamma\gamma'$. Thus D'' is a derivation of $L_n\gamma$ from $M\gamma$.

We have to show that D'' is c $\mathcal{I}^g$-derivation, i.e. that the used inference rule in the last step is contained in $\mathcal{I}^g$. Since $D_{n-1}\gamma$ is ground (as concluded above) and $L_{n-1}\gamma\gamma'$ is ground (by definition of γ') and $C_{n-1}\sigma\gamma\gamma'$ is ground (because $C_{n-1}\sigma\gamma\gamma' = L_n\gamma$) it follows with (*) that $(P \to C)\gamma\gamma'$ is also ground. Hence from $P \to C \in \mathcal{I}$ it follows $(P \to C)\gamma\gamma' \in \mathcal{I}^g$, and so D'' is a $\mathcal{I}^g$-derivation. Setting $\gamma_{n-1} = \gamma\gamma'$ shows that D'' is the desired derivation.

Proposition 5.3.3 (Sufficient $\succ_{Lin}$-Redundancy Criterion). *Let $\mathcal{I}$ be an inference system and $P \to_w C$ be an inference rule. Suppose that for every $L \in P$ there is a linear $\mathcal{I} \setminus \{P \to_w C\}$-derivation from P*

$$(L = L_1) \overset{D_1}{\Longrightarrow} P_1 \to_{w_1} C_1 L_2 \overset{D_2}{\Longrightarrow} P_2 \to_{w_2} C_2 L_3 \cdots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} P_{n-1} \to_{w_{n-1}} C_{n-1} (L_n = C)$$

with $n \geq 1$ and such that for $i = 1, \ldots, n-1$ it holds $\langle P \setminus \{\!\{L\}\!\}, w\rangle \not\succ_{MW} \langle D_i, w_i\rangle$ In this comparison the sequence D_i of literals is to be read as a multiset. Then $P \to_w C$ is $\succ_{Lin}$-redundant in $\mathcal{I}$ for derivations.

Proof. By contradiction. Hence suppose the assumptions of the proposition hold and $P \to C$ is not $\succ_{Lin}$-d-redundant in $\mathcal{I}$. Then for some inference system $\mathcal{J} \supseteq \mathcal{I}$, some ground literal L_1, some ground literal set M and some ground literal L_n a derivation $D = (L_1 \Rightarrow^*_{(\mathcal{J} \cup \{P \to C\})^g, M} L_n)$ exists but a derivation $D^- = (L_1 \Rightarrow^*_{(\mathcal{J} \setminus \{P \to C\})^g, M} L_n)$ with $D^- \prec_{Lin} D$ (*) does not exist.

We are given that at least one ground instance of $P \to_w C$ is used in some inference step $G = D|_{p,i,i+1}$ in D. W.l.o.g assume that no further ground instance of $P \to_w C$ is used in G (by tracing into D and its subderivations such a "bottommost" derivation can always be located).

We will show that a $(\mathcal{J} \setminus \{P \to_w C\})^g$-derivation F with $F \prec_{Lin} G$ which can replace G in D, i.e. we build $D^- = D[F]_{p,i,i+1}$. Since $\succ_{Lin}$ is monotonic (Proposition 5.3.2) it then holds $D^- \prec_{Lin} D$. Since F does not use a ground instance of $P \to_w C$ the number of usages of ground instances of $P \to_w C$ in D^- is 1 less than in D. Hence repeated application of this procedure terminates and thus yields a $(\mathcal{J} \setminus \{P \to_w C\})^g$-derivation. But then by transitivity of $\succ_{Lin}$ we obtain a contradiction to (*).

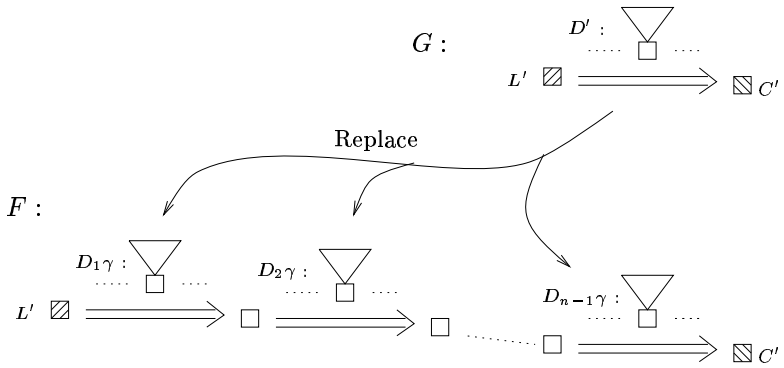


Figure A.1. Illustration of proof of Proposition 5.3.3.

Now for the construction of F (cf. Figure A.1): the inference step G can be written as

$$G = (L' \overset{D'}{\Longrightarrow}_{(L,N \to C)\gamma} C') \ ,$$

where $P = \{\!|L|\!\} \cup N$, γ is a ground substitution, $L' = L\gamma$, $C' = C\gamma$ and D' is a sequence of derivations of $N\gamma$; the complexity is

$$compl(G) = \{\!|0, compl(D')|\!\} \ .$$

From the assumption of the proposition we learn that a $\mathcal{I} \setminus \{P \to_w C\}$-derivation

$$F' = ((L = L_1) \overset{D_1}{\Longrightarrow} \ldots \overset{D_{n-1}}{\Longrightarrow} (L_n = C))$$

exists such that $\langle D_i, w_i \rangle \prec\!\!\prec_{MW} \langle N, w \rangle$ (for $1 \leq i \leq n$), where w_i is the weight of the inference rule used in the i-th inference step.

Applying the instantiation lemma to γ and F' (Lemma A.2.2) yields a $(\mathcal{I} \setminus \{P \to C\})^g$-derivation

$$F'' = ((L' = L_1\gamma) \overset{D_1\gamma}{\Longrightarrow} \ldots \overset{D_{n-1}\gamma}{\Longrightarrow} (L_n\gamma = C')) \ .$$

Note that with $\mathcal{I} \subseteq \mathcal{J}$ it holds that F'' is a $(\mathcal{J} \setminus \{P \to C\})^g$-derivation as well.

From $\langle D_i, w_i \rangle \prec\!\!\prec_{MW} \langle N, w \rangle$ it follows

$$\langle D_i\gamma, w_i \rangle \prec\!\!\prec_{MW} \langle N\gamma, w \rangle \ . \tag{A.50}$$

Since D' is a sequence of derivations of $N\gamma$ we can find for every sequence $D_i\gamma \subseteq N\gamma$ a sequence $D'_i \subseteq D'$ such that D'_i is a sequence of derivations of $D_i\gamma$. But then we can replace every side derivation $D_i\gamma$ in F'' by D'_i, yielding still a $(\mathcal{J} \setminus \{P \to_w C\})^g$-derivation (recall that we assume that no ground instance of $P \to_w C$ is used in G)

$$F = (L' \overset{D'_1}{\Longrightarrow} \ldots \overset{D'_n}{\Longrightarrow} C')$$

with complexity

$$compl(F) = \{\!|0, \langle compl(D'_1), w_1 \rangle, \ldots, \langle compl(D'_{n-1}), w_{n-1} \rangle|\!\} \ .$$

Now consider (A.50) again: either it holds $D_i \subset N$ which implies by construction $D'_i \subset D'$, which in turn implies $\langle D'_i, w_i \rangle \prec\!\!\prec_{NMW} \langle D', w \rangle$; or else it holds $D_i = N$ and $w_i < w$, which implies by construction $D'_i = D'$ and hence also $\langle D'_i, w_i \rangle \prec\!\!\prec_{NMW} \langle D', w \rangle$. It follows in any case $F \prec \mathit{LinG}$ which was to be shown.

A.2.3 Section 5.4 — Transformation Systems

Lemma 5.4.1. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Suppose $P \to C$ is $\succ$-c-redundant in some $\mathcal{I}$, and suppose that $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{J}$. Then $P \to C$ is also $\succ$-c-redundant in $\mathcal{J}$.*

Proof. $\mathcal{J}$ is obtained from $\mathcal{I}$ by deletion of a $\succ$-c-redundant rule (case 1), or by adding a new rule (case 2). We know that $P \to C$ is not of the form $\neg A \to \mathsf{F}$.

Case 1: We have $\mathcal{J} = \mathcal{I} \setminus \{P' \to C'\}$ for some rule $P' \to C' \in \mathcal{I}$ to be deleted. By definition of c-redundancy we have to show that whenever

$$D = (L_1 \Rightarrow^*_{((\mathcal{I}\setminus\{P'\to C'\})\cup\{P\to C\})^g, M} L_n) \tag{A.51}$$

and $P \to C$ is used in D, then a

$$D' = (L_1 \Rightarrow^*_{((\mathcal{I}\setminus\{P'\to C'\})\setminus\{P\to C\})^g, M} L_n) \tag{A.52}$$

exists such that $D' \prec D$.

From $\mathcal{I} \setminus \{P' \to C'\} \subset \mathcal{I}$ and (A.51) it follows $D = (L_1 \Rightarrow^*_{(\mathcal{I}\cup\{P\to C\})^g, M} L_n)$. We are given that $P \to C$ is $\succ$-c-redundant in $\mathcal{I}$. Hence a derivation $D' = (L_1 \Rightarrow^*_{(\mathcal{I}\setminus\{P\to C\})^g, M} L_n)$ exists such that $D' \prec D$. Now, if no ground instance of $P' \to C'$ is used in D' then D' is also a $((\mathcal{I} \setminus \{P' \to C'\}) \setminus \{P \to C\})^g$-derivation, which proves (A.52). Otherwise, application of Lemma A.2.4 below to D' also proves (A.52).

Case 2: We have $\mathcal{J} = \mathcal{I} \cup \{P' \to C'\}$ for some new rule $P' \to C'$. By definition of c-redundancy we have to show that whenever

$$D = (L_1 \Rightarrow^*_{((\mathcal{I}\cup\{P'\to C'\})\cup\{P\to C\})^g, M} L_n) \tag{A.53}$$

and $P \to C$ is used in that derivation then a derivation

$$D' = (L_1 \Rightarrow^*_{((\mathcal{I}\cup\{P'\to C'\})\setminus\{P\to C\})^g, M} L_n) \tag{A.54}$$

exists such that $D' \prec D$. Setting $\mathcal{J} = \mathcal{I} \cup \{P' \to C'\}$ in the definition of c-redundancy (Def. 5.3.3) renders this case trivial.

The following lemma is needed in the proof of Lemma 5.4.3.

Lemma A.2.4. *If*

1. *$D = (L \Rightarrow^*_{(\mathcal{I}\setminus\{P\to C\})^g} L')$, where $P \to C$ is a $\succ$-c-redundant inference rule in $\mathcal{I}$, and*
2. *if in D some ground instance of $P' \to C'$ is used, and*
3. *if $\mathcal{I} \vdash \mathcal{I} \setminus \{P' \to C'\} =: \mathcal{J}$ by deletion*

then a derivation

$$D' = (L \Rightarrow^*_{(\mathcal{J}\setminus\{P\to C\})^g} L')$$

exists with $D' \prec D$.

Proof. Informally: if $P' \to C'$ is deleted in $\mathcal{J}$ then it must be c-redundant in $\mathcal{I}$. By redundancy, the use of $P' \to C'$ in any derivation D' can be simulated by the remaining rules of $\mathcal{I}$. However, that simulating derivation, say D'' might use $P \to C$, which should not be used according to the lemma. On the other hand, $P \to C$ itself is given as c-redundant in $\mathcal{I}$ and hence can be simulated by the remaining rules. However, that derivation, say D''' possibly contains usages of $P' \to C'$ again. Continuing this process will *not* fall into a loop due to strictly reduced complexity ($D''' \prec D'' \prec D'$) in every new derivation and well-foundedness of $\succ$.

Now the formal proof: if $P \to C \notin \mathcal{I}$ then the lemma holds trivially by the definition of $\succ$-c-redundancy.

Otherwise we apply well-founded induction on derivation orderings.

With $\mathcal{I} \setminus \{P' \to C'\} \subseteq \mathcal{I}$ it follows $D = (L \Rightarrow^*_{\mathcal{I}^g} L')$. With $P' \to C'$ being deleted from $\mathcal{I}$, $P' \to C'$ must have been $\succ$-c-redundant in $\mathcal{I}$. We are given that a ground instance of $P' \to C'$ is used in D. Hence by definition of $\succ$-c-redundancy a derivation

$$D' = (L \Rightarrow^*_{(\mathcal{I}\setminus\{P'\to C'\})^g} L') \tag{A.55}$$

exists with $D' \prec D$. Now we distinguish two cases:

Case 1: No ground instance of $P \to C$ is used in D'. But then by the existence of D' we find with (A.55) that $D' = (L \Rightarrow^*_{((\mathcal{I}\setminus\{P\to C\})\setminus\{P'\to C'\})^g} L')$, which proves the lemma.

Case 2: A ground instance of $P \to C$ is used in D'. First note that from $\mathcal{I} \setminus \{P' \to C'\} \subseteq \mathcal{I}$ by (A.55) it follows $D' = (L \Rightarrow^*_{\mathcal{I}^g} L')$. We are given that $P \to C$ is $\succ$-c-redundant in $\mathcal{I}$. By definition of $\succ$-c-redundancy we conclude that a derivation $D'' = (L \Rightarrow^*_{(\mathcal{I}\setminus\{P\to C\})^g} L')$ exists such that $D'' \prec D' (\prec D)$. Now apply the induction hypothesis to D'' and conclude the result from transitivity of $\succ$.

Lemma 5.4.3. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be a $\mathcal{S}$-deduction. If for some k, $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$ then $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_\infty$.*

Proof. In order to prove the lemma suppose that

$$D = (L \Rightarrow^*_{(\mathcal{I}_\infty\cup\{P\to C\})^g,M} L') \tag{A.56}$$

which uses a ground instance of $P \to C$. We have to show that a derivation

$$D' = (L \Rightarrow^*_{(\mathcal{I}_\infty \setminus \{P \to C\})^g, M} L') \tag{A.57}$$

exists with $D' \prec D$. As a consequence of Proposition 5.4.1 some $\mathcal{I}_m$ $(m \geq 0)$ contains (at least) all those inference rules from $\mathcal{I}_\infty$, the ground instances of which are used in D. That is

$$D = (L \Rightarrow^*_{(\mathcal{I}_m \cup \{P \to C\})^g, M} L') \ . \tag{A.58}$$

We distinguish two cases:

Case 1: $m > k$. We are given that $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$. Applying Lemma 5.4.1 $m-k$ times we conclude that $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_m$ as well. Hence by definition of $\succ$-c-redundancy we find with (A.58) a derivation

$$D_m = (L \Rightarrow^*_{(\mathcal{I}_m \setminus \{P \to C\})^g, M} L') \tag{A.59}$$

with $D_m \prec D$.

Case 2: $m \leq k$. By Lemma 5.4.2 every inference rule in $\mathcal{I}_m$ is also contained in $\mathcal{I}_k$. Thus with (A.58) D is also $(\mathcal{I}_k \cup \{P \to C\})^g$-derivation. We are given that $P \to C$ is $\succ$-c-redundant in $\mathcal{I}_k$. Hence a derivation $D_k = (L \Rightarrow^*_{(\mathcal{I}_k \setminus \{P \to C\})^g, M} L'$ with $D_k \prec D)$.

By taking $n = max(m, k)$ in both cases a derivation

$$D_n = (L \Rightarrow^*_{(\mathcal{I}_n \setminus \{P \to C\})^g, M} L')$$

exists with $D_n \prec D$.

We *claim* that for some $l \geq n$ a derivation $D_l = (L \Rightarrow^*_{(\mathcal{I}_l \setminus \{P \to C\})^g, M} L')$ exists with $D_l \preceq D_n$, and the inference rules whose ground instances are used in D_l are never deleted afterwards, i.e. if $(P' \to C')\gamma \in (\mathcal{I}_l \setminus \{P \to C\})^g$ is a ground instance used in D_l then $P' \to C' \in \mathcal{I}_j \setminus \{P \to C\}$ for every $j \geq l$.

This claim then proves (A.57) and thus the lemma by transitivity of $\succ$ and by the following line of reasoning: suppose $P' \to C' \in \mathcal{I}_j \setminus \{P \to C\}$ for every $j \geq l$ then

$$\begin{aligned} P' \to C' \in \bigcap_{j \geq l} (\mathcal{I}_j \setminus \{P \to C\}) &= (\bigcap_{j \geq l} \mathcal{I}_j) \setminus \{P \to C\} \\ &\subseteq (\bigcup_{l \geq 0} \bigcap_{j \geq l} \mathcal{I}_j) \setminus \{P \to C\} \\ &= \mathcal{I}_\infty \setminus \{P \to C\} \ . \end{aligned}$$

Hence if $(P' \to C')\gamma \in (\mathcal{I}_l \setminus \{P \to C\})^g$ is a used ground instance in D_l then also $(P' \to C')\gamma \in (\mathcal{I}_\infty \setminus \{P \to C\})^g$. Thus (A.57) follows.

It remains to prove the above claim. Starting with D_n we construct a sequence

$$D_n, D_{n+1}, D_{n+2}, \ldots$$

of derivations, where for $i > n$ we define

$$D_{i+1} = \begin{cases} D_i' & \text{if } \mathcal{I}_{i+1} \text{ is obtained from } \mathcal{I}_i \text{ by deletion of an inference rule } P'' \to C'', \text{ ground instances of which are used in } D_i, \text{ where } D_i' = (L \Rightarrow^*_{(\mathcal{I}_{i+1} \setminus \{P \to C\})^g, M} L') \text{ and } D_i' \prec D_i. \text{ Such a derivation } D_i' \text{ exists because with } P \to C \text{ being } \succ\text{-c-redundant in } \mathcal{I}_n \text{ it follows by Lemma 5.4.1 that } P \to C \text{ is } \succ\text{-c-redundant in } \mathcal{I}_n, \mathcal{I}_{n+1}, \ldots, \mathcal{I}_i. \text{ Now apply Lemma A.2.4.} \\ D_i & \text{else.} \end{cases}$$

Note that in this sequence, deleting an inference rule $P'' \to C''$, ground instances of which are used, results in a strictly smaller derivation $D_i' \prec D_i$. Since $\succ$ is a derivation ordering and hence well-founded, we arrive at the chain $D_n, D_{n+1}, \ldots, D_l, D_l, D_l, \ldots$. Thus deletion of used inferences will not be continued infinitely. Stated positively, every inference system $\mathcal{I}_l, \mathcal{I}_{l+1}, \mathcal{I}_{l+2}$ contains every inference rule, ground instances of which are used in D_l. Thus the claim above is proved, which concludes the proof of the lemma.

Lemma 5.4.5. *Let $\mathcal{S}$ be a transformation system with derivation ordering $\succ$. Let $\mathcal{I}_0 \vdash \mathcal{I}_1 \vdash \cdots$ be an $\mathcal{S}$-deduction. If there is a derivation*

$$D = (L \Rightarrow^*_{\mathcal{I}_k^g, M \cup Punit(\mathcal{I}_k^g)} L')$$

then there is also a derivation

$$D' = (L \Rightarrow^*_{\mathcal{I}_\infty^g, M \cup Punit(\mathcal{I}_\infty^g)} L')$$

with $D' \preceq D$.

Proof. The proof is similar to that of Lemma 5.4.3 above. As a consequence of Lemma 5.4.4 every input literal from $M \cup Punit(\mathcal{I}_k^g)$ is also contained in $M \cup Punit(\mathcal{I}_\infty^g)$. Thus from the assumption of the lemma it follows $D = (L \Rightarrow^*_{\mathcal{I}_k^g, M \cup Punit(\mathcal{I}_\infty^g)} L')$. For ease of notation define $N = M \cup Punit(\mathcal{I}_\infty^g)$.

We claim that for some $l \geq k$ a $D_l = (L \Rightarrow^*_{\mathcal{I}_l^g, N} L')$ with $D_l \preceq D$ exists, and the inference rules whose ground instances are used in D_l are never deleted afterwards, i.e. if $(P' \to C')\gamma \in \mathcal{I}_l^g$ is a ground instance used in D_l then $P' \to C' \in \mathcal{I}_j$ for every $j \geq l$.

This claim then proves the lemma by the following line of reasoning: suppose $P' \to C' \in \mathcal{I}_j$ for every $j \geq l$ then

$$P' \to C' \in \bigcap_{j \geq l} \mathcal{I}_j \subseteq \bigcup_{l \geq 0} \bigcap_{j \geq l} \mathcal{I}_j = \mathcal{I}_\infty .$$

Hence if $(P' \to C')\gamma \in \mathcal{I}_l^g$ is a used ground instance in D_l then also $(P' \to C')\gamma \in \mathcal{I}_\infty^g$. Thus the lemma follows.

It remains to prove the above claim. Starting with $D_k = D$ we construct a sequence $D_k, D_{k+1}, D_{k+2}, \ldots$ of derivations, where for $i > k$ we define

$$D_{i+1} = \begin{cases} D_i' & \text{if } \mathcal{I}_{i+1} \text{ is obtained from } \mathcal{I}_i \text{ by deletion of an inference rule } P'' \to C'', \text{ ground instances of which are used in } D_i, \text{ where } D_i' = (L \Rightarrow^*_{\mathcal{I}_{i+1}^g, M} L') \text{ and } D_i' \prec D_i. \text{ Such a derivation exists by definition of the deletion operation.} \\ D_i & \text{else.} \end{cases}$$

The rest of the proof is literally the same as in the proof of the previous Lemma 5.4.3, and hence is omitted.

A.2.4 Section 5.5 — Complexity-Reducing Transformation Systems

Proposition 5.5.1. *The transformation system Lin is order-normalizing wrt. LinG.*

Proof. Let $\mathcal{I}$ be an inference system and D be a ground $\mathcal{I}^g$ refutation of M which is not linear, i.e. $D \notin LinG$. We have to show that a $(\mathcal{I}')^g$-derivation $D' \prec_{Lin} D$ of M exists, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I}'$ is obtained from $\mathcal{I}$ by application of some mandatory transformation rule.

Since D is given as a non-linear derivation at least one side derivation is not a sequence of trivial derivations. Thus D can be written as

$$D = (L_1 \overset{D_1}{\Longrightarrow} L_2 \ldots \overset{D_{i-1}}{\Longrightarrow} \underbrace{L_i \overset{D_i}{\Longrightarrow} L_{i+1}}_{=:D''} \overset{D_{i+1}}{\Longrightarrow} \ldots L_{n-1} \overset{D_{n-1}}{\Longrightarrow} L_n) \ ,$$

where D_i is that critical side derivation and $D'' = D|_{\lambda,i,i+1}$. More precisely, the subderivation D'' is of the form

$$D'' = (L_i \overset{D^{K\gamma} D^{M\gamma}}{\Longrightarrow}_{(L,K,M \to L')\gamma} L_{i+1}) \ ,$$

where $L_i = L\gamma$, $D_i = D^{K\gamma} D^{M\gamma}$, $D^{K\gamma}$ is a non-trivial derivation of $K\gamma$, $D^{M\gamma}$ is a sequence of derivations of $M\gamma$ and $(L, K, M \to L')\gamma \in \mathcal{I}^g$ is a ground instance of $L, K, M \to L' \in \mathcal{I}$ (cf. Figure A.2). We will show that a $(\mathcal{I}')^g$-derivation $D''' \prec_{Lin} D''$ exists, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I}'$ is obtained from $\mathcal{I}$ by application of a mandatory transformation rule from Lin. Then we define $D' = D[D''']_{\lambda,i,i+1}$. By monotonicity of $\succ_{Lin}$ (Proposition 5.3.2) then it follows $D' \prec_{Lin} D$, which was to be shown.

In general, the derivation $D^{K\gamma}$ is of the form

$$D^{K\gamma} = (\underbrace{J_1 \overset{E_1}{\Longrightarrow} J_2 \ldots J_{m-1} \overset{E_{m-1}}{\Longrightarrow} J_m}_{=:D^{J\delta}} \overset{D^{N\delta}}{\Longrightarrow}_{(J,N \to J')\delta} J'\delta) \ ,$$

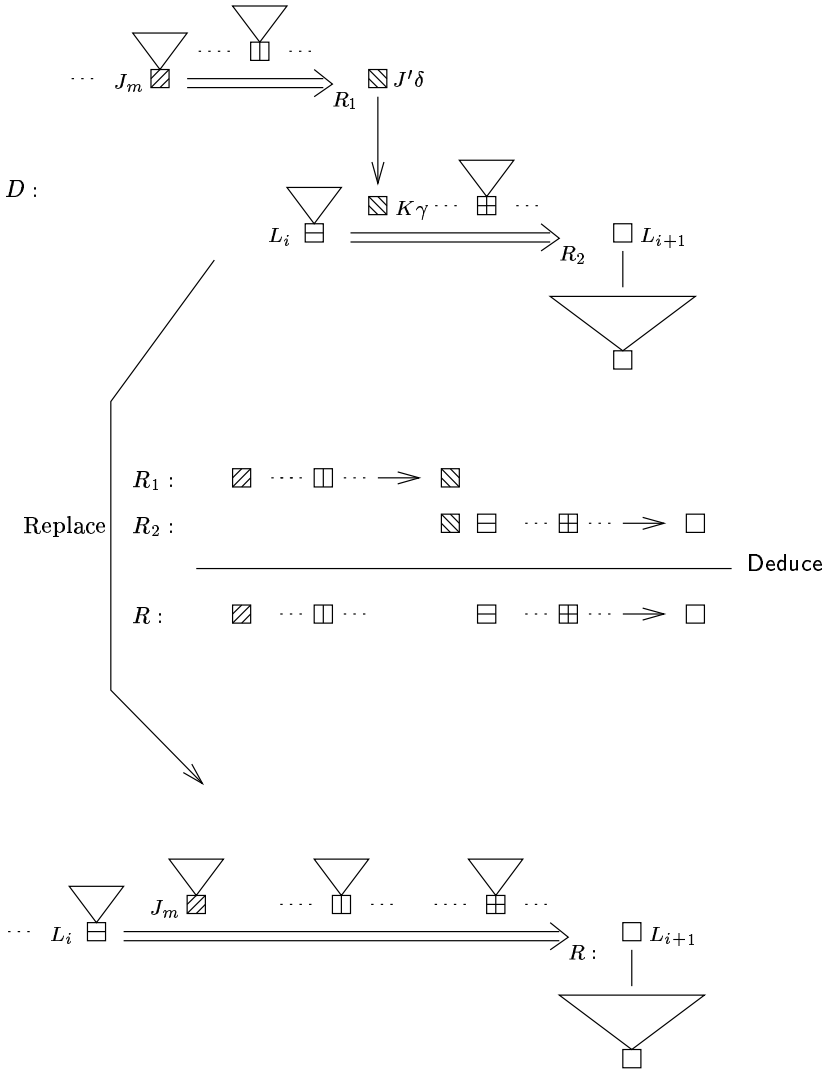


Figure A.2. Illustration of proof of Proposition 5.5.1 (ground case).

where $m \geq 1$, $J_m = J\delta$, $D^{N\delta}$ is a (possibly empty) sequence of derivations of $N\delta$, $J'\delta = K\gamma$ and $(J, N \to J')\delta \in \mathcal{I}^g$ is a ground instance of $J, N \to J' \in \mathcal{I}$. Without loss of generality suppose that $J, N \to J'$ is variable disjoint from $L, K, M \to L'$. Consequently we may assume that the domains of δ and γ are disjoint, too. Hence $(J, N \to J')\delta\gamma = (J, N \to J')\delta$ and $(L, K, M \to L')\delta\gamma = (L, K, M \to L')\gamma$. Together with with $J'\delta = K\gamma$ it follows that $J'\delta\gamma = K\delta\gamma$. In other words, $\delta\gamma$ is a unifier. Hence a MGU σ exists and a substitution ϕ such that $\delta\gamma = \sigma\phi$. By the existence of this MGU, the mandatory Deduce transformation rule can be applied to $L, K, M \to L'$ and $J, N \to J'$ by unifying J' and K with σ. The result is the inference rule $R = (L, J, N, M \to L')\sigma$.

Now either a variant of R is already contained in $\mathcal{I}$, and in this case define $\mathcal{I}' = \mathcal{I}$. Otherwise define $\mathcal{I}' = \mathcal{I} \cup \{R\}$. Since $R\phi$ is ground and $D^{J\delta}$, $D^{N\delta}$ and $D^{M\gamma}$ are appropriate ground $\mathcal{I}^g$-derivations (and hence also $(\mathcal{I}')^g$-derivations) the $(\mathcal{I}')^g$-derivation

$$D''' \;=\; (L_i \xRightarrow{D^{J\delta} D^{N\delta} D^{M\gamma}}_{R\phi} L_{i+1})$$

exists. Note that D''' and D'' coincide in top and derived literals. Hence the replacement as suggested above can be done. It remains to show $D''' \prec_{Lin} D''$. By definition of $\succ_{Lin}$ this is the same as to show $compl(D''') \prec_{NMW} compl(D'')$; for this proof the weights of the involved inference rules can be neglected, since decreasingness follows alone from properties of multiset orderings:

$$\begin{aligned}
compl(D'') &= \{\!\{0, compl(\{\!\{D^{K\gamma}\}\!\} \cup D^{M\gamma})\}\!\} \\
&= \{\!\{0, compl(D^{K\gamma}) \cup compl(D^{M\gamma})\}\!\} \\
&= \{\!\{0, \{\!\{0, compl(E_1), \ldots, compl(E_{m-1}), compl(D^{N\gamma})\}\!\} \\
&\quad \cup compl(D^{M\gamma})\}\!\} \\
&= \{\!\{0, compl(D^{J\delta}) \cup \{\!\{compl(D^{N\gamma})\}\!\} \cup compl(D^{M\gamma})\}\!\} \\
\succ_{NMW} \quad & \{\!\{0, compl(D^{J\delta}) \cup compl(D^{N\gamma}) \cup compl(D^{M\gamma})\}\!\} \\
&= \{\!\{0, compl(D^{J\delta} D^{N\gamma} D^{M\gamma})\}\!\} \\
&= compl(D''')
\end{aligned}$$

The transition $\succ_{NMW}$ is justified by property of nested multiset ordering (replacing a multiset by true subsets is decreasing). Since this remained to be shown the proof is done.

Proposition 5.5.3. *The transformation system Lin is Punit-normalizing wrt. LinG.*

Proof. Let $\mathcal{I}$ be an inference system and let

$$D = (T_1 \xRightarrow{D_1} T_2 \ldots T_n \xRightarrow{D_n} T_{n+1} \xRightarrow{D_{n+1}} T_{n+2})$$

be the given non-trivial linear ground $\mathcal{I}^g$ derivation from $M \cup Punit(\mathcal{I}^g)$ such that $Used(D) \cap Punit(\mathcal{I}^g) \neq \emptyset$. We have to show that a $(\mathcal{I}')^g$-derivation $D' \prec_{Lin} D$ of $M \cup Punit((\mathcal{I}')^g)$ exists, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I}'$ is obtained from $\mathcal{I}$ by application of some mandatory transformation rule, and the top literal of D' is T_1 or a literal from $M \cup Punit((\mathcal{I}')^g)$. Since D is given as non-trivial it holds $n \geq 0$.

Let $L\gamma \in Used(D) \cap Punit(\mathcal{I}^g)$ where $L\gamma$ is a ground instance of a literal $L \in Punit(\mathcal{I})$. Two cases apply: (1) $L\gamma$ occurs in some D_i ($i \in \{1 \ldots n+1\}$) or (2) $L\gamma = T_1$.

Case 1: (cf. Figure A.3) D contains an inference step of the form

$$D'' = D|_{\lambda,i,i+1} = (T_i \overset{L\gamma E_1 \cdots E_{m_i}}{\Longrightarrow} (T_i', L', E_1', \ldots, E_{m_i}' \rightarrow C)\gamma' C\gamma') \qquad (*)$$

where $(T_i', L', E_1', \ldots, E_{m_i}' \rightarrow C)\gamma' \in \mathcal{I}^g$ is a ground instance of the rule $T_i', L', E_1', \ldots, E_{m_i}' \rightarrow C \in \mathcal{I}$, $T_i'\gamma' = T_i$, $L'\gamma' = L\gamma$, $E_j'\gamma' = E_j$ (for $j = 1 \ldots m_i$) and $C\gamma' = T_{i+1}$ or $C\gamma' = \mathsf{F}$. The further reasoning holds for both cases. We will show how to deduce a new inference rule by which the application of $L\gamma$ in D'' can be omitted.

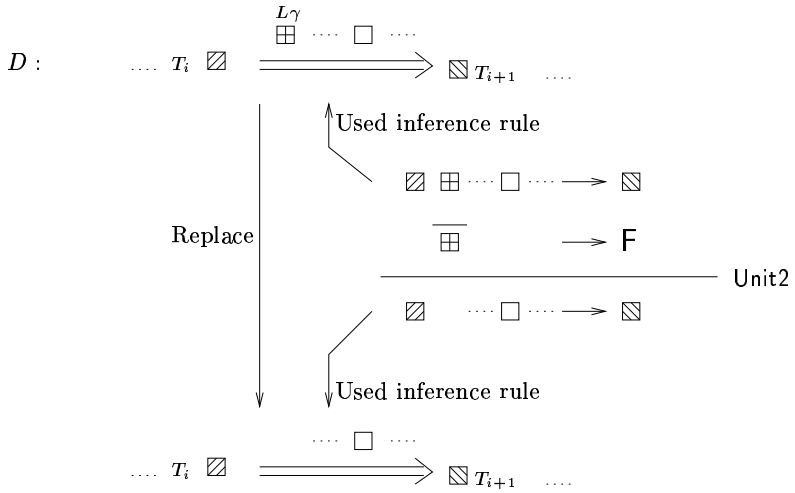


Figure A.3. A case in the proof of Proposition 5.5.3 (ground case).

We will show that there is a $(\mathcal{I}')^g$-derivation $D''' \prec_{Lin} D''$, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I}'$ is obtained from $\mathcal{I}$ by application of a mandatory transformation rule from Lin. Then we define $D' = D[D''']_{\lambda,i,i+1}$. By monotonicity of $\succ_{Lin}$ (Proposition 5.3.2) then it follows $D' \prec_{Lin} D$, which was to be shown.

Since $L \in Punit(\mathcal{I})$ by definition $\overline{L} \rightarrow \mathsf{F} \in \mathcal{I}$. Let $\overline{L''} \rightarrow \mathsf{F}$ be a new variant, variable disjoint from the premise $\{T_i', L', E_1', \ldots, E_{m_i}'\}$ of the applied infer-

ence rule. Since $L\gamma = L'\gamma'$ and L'' is a variant of L a ground substitution γ'' exists with $L''\gamma'' = L'\gamma'$. Since L'' is a new variant, γ' can be supposed not to act upon the variables of L''. Hence $L''\gamma'\gamma'' = L'\gamma'\gamma''$. Since $\gamma'\gamma''$ is a unifier for L'' and L' a most general unifier σ for L'' and L' exists and substitution δ such that $\sigma\delta = \gamma'\gamma''$ (**). By the existence of this MGU, the mandatory Unit2 transformation rule can be applied to $T_i', L', E_1', \ldots, E_{m_i}' \to C \in \mathcal{I}$ and $\overline{L''} \to \mathsf{F} \in \mathcal{I}$. The result is the new inference rule

$$R \;=\; (T_i', E_1', \ldots, E_{m_i}' \to C)\sigma \;\;.$$

Now either a variant of R already is contained in $\mathcal{I}$, and in this case define $\mathcal{I}' = \mathcal{I}$. Otherwise define $\mathcal{I}' = \mathcal{I} \cup \{R\}$.

Let δ' be the restriction of γ' to the domain $Var(\{\!|T_i', E_1', \ldots, E_{m_i}', C|\!\}) \setminus Var(L')$. Together with (**) it follows

$$\{\!|T_i', E_1', \ldots, E_{m_i}', C|\!\}\sigma\delta\delta' = \{\!|T_i', E_1', \ldots, E_{m_i}', C|\!\}\gamma'\gamma'' = \{\!|T_i', E_1', \ldots, E_{m_i}', C|\!\}\gamma' \;\;.$$

By these equalities we can build the desired derivation

$$D''' = (T_i \xRightarrow{E_1 \cdots E_{m_i}} {}_{R\delta\delta'} C\gamma')$$

with rule $R\delta\delta' \in \mathcal{I}^g$.

Note that D''' and D'' coincide in top and derived literals. Hence the replacement as suggested above can be done. It remains to show $D''' \prec_{Lin} D''$. By definition of $\succ_{Lin}$ this is the same as to show $compl(D''') \prec_{NMW} compl(D'')$; for this proof the weights of the involved inference rules can be neglected, since decreasingness follows alone from properties of multiset orderings:

$$\begin{array}{rcl} compl(D'') & = & \{\!|0, compl(L\gamma E_1 \cdots E_{m_i})|\!\} \\ & = & \{\!|0, \{\!|0|\!\} \cup compl(E_1 \cdots E_{m_i})|\!\} \\ & \succ_{NMW} & \{\!|0, compl(E_1 \cdots E_{m_i})|\!\} \\ & = & compl(D''') \end{array}$$

Case 2: ($L\gamma = T_1$). We distinguish the cases (2.1) $D_1 \neq \varepsilon$ (i.e. D_1 is not the empty sequence of derivations) and (2.2) $D_1 = \varepsilon$.

Case 2.1: D begins with the inference step

$$D'' = (L\gamma \xRightarrow{K E_1 \cdots E_{m_i}} {}_{(K', L', E_1', \ldots, E_{m_i}' \to C)\gamma'} C\gamma') \qquad (*)$$

where $(L', K', E_1', \ldots, E_{m_i}')\gamma' \to C\gamma' \in \mathcal{I}^g$ is a ground instance of the rule $L', K', E_1', \ldots, E_{m_i}' \to C \in \mathcal{I}$, $L'\gamma' = L\gamma$, $K'\gamma' = K$, $E_j'\gamma' = E_j$ (for $j = 1 \ldots m_i$) and $C\gamma' = T_2$ or $C\gamma' = \mathsf{F}$. The further reasoning holds for both cases. It is in close analogy to the case $L\gamma \in D_i$.

We will show that there is a $(\mathcal{I}')^g$-derivation $D''' \prec_{Lin} D''$, where $\mathcal{I}' = \mathcal{I}$ or $\mathcal{I}'$ is obtained from $\mathcal{I}$ by application of a mandatory transformation rule from Lin. Then we define $D' = D[D''']_{\lambda,1,2}$. By monotonicity of $\succ_{Lin}$ (Proposition 5.3.2) then it follows $D' \prec_{Lin} D$, which was to be shown.

Since $L \in Punit(\mathcal{I})$ by definition $\overline{L} \to \mathsf{F} \in \mathcal{I}$. Let $\overline{L''} \to \mathsf{F}$ be a new variant. As in case 1, the mandatory Unit2 transformation rule can be applied to $\overline{L''} \to \mathsf{F}$ and $L', K', E'_1, \ldots, E'_{m_i} \to C \in \mathcal{I}$, yielding

$$R = (K', E'_1, \ldots, E'_{m_i} \to C)\sigma \ ,$$

where σ is the MGU for L'' and L' used in that deduction step. Now either a variant of R already is contained in $\mathcal{I}$, and in this case define $\mathcal{I}' = \mathcal{I}$. Otherwise define $\mathcal{I}' = \mathcal{I} \cup \{R\}$.

As in case 1 there are ground substitutions $\delta\delta'$, γ' and γ'' such that

$$\begin{aligned} \{\!|K', E'_1, \ldots, E'_{m_i}, C|\!\}\sigma\delta\delta' &= \{\!|K', E'_1, \ldots, E'_{m_i}, C|\!\}\gamma'\gamma'' \\ &= \{\!|K', E'_1, \ldots, E'_{m_i}, C|\!\}\gamma' \end{aligned}$$

By these equalities we can build the desired derivation

$$D''' = (K \xRightarrow{E_1 \cdots E_{m_i}}_{R\delta\delta'} C\gamma')$$

with rule $R\delta\delta' \in \mathcal{I}^g$.

Note that D''' and D'' coincide in top and derived literals. Hence the replacement as suggested above can be done. It remains to show $D''' \prec_{Lin} D''$. This proof is literally the same as the corresponding proof in case 1 above, except that $L\gamma$ is to be replaced by K. This completes the proof for the case $D_1 \neq \varepsilon$.

Case 2.2: $(D_1 = \varepsilon)$. It holds that $n > 0$. Proof: Suppose, to the contrary that $n = 0$. Then the refutation consists of a single inference step with the rule $T_1 \to \mathsf{F} \in \mathcal{I}^g$. On the other hand from $T_1 \in Punit(\mathcal{I}^g)$ it follows $\overline{T_1} \to \mathsf{F} \in \mathcal{I}^g$. But then $\mathcal{I}$ would be unsatisfiable, since no interpretation can satisfy both, T_1 and $\overline{T_1}$. By this we would arrive at a contradiction to the satisfiability of the underlying theory $\mathcal{T}$ where $\mathcal{I}$ stems from.

The given derivation D can be written more specifically as ($n \geq 0$, cf. Figure A.4) :

$$D = (L\gamma \xRightarrow{\varepsilon}_{(L' \to T'_2)\gamma'} T_2 \ldots T_n \xRightarrow{D_n} T_{n+1} \xRightarrow{D_{n+1}} T_{n+2}) \qquad (*)$$

where $T_2 \neq \mathsf{F}$, $(L' \to T'_2)\gamma' \in \mathcal{I}^g$ is a ground instance of the rule $L' \to T_2 \in \mathcal{I}$, $L'\gamma' = L\gamma$ and $T'_2\gamma' = T_2$.

Since $L \in Punit(\mathcal{I})$ by definition $\overline{L} \to \mathsf{F} \in \mathcal{I}$. Let $\overline{L''} \to \mathsf{F}$ be a new variant, variable disjoint from the premise $\{\!|L'|\!\}$ of the applied inference rule. Since $L\gamma = L'\gamma'$ and L'' is a variant of L there is a ground substitution γ'' with $L''\gamma'' = L'\gamma'$. Since L'' is a new variant, γ' can be supposed not to act

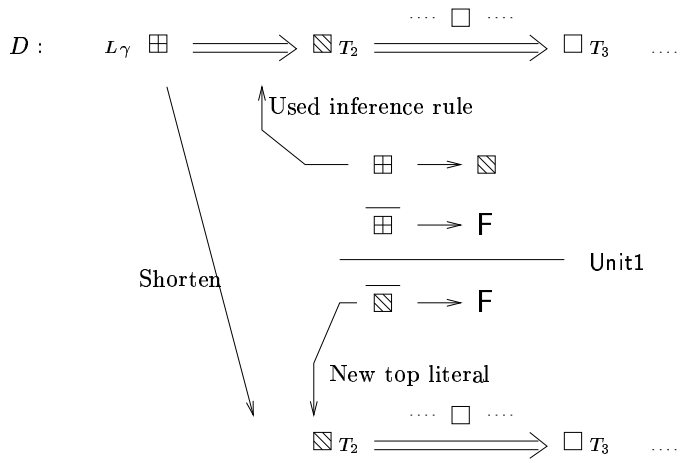


Figure A.4. A case in the proof of Lemma 5.5.3 (ground case)

upon the variables of L''. Hence $L''\gamma'\gamma'' = L'\gamma'\gamma''$. Since $\gamma'\gamma''$ is a unifier for L'' and L' there is a most general unifier σ for L'' and L' and substitution δ such that $\sigma\delta = \gamma'\gamma''$ (**). By the existence of this MGU, the mandatory Unit1 transformation rule can be applied to $L' \to T_2' \in \mathcal{I}$ and $\overline{L''} \to \mathsf{F} \in \mathcal{I}$. The result is the new inference rule

$$R \;=\; T_2'\sigma' \to \mathsf{F} \;\;.$$

Now either a variant of R is already contained in $\mathcal{I}$, and in this case define $\mathcal{I}' = \mathcal{I}$. Otherwise define $\mathcal{I}' = \mathcal{I} \cup \{R\}$.

Now let δ' be the restriction of γ' to the domain $Var(T_2') \setminus Var(L')$. Together with (**) it follows

$$T_2'\sigma\delta\delta' = T_2'\gamma'\gamma''(= T_2'\gamma' = T_2) \quad (***)$$

Thus with $R \in \mathcal{I}'$ it follows $T_2 \in Punit((\mathcal{I}')^g)$. By this fact we can cut off the first inference step in D, yielding

$$D' = D|_{2,n+2} = (T_2 \ldots T_n \overset{D_n}{\Longrightarrow} T_{n+1} \overset{D_{n+1}}{\Longrightarrow} T_{n+2}) \;\;,$$

which is a derivation from $M \cup Punit((\mathcal{I}')^g)$ as desired.

It remains to prove $D' \prec_{Lin} D$. By definition of $\succ_{Lin}$ this is the same as to show $compl(D') \prec_{NMW} compl(D)$; for this proof the weights of the involved inference rules can be neglected, since decreasingness follows alone from properties of multiset orderings:

$$\begin{array}{rcl} compl(D) & = & \{\!\{0, compl(\varepsilon), compl(D_2), \ldots, compl(D_{n+1})\}\!\} \\ & = & \{\!\{0, \emptyset, compl(D_2), \ldots, compl(D_{n+1})\}\!\} \\ & \succ_{NMW} & \{\!\{0, compl(D_2), \ldots, compl(D_{n+1})\}\!\} \\ & = & compl(D') \end{array}$$

By concluding this final case the lemma is proven.

Proposition 5.5.4. *Let $\mathcal{S}$ be a Punit-normalizing transformation system wrt. $\mathcal{N}$, and let $\mathcal{I}$ be a completed inference system wrt. $\mathcal{S}$. Whenever there is a ground derivation $D = (L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit(\mathcal{I}^g)} L')$ with $D \in \mathcal{N}$ then there is also a ground derivation $D' = (K \Rightarrow^*_{\mathcal{I}^g, M} L')$ with $D' \preceq D$, $D' \in \mathcal{N}$ and some $K \in M \cup \{L\}$.*

Proof. By well-founded induction on derivations wrt. the well-founded derivation ordering associated to $\mathcal{S}$. If $Used(D) \cap Punit(\mathcal{I}^g) = \emptyset$ then no literal from $Punit(\mathcal{I}^g)$ is used in D then D is also a derivation from M alone. Hence we take $D' = D$.

Otherwise, by definition of *Punit*-normalizing transformation systems there is a derivation $D'' = (L \Rightarrow^*_{(\mathcal{I}'')^g, M \cup Punit((\mathcal{I}'')^g)} L')$ with $D'' \prec D$, where (1) $\mathcal{I}'' = \mathcal{I}$ or (2) $\mathcal{I} \vdash_{\mathcal{S}} \mathcal{I} \cup \{P \to C\} = \mathcal{I}''$ by some mandatory transformation rule from $\mathcal{S}$.

In case 1 D'' is also a $\mathcal{I}^g$-derivation of $M \cup Punit(\mathcal{I}^g)$; now consider case 2: since $\mathcal{I}$ is completed it holds by Definition 5.4.3 $P \to C \in \mathcal{I}$ or (2.2) $P \to C$ is $\succ$-c-redundant in $\mathcal{I}$. In case 2.1 $\mathcal{I}'' = \mathcal{I}$ and hence D'' is also a $\mathcal{I}^g$-derivation of $M \cup Punit(\mathcal{I}^g)$. In case 2.2 $P \to C$ is not of the form $\neg A \to \mathsf{F}$ (such rules are by definition never c-redundant). Hence $Punit(\mathcal{I}'') = Punit(\mathcal{I})$. If no ground instance of $P \to C$ is used in D'' then D'' is also a $\mathcal{I}^g$-derivation of $M \cup Punit(\mathcal{I}^g)$, otherwise by definition of c-redundancy there is a ground derivation $D''' = (L \Rightarrow^*_{\mathcal{I}^g, M \cup Punit((\mathcal{I})^g)} L')$ with $D''' \prec D''$ (note that this is a $\mathcal{I}^g$-derivation). From $D''' \prec D'' \prec D$ it follows by downward closure of $\mathcal{N}$ also $D'' \in \mathcal{N}$ and $D''' \in \mathcal{N}$. Thus, in any case there is a $\mathcal{I}^g$-derivation of $M \cup Punit(\mathcal{I}^g)$ which is contained in $\mathcal{N}$ and which is strictly smaller wrt. $\succ$ than the given derivation.

Now apply the induction hypothesis to that derivation.

A.2.5 Section 5.6 — Completeness

Lemma 5.6.1 (Top Literal Lemma). *Let $\mathcal{I}$ be a completed inference system wrt. the transformation system Lin. Suppose there is a linear ground derivation $D = (L_1 \Rightarrow^*_{\mathcal{I}^g, M} L_n)$ with $L_1 \in M$. Let $T \in M$ such that $T \in Used(D)$. Then there is a linear ground derivation $D' = (T \Rightarrow^*_{\mathcal{I}^g, M} L_n)$.*

Proof. Let the given derivation be

$$D = (L_1 \overset{M_1}{\Longrightarrow} L_2 \ldots \overset{M_{k-1}}{\Longrightarrow} L_k \overset{T\ M_k}{\Longrightarrow}_{R_1 \gamma_1} L_{k+1} \overset{M_{k+1}}{\Longrightarrow} \ldots L_{n-1} \overset{M_{n-1}}{\Longrightarrow} L_n) \tag{A.60}$$

where $n > 1$ (otherwise the claim is trivial), $k \in \{1 \dots n\}$ and $R_1\gamma_1$ is a ground instance of $R_1 = L'_k, T', M'_k \to L'_{k+1} \in \mathcal{I}$, $L_k = L'_k\gamma_1$, $T = T'\gamma_1$, $M_k = M'_k\gamma_1$ and $L_{k+1} = L'_{k+1}\gamma_1$.

We do induction on the *top distance* k of the inference step using T.

Induction start ($k = 1$): The first inference step is $L_1 \overset{T\ M_1}{\Longrightarrow}_{R_1\gamma_1} L_2$. By swapping T and L_1 it can be replaced by the inference step $T \overset{L_1\ M_1}{\Longrightarrow}_{R_1\gamma_1} L_2$ which yields the desired linear derivation.

Induction step ($k - 1 \to k$): See Figure A.5 on Page 259 for illustration. Suppose $k > 1$, and as the induction hypothesis assume the claim to hold for derivations with top distance strictly smaller than k. D then can be written as

$$\overbrace{\underbrace{L_1 \overset{M_1}{\Longrightarrow} L_2 \dots \overset{M_{k-2}}{\Longrightarrow} L_{k-1}}_{D_1} \overset{M_{k-1}}{\Longrightarrow}_{R_2\gamma_2} L_k \overset{T\ M_k}{\Longrightarrow}_{R_1\gamma_1} \underbrace{L_{k+1} \overset{M_{k+1}}{\Longrightarrow} \dots \overset{M_{n-1}}{\Longrightarrow} L_n}_{D_3}}^{D_2}, \tag{A.61}$$

where $n \geq 3$ and $R_2\gamma_2$ is a ground instance of $R_2 = L'_{k-1}, M'_{k-1} \to L'_k \in \mathcal{I}$, $L_{k-1} = L'_{k-1}\gamma_2$, $M_{k-1} = M'_{k-1}\gamma_2$ and $L_k = L'_k\gamma_2$.

Without loss of generality suppose that R_1 is variable disjoint from R_2. Consequently, we may assume that the domains of γ_1 and γ_2 are disjoint, too. Hence $R_1\gamma_1\gamma_2 = R_1\gamma_1$ and $R_2\gamma_1\gamma_2 = R_2\gamma_2$. Together with $L'_k\gamma_1 = L_k = L''_k\gamma_2$ it follows that $L'_k\gamma_1\gamma_2 = L_k = L''_k\gamma_1\gamma_2$. In other words, $\gamma_1\gamma_2$ is a unifier. Hence there is a MGU σ and a substitution δ such that $\gamma_1\gamma_2 = \sigma\delta$. By the existence of this MGU, the Deduce transformation rule can be applied to R_1 and R_2 by unifying L'_k and L''_k with σ. The result is the inference rule $R = (T', L'_{k-1}, M'_{k-1}, M'_k \to L'_{k+1})\sigma$. Since Deduce is a mandatory transformation rule and $\mathcal{I}$ is completed (1) $R \in \mathcal{I}$ or (2) R is $\succ_{Lin}$-c-redundant in $\mathcal{I}$. In case (1) the derivation D_2 in (A.61) can be replaced by the one step derivation

$$L_{k-1} \overset{T\ M_{k-1}\ M_k}{\Longrightarrow}_{(T', L'_{k-1}, M'_{k-1}, M'_k \to L'_{k+1})\sigma\delta} L_{k+1}$$

using $R\delta$. Since δ is a ground substitution $R\delta \in \mathcal{I}^g$ and the replacement results in a $\mathcal{I}^g$ derivation with same structure. Furthermore, its top distance is $k - 1$. Hence we can apply the induction hypothesis to obtain the desired derivation.

In case (2) things are more complicated. First consider the $(\mathcal{I} \cup \{R\})^g$-derivation

$$T \overset{L_{k-1}\ M_{k-1}\ M_k}{\Longrightarrow}_{R\delta} L_{k+1}\ .$$

Since R is $\succ_{Lin}$-c-redundant in $\mathcal{I}$ there is a linear $\mathcal{I}^g$-derivation

$$D_4 = (T \Rightarrow^*_{\mathcal{I}^g, M_{k-1} \cup M_k \cup \{L_{k-1}\}} L_{k+1})\ .$$

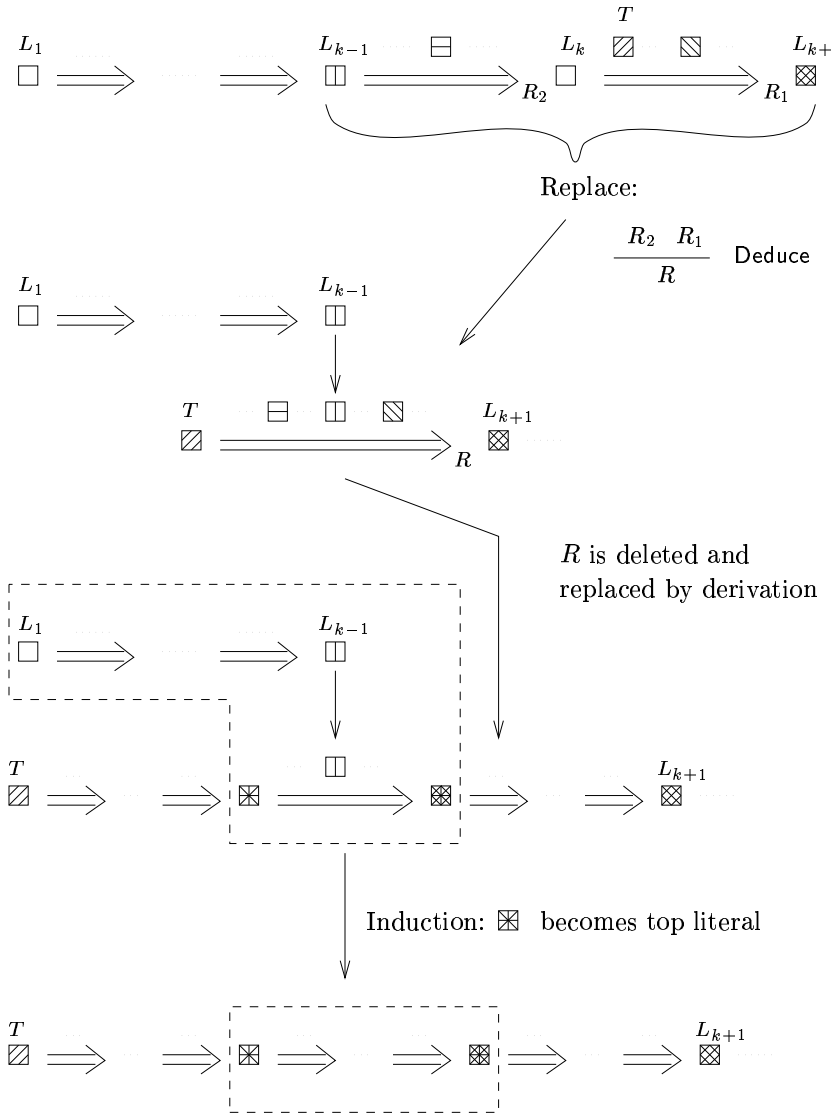


Figure A.5. Illustration of proof of Lemma 5.6.1 (ground case).

Note that $M_{k-1}, M_k \subseteq M$. Now concatenate to $D_5 = D_4 \cdot D_3$ and obtain a linear derivation of the form

$$D_5 = (T \Rightarrow^*_{\mathcal{I}^g, M \cup \{L_{k-1}\}} L_n)$$

Next the applications of the literal $\{L_{k-1}\}$ in D_5 have to be eliminated. This can be done one at another by the procedure described below. Once this is done we obtain the desired linear derivation of the form $T \Rightarrow^*_{\mathcal{I}^g, M} L_n$.

If L_{k-1} is not used in D_5 we are done. Otherwise let

$$D_6 = D_5|_{\lambda,l,l+1} = (K_l \overset{L_{k-1}\ N_l}{\Longrightarrow}_{R_l,\gamma_l} K_{l+1})$$

be such an inference step using L_{k-1}. Swapping K_l and L_{k-1} implies the existence of the $\mathcal{I}^g$-derivation

$$D_7 = (L_{k-1} \overset{K_l\ N_l}{\Longrightarrow}_{R_l,\gamma_l} K_{l+1})$$

Now concatenate $D_1 \cdot D_7$ to obtain the $\mathcal{I}^g$-derivation

$$D_8 = (L_1 \overset{M_1}{\Longrightarrow} L_2 \ldots \overset{M_{k-2}}{\Longrightarrow} L_{k-1} \overset{K_l\ N_l}{\Longrightarrow}_{R_l,\gamma_l} K_{l+1}) \ .$$

The top distance of K_l in D_8 is $k-1$ hence we can apply the induction hypothesis to D_8 and obtain a linear $\mathcal{I}^g$-derivation

$$D_9 = (K_l \Rightarrow^*_{\mathcal{I}^g, M} K_{l+1}) \ .$$

Next build $D_10 = D_5[D_9]_{\lambda,l,l+1}$, i.e. replace the inference step using L_{k-1} by a derivation not using L_{k-1}. Hence the number of applications of the literal L_{k-1} has decreased by 1, which guarantees the termination of the just described procedure when applied repeatedly in order to eliminate all applications of L_{k-1}. Hence the desired derivation exists.

Lifting Lemma for Background Refutations. In order to prove a lifting lemma for background refutations we need one preliminary result:

Lemma A.2.8. *Let α, β be substitutions and M be a literal set. Then*

$$\alpha(\beta|\, Var(M\alpha)) = \alpha\beta\ [Var(M)]$$

Proof. Let x be a variable. It suffices to show that both substitutions yield the same result when applied to x. We distinguish two disjoint cases.

1. $x \notin Var(M)$. Trivial, since both substitutions yield x.
2. $x \in Var(M)$. By definition of restriction of substitution it suffices to show

   $$x\alpha(\beta|\, Var(M\alpha)) = x\alpha\beta \ .$$

   From $x \in Var(M)$ it follows $Var(x\alpha) \subseteq Var(M\alpha)$ (*). Now we compute

   $$x\alpha\beta = x\alpha(\beta|\, Var(x\alpha)) \overset{(*)}{=} x\alpha(\beta|\, Var(M\alpha))$$

   which was to be shown.

Lemma A.2.9 (Lifting Lemma for Background Refutations). *Let M be a literal set, $L_1 \in M$ be literal, and γ be a ground substitution for M. For every ground background refutation $L_1\gamma \Rightarrow^*_{\mathcal{I}^g, M_\gamma} \square$ there is a first-order background refutation $L_1 \Rightarrow^*_{\mathcal{I},M,\sigma} \square$ such that $\sigma \leq \gamma \quad [Var(M)]$.*

Proof. Let the given refutation be

$$D \;=\; (L_1\gamma \overset{M_1\gamma}{\Longrightarrow}_{(L_1', M_1' \to L_2')\gamma'} L_2 \overset{M_2\gamma}{\Longrightarrow} L_3 \ldots L_n \overset{M_n\gamma}{\Longrightarrow} \square) \;.$$

The proof is by induction on the length n of the derivation.

Induction start ($n = 1$): Define $\gamma'' = \gamma\gamma'$. We can safely assume that all inference rules instances of which are used in D are all "new" variants. As a consequence we get that the domains of γ and γ' are disjoint. Hence it holds that

$$(\{\!|L_1|\!\} \cup M_1)^a\gamma'' = (\{\!|L_1'|\!\} \cup M_1')\gamma'',$$

where $(\{\!|L_1|\!\} \cup M_1)^a$ is suitable amplification of $\{\!|L_1|\!\} \cup M_1$. Thus there is also a multiset MGU σ_1 and a substitution δ_1 such that $\sigma_1\delta_1 = \gamma''$ (*). Using this MGU we can build the first-order refutation

$$D' \;=\; (L_1 \overset{M_1}{\Longrightarrow}_{L_1', M_1' \to L_2', \sigma_1} \square)$$

with amplification $(\{\!|L_1|\!\} \cup M_1)^a$ and answer substitution σ_1. It remains to show that σ_1 satisfies the claimed property. From (*) and $\gamma'' = \gamma\gamma'$ it follows $\sigma_1\delta_1 = \gamma'' = \gamma\gamma' = \gamma \; [Dom(\gamma)]$. The last identity holds because γ is a ground substitution. Since γ acts on every variable in M and in L_1 it holds $Dom(\gamma) \supseteq Var(M)$, and thus in particular $\sigma_1\delta_1 = \gamma \; [Var(M)]$. This is the same as $\sigma_1 \leq \gamma \quad [Var(M)]$ which was to be shown.

Induction step ($n-1 \to n$): Suppose $n > 1$ and as the induction hypothesis assume the result to hold for derivations with length $< n$. Consider the first inference step in D. Define σ_1 and δ_1 in the same way as in the induction start, i.e. $\sigma_1\delta_1 = \gamma\gamma'$ (*) and $\sigma_1\delta_1 = \gamma \; [Var(M)]$. Further it holds $L_2'\gamma' = L_2'\gamma\gamma'$ due to the new variant used in the first step. Thus, with (*) we have $L_2'\gamma' = L_2\sigma_1\delta_1$. Further, with the identity $L_2 = L_2'\gamma'$ and with $\sigma_1\delta_1 = \gamma \; [Var(M)]$ we can delete the first inference step from D and arrive at the refutation

$$L_2'\sigma_1\delta_1 \overset{M_2\sigma_1\delta_1}{\Longrightarrow} L_3 \ldots L_n \overset{M_n\sigma_1\delta_1}{\Longrightarrow} \square \;.$$

By the induction hypothesis we can lift this (still linear) refutation to a (linear) refutation

$$L_2'\sigma_1 \overset{M_2\sigma_1}{\Longrightarrow}_{R_2,\sigma_2} L_3 \ldots L_n \overset{M_n\sigma_1\sigma_2\cdots\sigma_{n-1}}{\Longrightarrow}_{R_n,\sigma_n} \square$$

with answer substitution

$$\sigma_2\sigma_3\cdots\sigma_n \leq \delta_1 \quad [Var(M\sigma_1)]$$

i.e. for some δ, $\sigma_2\sigma_3\cdots\sigma_n\delta = \delta_1\ [Var(M\sigma_1)]$ (**). Since σ_1 is an appropriate unifier (as in the induction start) we can prepend this derivation with the first-order inference step $L_1 \overset{M_1}{\Longrightarrow}_{L_1', M_1' \to L_2', \sigma_1} L_2'\sigma_1$ to obtain

$$L_1 \overset{M_1}{\Longrightarrow}_{L_1', M_1' \to L_2', \sigma_1} L_2'\sigma_1 \overset{M_2}{\Longrightarrow}_{R_2, \sigma_2} L_3 \ldots L_n \overset{M\sigma_1\sigma_2\cdots\sigma_{n-1}}{\Longrightarrow}_{R_n, \sigma_n} \square \ ,$$

which is a linear first-order refutation of L_1 as desired with answer substitution $\sigma_1\sigma_2\sigma_3\cdots\sigma_n$. It remains to show that the answer substitution satisfies the claimed property. Hence we compute

$$\begin{aligned}
\sigma_1\sigma_2\sigma_3\cdots\sigma_n\delta|\,Var(M) &= \sigma_1(\sigma_2\sigma_3\cdots\sigma_n\delta|\,Var(M\sigma_1))|\,Var(M) \\
&\qquad \text{(by Lemma A.2.8)} \\
&= \sigma_1(\delta_1|\,Var(M\sigma_1))|\,Var(M) \qquad \text{(by (**))} \\
&= \sigma_1\delta_1|\,Var(M) \qquad \text{(by Lemma A.2.8)} \\
&= \gamma|\,Var(M) \qquad \text{(see induction start).}
\end{aligned}$$

But then by definition $\sigma_1 \leq \gamma \quad [Var(M)]$ which was to be shown.

B. What is Where?

Notation

	All chapters
$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$	Connectives
$\forall, \exists$	Quantifiers
Σ	Signature
$\mathcal{PF}(\Sigma)$	The set of all formulas of signature Σ
$\mathcal{CPF}(\Sigma)$	The set of all closed formulas (i.e. sentences) of signature Σ
$\mathcal{T}$	Theory
$\forall F$, $\exists F$	Universal and existential closure of formula F
$a, b, \ldots$	Constant symbol
$f, g, \ldots$	Function symbol
$s, t, u, \ldots$	Term
$x, y, z, \ldots$	(First-order) variable
$A, B, C, \ldots$	Atom
$C, D, \ldots$	Clause
$P, Q, \ldots$	Predicate symbol or literal
$K, L, \ldots$	Literal
$\alpha, \beta, \delta,$	
$\phi, \gamma, \sigma, \tau, \mu$	Substitutions
ϵ	Empty substitution
$\succ$	(Partial Strict) Ordering
$\succeq$	$\succ \cup =$
$\gg_{\succ}$	Extension of $\succ$ to multisets

Chapter 2

$\mathcal{F}, \mathcal{P}, \mathcal{X}$	Set of function symbols, predicate symbols, variables.
$Term(\mathcal{F})$	The set of (ground) terms with function symbols from $\mathcal{F}$
$Term(\mathcal{F}, \mathcal{X})$	The set of terms with function symbols from $\mathcal{F}$ and variables from $\mathcal{X}$
F, G	Formula
S	Sentence
M	Set of sentences
$\mathcal{S}$	Structure
$\mathcal{U}$	Universe, as component of structure $\mathcal{S}$ also denoted by $\|\mathcal{S}\|$
$v_{\mathcal{X}}$	Assignment for variables $\mathcal{X}$
$\mathcal{I}$	Interpretation, its universe denoted by $\|\mathcal{I}\|$
$\langle \mathcal{I}, v_{\mathcal{X}} \rangle$	Interpretation plus assignment (extended interpretation)
$\mathcal{I}_{v_{\mathcal{X}}}(X)$	Value of X in $\langle \mathcal{I}, v_{\mathcal{X}} \rangle$
$\mathcal{U}_{\Sigma}$	Herbrand universe of Σ
$\mathcal{HB}(\Sigma)$	Herbrand base of Σ
$\mathcal{H}_{\Sigma}$	Herbrand interpretation
$Th(I)$	The theory of interpretation $\mathcal{I}$
$Cons(X)$	The theory of axiom set X

Chapter 3 + 4 + 5

c	computation rule
$\mathcal{D}$	Foreground calculi derivation
D, E	Linearizing completion derivation
$\mathcal{I}$,$\mathcal{J}$	Inference system
$\mathcal{K}$	Key set (of theory inference)
M,N	Clause set, literal set
p, q	Branch, read as a multiset
$[p]$, $[q]$	Branch, read as a sequence
$[L_1 \cdots L_k]$	Branch, labeled with literals $L_1, \ldots, L_k$
$\mathcal{P}$, $\mathcal{Q}$	Branch set
P	Premise part of inference rule
C	Conclusion of inference rule
$\mathcal{S}$	Transformation system
$\mathcal{R}$	Residue (in partial theory inference)
ϵ	Empty $\mathcal{I}$-derivation (Chapter 5)
$\succ_{MW}$	Ordering on multisets with weight
$\succ_{NMW}$	Ordering on nested multisets with weight

Note that some symbols are overloaded, e.g. C can be an atom or a clause. Confusion is less likely in these cases as the meaning can always be detected from the context. Where appropriate, sub- and superscripts will be used to enlarge the repository of symbols.

Table of Foreground Calculi

The following foreground calculi are treated in this text:

Abbreviation	*Name*, Inference Rules	*Definition*
ME	Model elimination ME-Ext, Red	Def. 3.2.3, page 53
CC	Connection calculus CC-Ext, Red	Def. 3.2.3, page 53
TTCC	Total theory connection calculus TTCC-Ext	Def. 4.2.3, page 85
TTCC-Link	Total theory connection calculus with link condition TTCC-Link-Ext	Def. 4.2.4, page 87
TME-Sem	Theory Model elimination — semantical version TME-Sem-Ext (TTME-Sem-Ext, PTME-Sem-Ext)	Def. 4.3.2, page 93
TTME-MSR	Total Theory Model elimination — MSR version TTME-MSR-Ext	Def. 4.4.3, page 106
PTME-I	Partial theory model elimination, inference system version PTME-I-Ext	Def. 4.5.4, page 120
PRTME-I	Partial restart theory model elimination, inference system version PDTME-I-Ext, Restart	Def. 4.6.3, page 131

Inferences and Derivation Relations

Notation	*Description*	*Definition*
$\mathcal{P} \vdash_{[p],\sigma,E} \mathcal{P}'$	Inference in Foreground Calculus: derive branch set $\mathcal{P}'$ from $\mathcal{P}$, with selected branch $[p]$, substitution σ and extending clauses E.	Def. 3.2.4, page 55
Extended Notations:		
$([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},\sigma,E} (\mathcal{Q}', \mathcal{Q})\sigma$	TTCC-Link-Ext inference with key set $\mathcal{K}$ and minimal $\mathcal{T}$-refuter σ	Def. 4.2.4, page 87
$([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},\langle\mathcal{R},\sigma\rangle,E} (\mathcal{Q}', \mathcal{Q})\sigma$	TME-Sem-Ext inference with key set $\mathcal{K}$ and minimal $\mathcal{T}$-residue $\langle\mathcal{R},\sigma\rangle$	Def. 4.3.2, page 93
$([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},\sigma,E} (\mathcal{Q}', \mathcal{Q})\sigma$	TTME-MSR-Ext inference with key set $\mathcal{K}$ and minimal most general $\mathcal{T}$-refuter $\sigma \in MSR_{\mathcal{T}}(\mathcal{K})$	Def. 4.4.3, page 106
$([p],\ \mathcal{Q}) \vdash_{[p],\mathcal{K},P\to C,\langle\mathcal{R},\sigma\rangle,E} (\mathcal{Q}', \mathcal{Q})\sigma$	PTME-$\mathcal{I}$-Ext inference with key set $\mathcal{K}$, new variant $P \to C$ of an inference rule from $\mathcal{I}$ and $\mathcal{T}$-residue $\langle\mathcal{R},\sigma\rangle$	Def 4.5.4, page 120
$([p_1],\ \mathcal{Q}_1) \vdash \cdots \vdash \mathcal{Q}_n$	Derivation in foreground calculus, possibly subscribed by selected branch etc.	Def. 3.2.4, page 55
$\mathcal{K} \Rightarrow_{P\to C,\,\sigma} \mathcal{R}\sigma$	Minimal first-order theory inference with rule $P \to C$	Def. 4.5.2, page 118
$L_1 \Rightarrow^*_{\mathcal{I},M,\sigma} L_{n+1}$	Background derivation of L_{n+1} from M with top literal L_1 and substitution σ	Def. 4.5.5, page 123
$P' \Rightarrow_{P\to_w C,\,\delta} C'$	Matching theory inference with rule $P \to_w C$ with weight w	Def. 5.2.2, page 158
$\mathcal{I}_0 \vdash_{\mathcal{S}} \mathcal{I}_1 \vdash_{\mathcal{S}} \cdots \vdash_{\mathcal{S}} \mathcal{I}_n \vdash_{\mathcal{S}} \cdots$	Deduction in transformation system $\mathcal{S}$	Def. 5.4.1,page 171

List of Figures

Bibliography

[Ait-Kaci and Nasr, 1986] H. Ait-Kaci and R. Nasr. LOGIN: A Logic Programming Language with Built-in Inheritance. *Journal of Logic Programming*, 3:293–345, 1986.

[Anderson and Bledsoe, 1970] R. Anderson and W. Bledsoe. A linear format for resolution with merging and a new technique for establishing completeness. *J. of the ACM*, 17:525–534, 1970.

[Andrews, 1976] Peter B. Andrews. Refutations by Matings. *IEEE Transactions on Computers*, C-25:193–214, 1976.

[Antoniou and Langetepe, 1994] G. Antoniou and E. Langetepe. Applying SLD-Resolution to a Class of Non-Horn Logic Programs. *Bulletin of the IGPL*, 2(2):231–243, 1994.

[Apt and van Emden, 1982] K.R. Apt and M.H. van Emden. Contributions to the Theory of Logic Programming. *Journal of the Association for Computing Machinery*, 29(3):841–862, 1982.

[Astrachan and Stickel, 1992] Owen L. Astrachan and Mark E. Stickel. Caching and Lemmaizing in Model Elimination Theorem Provers. In Kapur [1992], pages 224–238.

[Bachmair and Ganzinger, 1990] L. Bachmair and H. Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In Stickel [1990b], pages 427–441.

[Bachmair and Ganzinger, 1994] L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *Proc. 9th IEEE Symposium on Logic in Computer Science*, pages 384–393. IEEE Computer Society Press, 1994.

[Bachmair and Ganzinger, 1998a] Leo Bachmair and Harald Ganzinger. Elimination of equality via transformation with ordering constraints. In Claude Kirchner and Hélène Kirchner, editors, *Automated Deduction — CADE 15*, LNAI 1421, Lindau, Germany, July 1998. Springer-Verlag.

[Bachmair and Ganzinger, 1998b] Leo Bachmair and Harald Ganzinger. Ordered chaining calculi for first-order theories of transitive relations. *Journal of the Association for Computing Machinery*, 1998. Revised Version of MPI-I-95-2-009. To appear.

[Bachmair *et al.*, 1986] Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for equational proofs. *IEEE*, pages 346–357, 1986.

[Bachmair *et al.*, 1989] L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In H. Ait-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures 2: Rewrite Techniques*, pages 1–30. Academic Press, 1989.

[Bachmair *et al.*, 1992] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation and Superposition. In Kapur [1992], pages 462–476.

[Bachmair, 1987] Leo Bachmair. *Proof Methods for Equational Theories.* PhD thesis, University of Illinois at Urbana, U.S.A., 1987.

[Bachmair, 1991] L. Bachmair. *Canonical Equational Proofs.* Progress in Theoretical Computer Science. Birkhäuser, 1991.

[Baumgartner and Brüning, 1997] Peter Baumgartner and Stefan Brüning. A Disjunctive Positive Refinement of Model Elimination and its Application to Subsumption Deletion. *Journal of Automated Reasoning*, 19(2):205–262, 1997.

[Baumgartner and Furbach, 1993] P. Baumgartner and U. Furbach. Consolution as a Framework for Comparing Calculi. *Journal of Symbolic Computation*, 16(5):445–477, 1993. Academic Press.

[Baumgartner and Furbach, 1994a] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives and its Application to PTTP. *Journal of Automated Reasoning*, 13:339–359, 1994. Short version in: Proceedings of CADE-12, Springer LNAI 814, 1994, pp 87–101.

[Baumgartner and Furbach, 1994b] P. Baumgartner and U. Furbach. Model Elimination without Contrapositives. In A. Bundy, editor, *'Automated Deduction – CADE-12'*, volume 814 of *LNAI*, pages 87–101. Springer, 1994.

[Baumgartner and Furbach, 1994c] P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. In A. Bundy, editor, *Automated Deduction – CADE-12*, volume 814 of *LNAI*, pages 769–773. Springer, 1994. Available in the WWW, URL: `http://www.uni-koblenz.de/ag-ki/Systems/PROTEIN/`.

[Baumgartner and Petermann, 1998] Peter Baumgartner and Uwe Petermann. Theory Reasoning. In Bibel and Schmitt [1998].

[Baumgartner and Stolzenburg, 1995] P. Baumgartner and F. Stolzenburg. Constraint Model Elimination and a PTTP-Implementation. In R. Hähnle P. Baumgartner and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 918 of *Lecture Notes in Artificial Intelligence*, pages 201–216, 1995.

[Baumgartner *et al.*, 1992] P. Baumgartner, U. Furbach, and U. Petermann. A Unified Approach to Theory Reasoning. Research Report 15/92, University of Koblenz, 1992.

[Baumgartner *et al.*, 1995] P. Baumgartner, U. Furbach, and F. Stolzenburg. Model Elimination, Logic Programming and Computing Answers. In *14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, volume 1, 1995.

[Baumgartner *et al.*, 1996] P. Baumgartner, U. Furbach, and I. Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, number 1126 in LNAI. European Workshop on Logic in AI, Springer, 1996. (Long version in: *Fachberichte Informatik*, 8–96, Universität Koblenz-Landau).

[Baumgartner *et al.*, 1997] Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In *Proc. of IJCAI '97.* IJCAI, 1997.

[Baumgartner, 1990] P. Baumgartner. Combining Horn Clause Logic with Rewrite Rules. In V. Sgurev Ph. Jorrand, editor, *Artificial Intelligence IV – Methodology, Systems, Applications.* Norh Holland, 1990.

[Baumgartner, 1991a] P. Baumgartner. A Completeness Proof Technique for Resolution with Equality. In Th. Christaller, editor, *GWAI '91 – 15. Fachtagung für Künstliche Intelligenz*, pages 12–22. Springer, 1991. Informatik Fachberichte 285.

[Baumgartner, 1991b] P. Baumgartner. A Model Elimination Calculus with Built-in Theories. Fachbericht Informatik 7/91, Universität Koblenz, 1991.

[Baumgartner, 1992a] P. Baumgartner. A Model Elimination Calculus with Built-in Theories. In H.-J. Ohlbach, editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer, 1992. LNAI 671.

[Baumgartner, 1992b] P. Baumgartner. An Ordered Theory Resolution Calculus. In A. Voronkov, editor, *Logic Programming and Automated Reasoning (Proceedings)*, pages 119–130, St. Petersburg, Russia, July 1992. Springer. LNAI 624.

[Baumgartner, 1993] P. Baumgartner. Refinements of Theory Model Elimination and a Variant without Contrapositives. Research Report 8/93, University of Koblenz, 1993. (Short version in *Proc. ECAI 94, 1994, Wiley*).

[Baumgartner, 1994] P. Baumgartner. Refinements of Theory Model Elimination and a Variant without Contrapositives. In A.G. Cohn, editor, *11th European Conference on Artificial Intelligence, ECAI 94*. Wiley, 1994.

[Baumgartner, 1996] Peter Baumgartner. Linear and Unit-Resulting Refutations for Horn Theories. *Journal of Automated Reasoning*, 16(3):241–319, June 1996.

[Beckert and Hähnle, 1992] B. Beckert and R. Hähnle. An Improved Method for Adding Equality to Free Variable semantic Tableaux. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *LNCS*, pages 507–521. Springer, 1992.

[Beckert and Pape, 1996] B. Beckert and C. Pape. Incremental Theory Reasoning Methods for Semantic Tableaux. In *Theorem Proving with Analytic Tableaux and Related Methods (Tableaux 96)*. Springer, 1996.

[Beckert *et al.*, 1996] Bernhard Beckert, Reiner Hähnle, Peter Oel, and Martin Sulzmann. The tableau-based theorem prover $_3$TAP, version 4.0. In M.A. McRobbie and J.K. Slaney, editors, *Automated Deduction — CADE 13*, LNAI 1104, pages 303–307, New Brunswick, NJ, USA, July 1996. Springer-Verlag.

[Beckert, 1994] B. Beckert. A completion-based method for mixed universal and rigid E-unification. In Bundy [1994], pages 678–692.

[Beckert, 1998] Bernhard Beckert. Equality and other theory inferences. In D'Agostino et al. [1998].

[Benanav, 1990] Dan Benanav. Simultaneous paramodulation. In Stickel [1990b], pages 442–455.

[Bertling, 1990] Hubert Bertling. Knuth-Bendix Completion of Horn Clause Programs for Restricted Linear Resolution and Paramodulation. In S. Kaplan and M. Okada, editors, *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, pages 181–193. Springer-Verlag, June 1990. LNCS 516.

[Bibel and Schmitt, 1998] Wolfgang Bibel and Peter H. Schmitt, editors. *Automated Deduction. A basis for applications*. Kluwer Academic Publishers, 1998.

[Bibel *et al.*, 1994] Wolfgang Bibel, Stefan Brüning, Uwe Egly, and Thomas Rath. Komet. In Bundy [1994], pages 783–787.

[Bibel, 1981] Wolfgang Bibel. On Matrices with Connections. *Journal of the Association for Computing Machinery*, 28:633–645, 1981.

[Bibel, 1982a] W. Bibel. *Automated Theorem Proving*. Vieweg, 1982.

[Bibel, 1982b] Wolfgang Bibel. A Comparative Study of Several Proof Procedures. *Artificial Intelligence*, 18:269–293, 1982.

[Bibel, 1983] Wolfgang Bibel. Matings in Matrices. *Communications of the Association for Computing Machinery*, 26:844–852, 1983.

[Bibel, 1987] W. Bibel. *Automated Theorem Proving*. Vieweg, 2nd edition, 1987.

[Bibel, 1992] W. Bibel. *Deduktion*, volume 6.2 of *Handbuch der Informatik*. Oldenburg, 1992.

[Bollinger, 1991] T. Bollinger. A Model Elimination Calculus for Generalized Clauses. In *IJCAI*, 1991.

[Bonacina and Hsiang, 1992] Maria Paola Bonacina and Jieh Hsiang. Incompleteness of the RUE/NRF inference systems. Newsletter of the Association for Automated Reasoning, No. 20, pages 9–12, May 1992.

[Bonacina and Hsiang, 1994] Maria Paola Bonacina and Jieh Hsiang. Parallelization of deduction strategies: an analytical study. *Journal of Automated Reasoning*, 13:1–33, 1994.

[Bonacina and Hsiang, 1995] Maria Paola Bonacina and Jieh Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theoretical Computer Science*, 146:199–242, July 1995.

[Brachman and Schmolze, 1985] Ronald J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Brachmann *et al.*, 1983] R. Brachmann, R. Fikes, and H. Levesque. KRYPTON: a functional approach to knowledge representation. *IEEE Computer*, 16(10):67–73, October 1983.

[Brand, 1975] D. Brand. Proving theorems with the modification method. *SIAM Journal on Computing*, 4:412–430, 1975.

[Bronsard and Reddy, 1992] Francois Bronsard and Uday S. Reddy. Reduction Techniques for First-Order Reasoning. In M. Rusinowitch and J.L. Rémy, editors, *Proceedings of the Third International Workshop on Conditional Term Rewriting Systems*, pages 242–256. Springer-Verlag, July 1992. LNCS 656.

[Brown, 1978] Frank Malloy Brown. Towards the automation of set theory and its logic. *Artificial Intelligence*, 10(3):281–316, 1978.

[Brüning, 1995] S. Brüning. Exploiting Equivalences in Connection Calculi. *Journal of the IGPL*, 3(6):857–886, 1995.

[Bundy, 1994] Alan Bundy, editor. *Automated Deduction — CADE 12*, LNAI 814, Nancy, France, June 1994. Springer-Verlag.

[Bürckert, 1990] H.J. Bürckert. A Resolution Principle for Clauses with Constraints. In Stickel [1990b], pages 178–192.

[Bürckert, 1991] H.-J. Bürckert. *A Resolution Principle for Clauses with Restricted Quantifiers*, volume 568 of *LNAI*. Springer, 1991.

[Büttner, 1986] W. Büttner. Unification in the Datastructure Multisets. *Journal of Automated Reasoning*, 2:75–88, 1986.

[Chang and Lee, 1973] C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[Christoph Goller and Schumann, 1994] Klaus Mayr Christoph Goller, Reinhold Letz and Johann Schumann. Setheo v3.2: Recent developments — system abstract —. In Bundy [1994], pages 778–782.

[Chu and Plaisted, 1994] Heng Chu and David A. Plaisted. Semantically Guided First-Order Theorem Proving using Hyper-Linking. In Bundy [1994], pages 192–206.

[D'Agostino *et al.*, 1998] Marcello D'Agostino, Dov Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1998.

[Davis and Putnam, 1960] M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7:201–215, 1960.

[de Kogel, 1995] E. de Kogel. Rigid E-unification simplified. In R. Hähnle P. Baumgartner and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 918 of *Lecture Notes in Artificial Intelligence*, pages 17–30, 1995.

[Degtyarev and Voronkov, 1995a] A. Degtyarev and A. Voronkov. Equality Elimination for the Inverse Method and Extension Procedures. In C.S. Mellish, editor, *IJCAI 95: 14th International Conference on Artificial Intelligence*, 1995.

[Degtyarev and Voronkov, 1995b] A. Degtyarev and A. Voronkov. Simultaneous Rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, 1995.

[Dershowitz and Manna, 1979] N. Dershowitz and Z. Manna. Proving Termination with multiset orderings. *Comm. ACM*, 22:465–476, 1979.

[Dershowitz and Sivakumar, 1988] Nachum Dershowitz and G. Sivakumar. Goal-Directed Equation Solving. In *Proceedings of the AAAI-88*, 1988.

[Dershowitz, 1982] N. Dershowitz. Orderings for Term-Rewriting Systems. *Theoretical Computer Science*, 17:279–301, 1982.

[Dershowitz, 1985] N. Dershowitz. Computing with Rewrite Systems. *Information and Control*, 65:122 – 157, 1985.

[Dershowitz, 1987] Nachum Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3(1&2):69–116, February/April 1987.

[Dershowitz, 1990] Nachum Dershowitz. A Maximal-Literal Unit Strategy for Horn Clauses. In S. Kaplan and M. Okada, editors, *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, pages 14–25. Springer-Verlag, June 1990. LNCS 516.

[Dershowitz, 1991a] N. Dershowitz. Ordering-Based Strategies for Horn Clauses. In *Proc. IJCAI*, 1991.

[Dershowitz, 1991b] Nachum Dershowitz. Canonical Sets of Horn Clauses. In J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, pages 267–278, July 1991. LNCS 510.

[Digricoli and Harrison, 1986] Vincent J. Digricoli and Malcolm C. Harrison. Equality-Based binary Resolution. *Journal of the Association for Computing Machinery*, 1986.

[Dixon, 1973] J. Dixon. Z-Resolution: Theorem-Proving with Compiled Axioms. *Journal of the ACM*, 20(1):127–147, 1973.

[Eder, 1991] E. Eder. Consolution and its Relation with Resolution. In *Proc. IJCAI '91*, 1991.

[Eder, 1992] E. Eder. *Relative Complexities of First Order Languages.* Vieweg, 1992.

[Enderton, 1972] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Fages and Huet, 1986] F. Fages and G. Huet. Complete Sets of Unifiers and Matchers in Equational Theories. *Theoretical Computer Science*, 43, 1986.

[Fages, 1984] F. Fages. Associative-commutative unification. Rapports de Recherche 287, INRIA, April 1984.

[Fitting, 1990] M. Fitting. *First Order Logic and Automated Theorem Proving.* Texts and Monographs in Computer Science. Springer, 1990.

[Fitting, 1993] Melvin Fitting. Basic Modal Logic. In Gabbay et al. [1993], pages 365–448.

[Frisch, 1991] A. M. Frisch. The Substitutional Framework for Sorted Deduction: Fundamental Results on Hybrid Reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[Furbach *et al.*, 1989a] U. Furbach, S. Hölldobler, and J. Schreiber. Horn Equational Theories and Paramodulation. *Journal of Automated Reasoning*, 5:309–337, 1989.

[Furbach *et al.*, 1989b] U. Furbach, S. Hölldobler, and J. Schreiber. Paramodulation modulo equality. In *Proc. GWAI'89*, pages 107–116. Springer, IFB 216, 1989.

[Furbach *et al.*, 1989c] Ulrich Furbach, Steffen Hölldobler, and Joachim Schreiber. Horn equational theories and paramodulation. *Journal of Automated Reasoning*, 3:309–337, 1989.

[Furbach, 1991] U. Furbach. *Logische und Funktionale Programmierung — Grundlagen einer Kombination.* Künstliche Intelligenz. Vieweg, 1991.

[Gabbay *et al.*, 1993] Dov M. Gabbay, C.J. Hogger, and J.A. Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 1: Logical Foundations*. Oxford Science Publications, 1993.

[Gallier and Snyder, 1990] J. Gallier and W. Snyder. Designing Unification Procedures Using Transformations: A Survey. *Bulletin of the EATCS*, 40:273 – 326, 1990.

[Gallier *et al.*, 1987] J. H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid e-unification: Equational matings. In *Logics in Computer Science '87, Ithaca, New York*, 1987.

[Gallier *et al.*, 1992] Jean Gallier, Paliath Narendran, Stan Raatz, and Wayne Snyder. Theorem Proving Using Equational Matings and Rigid E-Unification. *Journal of the ACM*, 39(2):377–429, April 1992.

[Gallier, 1987] J. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Wiley, 1987.

[Ganzinger, 1991] Harald Ganzinger. A Completion Procedure for Conditional Equations. *Journal of Symbolic Computation*, 11:51–81, 1991.

[Gentzen, 1939] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 176–210, 405–431, 1939. English translation in: Szabo, M. (ed.), *The Collected Papers of Gerhard Gentzen*, pp. 68–131, North Holland, Amsterdam, 1969.

[Hanschke and Hinkelmann, 1992] P. Hanschke and K. Hinkelmann. Combining Terminological and Rule-based Reasoning for Abstraction Processes. In H.-J. Ohlbach, editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 144–155. Springer, 1992. LNAI 671.

[Heinemann and Weihrauch, 1991] B. Heinemann and K. Weihrauch. *Logik für Informatiker*. Leitfäden und Monographien der Informatik. B.G. Teubner, 1991. (in German).

[Hines, 1990] L. Hines. Str+ve$\subseteq$: The Str+ve-based Subset Prover. In Stickel [1990b], pages 193–206.

[Hines, 1992] L. Hines. The Central Variable Strategy of Str$\dot{+}$ve. In Kapur [1992], pages 35–49.

[Hölldobler, 1989] Steffen Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence. Subseries of Lecture Notes in Computer Science*. Springer, 1989.

[Hollunder, 1990] B. Hollunder. Hybrid Inferences in KL-ONE-based Knowledge Representation Systems. Research Report RR-90-6, DFKI, May 1990.

[Hopcroft and Ullman, 1979] A. Hopcroft and D. Ullman. *Formal languages and their relation to automata*. Addison-Wesley, 1979.

[Hsiang and Rusinowitch, 1986] J. Hsiang and M. Rusinowitch. A New Method for Establishing Refutational Completeness in Theorem Proving. In Jörg Siekmann, editor, *8th International Conference on Automated Deduction*, LNCS 230, pages 141–152, Oxford, England, 1986. Springer-Verlag.

[Hsiang and Rusinowitch, 1987] J. Hsiang and M. Rusinowitch. On word problems in equational theories. In *Proc. ICALP'87*, pages 54–71. Springer, LNCS 267, 1987.

[Hsiang and Rusinowitch, 1991] Jieh Hsiang and Michäel Rusinowitch. Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. *Journal of the ACM*, 38(3):559–587, 1991.

[Hsiang, 1985] J. Hsiang. Refutational theorem proving using term rewriting systems. *Artificial Intelligence*, 25:255–300, 1985.

[Huet and Oppen, 1980] Gérard Huet and Derek C. Oppen. Equations and Rewrite Rules. A Survey. In R. Book, editor, *Formal Languages: Perspectives and Open Problems*, pages 349–405. Academic Press, 1980.

[Hullot, 1980] J.M. Hullot. Canonical forms and unification. In *Proc. Conf. Automated Deduction*, pages 318–334, 1980.

[J.-P. Jouannaud, 1991] C. Kirchner J.-P. Jouannaud. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. In J.L. Lassez and G. Plotkin, editors, *Computational Logic — Essays in Honor of Alan Robinson*, pages 257–321. MIT Press, 1991.

[Jaffar and Lassez, 1987] J. Jaffar and J.-L. Lassez. Constrained Logic Programming. In *Proc. of the ACM Symp. on Principles of Programming Languages*, pages 111–119, 1987.

[Joyner, 1976] W.H. Joyner. Resolution Strategies as Decision Procedures. *Journal of the ACM*, 23(3):396–417, 1976.

[Kaplan, 1987] Stéphane Kaplan. Simplifying Conditional Term Rewriting Systems: Unification, Termination and Confluence. *J. of Symbolic Computation*, 4(3):295–334, 1987.

[Kapur, 1992] Deepak Kapur, editor. *11th International Conference on Automated Deduction*, LNAI 607. Springer-Verlag, 1992.

[Kerber *et al.*, 1993] Manfred Kerber, Erica Melis, and Jörg Siekmann. Reasoning with Assertions and Examples. Technical Report SEKI Report SR-93-10, Universität des Saarlandes, Fachbereich Informatik, 1993.

[Knight, 1989] Kevin Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124, March 1989.

[Knuth and Bendix, 1970] Donald E. Knuth and B. Bendix, Peter. Simple world problems in universal algebras, 1970.

[Kowalski and Kuehner, 1971] R. A. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[Kowalski, 1974] R. A. Kowalski. A proof procedure using connection graphs. *Journal of the ACM*, 22:572–595, 1974.

[Kunen, 1991] Kenneth Kunen. A completeness result for linked resolution. Technical Report CS-TR-91-1013, University of Wisconsin, Madison, February 1991.

[Lalement, 1993] R. Lalement. *Computation as Logic*. International Series in Computer Science. Prentice Hall, 1993.

[Lee and Plaisted, 1989] Shie-Jue Lee and David A. Plaisted. Reasoning with Predicate Replacement, 1989.

[Lee and Plaisted, 1992] S.-J. Lee and D. Plaisted. Eliminating Duplicates with the Hyper-Linking Strategy. *Journal of Automated Reasoning*, 9:25–42, 1992.

[Letz *et al.*, 1992] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2), 1992.

[Letz *et al.*, 1994] R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13, 1994.

[Letz, 1993] R. Letz. *First-Order Proof Calculi and Proof Procedures for Automated Deduction*. PhD thesis, Technische Hochschule Darmstadt, 1993.

[Lloyd, 1987] J. Lloyd. *Foundations of Logic Programming*. Symbolic Computation. Springer, second, extended edition, 1987.

[Loveland and Reed, 1989] D.W. Loveland and D.W. Reed. A near-horn prolog for compilation. Technical Report CS-1989-14, Duke University, 1989.

[Loveland, 1968] D. Loveland. Mechanical Theorem Proving by Model Elimination. *JACM*, 15(2), 1968.

[Loveland, 1970] D. Loveland. A Linear Format for Resolution. In *Symposium on Automatic Demonstration*, number 125 in Lecture Notes in Mathematics, pages 147–162, 1970.

[Loveland, 1978] D. Loveland. *Automated Theorem Proving - A Logical Basis.* North Holland, 1978.

[Lynch, 1995] Christopher Lynch. Paramodulation Without Duplication. In *Logics in Computer Science*, 1995.

[Manna and Waldinger, 1986] Z. Manna and R. Waldinger. Special Relations in Automated Deduction. *Journal of the ACM*, 33(1):1–59, 1986.

[Manna *et al.*, 1991] Z. Manna, M. Stickel, and R. Waldinger. Monotonicity Properties in Automated Deduction. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Essays in Honor of John McCarthy*, pages 261–280. Academic Press, 1991.

[Marcinkowski and Pacholski, 1992] J. Marcinkowski and L. Pacholski. Undecidability of the Horn Clause Implication Problem. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 354–362, 1992.

[Martelli *et al.*, 1986] A. Martelli, C. Moiso, and G. F. Rossi. An algorithm for unification in equational theories. In *Proc. of the Third IEEE Symposium on Logic Programming*, pages 180–186, 1986.

[Mayr, 1995] K. Mayr. Link Deletion in Model Elimination. In R. Hähnle P. Baumgartner and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 918 of *Lecture Notes in Artificial Intelligence*, pages 169–184, 1995.

[McCharen *et al.*, 1976] J. McCharen, R. Overbeek, and L. Wos. Complexity and related enhancements for automated theorem-proving programs. *Computers and Mathematics with Applications*, 2:1–16, 1976.

[McMichael, 1990] Alan F. McMichael. SLIM: An automated reasoner for equivalences, applied to set theory. In Stickel [1990b], pages 308–321.

[Morris, 1969] J. B. Morris. E-Resolution: An Extension of Resolution to Include the Equality Relation. In *Proc. IJCAI*, pages 287–294, 1969.

[Moser *et al.*, 1995] Max Moser, Christopher Lynch, and Joachim Steinebach. Model Elimination with Basic Ordered Paramodulation. Forschungsgruppe Automated Reasoning AR-95-11, Technische Universität München, 1995.

[Moser, 1993] M. Moser. Improving Transformation Systems for General E-Unification. In *Rewrite Techniques and Applications, RTA-95*, volume 690 of *LNCS*, pages 92–105. Springer, 1993.

[Murray and Rosenthal, 1987] N. Murray and E. Rosenthal. Theory Links: Applications to Automated Theorem Proving. *J. of Symbolic Computation*, 4:173–190, 1987.

[Myers, 1994] K.L. Myers. Hybrid Reasoning Using Universal Attachment. *Artifical Intelligence*, 67(2):329–375, June 1994.

[Neugebauer and Petermann, 1995] Gerd Neugebauer and Uwe Petermann. Specification of Inference Rules and Their Automatic Translation. In R. Hähnle P. Baumgartner and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 918 of *Lecture Notes in Artificial Intelligence*. Springer, 1995.

[Nieuwenhuis and Orejas, 1990] Robert Nieuwenhuis and Fernando Orejas. Clausal Rewriting. In S. Kaplan and M. Okada, editors, *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, pages 246–258. Springer-Verlag, June 1990. LNCS 516.

[Nieuwenhuis and Rubio, 1992] Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering Constrained Clauses. In Kapur [1992], pages 477–491.

[Nieuwenhuis and Rubio, 1995] R. Nieuwenhuis and A. Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19:321–351, 1995.

[Nieuwenhuis, 1993] R. Nieuwenhuis. Simple LPO Constraint Solving Methods. *Information Processing Newsletters*, 47:65–69, 1993.

[Nonnengart, 1995] Andreas Nonnengart. *A Resolution-Based Calculus for Temporal Logics*. PhD thesis, Universität des Saarlandes, 1995.

[Oberschelp, 1962] A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Math. Annalen*, 145:297–333, 1962.

[Ohlbach, 1987] Hans Jürgen Ohlbach. Link Inheritance in Abstract Clause Graphs. *Journal of Automated Reasoning*, 3(1):1–34, 1987.

[Ohlbach, 1990] H.-J. Ohlbach. Compilation of Recursive Two-Literal Clauses into Unification Algorithms. In V. Sgurev Ph. Jorrand, editor, *Artificial Intelligence IV – Methodology, Systems, Applications*. Norh Holland, 1990.

[Ohlbach, 1993] H.-J. Ohlbach. Translation Methods for Non-Classical Logics — An Overview. Research Report MPI-I-93-225, Max-Planck-Institut für Informatik, 1993.

[Oppacher and Suen, 1988] F. Oppacher and E. Suen. HARP: A Tableau-Based Theorem Prover. *Journal of Automated Reasoning*, 4:69–100, 1988.

[Paramasivam and Plaisted, 1995] M. Paramasivam and David Plaisted. Automated Deduction Techniques for Classification in Concept Languages. Technical report, University of North Carolina, 1995.

[Paul, 1985] Etienne Paul. Equational Methods in First Order Predicate Calculus. *Journal of Symbolic Computation*, 1(1), March 1985.

[Paul, 1986] E. Paul. On Solving the Equality Problem in Theories Defined by Horn Clauses. *Theoretical Computer Science*, 44:127–153, 1986.

[Petermann, 1991a] U. Petermann. Building in Equational Theories into the Connection Method. In *Proceedings of Symposium on Fundamentals of Artificial Intelligence Research*, Smolenice, 1991.

[Petermann, 1991b] U. Petermann. Petermann U., Building in Equational Theories into the Connection Method. In *Proceedings of the 1st Int. Symp. on Foundamentals of AI-Research, FAIR '91*, 1991.

[Petermann, 1992] U. Petermann. How to build in an open theory into connection calculi. *J. on Computer and Artificial Intelligence*, 1992.

[Petermann, 1993a] U. Petermann. Completeness of the pool calculus with an open built in theory. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *3rd Kurt Gödel Colloquium '93*, number 713 in Lecture Notes in Computer Science, pages 264–277. Springer-Verlag, 1993.

[Petermann, 1993b] Uwe Petermann. Building-in a theory into a connection calculus with positive refinement. Technical Report 25/1993, Uni Leipzig, Naturwissenschaftlich-Theoretisches Zentrum, 1993.

[Petermann, 1994] U. Petermann. A Complete Connection Calculus with Rigid E-Unification. In C. MacNish, D. Pearce, and L. Pereira, editors, *Logics in Artificial Intelligence*, volume 838 of *LNAI*, pages 152–166. Springer, 1994.

[Peterson, 1983] G. Peterson. A Technique for Establishing Completeness Results in Theorem Proving with Equality. *SIAM Journal on Computing*, 12(1):82–100, February 1983.

[Plaisted and Greenbaum, 1984] D. Plaisted and S. Greenbaum. Problem Representations for Back Chaining and Equality in Resolution Theorem Proving. In *Proceedings of the First Conference on Artificial Intelligence Applications*, 1984.

[Plaisted, 1988] D. Plaisted. Non-Horn Clause Logic Programming Without Contrapositives. *Journal of Automated Reasoning*, 4:287–325, 1988.

[Plaisted, 1990] D. Plaisted. A Sequent-Style Model Elimination Strategy and a Positive Refinement. *Journal of Automated Reasoning*, 4(6):389–402, 1990.

[Plaisted, 1993] D. Plaisted. Equational Reasoning and Term Rewriting Systems. In Gabbay et al. [1993], pages 273–364.

[Plaisted, 1994] David Plaisted. The Search Efficiency of Theorem Proving Strategies. In Bundy [1994].

[Plaisted, 1995] D. Plaisted. Special Cases and Substitutes for Rigid E-Unification. Research Report MPI-I-95-2-010, Max-Planck-Institut für Informatik, 1995.

[Plotkin, 1972] G. D. Plotkin. Building-In Equational Theories. *Machine Intelligence*, 7:73–90, 1972.

[Rabin, 1977] M. Rabin. Decidable Theories. In Jon Barwise, editor, *Studies in Logic and the Foundations of Mathematics*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.3, pages 595–630. North-Holland, 1977.

[Reeves, 1987] S. Reeves. Semantic Tableaux as a Framework for Automated Theorem Proving. In C.S. Mellish and J. Hallam, editors, *Advances in Artificial Intelligence (Proceedings of AISB)*, pages 125–139, 1987.

[Reif, 1992] W. Reif. The KIV-System: Systematic Construction of Verified Software. In D. Kapur, editor, *11th Conference on Automated Deduction. Proceedings*, Springer LNCS. Albany, NY, USA, 1992.

[Reiter, 1987] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

[Robinson and Wos, 1969] G. A. Robinson and L. Wos. Paramodulation and Theorem Proving in First Order Theories with Equality. In Meltzer and Mitchie, editors, *Machine Intelligence 4*. Edinburg University Press, 1969.

[Robinson, 1965a] J. A. Robinson. Automated deduction with hyper-resolution. *Internat. J. Comput. Math.*, 1:227–234, 1965.

[Robinson, 1965b] J.A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, January 1965.

[Schmidt-Schauß and Smolka, 1991] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1988] M. Schmidt-Schauß. Implication of Clauses is Undecidable. *Theoretical Computer Science*, 59:287–296, 1988.

[Schmidt-Schauß, 1989] M. Schmidt-Schauß. *Computational Aspects of an Order Sorted Logic with Term Declarations*, volume 395 of *Lecture Notes in Artificial Intelligence*. Springer, 1989.

[Schmitt and Wernecke, 1989] P.H. Schmitt and W. Wernecke. Tableau calculus for sorted logics. In *Sorts and Types in Artificial Intelligence*, volume 418 of *Lecture Notes in Artificial Intelligence*, pages 49–60. Springer, 1989.

[Schmitt, 1987] Peter H. Schmitt. The THOT theorem prover. Technical Report TR–87.09.007, IBM Heidelberg Scientific Center, 1987.

[Schumann, 1994] Johann Schumann. Delta — a bottom-up preprocessor for top-down theorem provers — system abstract —. In Bundy [1994], pages 774–777.

[Siekmann and Wrightson, 1983] Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning – Classical Papers on Computational Logic*. Symbolic Computation. Springer, 1983.

[Siekmann, 1989] Jörg H. Siekmann. Unification Theory. *Journal of Symbolic Computation*, 7(1):207–274, January 1989.

[Sikka and Genesereth, 1994] Vishal Sikka and Michael Genesereth. Integrating Specialized Procedures into Proof Systems. In *12th National Conference on Artificial Intelligence*. AAAI Press, 1994.

[Smullyan, 1968] R. Smullyan. *First Order Logic*. Springer, 1968.

[Snyder and Lynch, 1991] W. Snyder and C. Lynch. Goal directed strategies for paramodulation. In R. Book, editor, *Rewriting Techniques and Applications*, Lecture Notes in Computer Science No. 488, pages 15 – 28, Berlin, 1991. Springer.

[Snyder, 1991] W. Snyder. *A Proof Theory for General Unification.* Birkhäuser, 1991.

[Socher-Ambrosius, 1990] R. Socher-Ambrosius. Simplification and Reduction for Automated Theorem Proving. SEKI-Report SR-90-10, Universität Kaiserslautern, 1990. (Ph.D. Thesis).

[Socher, 1992] R. Socher. How to Avoid the Derivation of Redundant Clauses in Reasoning Systems. *Journal of Automated Reasoning*, 9:77–97, 1992.

[Steinbach, 1990] J. Steinbach. Improving Associative Path Orderings. In Stickel [1990b], pages 411–425.

[Stickel, 1985] M.E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Stickel, 1986] M. Stickel. An Introduction to Automated Deduction. In Bibel, Biermann, Delgrande, Huet, Jorrand, Shapiro, Mylopoulos, and Stickel, editors, *Fundamentals of Artificial Intelligence*, pages 75–132. Springer, 1986.

[Stickel, 1988] M. Stickel. A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning*, 4:353–380, 1988.

[Stickel, 1989] M. Stickel. A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog. Technical note 464, SRI International, 1989.

[Stickel, 1990a] M. Stickel. A Prolog Technology Theorem Prover. In Stickel [1990b], pages 673–675.

[Stickel, 1990b] Mark E. Stickel, editor. *10th International Conference on Automated Deduction*, LNAI 449, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.

[Sutcliffe *et al.*, 1994] G. Sutcliffe, C. Suttner, and T. Yemenis. The TPTP problem library. In Bundy [1994].

[Tammet, 1992] T. Tammet. Resolution Methods for Decision Problems and Finite-Model Building. Technical report, Chalmers University of Technology, Göteborg, Department of Computer Science, 1992. (Ph.D. Thesis).

[Tarver, 1990] Mark Tarver. An examination of the Prolog technology theorem-prover. In Stickel [1990b], pages 322–335.

[van Dalen, 1980] D. van Dalen. *Logic and Structure.* Universitext. Springer, 2nd edition, 1980.

[Veroff and Wos, 1992] R. Veroff and L. Wos. The Linked Inference Principle, I: the Formal Treatment. *Journal of Automated Reasoning*, 8(2), 1992.

[Voronkov and Degtyarev, 1996] Andrei Voronkov and Anatoli Degtyarev. What You Always Wanted to Know About Rigid E-Unification. UPMAIL Technical Report 125, Uppsala University, Computing Science Department, April 1996.

[Walther, 1983] C. Walther. A Many-Sorted Calculus Based on Resolution and Paramodulation. In *Proc. 8th IJCAI*, Karlsruhe, 1983.

[Walther, 1987] C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Pitman Ltd., 1987.

[Weidenbach, 1993] C. Weidenbach. Extending the Resolution Method with Sorts. In *13th International Joint Conference on Artificial Intelligence, IJCAI 93*, pages 60–65. Morgan Kaufmann, 1993.

[Weidenbach, 1994] Christoph Weidenbach. Minimal Resolution. Technical report, Max-Planck-Institut für Informatik, Saarbrücken, 1994.

[Weidenbach, 1995] C. Weidenbach. First-Order Tableaux with Sorts. *Journal of the IGPL*, 3(6), 1995.

[Wos *et al.*, 1984] L. Wos, R. Veroff, B. Smith, and W. McCune. The Linked Inference Principle, II: The User's Viewpoint. In R. E. Shostak, editor, *Proceedings of the 7^{th} Conference on Automated Deduction, Napa, CA, USA, May 1984*,

volume 170 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Heidelberg, New-York, 1984.

[Zhang and Kapur, 1988] H. Zhang and D. Kapur. First-Order Theorem Proving Using Conditional Rewrite Rules. In E. Lusk and R. Overbeek, editors, *9th International Conference on Automated Deduction*, LNCS 310, pages 1–20, Argonne National Laboratory, 1988. Springer-Verlag.

Index