A Unified Approach to Theory Reasoning

Peter Baumgartner, Ulrich Furbach
Universität Koblenz
Rheinau 3–4
W-5400 Koblenz
{peter,uli}@infko.uni-koblenz.de
Uwe Petermann
Universitaet Leipzig
Augustusplatz 10/11
O-7010 Leipzig
petermann@informatik.uni-leipzig.dbp.de

Abstract. *Theory reasoning* is a kind of two-level reasoning in automated theorem proving, where the knowledge of a given domain or theory is separated and treated by special purpose inference rules. We define a classification for the various approaches for theory reasoning which is based on the syntactic concepts of *literal level — term level — variable level*. The main part is a review of theory extensions of common calculi (resolution, model elimination and a connection method). In order to see the relationships among these calculi, we define a super-calculus called *theory consolution*. Completeness of the various theory calculi is proven. Finally, due to its relevance in automated reasoning, we describe current ways of equality handling.

Contents

1 Introduction

One of the most traditional disciplines in AI is Theorem Proving. In the early days it was concentrated mainly on developing general proof procedures for predicate logic. According to the shift within wide parts of AI-research towards special domain dependent systems, automated reasoning and theorem proving nowadays aim at incorporating specialized and efficient moduls, which are suited for handling special parts or domains of knowledge. Whenever this is done in a formal way we will understand those moduls as a means to built-in theories and speak of "theory reasoning".

A very prominent example for efficient handling of a theory 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 substitutivity 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) or RUE-resolution (Digricoli and Harrison, 1986).

Another very well-investigated example for theory handling is the design of calculi and proof procedures, which use *many-* or *order-sorted* logics (Bläsius and Bürckert, 1989). Here, the aim is to take care of a sort hierarchy in a direct way, e.g. by using a special unification procedure. This is in contrast to relativation approaches which transform the sort information into formulas of the unsorted logics.

While the above two examples were concerned mainly with an efficient treatment of a theory, there is as well a concern in the field of knowledge engineering of keeping *different kinds of knowledge* apart. Often, one wants to separate taxonomical from assertional knowledge: taxonomical knowledge is used as a special theory, which has to be handled outside the deduction mechanism which processes the assertional knowledge. 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), which will be discussed later on. Nowadays numerous works on defining *concept* languages with well-understood semantics for the definition of taxonomical knowledge exist (Hollunder, 1990).

Viewpoint of this paper

The aim of this paper is twofold: Firstly, we want to classify the various kinds of treating theories within deduction systems, and secondly, we want to compare a special subclass of these methods. It is easier to compare the theory reasoning calculi if a common input language can be presupposed. Fortunately, the calculi to be discussed operate on formulas in a standard language, which is, of course, *clause logic*. Thus clause logic will serve us well as a common input language, too. However there are minor differences: sometimes clauses are thought of as *sets* of literals, sometimes of *multi*sets and sometimes of *sequences*. Fortunately these differences are not essential for the intuitive understanding of theory reasoning, and so the discussion of these differences can be postponed to the formal section. As a coincidence most work is done in a *refutational setting*, but not in an *affirmative* one. Due to their perfect duality, it suffices to restrict attention to one of these concepts. This will be the more common *refutational* setting.

Theory reasoning always deals with two kinds of reasoning: background reasoning for the theory, and foreground reasoning for the actual problem specification. We are mostly interested in studying the interface between foreground and background reasoning. We give a formal description of this interface which applies to a wide class of theories. We will not focus on the question of how to built dedicated background reasoners for *special* theories (equality will be an exception).

It has to be said what kinds of theories we are interested in. The "upper bound" is given by the *universal theories*, i.e. theories that can be axiomatized by a set of formulas that does not contain $\exists$-quantifiers. Universal theories are expressive enough to formulate e.g. equality or interesting taxonomic theories. Moreover, 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 up to an extension of the signature by Skolem functions. Universal theories also mark the limit of what can be built into a calculus preserving the completeness of calculus (cf. (Petermann, 1991b)).

We do not treat reasoning in single models, like real numbers and their arithmetic, or classes of models. Such extensions of theory reasoning have been investigated in (Bürckert, 1990a).

Due to the great variety of approaches for theory reasoning, we prefer to bring in structure by classifying the various approaches. Of course there are plenty ways of doing. The classification we use is by *level of connection*. In order to explain this term it is necessary to recall the nature of theory reasoning as interfacing background reasoning and foreground reasoning. Now, by "level of connection" we mean the common subpart of the foreground and the background language, that is used for their interfacing. To be concrete, we will distinguish the three levels *literal level*, *term level* and *variable level*. Figure (1) is a classification of the approaches to be described with respect to level of connection. The *literal level* is the most general of all; it allows for theory reasoning with literals with *different* predicate symbols (general theory reasoning

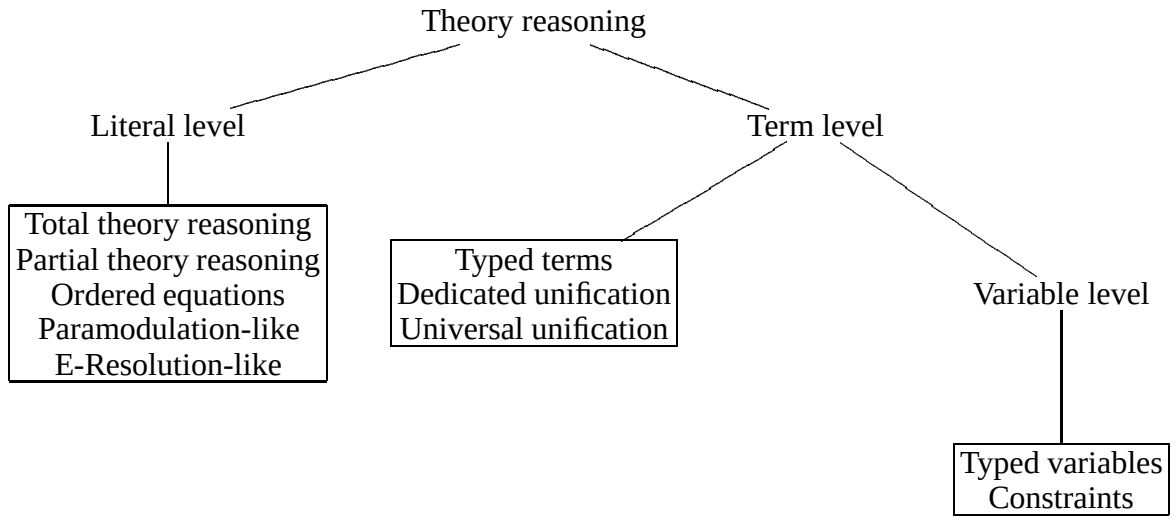


Figure 1: Classification wrt. level of connection

approaches and equality). This is different to *term level* theory reasoning, where unification on terms is replaced by some unification modulo a theory (typing, dedicated unification, universal unification). A further specialization is the *variable level* theory reasoning which is bounded to variables (typing of variables, constraints).

The methods subsumed by "theory reasoning" in figure (1) are subject to enormous research activities and results. Thus it would be too a big task to give a deep overview of all of them. Since there are excellent overviews of *term level* (usually known as "unification theory") (e.g. (Siekmann, 1989)) and *constraints* (see e.g. (Meseguer, 1989), or (Van Hentenryck, 1989) for a textbook on constraints in logic programming), we will concentrate on the *literal level*.

Our strategy for describing the literal level is essentially as follows: we will define a formal framework, called *theory consolution*, and show how it can be instantiated to the various known theory calculi. By this technique, we hope to give an understanding of the similarities and differences among the calculi. This technique of instantiating the consolution framework to other calculi has been applied by two of the authors to non-theory cases before in (Baumgartner and Furbach, 1992).

For a related overview of modifications to the resolution calculus, such as UR-resolution (McCharen et al., 1976) and Hyperresolution, but also theory resolution, see (Eisinger and Ohlbach, 1993). This work also covers various kinds of theories (equality, theories compiled from the axioms, taxonomically represented theories).

This work is structured as follows: in the next section the various theory reasoning approaches will be described informally. This tour touches all of them. In section 3 we will turn to a more

formal presentation of the literal level. As just mentioned, this will be done on the basis of one special calculus, namely the consolution calculus. It will be presented first in its non-theory version and finally we define its total and its partial variant. This calculi will be used afterwards to discuss theory resolution, theory model elimination and a theory connection method. The completeness of these calculi is proven rigorously by mapping our presentation of the calculi to corresponding representations, for which we have completeness results. Section 4 contains a discussion on equality handling.

2 The Tour: *Literal level – Term level – Variable level*

In this section we will present several methods of theory reasoning according to our classification. Consider again figure (1) in the introduction. The inner nodes can be briefly characterized as follows:

Literal level: Certain literals from selected clauses are passed to the theory reasoner. For example, if the theory is instantiated to "strict ordering" then the theory reasoner might get the two literals (not the different predicate symbols) $a < b$ and $a > x$ and decide that applying the substitution $\{x \leftarrow b\}$ to their conjunction gives a formula which is unsatisfiable in the theory of strict orderings.

Term level: The argument terms of certain literals from selected clauses are passed to the theory reasoner. Typically, *two* complementary literals are selected, and the theory reasoner has to check for a pairwise theory-unifiability of the argument terms. For example, if f is a commutative function symbol, then the theory reasoner might decide that the arguments of $P(f(a,b))$ and $\neg P(f(x,a))$ are unifiable by the substitution $\{x \leftarrow b\}$.

Variable level: Since variables are terms, this is a subcase of the previous case. Here, the legal assignments for variables are restricted in some way, e.g. by domain restrictions. For example if $P(x)$ and $\neg P(y)$ are given, and the domain of x is $\{a,b\}$ and the domain of y is $\{b,c\}$ then the substitution $\{x \leftarrow b, y \leftarrow b\}$ is in accordance with the domains and makes the above literal set unsatisfiable.

Obviously, the methods in the same level of connection deserve further structuring. Let us start with general literal level theory reasoning.

2.1 Literal Level Theory Reasoning

The basic idea of literal level theory reasoning is at best explained in comparison to ordinary, i.e. non-theory reasoning: in ordinary reasoning, clauses containing *syntactically* complementary

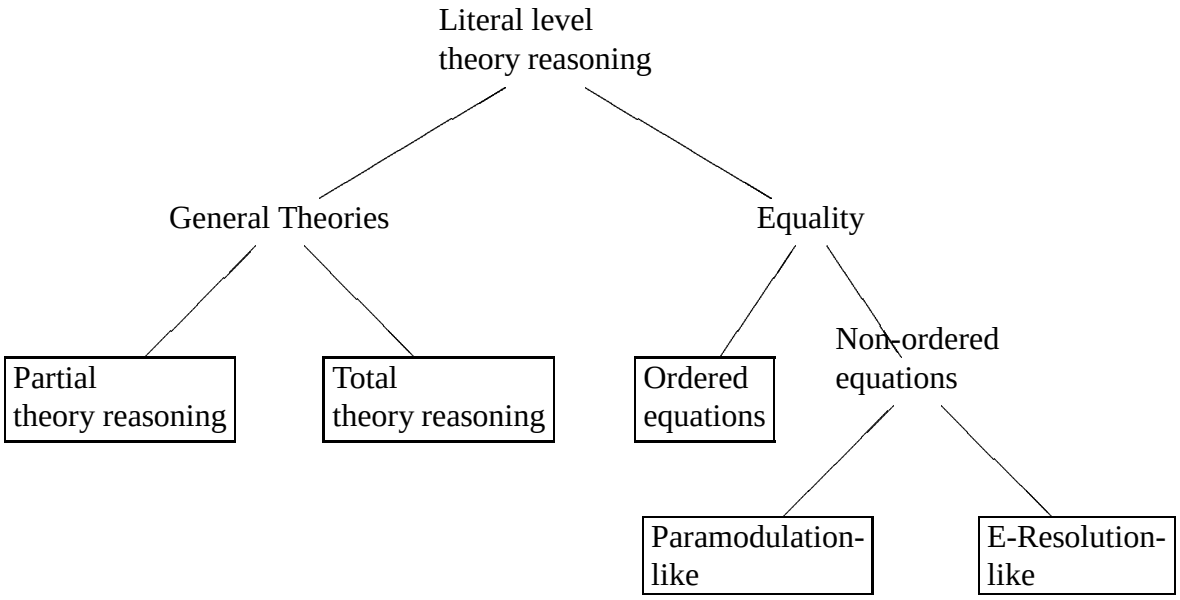


Figure 2: Classification of "Literal level"

literals are used for inferences, whereas in theory reasoning this is done with *semantically* complementary literals. Here "semantically complementary" means roughly "unsatisfiable in the given theory". A precise characterization is given in the next section.

Figure (2) depicts a classification of literal level theory reasoning. On the one hand we will consider *general theory reasoning* approaches. There, no fixed theory is presupposed, and these frameworks can be instantiated with a great variety of theories. On the other hand we will devote an extra section to *equality*, i.e. general theory reasoning that is instantiated with the theory of equality. This partition is motivated by the enormous research dedicated to equality.

General literal level theory reasoning (simply called "theory reasoning" from now on in this subsection) is the most general technique of all. It is a scheme of building in arbitrary universal theories into first order calculi. Still on this very general level two variants can be distinguished (figure 2):

Partial theory reasoning: Viewed operationally, in a partial reasoning step the background reasoner is passed from the foreground reasoner a set of literals and returns a formula. This formula is a new subgoal to be proved. It is also called *residue*. For example, if the theory is "strict ordering" and we are given the literals $S = \{a < b, b < c, c < d\}$ and the "goal" $\neg a < d$. By transitivity of $<$ the literal $a < d$ is a logical consequence of S, and this literal immediately contradicts the goal. To show this fact with partial theory reasoning the background reasoner might be passed the goal $\neg a < d$ and the literal $a < b$ from S; then it returns the residue $\neg b < d$. For the next step this residue plays the same

role as the goal before. This kind of reasoning is repeated until a contradiction becomes immediate.

Semantically, a residue states a logical consequence of the passed literals, or, in other words, the negation of the residue together with the passed literals is unsatisfiable in the theory. As usual, in the non-ground case substitutions are involved. See section (3) for a precise treatment of residues and so-called "theory refuting substitutions".

Total theory reasoning: Total theory reasoning is the same as partial theory reasoning, except that the residue must be empty. Thus, the literal set passed to the background reasoner must be unsatisfiable by itself. For example, the literals $a < b$ and $a > b$ are unsatisfiable in the theory of strict orderings, and thus are subject to a total theory reasoning step.

The distinction between these kinds of theory reasoning is important for several reasons: firstly, in partial theory reasoning we have the unique situation that the theory reasoner returns a formula to the caller, and not just a substitution or a yes-no result. Thus the coupling is more symmetric than the other approaches on the term level.
Secondly, for *undecidable* theories total theory reasoning requires too much from the theory reasoner, i.e. the inference must neccessarily be undecidable. As a consequence, the notion of "derivation" is undecidable — a highly undesirable property. This implies for implementations that the background reasoner cannot be "called" as a procedure that is guaranteed to terminate. But even if the theory is decidable there remain problems with total theory reasoning. For example let us consider the theory of equality. Though this theory admits a decision procedure for the background reasoner, as e.g. rigid $\mathcal{E}$-unification (cf. (Gallier and Snyder, 1990) and section 4 of the present paper), it cannot be predicted by the foreground reasoner how many variants of literals from clauses constitute a contradictory set. In other words: it is hard to find good candidates for contradictory sets. On the other hand, it may not be difficult for the foreground reasoner to detect the potential kernel of a contradictory set. Moreover, the theory reasoner might guide the search for literals which complete the kernel to a contradictory set. This scheme of co-operation with a search guiding role of the theory reasoner is the idea of partial theory reasoning. The residue returned by the partial reasoner gives advice to the general reasoner for the search for appropriate literals.

The generality of theory reasoning is both a strength and a weakness: it is a strength, because it subsumes all other techniques if an appropriate theory reasoner is given — and the generality is a weakness because it cannot propose how to come to efficient theory reasoners, which are usually domain dependent.

Since Stickel's pioneering work for the resolution calculus (Stickel, 1985; Stickel, 1983), the scheme was ported to many calculi. It was done for matrix methods in (Murray and Rosenthal, 1987), for the connection method in (Petermann, 1990) and for model elimination in (Baumgartner, 1992a). The latter two papers contain completeness results for the first-order case. The primary concern of these works is the *combination* of the main calculus with the theory reasoner; it is not the construction of efficient theory reasoners.

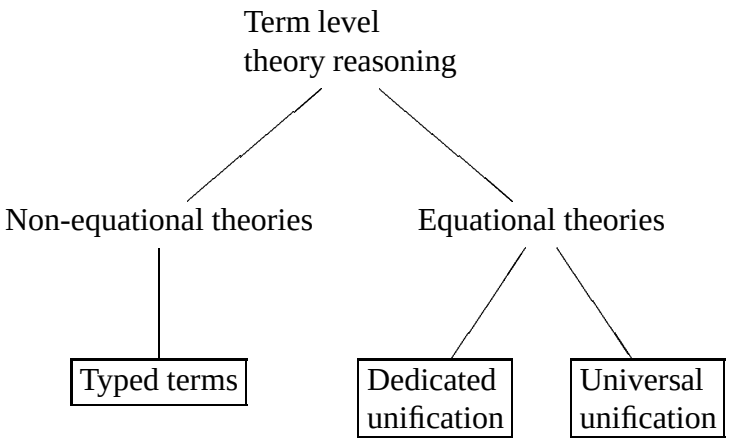


Figure 3: Classification of "Term level"

2.2 Term Level Theory Reasoning

Term level theory reasoning is better known under the name *unification modulo a theory*, or simply *theory unification*.

Since there are excellent overviews of this area, we will only supply a very brief description here. The reader interested in this topic is referred to (Bläsius and Bürckert, 1989; Siekmann, 1989) for more details.

Like general theory reasoning it generalizes a syntactical concept to a semantical one, by replacing the syntactical unification algorithm with a unification algorithm for the theory.

Theory unification differs from theory reasoning in two aspects:

- Theory *reasoning* is carried out on the literal level, whereas theory *unification* is carried out on the term level. Thus, the predicate symbols of the literals selected for the theory reasoning step may be different, while in the theory unification step they must be equal.
- The partial variant of theory reasoning computes with residues, i.e. the theory reasoner may establish new subgoals and return them to the main procedure. In theory unification such a concept is missing.

Figure (3) depicts a classification of theory unification, where the kind of theory is used as a classification criterion:

Equational theories: Since equations (pairs of the form $s = t$) occur so frequently in mathematics and in nearly every form of reasoning, it is not surprising that much research has been spent on the automatization of equational theories. Here the theory is axiomatized by a set of (conditional)equations E. Only $\mathcal{E}$-models are considered, i.e. models of the theory of equality $\mathcal{E}$ (section (4)).

Thus in equational unification the following *$\mathcal{E}$-unification problem* has to be solved:

> Given a set E of equations and some pairs $\langle u_i, v_i \rangle$. Is there a substitution σ such that every $\mathcal{E}$-model satisfying E also satisfies $u_i\sigma = v_i\sigma$, or shortly, that all equations $u_i\sigma = v_i\sigma$ are $\mathcal{E}$-consequences of E.

Such a σ is called *$\mathcal{E}$-unifier for $\langle u_i, v_i \rangle$ in the theory E*. The $\mathcal{E}$-unification problem can equivalently be formulated in a more operational fashion. It is roughly as follows: in order to solve the above problem, try to transform u_i into v_i by replacement of subterms with equal terms as given by E. The $\mathcal{E}$-unifier is obtained by collecting the substitutions along the way.

Some questions coming up immediately are the following: is the $\mathcal{E}$-unification problem decidable? (No); how many such $\mathcal{E}$-unifiers exist? (Countable many); if more than one exists, can we compute a solution base, i.e. a reasonable small set that implicitly includes all other solutions?

A problem similar to $\mathcal{E}$-unification is *rigid $\mathcal{E}$-unification*. This is a resource-bounded form of $\mathcal{E}$-unification which, loosely, forbids drawing more than one instance of an equation in E when solving an $\mathcal{E}$-unification problem. Rigid $\mathcal{E}$-unification is relevant for building in *the* theory of equality into general theory reasoning calculi and is discussed in section (4).

Unification procedures for equational theories can be further distinguished: on the one hand, there are *dedicated* unification algorithms, which are special purpose theory unification algorithms for one single equational theory (see e.g. (Petermann, 1991a)). For example, there are such algorithms for associative and for commutative theories (see (Bürckert et al., 1988) as an anchor). From a practical point of view it would be very pleasing to combine given unification algorithms for different theories in order to obtain a unification algorithm for the theories' union. Unfortunately this is a highly non-trivial task. An advanced result allows for the combination of theories with disjunct function symbols, but common constant symbols (Ringeissen, 1992).

On the other hand we have *universal* unification algorithms, that work for a wide class of equational theories (see (Gallier and Snyder, 1990; Gallier and Snyder, 1989; J.-P. Jouannaud, 1991)). Further advantage can be taken if the equations can be directed into rewrite rules. Then the *Narrowing*-technique can be used, which is the first order version of rewriting (Hullot, 1980; W. Nutt and Smolka, 1987; Hölldobler, 1989). General overviews of theory unification can be found in (J.-P. Jouannaud, 1991; Siekmann, 1989), and a recent text book is (Snyder, 1991).

Non-equational theories – typing: The most investigated non-equational theories (at least, conceptually) are those which employ a type hierarchy on terms, as already discussed in the introduction. This is also known as order-sorted unification. Roughly, if $s_1 : T_1$

means that the term s_1 is of type T_1, then the unification of the expressions $s_1 : T_1$ and $s_2 : T_2$ succeeds if s_1 and s_2 (Robinson-)unify, and if T_1 and T_2 can both be restricted to a common subtype. See (J.-P. Jouannaud, 1991) for an overview. Order-sorted logics has been developed since the 60s (Oberschelp, 1962), and nowadays numerous proof-procedures exist, which demonstrate that order-sortedness increases efficiency significantly.

2.3 Variable Level Theory Reasoning – Constraints

In variable level theory reasoning, the set of legal values for variables is limited. This is usually achieved by *constraints*. Syntactically, constraints are formulas that are attached to some variables in clauses, and semantically they filter out valid assignments for the variables. During inferences the constraints of the unified variables are combined, and eventually, but not necessarily immediately, the combined constraints must be solved. In other words, constraints may be treated lazily. This is the approach taken in (Bürckert, 1990a). Constraints are mostly investigated in the context of logic programming and Prolog (Van Hentenryck, 1989; Jaffar and Lassez, 1987), which is so successful that *constraint logic programming* has been established as a field of its own.

3 Literal Level Theory Reasoning

As mentioned previously, this work is strongly biased towards the literal level theory reasoning. This section describes several general calculi for literal level theory reasoning in a formal way. In order to see the similarities and differences among them, we have decided to define the calculi as instances of some particular common framework. Thus, many notions, such as “derivation”, “theory refuting substitution” and “residue” only have to be defined once. The common framework is called “theory consolution” and it generalizes the non-theory consolution calculus as defined in (Eder, 1991). This calculus was designed as a generalization of both connection calculi and resolution calculi. In (Baumgartner and Furbach, 1992) it has been proved to be useful as framework to define and to compare various other calculi. Thus it is not surprising that its theory-generalization is well-suited for our purpose.

This section is structured as follow. We introduce the consolution calculus along the lines of (Eder, 1991; Baumgartner and Furbach, 1992) and then define a theory version, both in a total and a partial variant. These calculi are then modified to obtain theory resolution, theory model elimination and a theory connection method in order. The partial variant thereof is derived only for the case of the resolution calculus. Since the instantiations of theory consolution in the partial case to model elimination and connection method can be done analogously, we assume that it is sufficient to demonstrate this construction only once.

But before we start the discussion, let us introduce an example theory that will serve us commonly for all calculi throughout this section. It is the first-order representation of some taxonomical knowledge about persons. We are not concerned with how this theory is represented in a real system.

$\mathcal{T}$:

(T-1) $\forall x((mammal(x) \wedge thinker(x)) \rightarrow person(x))$
(T-2) $\forall x(woman(x) \rightarrow person(x))$
(T-3) $\forall x(man(x) \rightarrow person(x))$

Besides this theory some concrete situation is needed. We will often use the following clauses in conjunction with the theory[1]:

$\mathcal{C}$:

(1) $\neg person(fred) \vee \neg man(y)$
(2) $thinker(fred)$
(3) $mammal(x) \vee man(z)$

3.1 Consolution

In this section we will briefly recall the consolution calculus as defined in (Eder, 1991). Since we want the calculus as a starting point for the description of other calculi, we feel the need to modify the original definition as it is given by Eder. For a careful treatment and discussion of these divergences, the reader is referred to (Baumgartner and Furbach, 1992); in the present paper we use consolution in the already modified version.

3.1.1 The Idea of Consolution

Consolution can be seen as a procedure for converting a formula given in one normal form into another normal form: assume we are given a (for simplicity: ground) formula in disjunctive normal form (DNF) and want to prove its validity. This can be done by converting it in a first step into conjunctive normal form (CNF). The second step then uses the fact that a formula in CNF is valid iff every conjunct contains complementary literals. Thus, a simple test for complementary literals in every conjunct suffices to decide the validity of the CNF and also the

[1] This example is a bit contrived, but will serve us well in the formal part below

DNF. Now, with some additional optimizations this is just how consolution works. Consider for example the DNF-formula

$$\underbrace{(P \wedge Q) \vee (\neg P \vee Q)} \vee \neg Q$$

Conversion to CNF can be begun by applying the law of distributivity to the underbraced part, yielding

$$((P \vee \neg P) \wedge (P \vee Q) \wedge (Q \vee \neg P) \wedge (Q \vee Q)) \vee \neg Q$$

This operation is also carried out as a first step in an consolution inference. The subsequent steps deal with the above-mentioned optimizations: first, disjuncts such as $P \vee \neg P$ which contain complementary literals are tautological and thus can be removed. Second, disjuncts may be shortened; for example, $(Q \vee \neg P)$ may be replaced by Q. This corresponds to the "weakening" rule in Gentzen's sequent calculus (see e.g. (Gallier, 1987) for the sequent calculus). However, it may cause incompleteness by throwing away the "wrong" literal, i.e. the literal that contributes to a complementary pair in a later stage. Third, $Q \vee Q$ can be replaced by Q. This rule corresponds to the "contraction" rule in the sequent calculus. It is implicitly present in consolution by means of the set data structure, which collapses multiple occurences of literals into a single one. Similarly, identical conjuncts such as Q in $Q \wedge Q$ can be contracted to a single one. Carrying out these suggested operations results in the formula

$$((P \vee Q) \wedge Q) \vee \neg Q$$

Now it is easy to see that the next step produces the "empty" disjunct, which is a proof for the validity of the given formula.

Consolution is slightly more general than just explained: instead of logical formulas in DNF, consolution works on sets of clauses, where a clause is a set of literals. The semantics of clause sets is then obtained by interpreting the outer commas by "$\vee$" and the inner commas by "$\wedge$". The clause set data structure is more general, since the interpretation of the outer comma and inner comma can be exchanged. In other words, one starts with a CNF instead of a DNF. A derivation of the empty clause can then be interpreted as proof of the unsatisfiablity of the DNF-formula, instead of a proof of the validity of a (logically different) CNF-formula. This duality is not specific to consolution but applies to every calculus with clause sets as data structure. It gives us the freedom to directly relate derivations in e.g. model elimination (which is usually formulated in the refutational setting) and consolution (which was formulated in Eder's theorem in the affirmative setting).

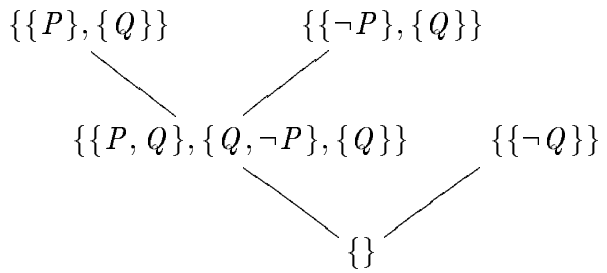
Consolution can also be explained from the background of the connection method (cf. (Bibel, 1987)). Here, clause are called *matrices*, and the method is concerned with proving that every path through this matrix contains two complementary literals, called connections in this framework (a *path* through a matrix is built by selecting exactly one literal from every clause in the matrix).

To illustrate consolution we use an example from (Eder, 1991):

The three clauses $\{P, Q\}$, $\{\neg P, Q\}$ and $\neg Q$ are represented in the connection method as a matrix M:

$$\begin{array}{ccc} P & \neg P & \neg Q \\ Q & Q & \end{array}$$

Thus the possible paths through this matrix are $\{P, \neg P, \neg Q\}$, $\{P, Q, \neg Q\}$, $\{Q, \neg P, \neg Q\}$ and $\{Q, \neg Q\}$. Consolution shares with the connection method the idea of showing that every path contains a connection. Consolution does so by combining partial paths through a matrix to even longer partial paths and thereby ruling out paths containing a connection. The following tree is a proof tree in consolution. The nodes are marked with path sets, e.g. $\{\{P, Q\}, \{Q, \neg P\}, \{Q\}\}$ is a set with three partial paths through the two leftmost clauses in the matrix M. Now, in an inference the cross product of the elements of the parent nodes is built, and paths containing connections are deleted.



The root of this tree is the empty path set, which proves that all paths through M are complementary.

3.1.2 Formal Definition

To introduce consolution formally we need the following definitions.

A *connection* in a set of clauses is a pair of literals which can be made complementary by instantiation. Let $\{C_1, \ldots, C_n\}$ be a clause set. A *path through* $\{C_1, \ldots, C_n\}$ is a finite sequence of literals

$$L_1 \circ \cdots \circ L_n$$

where every L_i is a literal in C_i (for all $i = 1 \ldots n$). Define

$$last(L_1 \circ \cdots \circ L_n) = L_n \qquad \text{if } n \geq 1$$

and

$$front(L_1 \circ \cdots \circ L_n) = L_1 \circ \cdots \circ L_{n-1} \qquad \text{if } n \geq 1$$

A path $p = L_1 \circ \cdots \circ L_n$ *immediately extends to a path q* iff $p \rightsquigarrow q$ iff

$$\exists L \; \exists i (0 \leq i \leq n) : q = L_1 \circ \cdots \circ L_i \circ L \circ L_{i+1} \circ \cdots \circ L_n$$

For the transitive closure we say that a path p *extends to a path q* iff $p \rightsquigarrow^+ q$. The partial order $\leq$ on paths is defined as

$$p \leq q \quad \text{iff} \quad p = q \text{ or else } p \rightsquigarrow^+ q$$

A *path set* is a finite multiset of paths. *Multisets* are like sets, but allow multiple occurrences of identical elements. Formally, a multiset can be introduced as a function N over a certain domain that maps every element of the domain to a natural number. For convenience we will use set-notations. 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 $\{a, a, a, b\}$. As operators for multisets we will use the usual set operators with the obvious intended meaning.

In (Eder, 1991) paths are simply sets and thus the above operations $last$, $front$ and extension are unnecessary or rather their effect can be achieved by the usual set operations. Furthermore, the original calculus computes with sets of paths instead of multisets. In (Baumgartner and Furbach, 1992) we argue that our modified calculus is the appropriate formal base for expressing other calculi. Since the arguments given for the modifications carry over to the theory case, we refer the reader to (Baumgartner and Furbach, 1992) for that discussion. The next definitions are adaptions of the ones in (Eder, 1991) towards our data structures.

If C is a clause $C = \{L_1, \ldots, L_n\}$ then the *path set of C* is given by the path set

$$\mathcal{P}_C = \{L_1, \ldots, L_n\}$$

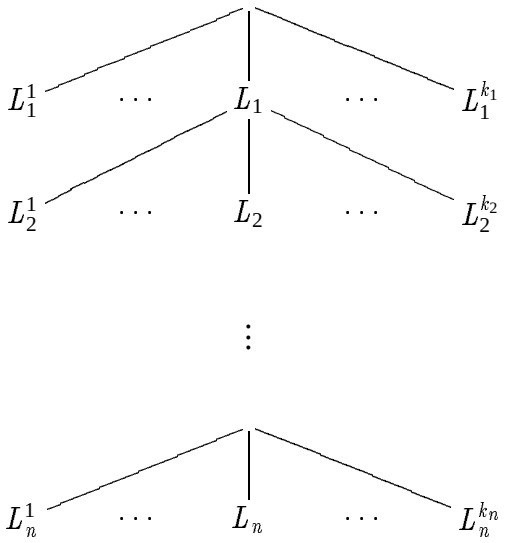
The *product pq* of two paths p and q is the path $p \circ q$. The *product* $\mathcal{P}\mathcal{Q}$ of two path sets $\mathcal{P}$ and $\mathcal{Q}$ is defined as

$$\mathcal{P}\mathcal{Q} = \{pq | p \in \mathcal{P} \text{ and } q \in \mathcal{Q}\}$$

For ease of notation we write $p \cdot \mathcal{P}$ as an abbreviation for $\{p\} \cdot \mathcal{P}$.

In the sequel we are also concerned with trees, whose nodes are labelled with literals, except for the root which remains unlabelled. Such trees can be conveniently represented by path sets by simply taking the sequence of the labels $L_1, \ldots, L_n$ of a branch in the tree as a path $L_1 \circ \cdots \circ L_n$ in the corresponding path set. Then the $last$ operation of above denotes the leaf of a branch. This representation is not into: although every such tree can be represented as a path set, there are path sets that may stem from non-isomorphic trees.

In order to define the model elimination and matrix calculi later in this section, we have to introduce *ferns*, which are trees whose shape is pictorially as follows:



More formally, a *fern F of* $C_1, \ldots, C_n$ *with trunk* $L_1 \circ \cdots \circ L_n$ is the smallest path set F satisfying the following conditions:

1. C_i is a clause containing the literal L_i, for all $i = 1 \ldots n$.
2. $L_1 \circ \cdots \circ L_n \in F$
3. $L_1 \circ \cdots \circ L_{i-1} \circ L \in F$, for all $i = 1 \ldots n$, for all $L \in C_i - \{L_i\}$

The following concepts will be used in the inference rule below:

Definition 3.1 (Spanning MGU) A substitution σ is a *spanning MGU for a path set Q* iff σ is a most general substitution such that every element in $Q\sigma$ contains syntactical complementary literals. □

Definition 3.2 (Shortening of paths) A path set $\mathcal{Q}$ is obtained from a path set $\mathcal{P}$ by *shortening of paths* if there is a surjective mapping $f : \mathcal{P} \to \mathcal{Q}$ such that $f(p) \leq p$ holds for all $p \in \mathcal{P}$. □

Definition 3.3 (Simplification) A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *simplification* iff

A) there exists a spanning MGU σ for some subset $Q \subseteq P$, and

B) $\mathcal{P}^B$ is obtained from $\mathcal{P}\sigma$ by deleting zero or more paths containing complementary literals, and

C) $\mathcal{P}^C$ is obtained from $\mathcal{P}^B$ by shortening of zero or more paths, and

D) $\mathcal{R}$ is obtained from $\mathcal{P}^C$ in the following way: for every path $p \in \mathcal{P}^C$ zero or more, but not all paths are deleted that are equal to p as a set of literals.

□

The motivation for the term "equal as a set of literals" in item D comes from the desire to simulate the behaviour of *sets* in Eder's original consolution. See (Baumgartner and Furbach, 1992) for details.

Definition 3.4 (Consolution inference) (Sequence Consolution) The inference rule *sequence consolution* is defined as follows

$$\frac{\mathcal{P} \quad \mathcal{Q}}{\mathcal{R}}$$

if there exists a variant $\mathcal{Q}\prime$ of $\mathcal{Q}$ which does not have variables in common with $\mathcal{P}$ such that $\mathcal{R}$ is obtained from $\mathcal{P} \cdot \mathcal{Q}\prime$ by simplification. $\mathcal{R}$ is called a *sequence consolvent* of $\mathcal{P}$ and $\mathcal{Q}$.

A *derivation* of a matrix M is a finite sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ of path sets such that the following conditions hold:

1. For all $k = 1, \ldots, n$, the set $\mathcal{P}_k$
   (a) is a path set $\mathcal{P}_C$ of a clause $C \in M$, or
   (b) is a consolvent of $\mathcal{P}_i$ and a new variant of $\mathcal{P}_j$ for some $i, j < k$.
2. $\mathcal{P}_n = \{\}$

In order to express linear calculi, such as *model elimination* a slightly modified definition is necessary. Thus, a *linear derivation* of a matrix M is a finite sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ of path sets such that the following conditions hold:

1. $\mathcal{P}_0$ is a path set $\mathcal{P}_C$ of a clause $C \in M$.
2. For all $k = 1, \ldots, n$, the set $\mathcal{P}_k$ is a consolvent of $\mathcal{P}_{k-1}$ and a new variant of a path set $\mathcal{P}_C$ of a clause $C \in M$.
3. $\mathcal{P}_n = \{\}$

□

We will not give an example of consolution here, because it would be subsumed by the theory consolution examples presented below. Instead we conclude non-theory consolution with the following theorem.

Theorem 3.1 (Soundness and completeness of consolution) *(Eder, 1991) A formula in disjunctive normal form is valid if and only if there is a derivation of its matrix by consolution.*

Together with a theorem from (Baumgartner and Furbach, 1992) which states that every consolution derivation can be stepwisely simulated by a sequence consolution derivation, the completeness of sequence consolution follows.

3.2 Theory Consolution

As motivated in the introduction we take apart the knowledge of the domain (i.e. the *theory*) from the program clauses. Formally, a *theory* is a satisfiable set of universally quantified formulas.

A $\mathcal{T}$*-interpretation* is an interpretation satisfying the theory $\mathcal{T}$. A $\mathcal{T}$-interpretation I $\mathcal{T}$-satisfies a clause set M iff I simultaneously assigns *true* to all ground instances of the clauses in M. $\mathcal{T}$-(un-)satisfiability and $\mathcal{T}$-validity of clause sets are defined on top of this notion as usual.

The restriction to universally quantified theories, shortly universal theories, is necessary because precisely for those theories a Herbrand theorem of the following form holds (Petermann, 1991b).

Theorem 3.2 *A clause set M is $\mathcal{T}$-unsatisfiable if and only if there is a finite set of ground instances of clauses from M which is $\mathcal{T}$-unsatisfiable.*

Similar to the non-theory case a Herbrand theorem of this form is the basis for any completeness proof for a calculus which relies on the co-operation of foreground and background reasoning. The restriction is not serious in principle because every theory may be substituted by an equivalent universal theory. However, equivalent means here equivalence with respect to theory-satisfiability and up to the enrichment of the signature by Skolem functions.

3.2.1 The Interface Between General and Dedicated Reasoner

In the present subsection we define the interface between the foreground reasoner, consolution, and the background reasoner. This interface is constituted by three notions. **Firstly,** we have to generalize the concept of "complementary pair of literals" to the theory case. Unlike in the non-theory case, there is no general syntactic characterization in the theory case. Therefore we will give a semantic pendant which is that of a "theory complementary set of literals". **Secondly**, we have to generalize the notion of "unifier". The task of the background reasoner is to construct from candidates given by the foreground reasoner "theory complementary sets of literals" which play the role of elementary arguments in the course of the refutation. This construction is carried out by instantiating the candidates. We will call the respective substitutions "theory refuters". **Thirdly,** in order to be able to treat partial theory reasoning too we introduce the notion of "theory residue".

Definition 3.5 Let $S = \{L_1, \ldots, L_n\}$ be a literal set. S is called $\mathcal{T}$*-complementary* iff the existentially quantified conjunction $\exists(L_1 \wedge \ldots \wedge L_n)$ is $\mathcal{T}$-unsatisfiable. A $\mathcal{T}$-complementary set is called *minimal $\mathcal{T}$-complementary* iff every true subset is not $\mathcal{T}$-complementary. □

Equivalently to this definition it holds that $\{L_1, \ldots, L_n\}$ is $\mathcal{T}$-complementary iff every ground instance of $L_1 \wedge \ldots \wedge L_n$ is $\mathcal{T}$-unsatisfiable iff the universally quantified disjunction $\forall(\overline{L_1} \vee$

$\ldots \vee \overline{L_n}$) is $\mathcal{T}$-valid.

There is a subtle difference between the $\mathcal{T}$-complementary of a literal set and its $\mathcal{T}$-unsatisfiability, when read as a set of unit clauses, i.e. if the variables were $\forall$-quantified. These notions are the same only for ground sets. Consider, for example, a language with at least two constant symbols a and b and the "empty" theory $\emptyset$. Then $S = \{P(x), \neg P(y)\}$ is, when read as a clause set, $\emptyset$-unsatisfiable, but S is not $\emptyset$-complementary, because the conjunction $\exists x, y(P(x) \wedge \neg P(y))$ is not $\emptyset$-valid (because the interpretation with $I(P(a)) = false$ and $I(P(b)) = true$ is a model). However, when applying the substitution $\sigma = \{x \leftarrow y\}$ to S the resulting set $S\sigma$ is $\emptyset$-complementary.

The importance of "complementary" arises from its application in inference rules, such as resolution, which for soundness reasons have to be built on top of complementarism. Since we deal with theory inference rules, we had to extend the usual notion of "complementarism" to "$\mathcal{T}$-complementarism". As a further example consider the theory $\mathcal{E}$ of equality (section 4). Then $S = \{P(x), y = f(y), \neg P(f(f(a)))\}$ is $\mathcal{E}$-unsatisfiable but not $\mathcal{E}$-complementary. However after application of the substitution $\sigma = \{x \leftarrow a, y \leftarrow a\}$, the set $S\sigma$ is $\mathcal{E}$-complementary. Such substitutions will be called *refuters.* As with non-theory consolution, the theory consolution derivations should be computed at a most general level; this is achieved by *most general refuters.* More formally we define:

Definition 3.6 A substitution σ is a *$\mathcal{T}$-refuter* for S iff $S\sigma$ is $\mathcal{T}$-complementary. Conversely, S is called $\mathcal{T}$-refutable iff a $\mathcal{T}$-refuter exists for it. If $S\sigma$ is minimal $\mathcal{T}$-complementary then S is also called minimal $\mathcal{T}$-refutable by σ. A set of substitutions is a *complete set of $\mathcal{T}$-refuters for* S (or short: $CSR_{\mathcal{T}}(S)$) iff

1. for all $\sigma \in CSR(S)$: σ is a $\mathcal{T}$-refuter for S *(Correctness)*
2. for all $\mathcal{T}$-refuters θ for S:
   there is a $\sigma \in CSR(S)$ and a substitution σ' such that $\theta|var(S) = (\sigma\sigma')|var(S)$
   (Completeness)

□

A "partial" variant of $\mathcal{T}$-refuters is as follows:

Definition 3.7 A pair (σ, R), where σ is a substitution and R is a literal, is a *$\mathcal{T}$-residue of S* iff $S\sigma \cup \{\overline{R}\}$ is minimal $\mathcal{T}$-complementary. □

The prefix "$\mathcal{T}$-" is often omitted in the sequel.

In this context it might be interesting to know that our notion of theory refuter generalizes the notion of *rigid E-unifier* (Gallier et al., 1990) to more general theories than equality (see (Bürckert, 1990a) for a proof). A dual notion, "unifier with respect to $\mathcal{T}$-complementary literal sets", has been studied within an affirmative setting (Petermann, 1991b). Constraint theories, equational theories and simple taxonomic theories have been discussed there as special cases.

To give an example of theory refuters assume that the theory consists solely of the following formula:

$$\mathcal{T} = \{\forall x : man(x) \rightarrow person(x)\}$$

Now consider the literal set

$$S = \{man(x),\ \neg person(father(y))\}$$

Then

$$\sigma = \{x \leftarrow father(y)\}$$

is a $\mathcal{T}$-refuter for S, because the formula

$$S\sigma = \exists y(man(father(y)) \wedge \neg person(father(y)))$$

is $\mathcal{T}$-unsatisfiable. $S\sigma$ is even minimal $\mathcal{T}$-complementary, as any true subset of $S\sigma$ can be ground instantiated to a $\mathcal{T}$-satisfiable set. The substitution $\gamma = \{x \leftarrow father(z)\}$ is not a $\mathcal{T}$-refuter for S, because $\exists y, z(man(father(z)) \wedge \neg person(father(y)))$ is not $\mathcal{T}$-unsatisfiable. This can be seen by replacing, say, z by a and y by b and finding a model.

The semantics of a residue (L, σ) of S is given as follows: L is a logical consequence of $S\sigma$. Operationally, L is a new goal to be proved. For example let $S' = \{P(x), y = f(y)\}$. Then $(\{x \leftarrow y\}, P(f(y)))$ is an $\mathcal{E}$-residue of S', since

$$S'\{x \leftarrow y\} \cup \{\neg P(f(y))\} = \{P(y), y = f(y), \neg P(f(y))\}$$

is minimal $\mathcal{E}$-complementary.

If desired, a *minimality* requirement stating that no refuter is an instance of another can be added to the definition of CSR. However it is not required for correctness or completeness issues; even more it may be advisable to leave minimality away, as there are cases where a complete set of *minimal* refuters may not exist (See (Fages and Huet, 1986) for a proof in the context of theory unifiers).

3.2.2 Formal Definition of Theory Consolution

Being equipped with the definition of the consolution calculus, a generalization to a *theory consolution calculus* is fairly straightforward now.

At first the notion of *product* has to be generalized: the product $p_1 p_2 \ldots p_n$ of n paths $p_1, p_2, \ldots, p_n$ is the path $p_1 \circ p_2 \circ \cdots \circ p_n$. The product $\mathcal{P}_1\mathcal{P}_2 \ldots \mathcal{P}_n$ of n path sets $\mathcal{P}_1, \mathcal{P}_2, \ldots, P_n$ is defined as

$$\mathcal{P}_1\mathcal{P}_2 \ldots \mathcal{P}_n = \{p_1 p_2 \ldots p_n | p_i \in \mathcal{P}_i \text{ for all } i = 1 \ldots n\}$$

Definition 3.8 (Spanning $\mathcal{T}$-refuter) A substitution σ is a *spanning $\mathcal{T}$-refuter for a path set* $\mathcal{Q}$ iff σ is a $\mathcal{T}$-refuter for every $q \in \mathcal{Q}$ when q is read as a set of literals. □

Simplification is generalized in the following way:

Definition 3.9 (Total theory simplification) A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *total theory simplification* iff

A) the path set $\mathcal{P}^A$ is obtained from $\mathcal{P}$ by application of some substitution σ to some $\mathcal{Q} \subseteq \mathcal{P}$.

B) the path set $\mathcal{P}^B$ is obtained from $\mathcal{P}^A$ by deleting zero or more paths containing $\mathcal{T}$-complementary literals.

C) $\mathcal{P}^C$ is obtained from $\mathcal{P}^B$ by shortening of zero or more paths, and

D) $\mathcal{R}$ is obtained from $\mathcal{P}^C$ in the following way: for every path $p \in \mathcal{P}^C$ zero or more, but not all paths are deleted that are equal to p as a set of literals.

□

The substitution σ applied in A) need not necessarily be a $\mathcal{T}$-refuter or to be a most general substitution. This enables e.g. application of a factorisation substitution as in resolution.

Deletion of duplicate paths modulo ordering of literals in D) allows to simulate the set data structure of paths and of path sets in consolution.

Now the inference rule can be defined. It generalizes from 2 to n path sets in the premise:

Definition 3.10 (Theory consolution inference) The *inference rule*:

$$\frac{\mathcal{P}_1 \quad \ldots \quad \mathcal{P}_n}{\mathcal{R}}$$

if the $\mathcal{P}_i$ are pairwise disjoint path sets, and $\mathcal{R}$ is obtained from the product $\mathcal{P}_1 \ldots \mathcal{P}_n$ by simplification. We say then that the path set $\mathcal{R}$ is a *theory consolvent* of the path sets $\mathcal{P}_1, \ldots, \mathcal{P}_n$. □

Next the notion of *derivation* has to be adapted.

Definition 3.11 A *theory derivation* of a matrix M is a finite sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ of path sets such that the following conditions hold:

1. For all $k = 1, \ldots, n$, the set $\mathcal{P}_k$
   (a) is a path set $\mathcal{P}_C$ of a clause $C \in M$, or
   (b) is a theory consolvent of new variants of path sets $\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k}$ where $i_1, \ldots i_k < k$.

2. $\mathcal{P}_n = \{\}$

A *linear theory derivation* of a matrix M is a finite sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ of path sets such that the following conditions hold:

1. $\mathcal{P}_0$ is a path set of a clause $C \in M$.
2. For all $k = 1, \ldots, n$, the set $\mathcal{P}_k$ is a theory consolvent of $\mathcal{P}_{k-1}$ and new variants of path sets $\mathcal{P}_{C_1}, \ldots, \mathcal{P}_{C_n}$ where the C_i (for $i = 1 \ldots n$) are clauses in M.
3. $\mathcal{P}_n = \{\}$

□

It should have become clear from the preceding definitions that consolution has some open parameters (the commitment to the derivation strategy, and the simplification). Thus one cannot speak of *the* consolution calculus. Instead consolution should be interpreted as a method that can be instantiated to several calculi. This will be done in the following sections in a goal-oriented way.

3.2.3 Example

This example is intended to trace through the several intermediate steps in a consolution inference. The parameter settings are chosen "at random".

Consider this excerpt from the example in the beginning of this section, consisting of the theory $\mathcal{T}$ and the clause set $\mathcal{C}$:

$\mathcal{T}$: (T-1) $\forall x((mammal(x) \wedge thinker(x)) \rightarrow person(x))$
(T-3) $\forall x(man(x) \rightarrow person(x))$

$\mathcal{C}$: (1) $\{\neg person(fred), \neg man(y)\}$
(2) $\{thinker(fred)\}$
(3) $\{mammal(x), man(z)\}$

We will show a total inference step, using clauses (1), (2) and (3). Then the following inference can be made:[2]

$$\frac{\begin{array}{l}\neg person(fred), \neg man(y) \\ thinker(fred) \\ mammal(x), man(z)\end{array}}{\neg man(y) \circ thinker(fred)}$$

[2] In this presentation style the input path sets are written in separate rows; the set braces are left away.

Let us have a detailed look at how this result can be achieved: in a first step the product of the path sets of clauses (1), (2) and (3) is computed:

$$\mathcal{Q} = \mathcal{P}_1\mathcal{P}_2\mathcal{P}_3 = \{\ \underline{\neg person(fred) \circ thinker(fred) \circ mammal(x)},\\ \underline{\neg person(fred) \circ thinker(fred) \circ man(z)},\\ \neg man(y) \circ thinker(fred) \circ mammal(x),\\ \neg man(y) \circ thinker(fred) \circ man(z)\}$$

Then $\mathcal{Q}$ is simplified:

A) Due to (T-1), a $\mathcal{T}$-refuter for the first underlined path is $\{x \leftarrow fred\}$, and due to (T-3), $\{z \leftarrow fred\}$ is a is a $\mathcal{T}$-refuter for the second path. In total we build $\sigma = \{x \leftarrow fred, z \leftarrow fred\}$ and apply it to $\mathcal{Q}$.

B) $\mathcal{R}^B$ is obtained from $\mathcal{Q}\sigma$ by deletion of the underlined paths.

$$\mathcal{R}^B = \{\ \underline{\neg man(y) \circ thinker(fred)} \circ mammal(fred),\\ \underline{\neg man(y) \circ thinker(fred)} \circ man(fred)\}$$

C) Shorten to the underlined paths in $\mathcal{R}^B$ and obtain:

$$\mathcal{R}^C = \{\ \neg man(y) \circ thinker(fred),\\ \neg man(y) \circ thinker(fred)\}$$

D) Obtain the above consolvent from $\mathcal{R}^C$ by deleting one of the duplicate occurrences.

3.3 Partial Theory Consolution

The calculus from above does not compute with residues. A partial variant can be defined by replacing "application of a substitution" in total theory consolution with "appending a residue to a path".

Definition 3.12 (Extension with a $\mathcal{T}$-residue) A path $p\sigma \circ Res$ is obtained from a path p by *extension with a $\mathcal{T}$-residue* (σ, Res) iff (σ, Res) is a $\mathcal{T}$-residue of the literal set of p. □

Definition 3.13 (Partial theory consolution) The inference rule *partial theory consolution* is defined in the same way as total theory resolution, except that in simplification step A) is changed in the following way:

A) the path set $\mathcal{P}^A$ is obtained from $\mathcal{P}$ by replacing a single occurence of a path $p \in \mathcal{P}$ by $p\sigma \circ Res$, which is obtained from p by extension with a $\mathcal{T}$-residue.

The *partial theory consolution* calculus consists of the inference rules *total theory consolution* and *partial theory consolution* □

In (Stickel, 1985) residues are introduce on the ground level in a dual way and more generally as a *set* of literals instead of a single literal: a set $\{L_1, \ldots, L_k\}$ is a $\mathcal{T}$-residue of a formula M iff $M \wedge L_1 \wedge \ldots \wedge L_k$ is $\mathcal{T}$-complementary. Equivalently, this means that the disjunction $\overline{L_1} \vee \ldots \vee \overline{L_k}$ is a logical $\mathcal{T}$-consequence of M. Since this generalisation is straightforward we have omitted it for the sake of simplicity.

For an example consider again the same theory and clauses (1) and (3) as above in total theory consolution. Again, the open parameters of consolution are set at random. Selecting from clauses (1) and (3) the literals $\neg person(fred)$ and $mammal(x)$ (respectively) the following inference can be made:

$$\frac{\begin{array}{l}\neg person(fred), \neg man(y) \\ mammal(x), man(z)\end{array}}{man(z), \neg man(y), \neg thinker(fred)}$$

Let us have a detailed look at how this result can be achieved: in a first step the product of the path sets of clauses (1) and (3) is computed:

$$\begin{array}{rl}\mathcal{Q} = \mathcal{P}_1\mathcal{P}_3 = \{ & \underline{\neg person(fred) \circ mammal(x)}, \\ & \neg person(fred) \circ man(z), \\ & \neg man(y) \circ mammal(x), \\ & \neg man(y) \circ man(z)\}\end{array}$$

Then $\mathcal{Q}$ is simplified:

A) The above underlined path is selected for extension with a residue (σ, R); a residue for this path is $(\{x \leftarrow fred\}, \neg thinker(fred))$. Applying the substitution $x \leftarrow fred$ and extending the selected path with $\neg thinker(fred)$ yields:

$$\begin{array}{rl}\mathcal{R}^A = \{ & \neg person(fred) \circ mammal(fred) \circ \underline{\neg thinker(fred)}, \\ & \neg person(fred) \circ \underline{man(z)}, \\ & \underline{\neg man(y)} \circ mammal(fred), \\ & \underline{\neg man(y)} \circ man(z)\}\end{array}$$

B) $\mathcal{R}^B = \mathcal{R}^A$, i.e. no path is deleted.

C) Shorten to the underlined paths in $\mathcal{R}^B$ and obtain:

$$\begin{array}{rl}\mathcal{R}^C = \{ & \neg thinker(fred), \\ & man(z), \\ & \neg man(y), \\ & \neg man(y)\}\end{array}$$

D) Obtain the partial theory consolvent as exposed above from $\mathcal{R}^C$ by deleting one of the duplicate occurrences of $\neg man(y)$

Now we have defined the formal framework that can be instantiated to the various theory calculi. Let us quickly summarize what "parameters" have to be instantiated in the theory consolution calculus:

- Concerning *derivation*: The function f and the commitment to a non-linear or linear strategy.
- Concerning *simplification*: The parameters of step A (application of a substitution or extension with a residue), step B (deleting complementary paths), step C (shortening of paths) and step D (deleting occurrences).

3.4 Theory Resolution

We assume the reader to be familiar with the resolution calculus and its terminology (see e.g. (Loveland, 1978; Chang and Lee, 1973) for introductory textbooks). First we will explain the original version of theory resolution (Stickel, 1985). Then, after the formal definition based on theory consolution is given, completeness will be proved.

3.4.1 Informal Explanation

In (Stickel, 1985) several variants of a theory resolution calculus are defined. One of them is called *narrow total theory resolution*. We will demonstrate a first-order version of this rule. It takes n clauses as inputs and selects one literal from each of the n clauses such that these literals can be instantiated to a theory-complementary set. Then the resolvent is built as in ordinary resolution by applying the substitution and collecting the rest of the clauses. Thus narrow theory resolution is a straightforward "semantical" generalization of the ordinary resolution rule.

Example. This is an excerpt from the example in the introduction:

$\mathcal{T}$: (T-1) $\forall x((mammal(x) \wedge thinker(x)) \rightarrow person(x))$
(T-3) $\forall x(man(x) \rightarrow person(x))$

$\mathcal{C}$: (1) $\{\neg person(fred), \neg man(y)\}$
(2) $\{thinker(fred)\}$
(3) $\{mammal(x), man(z)\}$

We will show an inference step, using from clauses (1), (2) and (3) the literals $\neg person(fred)$, $thinker(fred)$ and $mammal(x)$ (respectively) as a $\mathcal{T}$-refutable literal set. It can easily be seen that $\{X \leftarrow fred\}$ is a $\mathcal{T}$-refuter for this set (join the clause form of (T-1) to this set and find

a resolution refutation). Hence the following resolvent can be built (the selected literals are underlined) :

$$\frac{\begin{array}{l}\{\underline{\neg person(fred)}, \neg man(y)\} \\ \{\underline{thinker(fred)}\} \\ \{\underline{mammal(x)}, man(z)\}\end{array}}{\{\neg man(y), man(z)\}} \quad \text{with refuter } \sigma = \{x \leftarrow \mathit{fred}\}$$

It should be noted that sometimes *factorization* has to be carried out for completeness reasons. This can either be done by an extra inference rule, or by incorporating factorization into the resolution inference rule. Since factorization is not central for the idea of theory reasoning, we will skip over it for the moment.

The *partial* variant of theory resolution is similar to the total theory resolution rule, except that the resolvent additionally includes a residue in the resolvent. Stickel's ground calculus allows as residues *disjunctions* $B_1 \vee \ldots \vee B_k$ of literals. The disjunction $B = B_1 \vee \ldots \vee B_k$ is a residue of a conjunction $A = A_1 \wedge \ldots \wedge A_m$ if A implies B. Here we will restrict ourselves to the important special case where the residue is a single literal B_1. A generalization would be straightforward if desired.

Example. Using the example of above, we have that

$$(\{x \leftarrow \mathit{fred}\}, \neg thinker(fred))$$

is a residue of

$$\{\neg person(fred), mammal(x)\}$$

which is built from literals in clauses (1) and (3). Thus we arrive at the following partial theory resolution inference:

$$\frac{\begin{array}{l}\{\underline{\neg person(fred)}, \neg man(y)\} \\ \{\underline{mammal(x)}, man(z)\}\end{array}}{\{\neg thinker(fred) \neg man(y), man(z)\}} \quad \text{with residue } (\{x \leftarrow \mathit{fred}\}, \neg \mathit{thinker}(\mathit{fred}))$$

Now resolving this resolvent in an ordinary resolution step against (2) we could arrive at the same clause as in the above total theory step. As a general property one could say that the purpose of partial theory reasoning is to approximate a total theory reasoning step in a sequence of partial steps, followed by one single total step.

A problem of Stickel's original definition is that it allows one to derive many redundant clauses. For example, in the theory of strict orderings from $a < b$ and $b < c$ one might infer the residue $a < c$, but also residues like $\neg a < x \vee x < c$, $\neg a < x \vee \neg x < x' \vee x' < c$, ... Thus in practice it is inevitable to seek for suitable restrictions. See again (Stickel, 1985) for a more detailed discussion how redundant residues can be omitted.

So far we have discussed *narrow* theory resolution. In (Stickel, 1985) a more general variant

called *wide* theory resolution is defined. It differs from narrow theory resolution in that no longer are only single literals of the clauses considered in inferences, but instead sub*clauses* are passed to the theory reasoner. This however complicates the theory reasoner, since it must operate on clause sets instead of literal sets. Similar to the narrow case, a total and a partial variant can be defined.

3.4.2 Formal Definition

The total theory resolution inference rule can be defined as an instance of theory consolution. For the total version we need two inference rules: *total theory resolution* and *factorisation*. For the former n clauses have to be chosen and they have to be put into a format suitable for the theory consolution operation. The simplification operation in theory consolution has to be defined in such a way that the consolvent corresponds precisely to the theory resolvent in theory resolution. More formally we arrive at the following inference rule:

Definition 3.14 (Total theory resolution) The inference rule *total theory resolution* is defined as follows:

$$\frac{\begin{array}{c}\mathcal{P}_{\{L_1\}\cup R_1}\\ \vdots \\ \mathcal{P}_{\{L_n\}\cup R_n}\end{array}}{\mathcal{R}}$$

where

1. $\{L_i\} \cup R_i$ are pairwise variable disjoint clauses (for $i = 1 \ldots n,\ n \geq 1$).
2. $\{L_1, \ldots, L_n\}$ is $\mathcal{T}$-refutable.
3. The simplification of the product

   $$\mathcal{Q} = \mathcal{P}_{\{L_1\}\cup R_1} \cdot \quad \ldots \quad \cdot \mathcal{P}_{\{L_n\}\cup R_n}$$

   is done in the following way:

   A) $Q\sigma$ is obtained from $\mathcal{Q}$ by application of the $\mathcal{T}$-refuter σ of $\{L_1, \ldots, L_n\}$ (which exists by 2.)

   B) Delete every path $(L_1 \circ \cdots \circ L_n)\sigma$ from $\mathcal{Q}\sigma$ and obtain $\mathcal{R}^B$.

   C) $\mathcal{R}^C$ is obtained from $\mathcal{R}^B$ by shortening every path in $\mathcal{R}^B$ according to

   $$f(K_1 \circ \cdots \circ K_m) = K_i$$

   where $i \in \{1, \ldots, m\}$ such that $K_i \notin \{L_1, \ldots, L_n\}\sigma$

   D) $\mathcal{R}$ is obtained from $\mathcal{R}^C$ by deleting for every path all its duplicate occurences.

□

In simplification step C) the path sets computed so far are shortened to path sets of length 1. Thus resolution "forgets" information that might possibly be useful later. Indeed, matrix oriented calculi keep this information and make use of it. Below we will proof that this definition is indeed correct wrt. the standard theory resolution inference. More precisely we will show that

$$\mathcal{R} = \mathcal{P}_{(R_1\sigma - \{L_1\sigma\}) \cup \ldots \cup (R_n\sigma - \{L_n\sigma\})}$$

As with ordinary resolution, theory resolution requires for completeness reasons *factoring*, i.e. sometimes in a refutation a parent clause C must be instantiated to $C\sigma$ where σ is a most general unifier of some literals in C. This can be achieved in the consolution framework by instantiating the theory consolution inference in the following way:

Definition 3.15 (Factorisation) The inference rule *Factorisation* is defined as follows:

$$\frac{\mathcal{P}_C}{\mathcal{R}}$$

where

1. σ is a most general unifier for some $\{L_1, \ldots, L_n\} \subseteq C$.
2. The simplification of $\mathcal{P}_C$ is done in the following way:
   A) Obtain $\mathcal{P}^A = \mathcal{P}_C\sigma$.
   B) No path is deleted, i.e. $\mathcal{P}^A = \mathcal{P}^B$.
   C) No shortening is applied, i.e. $f(p) = p$ for every path. This yields $\mathcal{P}^C = \mathcal{P}^B$.
   D) $\mathcal{R}$ is obtained from $\mathcal{R}^C$ by deleting for every path all its duplicate occurences.

□

Building on these inference rules we define:

Definition 3.16 A *total theory resolution refutation* is defined as a *theory consolution derivation* where every theory consolution inference either is a total theory resolution inference or a factorisation inference. □

Example. Consider the theory and clauses as used above in the informal presentation of theory resolution. We will redo the above inference step. So we consider again from clauses

(1), (2) and (3) the literals $\neg person(fred)$, $thinker(fred)$ and $mammal(x)$ (respectively) as $\mathcal{T}$-complementary literal set. Then the following inference can be made:

$$\frac{\begin{array}{l}\neg person(fred), \neg man(y)\\ thinker(fred)\\ mammal(x), man(z)\end{array}}{man(z), \neg man(y)}$$

Observe that the resulting path set encodes the same clause as in the example of the above informal presentation. Let us have a detailed look at how this result can be achieved: in a first step the product of the path sets is computed:

$$\begin{array}{rl}\mathcal{Q} = \mathcal{P}_1\mathcal{P}_2\mathcal{P}_3 = \{ & \underline{\neg person(fred) \circ thinker(fred) \circ mammal(x)},\\ & \underline{\neg person(fred) \circ thinker(fred) \circ man(z)},\\ & \neg man(y) \circ thinker(fred) \circ mammal(x),\\ & \neg man(y) \circ thinker(fred) \circ man(z)\}\end{array}$$

Then $\mathcal{Q}$ is simplified:

A) The $\mathcal{T}$-refuter $\{x \leftarrow fred\}$ for the literals in the underlined path

$$\neg person(fred) \circ thinker(fred) \circ mammal(x) \qquad (*)$$

is applied to $\mathcal{Q}$.

B) $\mathcal{R}^B$ is obtained from $\mathcal{Q}\sigma$ by deletion of $(*)$

$$\begin{array}{rl}\mathcal{R}^B = \{ & \neg person(fred) \circ thinker(fred) \circ \underline{man(z)}, \quad \}\\ & \underline{\neg man(y)} \circ thinker(fred) \circ mammal(fred),\\ & \underline{\neg man(y)} \circ thinker(fred) \circ man(z)\}\end{array}$$

C) Shorten to the underlined paths in $\mathcal{R}^B$ and obtain, modulo multiplicity of paths:

$$\begin{array}{rl}\mathcal{R}^C = \{ & man(z),\\ & \neg man(y),\\ & \underline{\neg man(y)}\}\end{array}$$

It can be verified that this shortening is indeed achieved by the specification of the definition.

D) Obtain R from $\mathcal{R}^C$ by deleting the underlined path.

The partial variant of theory resolution is obtained in much the same way:

Definition 3.17 (Partial theory resolution) The inference rule *partial theory resolution* is defined as follows:

$$\frac{\begin{array}{c}\mathcal{P}_{R_1\cup\{L_1\}}\\ \vdots\\ \mathcal{P}_{R_n\cup\{L_n\}}\end{array}}{\mathcal{R}}$$

where

1. $\{L_i\} \cup R_i$ are pairwise variable disjoint clauses (for $i = 1 \ldots n$).
2. There is a most general residue (σ, Res) for $\{L_1, \ldots, L_n\}$.
3. The simplification of the product

   $$\mathcal{Q} = (\mathcal{P}_{\{L_1\}\cup R_1}) \cdot \quad \ldots \quad \cdot (\mathcal{P}_{\{L_n\}\cup R_n})$$

   is done in the following way:

   A) $\mathcal{Q}$ has the form

   $$\mathcal{Q} = \{L_1 \circ \cdots \circ L_n\} \cup \mathcal{P}'$$

   Obtain

   $$\{(L_1 \circ \cdots \circ L_n)\sigma \circ Res\} \cup \mathcal{P}\sigma'$$

   from $\mathcal{Q}$ by extension with a $\mathcal{T}$-residue (σ, Res) for $L_1 \circ \cdots \circ L_n$ (which exists by 2).

   B) Let $\mathcal{R}^B = \mathcal{R}^A$, i.e. do not delete any complementary path.

   C) $\mathcal{R}^C$ is obtained from $\mathcal{R}^B$ by shortening every path in $\mathcal{R}^B$ according to

   $$f(K_1 \circ \cdots \circ K_m) = K_i$$

   where $i \in \{1, \ldots, m\}$ such that $K_i \notin \{L_1, \ldots, L_n\}\sigma$

   D) $\mathcal{R}$ is obtained from $\mathcal{R}^C$ by deleting for every path all its duplicate occurences.

A *partial theory resolution refutation* is defined as a *theory consolution derivation* where every theory consolution inference is an "total theory resolution" or "partial theory resolution" □

By shortening the path $(L_1 \circ \cdots \circ L_n \circ)\sigma \circ Res$ to the path Res (due to simplification step C) at leat one occurence of such a path must be contained in the result) and shortening the other paths to $\mathcal{P}_{R_1\cup\ldots\cup R_n}\sigma$ as in total theory resolution, and finally "collapsing" all multiple occurrences of paths into one occurrence, we obtain precisely the path set corresponding to the resolvent of the

traditional inference rule.

Example. Consider again the same theory and clauses as above. We will show a partial theory resolution step, using from clauses (1) and (3) the literals $\neg person(fred)$ and $mammal(x)$ (respectively) to derive the residue $\neg thinker(fred)$. Then the following inference can be made:

$$\frac{\begin{array}{l}\neg person(fred), \neg man(y)\\ mammal(x), man(z)\end{array}}{\neg thinker(fred), man(z), \neg man(y)}$$

Observe again that the resulting path set encodes the same clause as in the example of the above informal presentation. Let us have a detailed look at how this result can be achieved: in a first step the product of the path sets is computed:

$$\begin{array}{rl}\mathcal{Q} = \{ & \underline{\neg person(fred) \circ mammal(x)},\\ & \neg person(fred) \circ man(z),\\ & \neg man(y) \circ mammal(x),\\ & \neg man(y) \circ man(z)\}\end{array}$$

Then $\mathcal{Q}$ is simplified:

A) The above underlined path

$$p = \neg person(fred) \circ mammal(x)$$

is selected for finding a most general residue; a most general residue for p is

$$Res = (\{X \leftarrow fred\}, \neg thinker(fred))$$

So we obtain by replacement of $p\sigma$ with

$$q = \neg person(fred) \circ mammal(fred) \circ \neg thinker(fred)$$

the path set

$$\begin{array}{rl}\mathcal{R}^A = \{ & \neg person(fred) \circ mammal(fred) \circ \underline{\neg thinker(fred)},\\ & \neg person(fred) \circ \underline{man(z)},\\ & \underline{\neg man(y)} \circ mammal(fred),\\ & \underline{\neg man(y)} \circ man(z)\}\end{array}$$

B) Let $\mathcal{R}^B = \mathcal{R}^A$, i.e. do not delete any complementary path.

C) The result of the inference rule must be the path set corresponding to the clause $\neg person(fred) \vee man(z) \vee \neg man(y)$. This path set, modulo multiplicity of paths, is obtained from $\mathcal{R}^B$ by shortening the paths such that the underlined subpaths in B) are kept. This results in the path set

$$\mathcal{R}^C = \{\neg thinker(fred), man(z), man(z), \neg man(y)\}$$

D) delete one occurrence of $man(z)$ in $\mathcal{R}^C$ to obtain $\mathcal{R}$.

3.4.3 Completeness

Completeness of the consolution-style total theory resolution calculus is obtained by simulating another theory resolution resolution calculus known to be complete. This strategy was also used for the non-theory case in (Eder, 1991). As a preliminary we cite from (Baumgartner, 1992b) a theory resolution calculus. The calculus defined there takes advantage of ordering restrictions which we will omit below. This is legal for our purposes, since an ordered refutation always is an unordered one as well. Thus, the completeness result for the ordered calculus as developed in (Baumgartner, 1992b) holds as well for the unordered case.

Definition 3.18 ((Baumgartner, 1992b), Clausal Theory Resolution Calculus) The inference rules of the *clausal theory resolution (CTR) calculus* are defined as follows:

Clausal factoring:

$$\frac{C}{C\sigma} \quad \begin{cases} \text{if } \sigma \text{ is a most general (syntactical) unifier} \\ \text{for some } \{L_1, \ldots, L_n\} \subseteq C \end{cases}$$

Clausal theory resolution:

$$\frac{C_1 \quad \ldots \quad C_n}{(C_1\sigma - \{L_1\sigma\}) \cup \ldots \cup (C_n\sigma - \{L_n\sigma\})} \quad \begin{cases} \text{if } \sigma \text{ is a } \mathcal{T}\text{-refuter for } \{L_1, \ldots, L_n\} \text{ for} \\ \text{some } L_1 \in C_1, \ldots, L_n \in C_n \end{cases}$$

In these inference rules, the L_i's are called the *selected literals*. Let M be a clause set. A CTR-derivation of C_n from M is a sequence $C_1, \ldots, C_n$ where each $C_i \in M$ or is obtained by an application of the above inference rules to k variable disjoint copies of clauses $C_{j_1} \ldots C_{j_k}$ where $j_1 < i, \ldots, j_k < i$. A *refutation of M* is a derivation of the empty clause. □

This calculus is complete:

Theorem 3.3 ((Baumgartner, 1992b), Completeness of clausal theory resolution) *Let $\mathcal{T}$ be a theory and M be a $\mathcal{T}$-unsatisfiable clause set. Then there exists an CTR-refutation of M.*

Building on this, we can prove the consolution-style version complete:

Theorem 3.4 (Completeness of Total Theory Resolution) *Let $\mathcal{T}$ be a theory and M be a $\mathcal{T}$-unsatisfiable clause set. Then there exists a total theory resolution refutation of M.*

Proof. The proof is in analogy to the corresponding theorem for the non-theory case in (Eder, 1991). By the previous theorem it suffices to show that every clausal theory resolution refutation can be simulated by step within total theory resolution.

Since the definitions of derivation are the same in total theory resolution and clausal theory resolution the only non-trivial task is to show how to simulate the inference rules:

1. Clausal theory resolution: Let

$$Res = (R_1\sigma - \{L_1\sigma\}) \cup \ldots \cup (R_n\sigma - \{L_n\sigma\})$$

be the result of a clausal theory resolution step with selected literal $L_1, \ldots, L_n$ and refuter σ. We have to show that Res is the same set of literals as the total theory consolvent $\mathcal{R}$ of $\mathcal{P}_{R_1 \cup \{L_1\}}, \ldots, \mathcal{P}_{R_n \cup \{L_n\}}$, i.e. we have to show that $Res = \mathcal{R}$. Since $\mathcal{R}^C$ is the same as $\mathcal{R}$ modulo multiplicity of occurences of same literals this is equivalent to prove

$$L \in Res \text{ iff } L \in \mathcal{R}^C \tag{1}$$

By the given clausal resolution step we know that $\{L_1, \ldots, L_n\}$ is $\mathcal{T}$-refutable by σ. Hence condition 2) in the definition of total theory resolution is satisfied. Then in simplification, the product Q is built and σ is applied in step A). The resulting multiset $Q\sigma$ can be written as

$$Q\sigma = \{(K_1 \circ \cdots \circ K_n)\sigma \mid K_i \in \{L_i\} \cup R_i\} \tag{2}$$

In Step B) $\mathcal{T}$-complementary paths are deleted which yields

$$\mathcal{R}^B = \{p \mid p \in Q\sigma, p \neq (L_1 \circ \cdots L_n)\sigma\}$$

Then in step C) we obtain

$$\mathcal{R}^C = \{f(p) \mid p \in Q\sigma, p \neq (L_1 \circ \cdots L_n)\sigma\}$$

Note that by definition, f has to select a literal that is unequal to $L_1\sigma, \ldots, L_n\sigma$ but paths containing only these literals are deleted. Thus f is well-defined.

Now we can prove (1):

$$\begin{array}{lll} L \in Res & \text{iff} & \exists i : L \in R_i\sigma - \{L_i\sigma\} \\ & \text{iff} & (\text{since } R_i\sigma - \{L_i\sigma\} \neq \{\}) \; L \neq L_i\sigma \\ & \text{iff} & (\text{by def. of } \mathcal{R}^B) \; p_i = (L_1 \circ \cdots \circ L_{i-1})\sigma \circ L \circ (L_{i+1} \circ \cdots \circ L_n)\sigma \in \mathcal{R}^B \\ & \text{iff} & (\text{by def. of } f) \; f(p_i) = L \\ & \text{iff} & (\text{by def. of } \mathcal{R}^C) \; L \in \mathcal{R}^C \end{array}$$

2. Clausal factoring: Let $C\sigma$ be the result of a clausal factorisation step applied to C. We have to show that $C\sigma$ is the same set of literals $\mathcal{R}$ as obtained by factorisation of $\mathcal{P}_C$, i.e. we have to show that $C\sigma = \mathcal{R}$. Since $\mathcal{R}^C = \mathcal{R}^B = \mathcal{R}^A = \mathcal{P}_C\sigma$ is the same as $\mathcal{R}$ modulo multiplicity of occurences of same literals this is equivalent to prove

$$L \in C\sigma \text{ iff } L \in \mathcal{P}_C\sigma \tag{3}$$

This however is a trivial consequence of

$$L \in C \text{ iff } L \in \mathcal{P}_C$$

□

3.4.4 Constrained Resolution

In (Bürckert, 1990a) the resolution calculus is extended with a framework for *constraints*, i.e. semantic restrictions for quantified variables. The goal of this section is to show the relation between this resolution principle and theory resolution as defined above.

In order to sketch the idea of the constrained resolution calculus (see (Bürckert, 1990b; Bürckert, 1990a; Bürckert, 1992) for details) consider a clause like

$$\forall x(\neg person(x) \vee sleep(x) \vee eat(x))$$

stating that "persons sleep or eat". In the spirit of constrained resolution this should be expressed as a formula (x sleeps or eats) over a restricted domain (x is a person), i.e. the clause:

$$\forall(x : person(x))(sleep(x) \vee eat(x))$$

As in theory resolution, constrained resolution distinguishes between foreground reasoning and a background reasoning. The background reasoning is formalized by a *restricted quantification system (RQS)*, which consists of a signature Δ with equality, a theory over Δ (called the *restriction theory* $\mathcal{R}$), and a set of open Δ-formulae (called the *restrictions*). The restriction theory might be given by a class of Δ-structures or by some Δ-axioms. The restrictions, such as $person(x)$ in the above example, play the role of an interface between foreground and background reasoner.

The foreground language is defined by RQ-formulas over a signature Σ, which consist of the usual first order formulas built from Σ and of formulas $\forall(x : R)F$ and $\exists(x : R)F$ where F is a RQ-formula and R is a restriction. The signatures Σ and Δ must share the same set of function symbols (they can always be extended in such a way) but must be disjoint in the predicate symbols. Then the model-theoretic semantics of the RQ-formulas is confined to those models that also obey the restriction theory $\mathcal{R}$.

Let us continue the above example. Here an RQS named **Tax** might consist of the following ingredients:

Tax:	Δ_T:	Predicate symbols: $man, person$, function symbols: joe
	Axioms for $\mathcal{R}_T$:	$\{man(joe), \forall x(man(x) \rightarrow person(x))\}$
	Restrictions:	$\{person(x)\}$

In order to complete the example we need a signature Σ and a set of RQ-formulas. Building on the above clause and leaving Σ implicit, we give the following set of RQ-formulas to a foreground reasoner:

(1') $\forall(x : person(x))(sleep(x) \vee eat(x))$

(2') $\neg sleep(joe)$

As an alternative notation for RQ-clauses one usually leaves quantifiers implicit and separates the restrictions with '$//$' from the rest of the clause:

$\mathcal{C}$:

(1) $sleep(x) \vee eat(x) \quad // \quad person(x)$
(2) $\neg sleep(joe) \quad // \quad true$

The semantics of an RQ-clause $P \ // \ Q$ can alternatively be given by relativizing to the implication $Q \rightarrow P$.

Two RQ-clauses can be resolved by the RQ-resolution rule, which can be described as follows: select from disjoint variants of the clauses two literals $P(s_1, \ldots, s_n)$ and $\neg P(t_1, \ldots, t_n)$ (respectively), and build the resolvent by joining the rest of the clauses; the restriction of the resolvent is obtained by inheriting the restrictions of the parent clauses together with the confrontations $s_i = t_i$ of the arguments of the selected literals. As an optimization it suffices to consider only those resolvents whose restriction is $\mathcal{R}$-satisfiable. Thus we can build from (1) and (2) by selecting the $sleep$-literals the resolvent

(3) $eat(x) \quad // \quad person(x) \wedge x = joe$

As a difference to ordinary resolution it does not suffice to derive a *single* empty clause in order to obtain a refutation. A refutation has been found if for each model $\mathcal{A}$ in the constrained theory $\mathcal{R}$ there is an empty RQ-clause $\Box \quad // \quad R$ such that R is satisfied by $\mathcal{A}$, i.e. $\mathcal{A} \models R$. In other words, if it can be shown that every model in $\mathcal{R}$ implies a contradiction (the empty clause) then the RQ-clause set is $\mathcal{R}$-unsatisfiable.

Constrained resolution derivations can roughly be mapped to theory resolution derivations as follows: for every set S of selected literals in a theory resolution derivation in the corresponding constrained resolution derivation a restriction S comes up, where S is interpreted as a conjunction of literals. Since the solution of the restrictions can be delayed until all other literals are resolved away, constrained resolution can be seen as a "lazy" strategy for theory resolution. Note however that *no* instantiation takes place in RQ-inferences. For example, in (3) the variable x is not instantiated to joe. Things are different in theory resolution. Let us compute the same example in theory resolution. The theory consists of the axioms $\mathcal{R}_T$, and the clauses specification is as follows:

(1") $\neg person(x) \vee sleep(x) \vee eat(x)$
(2") $\neg sleep(joe)$

In constrained resolution the $person$-literal is shifted to the constraint part as demonstrated above. So only one resolvent can be built. Theory resolution admits two possible resolvents. As a first possibility we could derive in a syntactic step

$$\frac{\neg person(x) \vee \underline{sleep(x)} \vee eat(x) \qquad \underline{\neg sleep(joe)}}{person(joe) \vee eat(joe)}$$

with $\mathcal{R}_T$ refuter $\{x \leftarrow joe\}$. Since the theory-literal $person(x)$ is not selected this inference roughly corresponds to the constrained resolution inference. As another possibility we could select $person(x)$ in (1") and obtain

$$\frac{\underline{\neg person(x)} \vee sleep(x) \vee eat(x)}{sleep(joe) \vee eat(joe)}$$

with $\mathcal{R}_T$ refuter $\{x \leftarrow joe\}$, since $\{\neg person(joe)\}$ is $\mathcal{R}_T$-complementary. In both cases, however, x will be instantiated.

Here, constraint resolution is clearly advantageous compared to theory resolution. While theory resolution blindly has to guess a refuter[3], constrained resolution may delay this until more information is available.

However constrained resolution has a serious drawback. Informally, the RQ-formulas must not alter the meaning of a predicate symbol in Δ defined by the restriction theory $\mathcal{R}$. More technically, only interpretations for RQ-formulas are considered whose reduct to Δ is one of the models in $\mathcal{R}$, i.e. every RQ-interpretation is a conservative extension of a model in $\mathcal{R}$. As a sufficient criterion to achieve this, the disjointness of the signatures w.r.t. predicate symbols (see above) is used.

This property marks an important difference between constraint resolution and theory resolution, since in theory resolution also non-conservative extensions of the background theory are allowed: the foreground theory might "cut off" some models of the background theory. This is advantageous because it allows one to split definitions for certain predicate symbols across the background and the foreground theory. Take as an example equality, i.e. the background theory is equality, where one usually has (positive) equations also in the foreground specification.

[3]Consider e.g. AC-unification which has an exponential number of solutions in the number of variables

3.5 Theory Model Elimination

Model elimination is a refutational complete calculus for first order clause logic (Loveland, 1968). Not quite correctly, but instructively it can be seen as restricted version of linear resolution with ancestor resolution, where ancestor resolution is restricted to s-resolution (subsumption resolution: the ancestor clause minus its literal resolved upon must subsume the near parent clause). Furthermore, the clauses are sequences ("chains") rather than sets of literals. However, collapsing multiple occurrences of equal literals in a sequence is achieved; furthermore in its strong version, unifiable literals may be collapsed into a single occurrence by the factorization rule.

The calculus presented in this section is not exactly model elimination, even in the non-theory case; the main difference is that our calculus does not contain such improvements as collapsing multiple occurrences of equal literals into one occurrence. This could be achieved in our calculus by adding an additional factorisation inference rule; however we feel that this topic is not essential for our purposes here, so it will not be considered any further. On the other hand, for our calculus we have the *independence of the computation rule*, i.e. we may nondeterministically select the next subgoal to be processed. So, Loveland's original computation rule is covered. For lack of a better name, and because of the many similarities we prefer to speak of "model elimination". A discussion of the differences compared with Loveland's calculus can be found in (Baumgartner and Furbach, 1992). The theory model elimination calculus is introduced in (Baumgartner, 1992a).

Next we will informally explain a tree-oriented version of theory model elimination. Then, after the formal definition based on theory consolution is given completeness will be proved.

3.5.1 Informal Explanation

We will follow the lines from (Letz et al., 1992) and define the inference rules as tree-transforming operators. Then the calculus is much in the spirit of semantic tableau with unification for clauses (see (Fitting, 1990)), but with an important restriction. This restriction will be explained below and justifies using the new name — "tableau model elimination" — instead of qualifying it as "analytical tableaux for clauses with unification".

Equally, this calculus can be seen as a restriction of semantic tableau with unification for clauses (see (Fitting, 1990)). This restriction will be explained below.

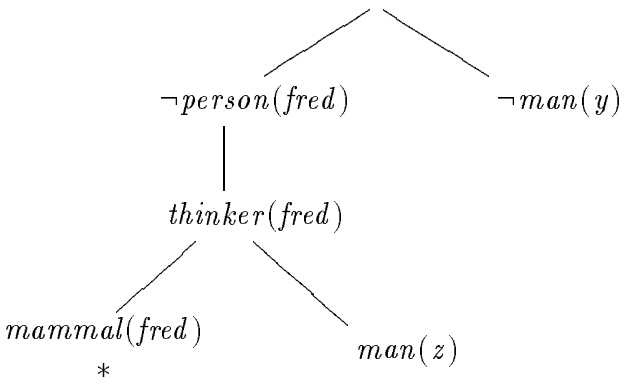
Example. This is an excerpt from the example in the introduction:

$\mathcal{T}$: (T-1) $\forall x((mammal(x) \wedge thinker(x)) \rightarrow person(x))$
(T-3) $\forall x(man(x) \rightarrow person(x))$

$\mathcal{C}$: (1) $\{\neg person(fred), \neg man(y)\}$
(2) $\{thinker(fred)\}$
(3) $\{mammal(x), man(z)\}$

Model elimination tries to construct a semantic tableau, where every branch is $\mathcal{T}$-complementary ("closed"). We have to pick a clause and construct an *initial tableau*; selecting clause (1) yields:

We will show a *theory extension step*, using from the left branch $\neg person(fred)$ and from clauses (2) and (3) the literals $thinker(fred)$ and $mammal(x)$ (respectively) as a $\mathcal{T}$-refutable literal set. It can easily be seen that $\{X \leftarrow fred\}$ is a $\mathcal{T}$-refuter for this set (join the clause form of (T-1) to this set and find a resolution refutation). Hence the tableau can be extended at the left branch with clauses (2) and (3) in such a way that a branch labelled with the complementary literal set comes up:



The path containing the set of $\mathcal{T}$-unsatisfiable literals is closed, i.e. it is marked with an asterix and the $\mathcal{T}$-refuter is applied to the whole tree.
Such a closing of a path can also be done by the second inference rule, the *reduction step*: if a path contains a set of literals that are theory complementary by some $\mathcal{T}$-refuter σ, it can be closed and the MGR is applied to the entire tree. The branch ending in $man(z)$ can be closed in a reduction step, because by (T-3) the literals $man(z)$ and $\neg person(fred)$ are $\mathcal{T}$-complementary by $\mathcal{T}$-refuter $\{z \leftarrow fred\}$.

This process has to be repeated until a tree is derived where all branches are marked with a star. Then a refutation is found.

3.5.2 Formal Definition

In the formalisation of this calculus we need only one inference rule, namely the extension rule, because the reduction rule will become an instance of the extension rule:

Definition 3.19 (Total theory model elimination) The inference rule *total theory extension* is defined as follows:

$$\frac{\begin{array}{l}\mathcal{P} \cup \{p\} \\ \mathcal{P}_{R_1 \cup \{L_1\}} \\ \vdots \\ \mathcal{P}_{R_n \cup \{L_n\}}\end{array}}{\mathcal{R}}$$

where

1. $\mathcal{P} \cup \{p\}$ is a path set.
2. $\{L_i\} \cup R_i$ are pairwise variable disjoint clauses (for $i = 1 \ldots n$).
3. for some subset q of the literal set of $\mathit{front}(p)$ the set $q \cup \{\mathit{last}(p), L_1, \ldots, L_n\}$ is minimal $\mathcal{T}$-refutable. Let $\mathcal{S}$ be a CSR for that set.
4. The simplification of the product

   $$\mathcal{Q} = (\mathcal{P} \cup \{p\})(\mathcal{P}_{R_1 \cup \{L_1\}} \cdot \quad \ldots \quad \cdot \mathcal{P}_{R_n \cup \{L_n\}})$$

   is done in the following way:

   A) $\mathcal{Q}\sigma$ is obtained from $\mathcal{Q}$ by application of a $\mathcal{T}$-refuter $\sigma \in \mathcal{S}$ for $\{q, L_1, \ldots, L_n\}$ (which exists by 2.)

   B) Delete $(p \circ L_1 \circ \cdots \circ L_n)\sigma$ from $\mathcal{Q}\sigma$ and obtain $\mathcal{R}^B$.

   C) Let $\mathcal{F}$ be the fern of $L_1 \vee R_1, \ldots, L_n \vee R_n$ with trunk $L_1 \circ \cdots \circ L_n$. Then obtain $\mathcal{R}^C$ from $\mathcal{R}^B$ by shortening to the path set that is specified as follows:

   $$\mathcal{R}^C = (\mathcal{P} \cup (\{p\}(\mathcal{F} \setminus \{L_1 \circ \cdots \circ L_n\}))) \, \sigma$$

   D) Delete some duplicate occurrences of some paths in $\mathcal{R}^C$ to obtain $\mathcal{R}$.

A *total model elimination refutation* is defined as a *linear theory consolution derivation* where every theory consolution inference is a "total theory model elimination". □

In terms of the traditional calculus, the path set $\mathcal{P} \cup \{p\}$ is intended to represent the tableau that is extended at its branch p. Then the effect of C) is to append to p the fern formed from the clauses to extend with. It is due to the design of "derivation" and "simplification" that

a sequence of tableau in a (traditional) theory model elimination derivation can be mirrored by a sequence of path sets in the consolution-style theory model elimination. Of particular importance here is step D) in simplification, where it is not exactly specified how many duplicate paths should be removed: at the one extreme, if all but one occurrences of a branch are removed then in effect more paths may get closed than in the corresponding traditional extension step; at the other extreme, if no path is removed then, speaking in the terminology of the original calculus, some additional branches may come up which are copies of already present branches. Thus, if the original calculus is to be precisely expressed in terms of the consolutions framework, step D) must be somewhere "in the middle" between both extremes.

Condition 3. might seem a bit complicated. Informally it expresses the requirement that the leaf, as well as all literals in the extending branch together with some literals in the extended branch form a minimal $\mathcal{T}$-complementary set. Thus this condition generalizes the condition in the non-theory case that in an extension step a leaf A must be closed with a clause containing $\neg A$ (of course this is the ground case and has to be lifted).

In the tree calculus by the extension step a branch gets closed. The corresponding action here is to delete that branch in simplification step B).

Note that "Reduction step" is an instance of extension step in case $n = 0$.

Example. Consider the theory and clauses as used above in the informal presentation. We will redo the above inference step. The initial tree to which we apply the extension step is represented by the following path set obtained from (1):

$$\mathcal{P} \cup \{p\} = \{\neg person(fred), \neg man(y)\}$$

Here we choose $p = \neg person(fred)$. Clauses (2) and (3) yield the following path sets:

$$\mathcal{P}_1 = \{thinker(fred)\}$$
$$\mathcal{P}_2 = \{mammal(x), man(z)\}$$

The path sets $\mathcal{P} \cup \{p\}$ together with the path sets $\mathcal{P}_1$ and $\mathcal{P}_2$ are to be combined in the following inference step:

$$\frac{\begin{array}{l}\neg person(fred), \neg man(y)\\ thinker(fred)\\ mammal(x), man(z)\end{array}}{\neg person(fred) \circ thinker(fred) \circ man(z), \neg man(y)}$$

Observe that the resulting path set encodes the same tableaux as in the example of the above informal presentation. Let us have a detailed look at how this result can be achieved: in a first

step the product of the path sets is computed:

$$\mathcal{Q} = (\mathcal{P} \cup \{p\})\mathcal{P}_1\mathcal{P}_2 = \{\ \underline{\neg person(fred) \circ thinker(fred) \circ mammal(x)},\\ \neg person(fred) \circ thinker(fred) \circ man(z),\\ \neg man(y) \circ thinker(fred) \circ mammal(x),\\ \neg man(y) \circ thinker(fred) \circ man(z)\}$$

Then $\mathcal{Q}$ is simplified:

A) The above underlined path

$$\neg person(fred) \circ thinker(fred) \circ mammal(x) \qquad (*)$$

is selected for finding a $\mathcal{T}$-refuter σ; a $\mathcal{T}$-refuter for this path is $\{x \leftarrow fred\}$.

B) $\mathcal{R}^B$ is obtained from $\mathcal{Q}\sigma$ by deletion of $(*)$

$$\mathcal{R}^B = \{\ \underline{\neg person(fred) \circ thinker(fred) \circ man(z)},\\ \underline{\neg man(y)} \circ thinker(fred) \circ mammal(fred),\\ \underline{\neg man(y)} \circ thinker(fred) \circ man(z)\}$$

C) Shorten to the underlined paths in $\mathcal{R}^B$ and obtain, modulo multiplicity of paths:

$$\mathcal{R}^C = \{\ \neg person(fred) \circ thinker(fred) \circ man(z),\\ \neg man(y),\\ \underline{\neg man(y)}\}$$

It can be verified that this shortening is indeed achieved by the specification of the definition.

D) Obtain R from $\mathcal{R}^C$ by deleting the underlined path.

The partial variant of theory model elimination can be defined in much the same way as total theory resolution is modified to partial theory resolution. This is omitted here.

3.5.3 Completeness

Completeness of the consolution-style theory model elimination is obtained by simulating another theory model elimination calculus known to be complete, namely the calculus from (Baumgartner, 1992a). To distinguish it from the above introduced model elimination, we will call this calculus tableaux model elimination.

Tableaux Model Elimination

In (Baumgartner, 1992a) tableaux are introduced as so called *literal trees*, i.e. multiset of branches, which form a tree, the nodes of which are labelled with literals. Note that without this distinction between nodes and their labels, we get our multisets of paths. A literal tree T' is obtained from a literal tree T by *extension with a clause* $L_1 \vee \ldots \vee L_n$ *at a branch* b iff

$$T' = T - \{b\} \cup \{b \circ l_i \mid i = 1 \ldots n \text{ and } l_i \text{ is labelled with } L_i\}$$

In this case we also say that T' contains a clause $L_1 \vee \ldots \vee L_k$ rooted at b.

The term "to close a branch" means to attach an additional label "$\star$" to its leaf in order to indicate that the branch is proved to be $\mathcal{T}$ -complementary. A branch is *open* iff it is not labelled in that way.

Definition 3.20 (Total Theory Tableau Model Elimination) Let M be a clause set and $\mathcal{T}$ be a theory. An *initial model elimination tableau for M with top clause C* is a literal tree that results from extending the empty tree (the tree that contains only the empty branch) with the clause C.

A *model elimination tableau (ME tableau) for M* is either an initial ME tableau or a literal tree obtained by a single application of one of the following inference rules to a ME tableau T:

Extension step: Let $b = L_1 \circ \ldots \circ L_{k-1} \circ L_k$ be an open branch in T . Suppose there exist new variants $C_i = K_i^1 \vee \ldots \vee K_i^{m_i}$ $(i = 1 \ldots n)$ of clauses in M. These clauses are called the *extending clauses* and the sequence $K_1^1 \circ \cdots \circ K_n^1$ is called the *extending literals*.

In order to describe the appending of the extending clauses, we define the literal tree T_n and the "actual branch to extend", b_n, recursively as follows: if $n = 0$ then $T_0 := T$ and $b_0 := b$, else T_n is the literal tree obtained from T_{n-1} by extending with the clause C_n at b_{n-1} and $b_n := b_{n-1} \circ K_n^1$.

Let $\mathcal{K}$ be a subset of the literal set of b_n with $L_k, K_1^1, \ldots, K_n^1 \in \mathcal{K}$. Borrowing a notion from (Stickel, 1985), $\mathcal{K}$ is called the *key set*. If there exists a most general $\mathcal{T}$-unifier σ for $\mathcal{K}$, and $\mathcal{K}\sigma$ is minimal $\mathcal{T}$-complementary, then total theory extension yields the literal tree $T_n\sigma$, and the branch $b_n\sigma \in T_n\sigma$ is closed.

A total extension step with $n = 0$ is also called **reduction step**.[4] A *derivation from M with top clause C and length n* is a finite sequence of tableaux $T_0, T_1, \ldots, T_n$, where T_0 is an initial tableau for M with top clause C, and for $i = 1 \ldots n$ T_i is the tableau obtained from T_{i-1} by one single application of one of the above inference rules with new variants of clauses from M. If additionally in T_n every branch is closed then this derivation is called a *refutation of M*. □

[4] This notion is kept for historical reasons

Tableau Model Elimination and Model Elimination

The data structures of the two calculi – tableau model elimination and model elimination – are already identical, since the trees from the tableau oriented calculus are defined as multisets of paths. A key concept of tableau model elimination is the extension of a tree; we can simulate this operation very naturally with path multiplication from consolution:

Lemma 3.1 *Let T' be a literal tree obtained from a literal tree T by* extension with a clause $C = L_1 \vee \ldots \vee L_n$ at a branch *b, i.e.*

$$T' = T - \{b\} \cup \{b \circ l_i \mid i = 1 \ldots n\} \text{ and } l_i \text{ is labelled with } L_i,$$

then

$$T' = T - \{b\} \cup (\{b\} \cdot \mathcal{P}_C)$$

Theorem 3.5 *Every total theory tableau model elimination derivation $T_0, T_1, \cdots, T_n$ is a total theory model elimination derivation.*

Proof. Let T_i be derived from T_{i-1} by extension of path b with extending clauses $C_1, \cdots, C_m$ and extending literals $K_1, \cdots, K_m$. Let σ be the $\mathcal{T}$-unifier for K, where K is a subset of the extended path b' (which is obtained by extension of b as defined in the definition of extension), such that $last(b), K_1, \cdots, K_m \in K$. Hence, we can write $T_{i-1} = \mathcal{P} \cup \{b\}$ with an appropriate path set $\mathcal{P}$ and $\mathcal{P}_{C_j} = \mathcal{P}_{R_j \cup \{K_j\}}$ with $C_j = R_j \cup \{K_j\}$ for $1 \leq j \leq m$.

Note that the path b' is obtained by successive extensions of b with $C_1, \cdots, C_m$ at the literals K_i; i.e. the path b' consists of an initial part b followed by the trunk of the fern F of $C_1, \cdots, C_m$.

Now the path sets $T_{i-1}, \mathcal{P}_{C_1}, \cdots, P_{C_m}$ are the premise of the theory model inference rule and we have to show that there exists a conclusion $\mathcal{R}$, which is equal to T_i. According to the theory model elimination rule we first have to compute the product

$$Q = (\mathcal{P} \cup \{b\})(\mathcal{P}_{R_1 \cup \{K_1\}} \cdot \ldots \cdot \mathcal{P}_{R_m \cup \{K_m\}})$$

which is simplified as follows:
A) $Q\sigma$ is obtained by applying the $\mathcal{T}$-refuter σ.
B) The closing of b' corresponds to the deletion of the path $(b \circ K_1 \circ \ldots \circ K_m)\sigma$ and thus we get $\mathcal{R}^B$
C)From $\mathcal{R}^B$ we have to shorten to the path set $\mathcal{P} \cup (\{b\}(F \setminus \{K_1 \circ \ldots \circ K_m\}))\sigma$
D) Delete those multiple occurrences of paths to obtain $\mathcal{P} \cup (\{b\}(T_m \setminus \{K_1 \circ \ldots \circ K_m\}))\sigma$. □

Togehter with the completeness result for total theory tableau model elimination we get completeness of theory model elimination. Note that the following result is formulated for partial theory tableau model elimination in (Baumgartner, 1992a). This requires a rather technical condition (called acceptable), which can be ommitted in the case of total derivations. Since we restrict ourselves to that case, we cite a simplified version of the completeness theorem.

Theorem 3.6 ((Baumgartner, 1992a)) *Let $\mathcal{T}$ be a theory and M be a $\mathcal{T}$-unsatisfiable clause set. Let $C \in M$ be such that C is contained in some minimal $\mathcal{T}$-unsatisfiable subset of M. Then there exists a total theory tableau model elimination refutation of M with top clause C.*

Theorem 3.7 (Completeness of total theory model elimination) *Let $\mathcal{T}$ be a theory and M be a $\mathcal{T}$-unsatisfiable clause set. Then there exists a total theory model elimination refutation of M.*

Proof. From the completeness of theory tableau model elimination (theorem 3.6) we know, that there exists a theory tableau model elimination derivation of the empty clause and from theorem 3.5 we conclude that there is a theory model elimination refutation as well. □

3.6 Theory connection method

In the present section we discuss a theory version of the connection method according to W. Bibel. According to the conventions of this paper we will consider only formulas in clause form. The connection method is not restricted to that class of formulas. We will present the classical version of the connection method following (Bibel, 1987)[5]. However, we will transcribe this version into a refutational form. This way we allow the reader to concentrate on the essential differences and similarities of the connection method in comparision with the other calculi discussed in this paper.

3.6.1 Informal explanation

Let us introduce some notions which are usually used in the context of the connection method.

Let us recall from 3.1.1 that a set of clauses is called a matrix. The following notion generalizes the notion of a connection from subsubsection 3.1.2. A $\mathcal{T}$ *-connection* in a matrix M is a set of literals which contains from each clause from M at most one literal and which can be made $\mathcal{T}$ -complementary by instantiation. If the set of the elements of a path p through a matrix contains a set u of literals then u is said to *span* p. Let U be a set of sets of literals. Then U *spans the matrix* M if for every path p through M there is a $u \in U$ that spans p. Now we can formulate the idea of the connection method.

In order to refute a clause set C the connection method tries to find a set M of instances of clauses from C such that there is some set of $\mathcal{T}$ -complementary sets of literals which is spanning M. Under the name connection method there exists a variety of algorithms for constructing systematically such an instance M. The approaches differ mainly in the data structure which is used to represent the set of paths $\mathcal{P}$ which should be checked for the existence of a spanning $\mathcal{T}$-complementary set. In the version presented here $\mathcal{P}$ is the fern of

[5] For a more recent introductory presentation see (Bibel, 1992).

$C_1, \ldots, C_n$ with trunk $L_1 \circ \ldots \circ L_n$ where for every $i = 1, \ldots, n$ C_i is a subclause of a clause from M and for $i, j,\ 1 \leq \ldots < i < j \leq \ldots n$ C_i and C_j must not be subclauses of the same clause in M. Then $L_1 \circ \ldots \circ L_{n-1}$ is called the *actual path* and L_n the *actual goal*. This notion reflects that the procedure tries as the next step to span the trunk or its extension.

Now let us consider a the following clause set C and let us track some steps of a derivation based on the connection method with the built-in theory $\mathcal{T}$ from our running example:

$\mathcal{T}$:

(T-1) $\forall x((mammal(x) \wedge thinker(x)) \rightarrow person(x))$
(T-3) $\forall x(man(x) \rightarrow person(x))$

$\mathcal{C}$:

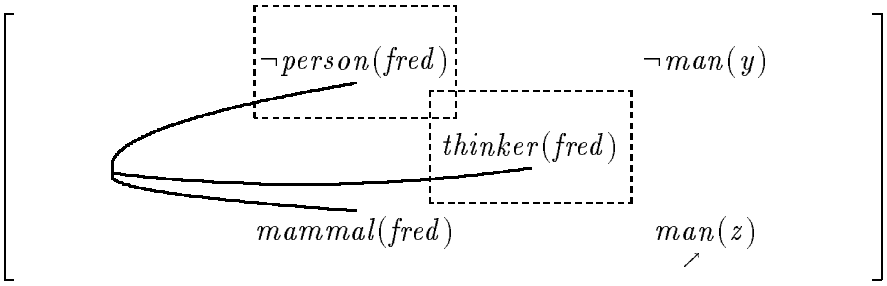
(1) $\neg person(fred), \neg man(y)$
(2) $thinker(fred)$
(3) $mammal(x), man(z)$

The clause set M which is actually under consideration is represented as a *matrix* where each row represents the elements of a clause. We start the derivation with the fern of one clause, namely clause (1), with trunk $\neg person(fred)$. The elements of the actual path will be in a dashed box. The actual goal will be pointed by a small arrow. The elements of each of the clauses which form the fern are placed on the right of the corresponding elements of the trunk. Thus, the initial matrix looks as follows.

$$\left[\begin{array}{ll} \underset{\nearrow}{\neg person(fred)} & \neg man(y) \end{array}\right] \tag{4}$$

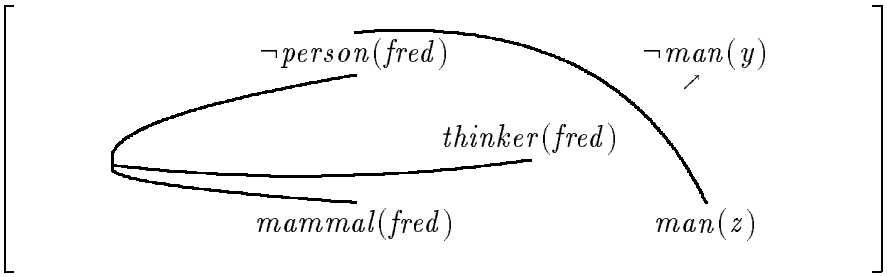
In our example we can try an *extension step* by entering new clause copies to the initial clause set at the bottom of the matrix and applying a substitution to the extended matrix. This extension must yield a $\mathcal{T}$ -refutable extension of the actual path via the actual goal. The actual goal and the literals $thinker(fred)$ and $mammal(x)$ from clauses (2) and (3) (respectively) form a $\mathcal{T}$ -connection. A $\mathcal{T}$ -refuter for that set is $\sigma_1 = \{x \leftarrow fred\}$. So entering copies of clauses (2) and (3), applying σ to the matrix, marking the $\mathcal{T}$ -complementary path with a connection (the triple arc) and putting the elements of the new actual path into dashed boxes

yields the following matrix.



(5)

It remains to span the paths in the fern determined by the new actual path. As next step we try a second inference rule, *reduction* that allows inferencing without adding new clauses to the matrix. Consider $\neg person(fred)$ from the first row and $man(z)$ from the third row. By theory clause $(T-3)$ they are $\mathcal{T}$ -refutable with $\mathcal{T}$ -refuter $\sigma_2 = \{z \leftarrow fred\}$. So σ_2 can be applied to the matrix and the $\mathcal{T}$ -connection can be established as shown below. Now we have spanned all paths starting from $\neg person(fred)$. Thus, it remains to span paths starting from $\neg man(y)$. The actual path now is empty and the actual goal is $\neg man(y)$. We arrive at the following matrix.



(6)

Here we leave our sample derivation. We only mention that the derivation will be completed if the fern will have length zero. Then every path through the matrix is spanned by a $\mathcal{T}$ -complementary set of literals.

3.6.2 Formal definition

The extension and the reduction step of the theory connection calculus may be expressed by one single instantiation of the consolution rule.

Definition 3.21 (Total theory connection method) Let $\mathcal{T}$ be a theory and $\mathcal{C}$ a set of clauses. The inference rule *total theory inference* is defined as follows:

$$\frac{\begin{array}{c}\mathcal{P}_1 \cup \{p_1\} \\ \mathcal{P}_{R_2 \cup \{L_2\}} \\ \vdots \\ \mathcal{P}_{R_n \cup \{L_n\}}\end{array}}{\mathcal{R}}$$

where

1. $\mathcal{P}_1 \cup \{p_1\}$ is a fern of $C_1, \ldots, C_m$ with trunc p_1 where for $j = 1, \ldots, m$ C_j is a subclause of an instance of a clause from C.
2. $\{L_i\} \cup R_i$ are pairwise variable disjoint copies of clauses from C (for $i = 2 \ldots n$).
3. for some subset q of the literal set of p_1 the set $\{q, L_2, \ldots, L_n\}$ is minimal $\mathcal{T}$ -refutable. Let $\mathcal{S}$ be a CSR for that set.
4. The simplification of the product

   $$\mathcal{Q} = (\mathcal{P}_1 \cup \{p_1\})(\mathcal{P}_{R_2 \cup \{L_2\}}) \quad \ldots \quad (\mathcal{P}_{R_n \cup \{L_n\}})$$

   is done in the following way:

   A) $\mathcal{Q}\sigma$ is obtained from $\mathcal{Q}$ by finding of a $\mathcal{T}$ -refuter $\sigma \in \mathcal{S}$ for $\{q, L_2, \ldots, L_n\}$ (which exists by 3.)

   B) All paths $q' \in \mathcal{Q}\sigma$ satisfying the conditions

   $$\mathit{tail}(p_1) < q', \ \ \mathit{length}(q') = \mathit{length}(p_1) + n \ \text{ and } \ \{q, L_2, \ldots, L_n\}\sigma \subseteq q'\sigma$$

   will be deleted. Call the resulting set $\mathcal{R}^B$.

   C) If F is the fern of $\{L_2\} \cup R_2, \ldots, \{L_n\} \cup R_n$ with trunk $L_2 \circ \cdots \circ L_n$ then obtain $\mathcal{R}^C$ from $\mathcal{R}^B$ by shortening to the path set that is specified as follows:

   $$\langle \mathcal{R}^B \cap (\mathcal{P}_1 \cup (\{p_1\}(F \setminus \{L_2 \circ \cdots \circ L_n\})))\rangle\sigma$$

   In step B) some paths p may have been deleted such that $K = \mathit{last}(p)$ satisfies $K \in C_n$ and $K \neq L_n$. Those paths belong to

   $$\mathcal{P}_1 \cup (\{p_1\}(F \setminus \{L_2 \circ \cdots \circ L_n\}))$$

   Therefore we have to cut with $\mathcal{R}_B$.

   D) Let $\mathcal{R} = \mathcal{R}^C$.

A *total theory connection derivation* is defined as a *linear theory consolution derivation* where every theory consolution inference is a "total theory connection inference". □

Example. In order to illustrate the previous definition let us consider again the theory and clause set as used above in the informal presentation. We discussed in detail the first inference step, an extension, in order to illustrate the model elimination case. Now for illustrating the formalization of the theory connection method we will redo the second inference step of our sample run which is a reduction. After the first step we arrive at the following path set[6]:

$$\mathcal{P}_2 \cup \{p_2\} = \{\neg person(fred) \circ thinker(fred) \circ man(z), \neg man(y)\}$$

Here we choose $p_2 = \neg person(fred) \circ thinker(fred) \circ man(z)$. The path set $\mathcal{P}_2 \cup \{p_2\}$ will be processed in the following inference step:

$$\frac{\{\underline{\neg person(fred) \circ thinker(fred) \circ man(z)}, \neg man(y)\}}{\{\neg man(y)\}} \qquad (*)$$

The resulting path set encodes the same path set as in the example of the above informal presentation. Let us have a detailled view how this result can be achieved: Since our product consists of one factor we have $\mathcal{Q} = \mathcal{P}_2 \cup \{p_2\}$. $\mathcal{Q}$ is simplified as follows: The instance of the consolution rule has the form

$$\frac{\mathcal{P}_2 \cup \{p_2\}}{\mathcal{R}}$$

where

1. p_2 denotes the path which is underlined in $(*)$. It has a maximal length in $\mathcal{P}_2 \cup \{p_2\}$.
2. $n = 0$.
3. The subset q is $\{\neg person(fred), man(z)\}$.
4. A) The above underlined path

   $$\neg person(fred) \circ thinker(fred) \circ man(z) \qquad (*)$$

   is selected for finding a $\mathcal{T}$-refuter $\sigma = \{z \leftarrow fred\}$ for the $\mathcal{T}$-connection $\{\neg person(fred), man(z)\}$.

   B) $\mathcal{R}_B$ is obtained from $\mathcal{Q}\sigma$ by deletion of $(*)$

   $$\mathcal{R}_B = \{\ \neg man(y)\ \}$$

   C) Shortening is neither possible nor necessary.

   $$\mathcal{R}_C = \mathcal{R}_B$$

   D) Finally we have $R = \mathcal{R}_C$.

[6]This path set is the fern of $\{\neg person(fred), \neg man(y)\}$, $\{thinker(fred)\}$, $\{man(z)\}$ with trunk $\neg person(fred) \circ thinker(fred) \circ man(z)$

It is astonishing how closely related the theory model elimination and theory connection method are.

- Model elimination has a free selection function and thus allows the next subgoal to be chosen in a don't care nondeterministically manner. In the presented version of the connection method a trunk, i.e. a branch of *maximal length*, is selected for extension. This restriction makes sure that the set of unexamined paths obtained after the extension may be represented as a fern too (cf. proposition 1). In (Neugebauer and Schaub, 1991) there has been proposed another representation of that set by a set of *hooks* instead of a fern. A hook represents the fern of $\{L_1\}, \{L_2\}, \ldots, \{L_{n-1}\}, \{L_n\} \cup C_n$ with trunk $L_1 \circ L_2 \circ \cdots \circ L_{n-1} \circ L_n$. This representation allows the same flexibility like the tree representation which is used in the tableaux model elimination.

- In step A) of simplification the role of the last literal of the path to be extended is different: in the tree-notation of model elimination, the last literal corresponds to the leaf of the branch, and that literal must be "essential" in the $\mathcal{T}$ -complementary literal set (i.e. without it, the set is no longer $\mathcal{T}$ -complementary). In the presented version of the connection method, the last literal has no special meaning. Thus the model elimination extension rule is more restrictive than the connection method inference rule. In (Petermann, 1993b) has been proved that also the extension rule of the the connection method may be restricted in that way without lost of completeness. Thus, the branching rate in the search space might be restricted like in the model elimination.

- In step B) the connection method is more effective than model elimination, since all paths which reach the actual clause and which become $\mathcal{T}$ -complementary by the $\mathcal{T}$-refuter are deleted. In model elimination only one path is deleted.

There are many refinements of the connection method which cut the search space essentially. For an overview see (Bibel, 1987) and (Letz et al., 1992)

Now let us formulate some results about the total theory connection method. First of all we need a representation lemma 3.2 concerning ferns and two lemmata 3.3 and 3.4 concerning their structure properties. The three lemmata may be proved by simple verification. Those facts enable us to prove that a set of paths $\mathcal{R}$ obtained by a total theory connection inference from a fern $\mathcal{F}$ is again a fern. This gives a formal justification of our definition of the total connection inference rule.

Lemma 3.2 *Let F be the fern of $\{C_1\}, \ldots, \{C_n\}$ with trunk $L_1 \circ \cdots \circ L_n$. Then for each $i = 1, \ldots, n$ holds*

$$C_i = \{L_i\} \cup \{\mathit{last}(p) \mid p \in F,\ \mathit{length}(p) = i\}.$$

Lemma 3.3 *Every subset of a fern is a fern.*

Lemma 3.4 *Let $\mathcal{F}$ be a fern of $\{C_1\}, \ldots, \{C_n\}$ with trunk $p = L_1 \circ \cdots \circ L_n$ and let $\mathcal{G}$ be a fern of $\{C_{n+1}\}, \ldots, \{C_{n+m}\}$ with trunk $L_{n+1} \circ \cdots \circ L_{n+m}$. Then*

$$(\mathcal{F} \setminus \{p\}) \cup (\{p\}\mathcal{G})$$

is the fern of $\{C_1\}, \ldots, \{C_n\}, \{C_{n+1}\}, \ldots, \{C_{n+m}\}$ with trunk $L_1 \circ \cdots \circ L_n \circ L_{n+1} \circ \cdots \circ L_{n+m}$.

Proposition 1 *Let*

$$\dfrac{\begin{array}{c}\mathcal{P}_1 \cup \{p_1\} \\ \mathcal{P}_{R_2 \cup \{L_2\}} \\ \vdots \\ \mathcal{P}_{R_n \cup \{L_n\}}\end{array}}{\mathcal{R}}$$

be a total theory connection inference. Then R is a fern.

Proof. Let $p_1 = K_1 \circ \cdots \circ K_m$ where $m = 0$ is allowed. Then (cf. definition 3.21) $\mathcal{P}_1 \cup \{p_1\}$ is a fern of some clauses $C_1, \ldots, C_m$ with trunk p_1. Let F be the fern of $\{L_2\} \cup R_2, \ldots, \{L_n\} \cup R_n$ with trunk $L_2 \circ \cdots \circ L_n$. Then according to lemma 3.4

$$\mathcal{P}_1 \cup (\{p_1\}F)$$

is the fern of

$$C_1, \ldots, C_m, \{L_2\} \cup R_2, \ldots, \{L_n\} \cup R_n$$

with trunk $p_1 \circ L_2 \circ \cdots \circ L_n$. According to lemma 3.3 R is a fern because it is a subset of that fern. □

Finally we give proof outlines of the soundness and completeness theorem for the total connection method.

Theorem 3.8 *Soundness Let $\mathcal{T}$ be a theory and let $\mathcal{C}$ be a clause set. If there exist a copy of a clause of $\mathcal{C}$, an element $L \in \mathcal{C}$ and a total theory connection derivation which starts from the fern of $\mathcal{C}$ with trunk L then $\mathcal{C}$ is $\mathcal{T}$ -unsatisfiable.*

Proof. Let $\mathcal{T}$ be a theory and let $\mathcal{C}$ be a clause set. Let $F_1, \ldots, F_k$ be a total theory connection derivation and let M_i denote the set of copies of clauses from C which have been used as extension clauses up to the $(i-1)$-th inference for each $i = 1, \ldots, k$. Moreover let U_i denote the set of $\mathcal{T}$ -connections which have been found up to the $(i-1)$-th inference for each $i = 1, \ldots, k$. Claim: U_i spans all paths p through M_i such that there is no path $q \in \mathcal{F}_i$ with $q < p$ or $q = p$. In order to prove this claim it is sufficient to observe that F_1 contains all paths through M_1 and that in each simplifaction step

within substeps A and B only those paths are deleted that are spanned by the considered $\mathcal{T}$ -connection,

within substeps C path are only shortened and

within substep D the set of paths remains unchanged.

Clearly, if $\mathcal{F}_k = \emptyset$ then U_k spans M_k. □

Theorem 3.9 *Completeness Let $\mathcal{T}$ be a theory such that*

1. *the set of $\mathcal{T}$ -connections is decidable,*
2. *for each $\mathcal{T}$ -refutable literal set u a complete set of $\mathcal{T}$ -refuters is enumerable.*

If $\mathcal{C}$ is a $\mathcal{T}$ -unsatisfiable clause set then there exist a copy of a clause of $\mathcal{C}$, an element $L \in \mathcal{C}$ and a total theory connection derivation which starts from the fern of $\mathcal{C}$ with trunk L.

Proof. Let $\mathcal{T}$ be a theory and let $\mathcal{C}$ be a $\mathcal{T}$ -unsatisfiable clause set. >From the Herbrand theorem proved in (Petermann, 1991b) follows that there is a set of instances M of clauses of $\mathcal{C}$ and a set of $\mathcal{T}$ -complementary literal sets which is spanning M. There is a ground total connection derivation of M. This may be shown using properties of minimal spanning matings of ferns (cf. (Petermann, 1993b)). The decidability of the set of $\mathcal{T}$ -connections and the enumerability of the CSR for each u allow to lift this ground derivation to a total connection derivation where each inference is effective. □

Discussion:

The decidability of the set of $\mathcal{T}$ -connections is necessary for the decidability of the derivation relation. Sometimes it is not necessary to know all $\mathcal{T}$ -connections and to be able to enumerate $\mathcal{T}$ -refuters for all of them. For example there is a fragment of first-order logic with equality which is the image of the translation of multi-modal logics following (Debart et al., 1990). Different modal systems may be described by equational theories. Fortunately, after translation there occurs no equality sign in the formulas. This sinplifies considerably the the set of theory connections to be considered. In order to describe this phenomenon in (Petermann, 1993b) has been studied the notion of a complete set of theory connections.

The decidability whether $CSR_{\mathcal{T}}(u) = \emptyset$ means in an implementation that whenever the existence of a $\mathcal{T}$ -refuter for a $\mathcal{T}$ -connection is considered that in case of the non-existence there will be a defined answer and not a failure.

The restriction that the actual goal is element of the actually found connection can be proved by careful examination of minimal spanning sets of connection.

The partial variant of theory connection method can be defined in much the same way as total theory resolution is modified to partial theory resolution. This is omitted here. The interested reader is referred to (Petermann, 1993a). Paramodulation, relaxed paramodulation and RUE-resolution have been view as instances of a partial theory connection calculus. It is worth

mentioning that the residue may have more than one literal if not only the paramodulation is considered.

4 Equality

Being of such a fundamental importance in mathematics and nearly every area where deduction applies, the equality relation deserves special treatment. Thus it is not surprising that in the early days of theorem proving the equality relation was already built in. Viewed syntactically "equations" are simply literals, built from the predicate symbol "=" and written infix. Semantically, the interpretation of "=" is comitted to an equivalence relation that is closed under equality replacement for all function and predicate symbols of the language (i.e. a congruence relation). From the viewpoint of theory reasoning, we have to deal with *the* theory of equality $\mathcal{E}$, which can be axiomatized by the following scheme:

$\mathcal{E}: \forall x: \quad x = x$	(Reflexivity)
$\forall x, y: \quad x = y \rightarrow y = x$	(Symmetry)
$\forall x, y, z: \quad x = y \wedge y = z \rightarrow x = z$	(Transitivity)
$\forall x_1, \ldots, x_n, y: \quad x_i = y \rightarrow$ $f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) =$ $f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$	for all n-ary function symbols f and all $1 \leq i \leq n$ (f-Substitutivity)
$\forall x_1, \ldots, x_n, y: \quad (x_i = y \wedge$ $P(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)) \rightarrow$ $P(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n))$	for all n-ary predicate symbols P and all $1 \leq i \leq n$ (P-Substitutivity)

If we assume that we have only a finite number of predicate and function symbols (as is usually the case) then this theory of equality is finite too. Thus also "equality" could be dealt with in the trivial way: simply add the clause form of these axioms to a given specification and rely on the no-theory calculus. This however is prohibited for well-known efficiency reasons. The theory of equality should be handled at least as carefully as any other theory following the discussion of the previous chapters e.g. 3.

4.1 Dealing with Equality via Total Theory Reasoning

Here we discuss the possibility of building in the theory of equality by total theory reasoning following our discussion in section 2.1.

For a given literal set we have to accomplish an $\mathcal{E}$-refuter. For this purpose it might be useful to know of the syntactical form of $\mathcal{E}$-complementary literal sets. It is as follows (Petermann, 1991b):

> $E \cup p$ where E is a set of equations and p consists either of two literals $P(u_1, \ldots, u_n)$ and $\neg P(v_1, \ldots, v_n)$ or of one negated equation $\neg u_n = v_n$ with $n = 0$ such that

$$\models_{\mathcal{E}} \bigwedge E \rightarrow \bigwedge_{i=0}^{n} u_i = v_i$$

Both cases have an obvious intuitive meaning. The sides of each term pair $\langle u_i, v_i \rangle$ might become equal by subsequent substitutions of subterms which occur as one side of the equations in E by the other side of such an equation. Obviously this raises a contradiction. Due to this syntactic structure the dedicated reasoner has to solve the following problem:

> Given a set E of equations and some pairs of terms $\langle u_i, v_i \rangle$. Is there a substitution σ such that for all equations $u_i\sigma = v_i\sigma$ holds that $E\sigma \cup \{\neg u_i\sigma = v_i\sigma\}$ is $\mathcal{E}$-complementary?

It can be shown (Baumgartner, 1992b) that this problem is equivalent to the so-called *rigid $\mathcal{E}$-unification* problem (Gallier and Snyder, 1990). It is formulated as follows:

> Given a set E of equations and some pairs $\langle u_i, v_i \rangle$. Is there a substitution σ such that all equations $u_i\sigma = v_i\sigma$ are a logical $\mathcal{E}$-consequence of $E\sigma$, where all variables are treated as constants?

Let us consider the following clause specification in the spirit of the introductory example:

(4) $\neg male(x) \vee father(firstchild(x)) = x$
(5) $firstchild(fred) = sue$
(6) $male(fred)$

A useful query is for example to ask for the father of sue. An answer might be found by adding the clause $\neg father(sue) = y$ and refuting the thus obtained clause set M. This example will be processed by the various approaches below. Of course there is a trivial solution, namely $\{y \leftarrow father(sue)\}$ which simply says that the father of Sue is the father of Sue. We will suppose the deductions below to be controlled in such a way that the more interesting answer $\{y \leftarrow fred\}$ results. Refutations which return that answer rely on the fact that there are two literal sets

$$\{father(firstchild(x)) = x, firstchild(fred) = sue, \neg father(sue) = y\}$$

$$\{\neg male(x), male(fred)\}$$

which become $\mathcal{E}$-complementary after the application of the substitution $\sigma = \{x, y \leftarrow fred\}$. Using the notions from section 3.1.2 σ is an $\mathcal{E}$-refuter for both literal sets. Let us consider a total theory consolution refutation for M. We assume that the query $\{\neg father(sue) = y\}$ has been used as start clause. The refutation consists of two inferences.

$$\frac{\{\neg father(sue) = y\}, \{firstchild(fred) = sue\}, \{father(firstchild(x)) = x, \neg male(x)\}}{\{\neg father(sue) = fred \circ firstchild(fred) = sue \circ \neg male(fred)\}}$$

$$\frac{\{\neg father(sue) = fred \circ firstchild(fred) = sue \circ \neg male(fred)\}, \{male(fred)\}}{\emptyset}$$

Let consider in more detail the interaction of the general and the dedicated reasoner during the first inference. The substitution σ has to be found by the dedicated reasoner for the literal set

$$\mathcal{L} = \{father(firstchild(x)) = x, firstchild(fred) = sue, \neg father(sue) = y\}$$

which has been supplied by the general reasoner. In our example the set $\mathcal{L}$ is a good choice for a candidate for an $\mathcal{E}$-complementary set. If the general reasoner made such a good choice then the dedicated reasoner will return a rigid $\mathcal{E}$-unifier, σ in our case. Otherwise, the dedicated reasoner will return the definitive answer that the choice was bad. In other words, rigid $\mathcal{E}$-unification is decidable, although NP-complete (Gallier et al., 1990). The hard problem is that it is undecidable whether a set of equations E in the $\mathcal{E}$-complementary problem format exists, or how many instances of equations are needed (see above). For example, in order to extend the equation $c(f(a), b) = c(a, f(b))$ to an $\mathcal{E}$-complementary literal set two instances of the equation $f(x) = x$ are needed, which are $f(a) = a$ and $f(b) = b$. The substitution $\{x_1 \leftarrow a, x_2 \leftarrow b\}$ is the $\mathcal{E}$-refuter.

The completeness of both connection and a model elimination calculi with total equality reasoning follows from a general completeness theorem for total theory reasoning proved in (Petermann, 1991b). In (Gallier et al., 1987) rigid $\mathcal{E}$-unification has been built into the "method of matings".

E-resolution (Morris, 1969) is a slightly different realization of this kind of reasoning with equality. Here for a given literal set $p = \{L, K\}$ and clauses D and D', where $L \in D$ and $K \in D'$, both the set of equations E and the appropriate $\mathcal{E}$-refuter σ for $E \cup p$ are computed by exhaustive paramodulating (see below) into p all equations which exist somewhere in the given clause set until a trivial syntactical inconsistence is discovered. In order to obtain completeness a level saturation strategy has been used. As the result of the computation, the dedicated reasoner returns the set of equations $E = \{e_1, \ldots, e_n\}$, the $\mathcal{E}$-refuter σ and the clauses $\{C_1, \ldots, C_n\}$ such that a total equality reasoning step may be carried out which involves the clauses D, D' and $C_1, \ldots, C_n$.

4.2 Dealing with Equality via Partial Theory Reasoning

In the previous approach we had to guess in one step an $\mathcal{E}$-complementary literal set and to compute an $\mathcal{E}$-refuter. Now we break this huge step into smaller ones. Two main ways to do so have been studied.

Paramodulation-like setting: We apply equations by substituting "equals by equals" in order to transform terms which are suspected to be equal. We are allowed to use unification in order to make an equation applicable. This is rather a bottom-up approach.

RUE-resolution-like setting: We consider the "difference" between corresponding terms, in order to determine some equations which are sufficient for their equality. This is rather a top-down approach.

In both cases the considered rules are thought to produce either complementary pairs or negated equations. Both cases will be viewed now as instances of partial theory reasoning.

4.2.1 Paramodulation-like Calculi

Let us recall from section 4.1 that if an $\mathcal{E}$-refuter σ has been found for literal set $E \cup p$ then corresponding term pairs in $p\sigma$ may be made equal by substituting "equals by equals" using equations from $E\sigma$. In a *paramodulation-like* setting, the set $E \cup p$ and the $\mathcal{E}$-refuter will be approximated by number of single replacements of "equals by equals". To be equal means now "equal up to some substitution".

Let us examine the co-operation between general and dedicated reasoner in this situation. The general reasoner now has to supply literal sets

$$\{L[t'], t \doteq u\}$$

where t' denotes a subterm occurrence in L and $t \doteq u$ is a nondeterministic notation for either the literal $t = u$ or the literal $u = t$. L may contain the equality predicate. The dedicated reasoner has to return a pair

$$(\sigma, L[t' \leftarrow u])$$

if a most general unifier σ of t' and t exists. $L[t' \leftarrow u]$ means that one occurrence of t' in L is replaced by u. Obviously, the literal set

$$\mathcal{L} = \{L[t'], t \doteq u, \overline{L}[t' \leftarrow u]\}\sigma$$

is $\mathcal{E}$-complementary. Thus, the pair

$$(\sigma, L[t' \leftarrow u])$$

is a residue for the literal set

$$\{L[t'], t \doteq u\}.$$

A partial $\mathcal{E}$-reasoning step using this way of forming the residue will be called a *paramodulation* inference. t' is called the *term paramodulated into*.

Although well-known we prefer to give a short example of paramodulation in order to ease comparison with other approaches below. We return to our example from 4.1. We could compute the $father$ of sue by splitting the first total theory consolution step into one paramodulation and one extension step. This fragment of the derivation has the following form:

$$\frac{\{firstchild(fred) = sue\}, \{father(firstchild(x)) = x, \neg male(x)\}}{\{p_1, p_2\}}$$

$$\frac{\{p_1, p_2\}, \{\neg father(sue) = y\}}{\{firstchild(fred) = sue \circ \neg male(fred)\}}$$

where

$$\begin{aligned} p_1 &= firstchild(fred) = sue \\ &\quad \circ father(firstchild(fred)) = fred \circ father(sue) = fred \\ p_2 &= firstchild(fred) = sue \circ \neg male(fred) \end{aligned}$$

The more interesting first step in that derivation is a paramodulation into the term $firstchild(x)$. The pair

$$(\sigma', father(sue) = fred) \text{ with } \sigma' = \{x \leftarrow fred\}$$

is a residue for the set of literals

$$\{firstchild(fred) = sue, father(firstchild(x)) = x\}.$$

The path p_1 has been obtained by adjoining the path from the residue to the path

$$firstchild(fred) = sue \circ father(firstchild(fred)) = fred.$$

Obviously, the paramodulation inference is much simpler than the the computation of an $\mathcal{E}$-refuter. But this simplification must be paid for by a large search space introduced by the chaining of paramodulation steps. In the present version paramodulation cannot use any hint which equation should be applied for paramodulation in order to enable further reasoning steps to be made. Thus, the enormous search space will be entered without any guidance. Therefore, improvements are urgently needed (see below).

Paramodulation has been built in to many calculi. *Resolution* with paramodulation is complete. For the resolution calculus, we arrive at the following well-known form of the paramodulation inference rule (Robinson and Wos, 1969):

$$\frac{L[t'] \vee R_1 \qquad t \doteq u \vee R_2}{(L[t' \leftarrow u] \vee R_1 \vee R_2)\sigma}$$

with L, t', t and σ as above and, as usual, the two clauses involved in the inference step being variable disjoint.

Several rules which restrict the applicability of paramodulation have been invented in order to cut the search space. It is well known (see e.g. (Chang and Lee, 1973)) that linear paramodulation is complete provided that additionally the *functional reflexive axioms* are added (e.g. (Furbach et al., 1989; Hölldobler, 1989)). In (Peterson, 1983) it was shown that these axioms are unneccesary in the unrestricted (no linear or set of support restriction) resolution and paramodulation calculus. Furthermore he shows that it is not necessary to paramodulate into variable occurrences. By these restrictions two sources for substantial inefficiences are eliminated. The refutation from above satisfies this restriction. As anexample for a paramodulation step into a variable confer the following step, which paramodulates into the variable occurrence y.

$$\frac{\{\neg father(sue) = y\}, \{father(firstchild(x)) = x, \neg male(x)\}}{p_3, \neg father(sue) = y \circ \neg male(fred)}$$

where

$$p_3 = \neg father(sue) = y \circ father(firstchild(x)) = x \circ \neg father(sue) = father(firstchild(x))$$

In order to combine the best of two worlds, a set of support strategy and avoidance of paramodulation into variables, in (Snyder and Lynch, 1991) an inference rule "relaxed paramodulation" is defined. The calculus essentially consists of this rule, factorisation and a standard paramodulation rule that, however, does not paramodulate into variables. "Relaxed paramodulation" delays part of the unification and introduces the delayed parts as new subgoals into the resolvent. Thus, this rule may be applied even if paramodulation is not possible. The delay of unification is necessary for preserving the completeness. The following sample clause set is $\mathcal{E}$-unsatisfiable, but it cannot be refuted by a set of support strategy with paramodulation. Let

$$d = e$$
$$\neg f(c(d), c(e)) = g(c(d), c(e))$$

$$\neg f(c(d), c(e)) = h(c(d), c(e))$$
$$f(X, X) = g(X, X)\ ,\ f(Y, Y) = h(Y, Y)$$

be a set of equations with set of support $\{f(Y, Y) = h(Y, Y), f(X, X) = g(X, X)\}$. The reason for paramodulation (without substituting into variables) to fail is that there is no way to paramodulate from $f(Y, Y)$ into $f(c(d), c(e))$ nor from $g(X, X)$ into $g(c(d), c(e))$ nor from $h(Y, Y)$ into $h(c(d), c(e))$. However, relaxed paramodulation from, say, $f(Y, Y) = h(Y, Y)$ into $f(c(d), c(e))$ is possible. This inference returns $(\{\ \}, Y = c(d), Y = c(e), h(c(d), c(e)) = h(Y, Y))$ as a kind of a residue. All of these additional goals may be solved by use of the equation $d = e$. This technique is much in the spirit of RUE-resolution (Digricoli and Harrison, 1986) with the important restriction that the delayed subgoals (the corresponding concept in RUE-resolution is "disagreement set") can be restricted to variable-term pairs. Unfortunately the completeness of "relaxed paramodulation" with set of support strategy has not been proved yet.

As a concluding improvement of paramodulation we will briefly sketch *basic paramodulation* (Bachmair et al., 1992). The term "basic" was coined in (Hullot, 1980) in the context of narrowing (section 2.2) and means the same thing for paramodulation: in basic paramodulation, it is forbidden to paramodulate into occurrences introduced by previous inference steps. Stated positively, paramodulation is allowed only into such occurrences that are already present in the input set. For example, suppose that the clause $P(x) \vee Q(x)$ is resolved with $\neg Q(g(f(a)))$ yielding $P(g(f(a)))$. Then $P(g(f(a)))$ can be paramodulated with, say, $f(a) = a$ to $P(g(a))$; however, this is no basic paramodulation step, since $P(x)$ does not contain the f-term to be paramodulated upon. Thus, basic paramodulation is a serious restriction of the applicabibilty of paramodulation. Moreover, in basic paramodulation it is not necessary to paramodulate into variables; also it is compatible with ordering restrictions (section 4.2.3).

A paramodulation-like treatment of equality was carried out for the tableau method (Fitting, 1990) in (Jeffrey, 1967) and (Poppelstone, 1967), for model elimination (Loveland, 1978), for the connection method (Petermann, 1991a) and for connection graphs (Siekmann and Wrightson, 1980). Paramodulation with order-sorted theories has been considered in (Walther, 1983).

In the next parts of the paper we discuss in more detail two further alternatives for restrictions of the paramodulation rule which have been considered in the literature:

1. Make the application of equations more goal-oriented: choose first candidates for a complementary pair and look which equations should be used. Solve these equations by paramodulation.
2. Restrict the applicability of the paramodulation rule by order restrictions. For instance make sure that clauses are not growing w.r.t. a term ordering.

4.2.2 RUE-Resolution-like

Let us discuss in more detail one strategy of making paramodulation more efficient. The improvement is given by the following advice.

- Try to unify corresponding term pairs first.
- Paramodulate only into those terms which cannot be made equal this way.

Applying this strategy to the previous example the prover might "think": Let's try to make an $\mathcal{E}$-complementary literal set from $\neg father(sue) = y$ and $father(firstchild(x)) = x$.

Then the equation $firstchild(x) = sue$ would be needed to achieve that goal. The hint, that $firstchild(x) = sue$ should be solved as a subgoal follows from the failure of the unification algorithm[7]. Since on the outermost level the $father$-function symbols are the same in both terms, it suffices to prove that the argument terms of the left hand side, and the right hand side are equal. In other words, the left hand side has to be "decomposed"[8] into the new subgoal $\neg firstchild(x) = sue$. The right hand sides immediately unify by, say, $\{y \leftarrow x\}$. This example demonstrates the main idea of RUE-resolution: unify the terms "as much as possible", and prove equality of non-unifying terms later. In RUE-terminology the most general substitution involved in this is called a *most general partial unifier* and the non-unifying part is called the *disagreement set*. For a given unification problem several disagreement sets may exist. Let us examine an example related to that in subsection 4.1. In our terminology the example situation can be formulated as the path

$$\neg father(sue) = y \circ father(firstchild(x)) = x \circ firstchild(x) = sue$$

which becomes $\mathcal{E}$-complementary with the substitution $\sigma' = [y \leftarrow x]$. Thus, the pair

$$(\sigma', \neg firstchild(x) = sue)$$

forms a residue for the literal set

$$\{\neg father(sue) = y, father(firstchild(x)) = x\}.$$

The new goal $\neg firstchild(x) = sue$ may be solved together with the fact $firstchild(fred) = sue$. The corresponding two step consolution refutation fragment looks as follows. The first step is adding a residue, the second step is an extension. Let

$$\begin{aligned} q_1 &= \neg father(sue) = x \\ &\quad \circ father(firstchild(x)) = x \circ \neg firstchild(x) = sue \\ q_2 &= \neg father(sue) = x \circ \neg male(x) \end{aligned}$$

[7] Suppose the unification is according to the transformational approach to unification (J.-P. Jouannaud, 1991). Then the failure already occurs after one application of the decomposition rule.

[8] A unification algorithm based on an transformational approach which also includes a decomposition rule can be found in (J.-P. Jouannaud, 1991)

$$\frac{\{\neg father(sue) = y\}, \{father(firstchild(x)) = x, \neg male(x)\}}{\{q_1, q_2\}}$$

$$\frac{\{q_1, q_2\}, \{firstchild(fred) = sue\}}{\{\neg father(sue) = fred \circ \neg male(fred)\}}$$

Let us compare the present tactic with the paramodulation rule. The underlying $\mathcal{E}$-complementary literal set has a similar structure:

$$\{L(s[t]), t \doteq u, \overline{L}(s[t \leftarrow u])\}\sigma'$$

But now we construct the pair

$$(\sigma', \neg t \doteq u)$$

as the residue for the literal set

$$\{L(s[t]), \overline{L}(s[t \leftarrow u]\}.$$

The construction of the residue is now more goal-oriented. We have another restriction of the general rule for computing the residue.

It should be noted that in general it is not possible for completeness reasons to restrict to the "most specific" disagreement set. This is the disagreement set that results from unification "as much as possible". An example below will show this. This claim may illustrated by the following example (Anderson, 1970).

(1) $P(f(x), g(y)), P(h(x), i(y))$
(2) $\neg P(f(a), g(b))$
(3) $\neg P(h(b), i(b))$
(4) $f(a) = f(c)$
(5) $h(c) = h(b)$

A prover which is allowed to paramodulate only on unification failure comes up with a unifier containing

> either $\{x \leftarrow a\}$ (if $P(f(x), g(y))$ from the first and $\neg P(f(a), g(b))$ from the second clause are considered)
>
> or $\{x \leftarrow b\}$ (if $P(h(x), i(y))$ from the first and $\neg P(h(b), i(b))$ from the third clause are considered).

But the unique useful substitution $\{x \leftarrow c\}$ may be found only in a more flexible way. One improvement is the so called RUE-resolution due to Digricoli (Digricoli and Harrison, 1986).

There the prover is allowed to look ahead for paramodulation possibilities before going deeper applying the decomposition rule. In our example the prover is allowed to transform the unification problem $\{f(x), f(a)\}$ (called disagreement set in (Digricoli and Harrison, 1986)) into $\{f(x), f(c)\}$ using the fourth clause. This way the substitution $\{x \leftarrow c\}$ may be detected in fact. Nevertheless the inference rules are rather complicated and it is not known to the authors whether the overhead is not too much.

Another improvement is based on a level saturation strategy as given and proved to be complete in (Anderson, 1970). The connection graph calculus was extended with an equality handling in the spirit of RUE-resolution by means of equality graphs (EGC-procedure) in (Bläsius, 1987). A matrix calculus with a RUE-resolution-like inference rule has been proved to be complete in (Petermann, 1991c). A treatment of equality which is similar to the RUE-resolution was carried out for the tableau method (Fitting, 1990) in (Reeves, 1987).

4.2.3 Ordering Strategies

Except rigid $\mathcal{E}$-unification, for which experimental results are not yet available, all the methods described so far for equality handling could not prove to be sufficient for really hard practical problems. (Ohlbach and Siekmann, 1991) report various discouraging results with paramodulation and the EGC-procedure within the connection graph calculus. As a solution, recent research concentrates on the application of **ordering restrictions**. These methods can be seen as extensions of term rewriting and the Knuth-Bendix completion procedure (Knuth and Bendix, 1970). A good overview over term rewriting, completion and some of its extensions can be found in (Plaisted, 1993).

Calculi and proof procedures for full first order order-restricted equational theorem proving were proposed in (Bachmair, 1991; Bachmair and Ganzinger, 1990a; Bachmair and Ganzinger, 1990b; Zhang and Kapur, 1988; Hsiang and Rusinowitch, 1986; Hsiang and Rusinowitch, 1987; Kounalis and Rusinowitch, 1991; Rusinowitch, 1991; Kirchner et al., 1990; Socher-Ambrosius, 1990). These systems are basically paramodulation-like, however the possible inferences are highly restricted by orderings, which have to be given as an input parameter. More precisely, the given ordering $\succ$ must satisfy the following properties ("complete reduction ordering"):

1. $\succ$ is *stable under substitution*, i.e. $s \succ t$ implies $s\sigma \succ t\sigma$ for all s, t, σ.
2. $\succ$ is *monotonic*, i.e. $s \succ t$ implies $u[s] \succ u[t]$ for all s, t, u.
3. $\succ$ is ground-total.
4. $\succ$ is well-founded.

The inference rules below make heavy use of such a given ordering. One main application is to direct the use of equations, i.e. to allow only one side of an equation as a paramodulating term. Which side of an equation $t \doteq u$ is to be used, is determined by the ordering: if u

has to substitute the term t then u should be the smaller side of $t \doteq u$. However, in general only in the ground case can it be achieved that $s \neq t$ implies that either $s \succ t$ or $t \succ s$ (see (Dershowitz, 1987) for an overview about orderings). In order to compute with non-variable terms, the $\succ$-relation is "lifted" to the condition "$\not\succeq$"; this means that if in the ground case an inference were legal if $t \succ u$, then it is legal in the general case if $t \not\succeq u$. Similarly, for sets of literals the notion of a "greatest literal" lifts to that of a "maximal literal": a literal L is maximal relative to a set if for none of the literals K in that set holds $L \prec K$. Additionally the term ordering must be extended to equations and to literals; this can be done e.g. lexicographically.

Essentially the several order-restricted paramodulation calculi restrict the above paramodulation inference rule in the following way:

$$\frac{L[t'] \vee R_1 \qquad t \doteq u \vee R_2}{(L[t' \leftarrow u] \vee R_1 \vee R_2)\sigma}$$

if

1. $L[t']\sigma$ is maximal relative to $R_1\sigma$ and $(t \doteq u)\sigma$ is maximal relative to $R_2\sigma$ (only maximal literals in clauses may be selected for the inference; also comparison is done *after* application of σ, since after instantiation more information is available, possibly ruling out some inferences).
2. $t\sigma \not\succeq u\sigma$ (the substitution of $t\sigma$ by $u\sigma$ must not cause a growth of the literal) and if $L[t']\sigma$ is an equation $L[t']\sigma \equiv s[t']\sigma \doteq r\sigma$ then $s[t']\sigma \not\succeq r\sigma$ (it suffices to paramodulate into bigger sides of equations). In this case, the inference is called a *superposition* inference.
3. t' is not a variable occurrence.

Let us compute our example with this restricted variant. We need an ordering. For the sake of simplicity, assume that literals are ordered $male(s) \succ t_1 = t_2$ for any s, t_1, t_2, and for function symbols we order $firstchild \succ sue$, and assume e.g. a recursive path ordering based on this ordering (see (Dershowitz, 1987) for an overview about orderings). The ordering among the other function symbols does not matter. Suppose again that we want to refute

(7) $\neg y = father(sue)$.

According to restriction 3. we cannot paramodulate into y in (7). Furthermore we cannot paramodulate into sue with (5), because in (5) we have $firstchild(fred) \succ sue$ and such a step would violate restriction 2. Also we cannot use (4) to paramodulate into $father(sue)$ of (7), because in (4) $male(x) \succ father(firstchild(x)) = x$ and hence according to restriction 1. Only $\neg male(x)$ may be used in an inference. A legal step is to resolve (4) with (6), which yields:

(8) $father(firstchild(fred)) = fred$

Now (5) can be used to paramodulate into $firstchild(fred)$ in (8), which yields

(9) $father(sue) = fred$

Now paramodulating (9) into $father(sue)$ of (7), and a final resolution with $x = x$ completes the proof.

It should be indicated by this example that ordering restrictions are a very effective tool for disallowing many otherwise possible inferences. For example, the topmost step in the above paramodulation refutation is no legal inference in the restricted calculus, because it violates the restriction 1 and 2. As a general property these ordering restrictions seem to be only possible at the price of giving up the linear, top-down strategy as in the unordered case. However, as an improvement in this direction in (Socher-Ambrosius, 1992) a more goal-oriented completion strategy is proposed.

As a special case, the term ordering might be strong enough to direct all the equations into rewrite rules, which means that in inferences only the left side needs to be considered for replacement by the right side. Sometimes the input clauses, short of the query, can be closed under application of the above ordered inference rule and yield a *finite* clause set. This is easier to achieve if additionally simplification techniques are applied (see again (Bachmair and Ganzinger, 1991; Bachmair and Ganzinger, 1990b)). This finite set can be used to decide the word problem in the given equational theory. In the infinite case a semi-decision procedure results (see also (Hsiang and Rusinowitch, 1987) for such an "unfailing" procedure). This shows that such calculi are closely related to the Knuth-Bendix completion procedure; indeed, that procedure can be seen as an instance of them.

We are not aware of any research going on in a non-resolution setting.

5 Conclusion

We have presented a classification of theory reasoning methods for first order predicate calculi. We have distinguished between literal, term and variable level reasoning. The main focus of the paper is literal level theory reasoning, which is presented for various calculi in a uniform manner. For this we used the consolution calculus, which was defined in a partial and a total theory variant. This framework was then used in the main part of the paper to present theory resolution, theory model elimination and a theory connection method. For the total variants of these calculi we have proven completeness.

The advantages of such a uniform view of various calculi are both the possibility of using the same formal machinery for different calculi, and the ability to investigate differences in detail.

Since efficient equality handling is one of the central issues in theorem proving we have

discussed this in a special section. Again, we presented methods like paramodulation and RUE-resolution as special cases of partial theory reasoning in the framework of theory consolution.

References

Anderson, R. (1970). Completeness results for E-resolution. In *Proceedings AFIP 70, Spring Joint Comp. Conf., AFIPS Press, Reston VA*, pages 653–656.

Bachmair, L. (1991). *Canonical Equational Proofs*. Progress in Theoretical Computer Science. Birkhäuser.

Bachmair, L. and Ganzinger, H. (1990a). Completion of First-Order Clauses with Equality by Strict Superposition. In *Proc. Second Int. Workshop on Conditional and Typed Rewrite Systems, LNCS*. Springer.

Bachmair, L. and Ganzinger, H. (1990b). On Restrictions of Ordered Paramodulation with Simplification. In Stickel, M., editor, *Proc CADE 10, LNCS 449*, pages 427–441. Springer.

Bachmair, L. and Ganzinger, H. (1991). Rewrite-Based Equational Theorem Proving With Selection and Simplification. Technical Report MPI-I-91-208, Max-Planck-Institut für Informatik.

Bachmair, L., Ganzinger, H., Lynch, C., and Snyder, W. (1992). Basic Paramodulation and Superposition. In Kapur, D., editor, *Proc. 11th CADE*, pages 462–476. Springer.

Baumgartner, P. (1992a). A Model Elimination Calculus with Built-in Theories. In Ohlbach, H.-J., editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer. LNAI 671.

Baumgartner, P. (1992b). An Ordered Theory Resolution Calculus. In Voronkov, A., editor, *Logic Programming and Automated Reasoning (Proceedings)*, pages 119–130, St. Petersburg, Russia. Springer. LNAI 624.

Baumgartner, P. and Furbach, U. (1992). Consolution as a Framework for Comparing Calculi. Forschungsbericht 11/92, University of Koblenz. (to appear in *Journal of Symbolic Computation*).

Bibel, W. (1987). *Automated Theorem Proving*. Vieweg, 2nd edition.

Bibel, W. (1992). *Deduktion*, volume 6.2 of *Handbuch der Informatik*. Oldenburg.

Bläsius, K. (1987). Equality Reasoning Based on Graphs. SEKI-Report SR-87-01, Universität Kaiserslautern.

Bläsius, K. and Bürckert, H.-J. (1989). *Deduction Systems in Artificial Intelligence*. Horwood, Chichester.

Brachmann, R., Fikes, R., and Levesque, H. (1983). KRYPTON: a functional approach to knowledge representation. *IEEE Computer*, 16(10):67–73.

Bürckert, H. (1990a). A Resolution Principle for Clauses with Constraints. In Stickel, M. E., editor, *Proc CADE 10*, pages 178–192. Springer. LNCS/LNAI 449.

Bürckert, H.-J. (1990b). A Resolution Principle for Clauses with Constraints. Research Report RR-90-02, DFKI.

Bürckert, H.-J. (1992). *A Resolution Principle for Clauses with Restricted Quantifiers*, volume 568 of *LNAI*. Springer.

Bürckert, H.-J., Herold, A., Kapur, D., Siekmann, J., Stickel, M., Tepp, M., and Zhang, H. (1988). Opening the AC-Unification Race. *Journal of Automated Reasoning*, 4:465 – 474.

Chang, C. and Lee, R. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Debart, F., Enjalbert, P., and Lescot, M. (1990). Multi modal logic programming using equational and order-sorted logic. In Okada, M. and Kaplan, S., editors, *Proc. 2nd Conf. on Conditional and Typed Rewriting Systems*. Springer. LNCS.

Dershowitz, N. (1987). Termination of Rewriting. *Journal of Symbolic Computation*, 3(1&2):69–116.

Digricoli, V. J. and Harrison, M. C. (1986). Equality-Based binary Resolution. *Journal of the Association for Computing Machinery*.

Eder, E. (1991). Consolution and its Relation with Resolution. In *Proc. IJCAI '91*.

Eisinger, N. and Ohlbach, H.-J. (1993). Deduction systems based on resolution. In Gabbay, D. M., Hogger, C., and Robinson, J., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 1: Logical Foundations*, pages 184–263. Oxford Science Publications.

Fages, F. and Huet, G. (1986). Complete Sets of Unifiers and Matchers in Equational Theories. *Theoretical Computer Science*, 43.

Fitting, M. (1990). *First Order Logic and Automated Theorem Proving*. Texts and Monographs in Computer Science. Springer.

Furbach, U., Hölldobler, S., and Schreiber, J. (1989). Horn equational theories and paramodulation. *Journal of Automated Reasoning*, 3:309–337.

Gallier, J. (1987). *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Wiley.

Gallier, J., Narendran, P., Plaisted, D., and Snyder, W. (1990). Rigid E-unification: NP-Completeness and Applications to Equational Matings. *Information and Computation*, pages 129–195.

Gallier, J. and Snyder, W. (1989). Complete Sets of Transformations for General E-Unification. *Theoretical Computer Science*, 67:203 – 260.

Gallier, J. and Snyder, W. (1990). Designing Unification Procedures Using Transformations: A Survey. *Bulletin of the EATCS*, 40:273 – 326.

Gallier, J. H., Raatz, S., and Snyder, W. (1987). Theorem proving using rigid e-unification: Equational matings. In *Logics in Computer Science '87, Ithaca, New York*.

Hölldobler, S. (1989). *Foundations of Equational Logic Programming*, volume 353 of *Lecture Notes in Artificial Intelligence. Subseries of Lecture Notes in Computer Science*. Springer.

Hollunder, B. (1990). Hybrid Inferences in KL-ONE-based Knowledge Representation Systems. Research Report RR-90-6, DFKI.

Hsiang, J. and Rusinowitch, M. (1986). A New Method for Establishing Refutational Completeness in Theorem Proving. In *Proc. 8th CADE*, pages 141–152. Springer.

Hsiang, J. and Rusinowitch, M. (1987). On word problems in equational theories. In *Proc. ICALP'87*, pages 54–71. Springer, LNCS 267.

Hullot, J. (1980). Canonical forms and unification. In *Proc. Conf. Automated Deduction*, pages 318–334.

J.-P. Jouannaud, C. K. (1991). Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. In Lassez, J. and Plotkin, G., editors, *Computational Logic — Essays in Honor of Alan Robinson*, pages 257–321. MIT Press.

Jaffar, J. and Lassez, J.-L. (1987). Constrained Logic Programming. In *Proc. of the ACM Symp. on Principles of Programming Languages*, pages 111–119.

Jeffrey, R. C. (1967). *Formal Logic: Its Scope and Limits*. McGraw-Hill.

Kirchner, C., Kirchner, H., and Rusinowitch, M. (1990). Deduction with Sybolic Constrains. *Revue D'Intelligence Artificielle*, 4(3):11–51.

Knuth, D. E. and Bendix, Peter, B. (1970). Simple world problems in universal algebras.

Kounalis, E. and Rusinowitch, M. (1991). On Word Problems in Horn Theories. *Journal of Symbolic Computation*, 11:113–127.

Letz, R., Schumann, J., Bayerl, S., and Bibel, W. (1992). SETHEO: A High-Performace Theorem Prover. *Journal of Automated Reasoning*.

Loveland, D. (1968). Mechanical Theorem Proving by Model Elimination. *JACM*, 15(2).

Loveland, D. (1978). *Automated Theorem Proving - A Logical Basis*. North Holland.

McCharen, J., Overbeek, R., and Wos, L. (1976). Complexity and related enhancements for automated theorem-proving programs. *Computers and Mathematics with Applications*, 2:1–16.

Meseguer, P. (1989). Constraint Satisfaction Problems: An Overview. *AICOM*, 2(1).

Morris, J. B. (1969). E-Resolution: An Extension of Resolution to include the Equality Relation. In *Proc. IJCAI*, pages 287–294.

Murray, N. and Rosenthal, E. (1987). Theory Links: Applications to Automated Theorem Proving. *J. of Symbolic Computation*, 4:173–190.

Neugebauer, G. and Schaub, T. (1991). A Pool-Based Connection Calculus. Unpublished.

Oberschelp, A. (1962). Untersuchungen zur mehrwertigen Quantorenlogik. *Math. Annalen*, 145:297–333.

Ohlbach, H. and Siekmann, J. (1991). The Markgraf Karl Refutation Procedure. In Lassez, J. and Plotkin, G., editors, *Computational Logic — Essays in Honor of Alan Robinson*, pages 41–112. MIT Press.

Petermann, U. (1990). Towards a connection procedure with built in theories. In *JELIA 90*. European Workshop on Logic in AI, Springer, LNCS.

Petermann, U. (1991a). Building in equational theories into the connection method. In *Proceedings of Symposium on Fundamentals of Artificial Intelligence Research*, Smolenice.

Petermann, U. (1991b). How to build in an open theory into connection calculi. *submitted to J. on Computers and Artificial Intelligence*.

Petermann, U. (1991c). Petermann U., Building in Equational Theories into the Connection Method. In *Proceedings of FAIR '91, 1st Int. Workshop., Fundamentals of Artificial Intelligence Research, Smolenice, Czechoslovakia, Sept. q 8-13, 1991*, pages 185–199.

Petermann, U. (1993a). A framework for integrating equality reasoning into the extension procedure. In *Proceedings Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseille, 1993*, pages 195–207.

Petermann, U. (1993b). Completeness of the pool calculus with an open built in theory. In Gottlob, G., Leitsch, A., and Mundici, D., editors, *3rd Kurt Gödel Colloquium '93*, number 713 in Lecture Notes in Computer Science, pages 264–277. Springer-Verlag.

Peterson, G. (1983). A Technique for Establishing Completeness Results in Theorem Proving with Equality. *SIAM Journal on Computing*, 12(1):82–100.

Plaisted, D. (1993). Equational reasoning and term rewriting systems. In Gabbay, D. M., Hogger, C., and Robinson, J., editors, *Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 1: Logical Foundations*, pages 184–263. Oxford Science Publications.

Poppelstone, R. (1967). Beth tree methods in automated theorem proving. *Machine Intelligence*, pages 31–46.

Reeves, S. V. (1987). Adding equality to semantic tableaux. *Journal of Automated Reasoning*, 3:225–246.

Ringeissen, C. (1992). Unification in a combination of equational theories with shared constants and its application to Primal Algebras. In Voronkov, A., editor, *Proc. LPAR '92*. Springer.

Robinson, G. A. and Wos, L. (1969). Paramodulation and Theorem Proving in First Order Theories with Equality. In Meltzer and Mitchie, editors, *Machine Intelligence 4*. Edinburg University Press.

Rusinowitch, M. (1991). Theorem-proving with Resolution and Superposition. *Journal of Symbolic Computation*, 11:21–49.

Siekmann, J. and Wrightson, G. (1980). Paramodulated Connectiongraphs. *Acta Informatica*, 13:67–86.

Siekmann, J. H. (1989). Unification Theory. *Journal of Symbolic Computation*, 7(1):207–274.

Snyder, W. (1991). *A Proof Theory for General Unification*. Birkhäuser.

Snyder, W. and Lynch, C. (1991). Goal directed strategies for paramodulation. In Book, R., editor, *Rewriting Techniques and Applications*, Lecture Notes in Computer Science No. 488, pages 15 – 28, Berlin. Springer.

Socher-Ambrosius, R. (1990). Simplification and Reduction for Automated Theorem Proving. SEKI-Report SR-90-10, Universität Kaiserslautern.

Socher-Ambrosius, R. (1992). A Goal-Oriented Strategy Based on Completion. In Kapur, D., editor, *Proc. CADE 11*. Springer. LNAI 607.

Stickel, M. (1983). Theory Resolution: Building in Nonequational Theories. SRI International Research Report Technical Note 286, Artificial Intelligence Center.

Stickel, M. (1985). Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355.

Van Hentenryck, P. (1989). *Constraint Satisfaction in Logic Programming*. Logic Programming. The MIT Press, Cambridge, Massachusetts, USA / London, England, UK.

W. Nutt, P. R. and Smolka, G. (1987). Basic Narrowing Revisited. Seki-Report SR-87-07, Uni Kaiserslautern.

Walther, C. (1983). A Many-Sorted Calculus Based on Resolution and Paramodulation. In *Proc. 8th IJCAI*, Karlsruhe.

Zhang, H. and Kapur, D. (1988). First-Order Theorem Proving Using Conditional Rewrite Rules. In E. Lusk, R. O., editor, *Proc. 9th CADE, LNCS 310*, pages 1–20. Springer.