Refinements of Theory Model Elimination and a Variant without Contrapositives

Peter Baumgartner[1]

Abstract. Theory Reasoning means to build-in certain knowledge about a problem domain into a deduction system or calculus, such as model elimination. We present several complete versions of highly restricted theory model elimination (TME) calculi. These restrictions allow (1) to keep fewer path literals in extension steps than in related calculi, and (2) to discard proof attempts with multiple occurrences of literals along a path (i.e. regularity holds). On the other hand, we obtain by small modifications to TME versions which do not need contrapositives (à la Near-Horn Prolog). We show how regularity can be adapted for these versions. The *independence of the goal computation rule* holds for all variants. Comparative runtime results for our PTTP-implementations are supplied.

1 Introduction and Preliminaries

The *model elimination calculus* is a goal-oriented, linear and refutationally complete calculus for first order clause logic [15]. It is the base of numerous proof procedures for first order deduction. There are high speed theorem provers, like METEOR [1] or SETHEO [13] and there is a whole class of provers, namely Prolog technology theorem proving (PTTP) as introduced in [22], which rely on model elimination. In this paper we extend in several new ways model elimination towards theory reasoning.

Theory reasoning was introduced by M. Stickel within the general, non-linear resolution calculus [23]; for model elimination it is defined and investigated in [2]. Theory reasoning means to relieve a calculus from explicit reasoning in some domain (e.g. equality, partial orders, taxonomic reasoning) by taking apart the domain knowledge and treating it by special inference rules for theory reasoning ("background reasoner"). Theory reasoning is interesting because, first, proofs become more structured, and second, higher efficiency can be achieved by a clever reasoner that takes advantage of the theories' properties.

Theory reasoning is a very general scheme and has many applications, among them *reasoning with taxonomical knowledge* as in the Krypton system [10], building in *theory-unification* procedures and *equality reasoning* as by Paramodulation or E-resolution.

Another source for this work is the upcoming of various (non-theory) calculi which do not need all contrapositives.[2] *Near-Horn Prolog* ([16]) is motivated from the background of logic programming, and needs contrapositives for each positive literal in a clause only. The *(modified) simplified problem reduction format* [20] is more a theorem-proving system and needs only one single contrapositive per clause. A detailed comparison of these calculi and of another one, N-Prolog, can be found in [21]. In [7] we have made a small change to model elimination which also avoids contrapositives and allows for a PTTP-implementation.

The common idea behind all these calculi is to carry out *case analysis* wrt. the positive literals in a disjunctive clause. The initial goal then is to be proven for each case, and this kind of reasoning replaces the need for contrapositives. Proving by cases seems also to be a very natural way to do proofs. For example in proving theorems such as "if $x \neq 0$ then $x^2 > 0$" a human typically uses case analysis according to the axiom $X < 0 \lor X = 0 \lor -X < 0$.

Main Results. One major contribution of this paper are the various refinements of theory model elimination calculi ([2, 19]). One such refinement is the *saving of path literals*. Technically, the literals stemming from the extending clauses need in an extension step not to be included in the resulting paths. This is theoretically appealing because it gives a better understanding of the role of the different kinds of literals involved in inference steps; furthermore it is practically relevant since, occasionally, the time needed for finding a proof decreases as the local search space decreases. Such a restriction has not been consider before (cf. "related work" below).

Another improvement concerns the use of the *regularity restriction*. Regularity means, briefly, that in the course of a proof it is not necessary to derive an identical subgoal more than once. A nice property of this loop-detection feature is that it constitutes the base of a decision procedure for propositional logic. Regularity is easy to implement (at least approximatively) and it is one of the more effective restrictions for model elimination procedure. Our practical experiments (Section 4) strongly support this claim for the case of theory reasoning. While regularity is known to be compatible to non-theory model elimination ([12]), the corresponding result for the theory case has not been established so far.

While these modifications concern new refinements of the base calculus, we see the second main contribution in the design of a *theory model elimination calculus without contrapositives*. We will demonstrate that a slight change to the base calculus suffices to ensure completeness of the restriction to one single contrapositive per clause. Furthermore, the regularity restriction can be adapted in a completeness preserving way. Finally, the calculi may don't-care nondeterministically choose the next subgoal to be processed, i.e. the independence of the computation rule holds in a similar way as in Prolog's SLD-Resolution. The thus resulting calculus can be seen as an extension of the one presented in [7] towards theory reasoning. Furthermore, the PTTP implementation technique [22] used for that calculus can be extended to the theory case. See Section 4 for

[1] Universität Koblenz, Institut für Informatik, Rheinau 1, 56075 Koblenz, Germany, *E-mail:* `peter@infko.uni-koblenz.de`

[2] In systems *with* contrapositives n procedural counterparts (contrapositives) $L_i \, :- \, \overline{L}_1, \cdots, \overline{L}_{i-1}, \overline{L}_{i+1}, \cdots, \overline{L}_n$ for a clause $L_1 \lor \cdots \lor L_n$ have to be used. Each contrapositive represents a different entry point during the proof search into the clause. See [20] for a motivation for the avoidance of contrapositives.

experimental results.

Related Work. Theory reasoning was ported to many calculi. In [3] we showed that total theory resolution is compatible with ordering restrictions. Theory reasoning was defined for matrix methods in [17], for the connection graph calculus in [18] and for the connection method in [9, 19]. However there are significant differences between these calculi and model elimination: model elimination is a *linear* calculus, which means that an initially chosen goal clause is stepwisely processed until the refutation is found. In [2, 19] completeness results for theory model elimination can be found. However, these calculi neither make use of the above mentioned savings of path literals, nor do they incorporate the regularity restriction. As a consequence of all these considerations we need a new completeness proof and cannot use one of the existing ones.

Preliminaries. A *clause* is a multiset of literals, written as the disjunction $L_1 \vee \cdots \vee L_n$. As usual, clauses are considered implicitly as being universally quantified, and a clause set is considered logically as a conjunction of clauses. Instead of the *chains* of the original model elimination calculus we follow [6] and work in a branch-set setting. A *branch* is a finite sequence of literals, written by juxtaposing its constituents $L_1 L_2 \cdots L_n$. A *branch set* is a finite multiset of branches.

Concerning interpretations we restrict ourselves to *Herbrand-Interpretations* over a (most times implicitly) given finite signature Σ. Furthermore we suppose Σ to contain the 0-ary predicate symbol F which is to be interpreted by $false$ in every interpretation. We assume a *theory* to be given by a satisfiable set of universally quantified formulas, e.g. as a clause set (because a Herbrand-theorem holds). In the sequel $\mathcal{T}$ always denotes such a theory. As an example one might think of the theory of equality, given by the usual axioms. A *Herbrand $\mathcal{T}$-interpretation* I for a formula F is a Herbrand-interpretation over the joint signatures of $\mathcal{T}$ and F that satisfies $\mathcal{T}$, i.e. $I \models \mathcal{T}$. We write $I \models_{\mathcal{T}} F$ to indicate that I is a $\mathcal{T}$-interpretation and I satisfies F (i.e. I is a $\mathcal{T}$-model for F). Furthermore, F is called *$\mathcal{T}$-valid*, $\models_{\mathcal{T}} F$, iff every $\mathcal{T}$-interpretation satisfies F, and F is *$\mathcal{T}$-(un-)satisfiable* iff some (none) $\mathcal{T}$-interpretation satisfies F. As a consequence of these definition it holds $\models_{\mathcal{T}} F$ iff $\neg F$ is $\mathcal{T}$-unsatisfiable iff $\mathcal{T} \cup \{\neg F\}$ is unsatisfiable.

2 Theory Model Elimination Calculi

Theory reasoning comes in two variants [23]: *total* and *partial* theory reasoning. *Total* theory reasoning directly lifts the idea of finding syntactical complementary literals in inferences (e.g. resolution) to a semantic level. In *partial* theory reasoning semantical complementary sets are computed stepwise by means of intermediate goals (called "residues"). The partial variant is of particular interest for us because we have a technique to automatically construct background reasoners for partial theory model elimination [4].

Definition 2.1 (Theory model elimination (TME)) The inference rule *theory extension step* which transforms a branch set and some clauses in a new branch set is defined as follows:

$$\frac{\{K_1 \cdots K_m\} \cup \mathcal{Q} \qquad L_1 \vee R_1 \quad \cdots \quad L_n \vee R_n}{(\{K_1 \cdots K_m K \mid K \in Res \vee R_1 \vee \cdots \vee R_n\} \cup \mathcal{Q})\sigma} \ \textbf{Ext}$$

iff

1. $\{K_1 \cdots K_m\} \cup \mathcal{Q}$ is a branch set ($m \geq 1$, $K_1 \cdots K_m$ is called the *selected branch*, and K_m is called the *extended literal*), and
2. $L_i \vee R_i$ are clauses ($n \geq 0,\ i = 1 \ldots n$); the L_is are called *extending literals*, and R_i denotes the rest of the *extending clause* $L_i \vee R_i$, and
3. there exist indices $1 \leq j_1, \ldots, j_k \leq m-1$ and there exists a substitution σ such that

$$\models_{\mathcal{T}} \forall (K_{j_1} \wedge \cdots \wedge K_{j_k} \wedge K_m \wedge L_1 \wedge \cdots \wedge L_n \rightarrow Res)\sigma \qquad (1)$$

Here Res is a literal,[3] which is also called *residue* in this context. Following [23], the set $\{K_{j_1}, \ldots, K_{j_k}, K_m, L_1, \ldots, L_n\}$ is called the *key set* of the inference.

An extension step with $Res \equiv \mathsf{F}$ is called *total*, otherwise it is called *partial*. As a second inference rule we define *deletion step*:

$$\frac{\{K_1 \cdots K_m \mathsf{F}\} \cup \mathcal{Q}}{\mathcal{Q}} \ \textbf{Del}$$

Thus the deletion step allows to remove branches which are detected as contradictory (by means of a concluding F). The calculus of *total* theory model elimination (TME) consists of the inference rules *total theory extension step* and *deletion step*, and *partial* TME additionally consists of *partial theory extension step*.

A *(total, partial) TME derivation of $\mathcal{Q}_n$ from a clause set M* then is defined as a sequence ($n \geq 1$) $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ such that

1. $\mathcal{Q}_1 = \{L_1, \ldots, L_n\}$, i.e. a multiset of branches of length 1, for some $L_1 \vee \cdots \vee L_n \in M$; this clause is also called the *query*, and
2. for $i = 2 \ldots n$, $\mathcal{Q}_i$ is obtained (2a) either by applying the **Ext** inference rule to $\mathcal{Q}_{i-1}$ and some new variants of clauses from M, or else (2b) by applying the **Del** inference rule to $\mathcal{Q}_{i-1}$.

A *refutation of M* is a derivation of the empty branch set $\{\}$ from M.

A branch is called *regular* iff all the literals occuring in it are pairwise distinct (i.e. the branch contains no duplicate literals). A branch set is *regular* iff every of its branches is regular. A derivation is called *regular* iff every branch set of the derivation is regular. □

Informally, the implication (1) in Definition 2.1 means (roughly) that the residue is a logical consequence of some literals along the branch (including the extended literal) and the extending literals. This condition is needed to obtain a sound calculus.

In the full paper we require the implication (1) to be *minimal* (i.e. after deleting any element, (1) would no longer hold). For example, if the theory is "equality", the selected branch is $Pa\ \ Pb$ and the extending literals are $\{a = c, \neg Pc\}$ then a total extension step with key set $\{Pa, Pb, a = c, \neg Pc\}$ is possible. However, the implication $Pa \wedge Pb \wedge a = c \wedge \neg Pc \rightarrow \mathsf{F}$ is not minimal, as with deleting the extended literal $K_m = Pb$ the resulting implication $Pa \wedge a = c \wedge \neg Pc \rightarrow \mathsf{F}$ is still valid. Thus, this proposed extension step would not be allowed. In general, in implementations the local search space can be pruned considerably, as the minimality restriction allows to guide the search for the extending literals around the extended literal K_m.

Example 2.2 Consider the ground clause set $M = \{\neg A,\ B \vee C,\ D \vee \neg C\}$ and the theory $\mathcal{T} = \{B \rightarrow A,\ C \wedge D \rightarrow A\}$. Figure 1 contains a partial TME refutation of M with query $\neg A$. □

[3] Residues can be generalized to clauses as in [23], if it is of interest.

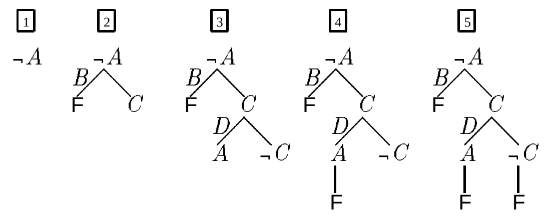

Figure 1. A TME refutation of M in a tree notation following [13]. $\boxed{1}$ is the tree resulting from the query $\neg A$. $\boxed{2}$ is obtained by a total extension step with $B \vee C$, making use of the fact that $\neg A \wedge B \rightarrow \mathsf{F}$ is $\mathcal{T}$-valid (because $(\neg A \wedge B \rightarrow \mathsf{F}) \equiv (B \rightarrow A)$). In this extension step (and similarly below) we have decorated the edge with the extending literal. $\boxed{3}$ is obtained by a partial extension step with clause $D \vee \neg C$, using the $\mathcal{T}$-validity of $C \wedge D \rightarrow A$. $\boxed{4}$ is obtained by a total extension step with the ancestor literal $\neg A$, which is possible since $\neg A \wedge A \rightarrow \mathsf{F}$ is valid in every theory. $\boxed{5}$ is obtained in a similar way.

Note that in extension steps the extending literals are not contained anywhere in the new branches. This is a difference to the works in [2] and [19] where these literals are appended to the extended branch one after the other.

So far, the **Ext**- and **Del**-inference rules operate on *some* selected branch. This would mean for implementations that choosing the selected branch is subject to backtracking. Clearly we would like to avoid this if possible. Indeed we have a free choice regarding the selected branch. As a further advantage of such a result, the selected branch can be choosen heuristically. Occasionally, *factoring* can be applied more successfully (see [12]) if such a subgoal reordering is allowed.

For the formalization, we borrow the notion of a "computation rule" from logic programming ([14]):

Definition 2.3 A *computation rule* is a function c from the set of branch sets to the set of branches such that $c(\mathcal{Q}) \in \mathcal{Q}$. Thus A computation rule can be used in derivations to determine the selected branch for the next inference step; we say that a derivation is a *derivation wrt. c* iff the selected branch in every of its inference steps is determined by c. □

For example, if a Prolog-like computation rule is desired, then always some longest branch is to be selected.

The following lemma is crucial; it can be proved for any of the calculi defined in this paper.

Lemma 2.4 (Independence of the computation rule – ground case) *Let $\mathcal{T}$ be a theory and M be a ground clause set. Suppose there exists a refutation of M with top clause C. Then for every computation rule c there exists a refutation of M wrt. c.*

As a consequence of our completeness proof techniques, which is a traditional ground-proof and lifting technique, it suffices to proof this result for the ground case only. The respective result for the first-order case with variables would be technically much more complicated (cf. [14]).

Completeness is stated for the ground case only. Although not quite trivial, lifting to the first order case can be carried out by generalizing standard techniques (see e.g. [11, 14]).

Theorem 2.5 (Ground Completeness of Regular Total TME) *Let $\mathcal{T}$ be a theory, c be a computation rule and let M be a $\mathcal{T}$-unsatisfiable ground 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 regular total TME refutation of M wrt. c with query C.*

For the ground proof the *excess literal parameter* technique can be used in a similar way as it is done for the non-theory version in [7]. Regularity is shown by restricting for every branch to be closed the available input clause set to those clauses that are free of literals occuring in that branch.

A similar completeness result exists for *partial* theory model elimination. The difficulty with the partial variant is that usually in extension steps not all $\mathcal{T}$-valid implications (condition 3 in definition of extension step) shall be considered for extension steps. Take e.g. equality: The paramodulation inference rule for equational reasoning is an instance of partial TME, but a paramodulation step does not compute all equational consequences of the given key set. Such restrictions can be formulated within the general framework defined in the full version ([5]). This framework allows to specify a "filtering out" of $\mathcal{T}$-valid implications which are not relevant for a derivation.

An important class of theories are *definite theories*, i.e. theories that are axiomatized by a set of definite clauses.[4] For such theories we find a special structure of implications:

Proposition 2.6 *For every definite theory $\mathcal{T}$ and for every $\mathcal{T}$-valid implication $\forall(L_1 \wedge \cdots \wedge L_n \rightarrow L_{n+1})$ it holds that either (1) all literals $L_1, \ldots, L_n$* and *the conclusion L_{n+1} are positive, or (2) one single literal L_i ($1 \leq i \leq n$) is negative* and *L_{n+1} is either negative or F.*

This special structure of implications can immediately be imposed on the corresponding implications in the definition of extension steps above. If additionally the input clause set is "Horn" then the ancestor literals of leafs need not to be stored and all extending literals are positive. This result generalizes a well-known property of ordinary model elimination, which states that no reduction steps are required (even possible) in the Horn case.

3 Restart Theory Model Elimination

Let us now modify the calculus given above, such that only a single contrapositive per clause is needed. The modifications work for both the total as well as the partial variant. Hence we will give up the distinction in this section.

For the modifications we have to presuppose *definite* theories (see the preceeding section). Furthermore, in order to get a complete calculus, we have to assume that there exists only one clause containing only negative literals, which furthermore does not contain variables, and this clause is to be used as query in refutations.[5] We note that Theorem 2.5 can be adapted for this new setting.

Definition 3.1 (Restart Theory Model Elimination) First, extension steps are restricted to operate on negative leafs only. For this define a *definite theory extension step* in the same way as theory extension step, except that additionally K_m is required to be a *negative* literal and all other key set literals $K_{j_1}, \ldots, K_{j_k}, L_1, \ldots, L_n$ are required to be positive literals.

In order to deal with *positive* leafs K_m define the inference rule *restart step* as follows:

[4] A definite clause contains exactly one positive literal.

[5] Without loss of generality this can be achieved by introducing a new clause $\leftarrow$ *goal* and transforming every purely negative clause $\neg B_1 \vee \cdots \vee \neg B_m$ into *goal* $\leftarrow B_1 \wedge \cdots \wedge B_m$.

$$\frac{\{K_1 \cdots K_m\} \cup \mathcal{Q}}{\{K_1 \cdots K_m K_1\} \cup \mathcal{Q}} \ \textbf{Rest}$$

iff K_m is a positive literal and $K_m \neq \mathsf{F}$.

The *strict Restart TME* calculus consists of the inference rules *definite theory extension step*, *restart step* and *deletion step*. The *non-strict* version consists additionally of *theory extension step*. □

If clauses are written in a programming language style $A_1, \cdots, A_n \leftarrow B_1, \cdots, B_m$ then it is clear that, for syntactical reasons, definite extension steps are possible only with head literals A's, but not with B's from the body. Thus it is possible to represent clauses as above *without* the need of augmenting them with all contrapositives; only contrapositives with conclusions (i.e. entry points) stemming from the positive literals are necessary. The price of the absence of contrapositives is that whenever a path ends with a positive literal, the root of the tree has to be copied. Then, the *only* inference applicable to that branch is a definite extension step. Occasionally a shorter refutation exists if non-definite extension steps are allowed as well; this motivates the need for the non-strict version.

Example 3.2 Consider again the (definite) theory $\mathcal{T}$ and ground clause set M of example 2.2 (Section 2). In Figure 1 the partial extension step from $\boxed{2}$ to $\boxed{3}$ is not allowed in Restart TME since the leaf C of the extended branch $\neg AC$ is positive. A restart step has to take place instead. Figure 2 contains a respective strict Restart TME refutation. □

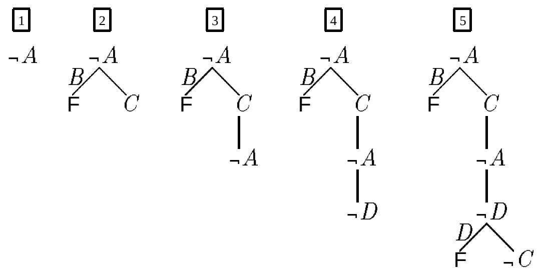

Figure 2. A strict partial Restart TME refutation of M (defined in Example 2.2 above). $\boxed{1}$ and $\boxed{2}$ are obtained as in Figure 1. $\boxed{3}$ is obtained by a restart step. $\boxed{4}$ is obtained by a partial definite extension step at $\neg A$ with the ancestor literal C and residue $\neg D$, using the $\mathcal{T}$-validity of $\neg A \wedge C \rightarrow \neg D$. $\boxed{5}$ is obtained by a total definite extension step with clause $D \vee \neg C$. The final deletion of branches ending in F is not depicted.

The restart calculus can further be weakened by introducing a *selection function* in the following way: a *selection function f* maps a clause $A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m$ with $n \geq 1$ to a literal $L \in \{A_1, \ldots, A_n\}$. Additionally, f is required to be *stable under lifting* which means that if f selects $L\gamma$ in the instance of the clause $(A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m)\gamma$ (for some substitution γ) then f selects L in $A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m$.[6] Now, for (definite) extension steps only selected literals may serve as extending literals. Thus, operationally, only one single contrapositive per input clause is needed. Notably, all this is complete:

[6] This property is needed for the completeness proof; it guarantees for lifting that the selection function will select on the first order level a literal whose ground instance was selected at the ground level.

Theorem 3.3 (Completeness of strict Restart TME) *Let M be a clause set with one single negative ground clause G, f be a selection function and c be a computation rule. Then there exists a strict Restart TME refutation of M with goal G.*

Proof. (Sketch) The proof is by splitting the given non-Horn clause set into Horn sets and assuming by completeness of TME respective refutations without reduction steps (cf. Proposition 2.6 and the remarks following the proposition). For the goal clause choose G in every refutation. Then these refutations are assembled back into the desired Restart TME refutation. There, reduction steps come in by replacing extension steps with splitted unit clauses by reduction steps to the literals where the restart occurred. □

The regularity restriction as it is usually defined for the non-restart versions (cf. Def. 2.1, "no literal occurs more than once in a branch") does no longer hold for Restart TME. This is rather easy to see since after a restart step it might be necessary to repeat – in parts – a refutation derived so far up to the restart step.

By analyzing in the proof of the previous theorem the process of assembling the refutations of the splitted Horn sets, as well as by certain semantical considerations, we learn that we can demand *blockwise regularity*. For this call the first literal of a branch the *goal literal*. Then a branch p is called *blockwise regular* iff (1) every pair of identical literals (unequal to the goal literal) in p is separated by at least one occurence of the goal literal, and, (2) all the positive literals occuring in p are pairwise distinct. For example, the refutation in Figure 2 is blockwise regular. Fortunately it holds:

Theorem 3.4 *Strict Restart TME with selection function is complete when restricted to blockwise regular refutations.*

4 Practical Experiments

We have implemented the calculi variants described in the preceeding sections. The system, called PROTEIN (*PRO*ver with a *T*heory *E*xtension *IN*terface, [8]), is implemented using the *PTTP implementation technique* ("Prolog Technology Theorem Prover", [22]) in ECLiPSe Prolog.

The implementation of *partial* theory reasoning is currently tailored for the method of *linearizing completion* [4]. Linearizing completion is a saturation technique that transforms a given Horn clause set $\mathcal{T}$ into a "completed" set, which admits (in resolution terminology) both linear and unit-resulting proofs for $\mathcal{T}$-unsatisfiable literal sets. Such a system then can be used as complete background reasoner for partial theory model elimination.

We ran several examples known from the literature with PROTEIN and a high-performance model elimination prover. Table 3 contains the runtime results (in seconds), obtained on a SPARC 10/40. The first four columns refer to different versions of PROTEIN. Column 5 contains data for Setheo [13] in its latest version (Version 3.0).

PROTEIN was run in default mode, except where indicated in Table 3. In default mode it includes the regularity restriction and the ground-reduction refinement. Setheo was also run in its default mode, which then makes use of the following refinements and constraints: subgoal reordering, purity, anti-lemmas, regularity, tautology and subsumption.

The example referred to as *Non-obvious* is taken from the October 1986 Newsletter of the *Association of Automated Reasoning*.[7] The selected theory here consists of a transitive and symmetric relation

[7] Entries such as `MSC/MSC006-1` refer to the respective TPTP-names [24]. All examples were drawn from that problem library without modification — only the theory part had to be selected by hand.

Example	ME	Restart-ME	TME	Restart-TME	Setheo
Non-obvious `MSC/MSC006-1`	0.3	2.7	1.6	1.1 0.15^{1}	0.5
Graph	10.8	∞	0.2	7.0 0.8^{2}	
$x \neq 0 \rightarrow x^2 > 0$	2.4	0.7	2.2	0.6	0.8
Pelletier 48 `SYN/SYN071-1`	5.9	1.2 0.6^{2}	0.4	0.9 $0.1^{1,2}$	0.2
Pelletier 49 `SYN/SYN072-1`	∞	297	1.6	1.5	∞
Pelletier 55 `PUZ/PUZ001-2`	392	∞	21	254	3.5
Lion&Unicorn `PUZ/PUZ005-1`	588	∞	21	∞	47
Wos 4 `GRP/GRP008-1`	22	20	0.3	26	13
Wos 10 `GRP/GRP001-1`	∞	-	14	-	850
Wos 11 `GRP/GRP013-1`	9.6	-	1.1	-	0.7
Wos 15 `GRP/GRP035-3`	384	-	47	-	478
Wos 16 `GRP/GRP036-3`	302	-	0.02	-	13
Wos 17 `GRP/GRP037-3`	∞	-	0.1	-	23

Entries: Numbers: runtimes (seconds) – ∞ no proof within reasonable time bound – "-" Not applicable –
Remarks: 1 – With selection function, 2 – With (back) factoring, 3 – With Lemmas

Figure 3. Runtime Results for various provers: *ME* – plain model elimination version of PROTEIN; *Restart-ME* – case-analysis style reasoning; *TME* and *Restart-TME* – respective versions with theory reasoning extensions.

p and a transitive relation q. In the *Graph* example a graph with a transitive and symmetric reachability relation is traversed. The example referred to by $x \neq 0 \rightarrow x^2 > 0$ is to prove this theorem (x is universally quantified) from calculus. Case analysis is carried out according to the axiom $X > 0 \quad \vee \quad X = 0 \quad \vee \quad -X > 0$.

For the theory variants of PROTEIN, the background calculus was obtained completely automatical by the linearizing completion tool in a preprocessing phase. For the theories we have selected appropriate Horn-subsets of the input clauses. The runtime of the linearizing completion tool was sufficiently small and need not be mentioned. In case linearizing completion would yield an infinite inference system for background reasoning – in the Wos examples from group theory – a finite approximation was used. The selected theory consists here of equality (except function substitution axioms) and the associativity of the group operation.

Concerning the search strategy we used iterative deepening with the costs of extension steps uniformly set to 1. The same costs are used for case analysis steps.

In the experiments the regularity restrictions turned out to be very advantageous for most examples. Both the non-restart and the restart-variants of TME are useful, with non-restart being in the average the preferrable choice. In general, the runtime results suggest to us that theory reasoning is a significant technique for automated theorem proving.

Acknowledgements. I would like to thank my colleagues J. Dix, U. Furbach and F. Stolzenburg for numerous discussions concerning theory model elimination, as well as for reading earlier versions of this paper.

REFERENCES

[1] Owen L. Astrachan and Mark E. Stickel, 'Caching and Lemmaizing in Model Elimination Theorem Provers', in *Proceedings of the 11th International Conference on Automated Deduction (CADE-11)*, ed., D. Kapur, pp. 224–238. Springer-Verlag, (June 1992). LNAI 607.

[2] P. Baumgartner, 'A Model Elimination Calculus with Built-in Theories', in *Proceedings of the 16-th German AI-Conference (GWAI-92)*, ed., H.-J. Ohlbach, pp. 30–42. Springer, (1992). LNAI 671.

[3] P. Baumgartner, 'An Ordered Theory Resolution Calculus', in *Logic Programming and Automated Reasoning (Proceedings)*, ed., A. Voronkov, pp. 119–130, St. Petersburg, Russia, (July 1992). Springer. LNAI 624.

[4] P. Baumgartner, 'Linear Completion: Combining the Linear and the Unit-Resulting Restrictions', Research Report 9/93, University of Koblenz, (1993).

[5] P. Baumgartner, 'Refinements of Theory Model Elimination and a Variant without Contrapositives', Research Report 8/93, University of Koblenz, (1993).

[6] P. Baumgartner and U. Furbach, 'Consolution as a Framework for Comparing Calculi', *Journal of Symbolic Computation*, **16**(5), (1993).

[7] P. Baumgartner and U. Furbach, 'Model Elimination without Contrapositives', in *Proc. 12th International Conference on Automated Deduction*. Springer, (1994).

[8] P. Baumgartner and U. Furbach, 'PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface', in *12th International Conference on Automated Deduction*. Springer, (1994).

[9] W. Bibel, *Automated Theorem Proving*, Vieweg, 2nd edn., 1987.

[10] R. Brachman, V. Gilbert, and H. Levesque, 'An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of Krypton', in *Proc. IJCAI*, (1985).

[11] C. Chang and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[12] R. Letz, K. Mayr, and C. Goller, 'Controlled Integrations of the Cut Rule into Connection Tableau Calculi', *Journal of Automated Reasoning*, (1994). (to appear).

[13] R. Letz, J. Schumann, S. Bayerl, and W. Bibel, 'SETHEO: A High-Performace Theorem Prover', *Journal of Automated Reasoning*, **8**(2), (1992).

[14] J. Lloyd, *Foundations of Logic Programming*, Symbolic Computation, Springer, second, extended edn., 1987.

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

[16] D.W. Loveland, 'Near-Horn Prolog', in *Proc. of the 4th Int. Conf. on Logic Programming*, ed., J.-L. Lassez, pp. 456–469. The MIT Press, (1987).

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

[18] Hans Jürgen Ohlbach, 'Link Inheritance in Abstract Clause Graphs', *Journal of Automated Reasoning*, **3**(1), 1–34, (1987).

[19] U. Petermann, 'Completeness of the pool calculus with an open built in theory', in *3rd Kurt Gödel Colloquium '93*, eds., Georg Gottlob, Alexander Leitsch, and Daniele Mundici, number 713 in Lecture Notes in Computer Science, pp. 264–277. Springer-Verlag, (1993).

[20] D. Plaisted, 'Non-Horn Clause Logic Programming Without Contrapositives', *Journal of Automated Reasoning*, **4**, 287–325, (1988).

[21] D. W. Reed and D. W. Loveland, 'A Comparison of Three Prolog Extensions', *Journal of Logic Programming*, **12**, 25–50, (1992).

[22] M. Stickel, 'A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler', *Journal of Automated Reasoning*, **4**, 353–380, (1988).

[23] M.E. Stickel, 'Automated Deduction by Theory Resolution', *Journal of Automated Reasoning*, **1**, 333–355, (1985).

[24] G. Sutcliffe, C. Suttner, and T. Yemenis, 'The TPTP problem library', in *Proc. CADE-12*. Springer, (1994).