Elimination of Equality via Transformation with Ordering Constraints

Leo Bachmair
Harald Ganzinger
Andrei Voronkov

MPI–I–97–2–012 December 1997

Authors' Addresses

Leo Bachmair
Department of Computer Science, SUNY at Stony Brook,
Stony Brook, NY 11794, U.S.A.
email: `leo@cs.sunysb.edu`, web: `www.cs.sunysb.edu/~leo`

Harald Ganzinger
Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken, Germany
email: `hg@mpi-sb.mpg.de`, web: `www.mpi-sb.mpg.de/~hg`

Andrei Voronkov
Computing Science Department, Uppsala University,
Box 311, S 751 05 Uppsala, Sweden
email: `voronkov@csd.uu.se`, web: `www.csd.uu.se/~voronkov`

Publication Notes

A shorter version of this report has been submitted for publication elsewhere.

Acknowledgements

This research was supported in part by the National Science Foundation under grants INT-9314412 and CCR-9510072, by the German Science Foundation (Deutsche Forschungsgemeinschaft) under grants Ga 261/7-1 and Ga 261/8-1, and by the ESPRIT Basic Research Working Group 22457 (Construction of Computational Logics II).

Abstract

We refine Brand's method for eliminating equality axioms by (i) imposing ordering constraints on auxiliary variables introduced during the transformation process and (ii) avoiding certain transformations of positive equations with a variable on the right-hand side. The refinements are both of theoretical and practical interest. For instance, the second refinement is implemented in Setheo and appears to be critical for that prover's performance on equational problems. The correctness of this variant of Brand's method was an open problem that is solved by the more general results in the present paper. The experimental results we obtained from a prototype implementation of our proposed method for the model elimination prover Protein also show some dramatic improvements of the proof search. Ordering constraints have already been widely used in equational theorem provers based on paramodulation. We prove the correctness of our refinements of Brand's method by establishing a suitable connection to basic paramodulation calculi and thereby shed new light on the connection between different approaches to equational theorem proving.

Keywords

Automated Theorem Proving, Equality Handling, Ordering Constraints, Model Elimination

Contents

1 Introduction

Efficient techniques for handling equality are a key component of automated reasoning systems. The most successful approaches to date are based on refinements of paramodulation, such as the superposition calculus, but these are unfortunately not fully compatible with tableau-based provers or model elimination methods. Various attempts have been made recently to improve the handling of equality in such provers (Moser, Lynch & Steinbach 1995, Degtyarev & Voronkov 1996*b*, Degtyarev & Voronkov 1996*a*), but they usually require subtle interactions between paramodulation-based and model elimination-based subcomponents and therefore are difficult to integrate into existing provers. Most current model elimination provers rely instead on preprocessing steps that transform formulas from logic with equality into logic without equality, see the survey (Schumann 1994).

The best-known transformation method is Brand's modification method (Brand 1975), which consists of three steps. First, terms are flattened by introducing new auxiliary variables, so that only variables occur as arguments of functions symbols. The axioms expressing the monotonicity properties of equality are not needed for the resulting flat clauses. Second, all symmetric variants of a clause (which are obtained by switching the arguments of equations) are added to the given set of clauses, so that the symmetry axioms of equality may be dispensed with. Third, the transitivity axioms are internalized by splitting positive equations $s{\approx}t$ into (clauses that represent) implications $t \approx x \to s \approx x$ with a new auxiliary variable x, called a "link" variable.

In this article, we improve Brand's modification in various ways. We systematically add ordering constraints during the transformation process, so as to be able to better control the theorem proving process on the transformed clauses. For example, a link variable x will be constrained via $s \succ x$ and $t \succeq x$ to terms smaller than s and smaller than or equal to t. Ordering constraints intuitively reflect assumptions about the form of equational proofs of $s{\approx}t$ and are related to rewrite techniques as used in paramodulation and superposition calculi. The rationale for transitivity elimination is that a sequence of equational replacements

$$s = s_0 \approx s_1 \approx \ldots \approx s_n = t$$

(using equations $s_i \approx s_{i+1}$) can be simulated by a sequence of resolution inferences from the goal clause $s \not\approx z \vee t \not\approx z$ and (clauses representing the) equivalences $s_i \approx x_i \leftrightarrow s_{i+1} \approx x_i$, plus a final resolution step with the reflexivity axiom $x \approx x$ that instantiates the link variables. The ordering constraints ensure that the variables x_i can only be instantiated by *minimal* terms among the s_i and block the search for alternative equational proofs that apply the same equations but differ in the instantiation of the link variables.

Aside from the ordering constraints, we also propose more subtle changes to the transformation process. In particular, we never split a positive equation $t \approx x$ where the right-hand side is already a variable. This may seem to be a minor technical aspect, but the optimization (Moser & Steinbach 1997) has been implemented in the Setheo model elimination theorem prover and is crucial for that prover's successful performance on many equational problems (Ibens & Letz 1997).[1] The completeness of this optimization (without any ordering constraints) had been an open problem[2] that follows from the more general results in the present paper. Our completeness proof is comparatively simple, but draws on rather non-trivial results about basic superposition, some of which have been obtained only very recently (Bachmair & Ganzinger 1997*b*). In essence, we show

[1]The optimized transformation avoids the generation of negative equations $x \not\approx y$ between two variables. Model elimination or resolution inferences with such literals correspond to paramodulation inferences into or from variables, most of which are redundant and ought to be avoided.

[2]The proof in (Moser & Steinbach 1997) contains a non-trivial gap which this paper closes.

how refutational proofs by strict basic superposition with flat clauses can be simulated by resolution with the corresponding transformed clauses. In addition to the theoretical results, we also report on experiments with model elimination theorem proving that provide encouraging evidence for the practical usefulness of the proposed constrained equality elimination method.

The structure of the paper is as follows. The next section contains preliminary material. The transformations are described in Section 3 and the main results about them in Section 4. In Section 5 we discuss the connection with other transformation methods, while in Section 6 we report on experiments with our method.

2 Preliminaries

The transformations described below will be applied to clauses with equality. We use the the symbol $\approx$ to denote the equality predicate and assume, for simplicity, that this is the only predicate in the original language.[3] A different symbol $\simeq$ is used to denote the predicate that replaces equality as part of the transformation process. Semantically, the difference between the two symbols is that $\approx$ is interpreted as a congruence relation, whereas no restrictions are imposed on the interpretation of $\simeq$. In other words, the original formulas with $\approx$ are interpreted in a logic with equality, whereas the transformed formulas with $\simeq$ are interpreted in a logic without equality. The aim is to design transformations so that the original clause set is satisfiable in an equality interpretation if, and only if, the transformed clause set is satisfiable in general.

Formally, a *clause* is a disjunction of literals; a literal being either an atomic formula or the negation thereof. Disjunction is associative and commutative, and hence clauses may be viewed as multisets of literals. The *empty clause* is denoted by $\Box$. By an *equational clause* we mean a clause that contains only $\approx$, but not $\simeq$. Satisfiability and logical consequence (denoted by $\models$) are defined in the usual way, with the proviso that the interpretation of $\approx$ has to be a congruence (while $\simeq$ may be interpreted as an arbitrary binary relation).[4]

Substitutions will be denoted by the letters σ, τ and ρ. The result of applying a substitution σ to an expression (e.g., a clause or term) E is denoted $E\sigma$. We write $E[s]$ to indicate that s is a subterm of E and write $E[t]$ to denote the expression obtained from E by replacing one specified occurrence of s by t. We also write $E(s)$ to indicate that s occurs in E and denote by $E(t)$ the result of simultaneously replacing *all* occurrences of s in E by t.

A *constraint* is a, possibly empty, conjunction of atomic formulas $s = t$ (called an *atomic equality constraint*) or $s \succ t$ or $s \succeq t$ (called *atomic ordering constraints*). The empty conjunction is denoted by $\top$. The letters γ and δ are used to denote constraints. A *constrained clause* is a pair of a clause C and a constraint γ, written as $C \cdot \gamma$. We call C the *clause part* and γ the *constraint part* of $C \cdot \gamma$.

A substitution σ is said to be a *solution of an atomic equality constraint* $s = t$ if $s\sigma$ and $t\sigma$ are syntactically identical. It is a *solution of an atomic ordering constraint* $s \succ t$ (with respect to a reduction ordering $>$ that is total on ground terms) if $s\sigma > t\sigma$; and a solution of $s \succeq t$ if it is a solution of $s = t$ or $s \succ t$. Finally, we say that σ is a *solution of a general constraint* γ if it is a solution of all atomic constraints in γ. A constraint is *satisfiable* if it has a solution.

A *ground instance* of a constrained clause $C \cdot \gamma$ is any ground clause $C\sigma$ such that the constraint $\gamma\sigma$ is satisfiable. A constrained clause $\mathcal{C}$ is *more general* than a constrained clause $\mathcal{D}$, denoted $\mathcal{D} \sqsubseteq \mathcal{C}$, if every ground instance of $\mathcal{D}$ is also a ground instance of $\mathcal{C}$. We call two constrained clauses $\mathcal{C}$ and

[3]The extension of the various transformation steps to non-equality literals is straightforward.

[4]On one or two occasions we will explicitly relax the restriction on the interpretation of $\approx$.

$\mathcal{D}$ *equivalent* if $\mathcal{C} \subseteq \mathcal{D}$ and $\mathcal{D} \subseteq \mathcal{C}$, i.e. when $\mathcal{C}$ and $\mathcal{D}$ have the same ground instances. A variable renaming is a (not necessarily injective) subsitution mapping variables to variables. Two syntactic expressions E and E' are said to have the *same skeleton* written $E \equiv E'$ whenever $E'\rho = E\rho$, for some renaming ρ.

Constraints γ_1 and γ_2 are *equivalent* with respect to a set V of variables if for every solution σ_1 of γ_1 there exists a solution σ_2 of γ_2 such that σ_1 and σ_2 agree on the variables in V, and vice versa. We shall identify constrained clauses $C \cdot \gamma_1$ and $C \cdot \gamma_2$ when the constraints γ_1 and γ_2 are equivalent with respect to the variables in C. In this case $C \cdot \gamma_1$ and $C \cdot \gamma_2$ are equivalent. We identify a constrained clause $C \cdot \top$ with the unconstrained clause C. A *contradiction* is a constrained clause $\square \cdot \gamma$ with an empty clause part such that the constraint γ is satisfiable. A clause is called *void* if its constraint is unsatisfiable. A void clause has no ground instances and therefore is redundant.

A set S of constrained clauses is *satisfiable* if the set of all its ground instances is satisfiable. Evidently, removal of void clauses and replacement of clauses by equivalent ones preserves the (un)satisfiability of S.

If $\mathcal{I}$ is an inference system and N is a set of clauses then $\mathcal{I}(N)$ denotes the set of clauses that can be derived by applying an inference rule in $\mathcal{I}$ to premises in N. Likewise, $\mathcal{I}^*(N)$ denotes the set of clauses that can be derived from N by repeated application of inferences in $\mathcal{I}$. In all calculi of this paper the premises of inference rules are assumed to have disjoint variables, which can be achieved by renaming.

3 Transformations

Given a set of equational clauses N, we apply various transformation rules and replace the equality predicate $\approx$ by the predicate $\simeq$ to obtain a modified clause set N', such that the transformed set N' is satisfiable if, and only if, the original set N is *equationally* satisfiable. Each part of the transformation process is designed to eliminate certain equality axioms and can be described by a set of (schematic) transformation rules to be applied to clauses. If R is a set of such transformation rules, we say that a (constrained) clause is in *R-normal form* if no rule in R can be applied to it. Most of the transformations described below define normal forms that are unique up to renaming of variables. If N is a set of (constrained) clauses, we denote by $R(N)$ the set of all R-normal forms of clauses in N.

3.1 Elimination of Monotonicity

A clause is said to be *flat* if variables are the only proper subterms of terms. Thus, $f(x)\not\approx y \vee h(x)\approx a$ is flat, but $f(f(x)) \approx x$ and $f(a) \approx x$ are not. A constrained clause $C \cdot \gamma$ is called flat if its clause part C is flat (but the constraint part γ may contain non-flat terms).

It is fairly straightforward to flatten clauses by abstracting subterms via introduction of new variables. This can be described by a set M of (schematic) transformation rules

$$C(s) \cdot \gamma \quad \Rightarrow \quad (s \not\approx x \vee C(x)) \cdot \gamma$$

where x is a variable not occurring in C and s is a non-variable term that occurs at least once as an argument of a function symbol in C. The rules in M are called *subterm abstraction rules*.

For example, the unit clause $i(x) * x \approx e$ contains one nested non-variable subterm, namely $i(x)$. Subterm abstraction yields a clause $i(x) \not\approx z \vee z * x \approx e$ that is unique up to renaming of the new variable z. The unit clause $i(x) * i(x) \approx i(x * x)$ contains two nested non-variable terms, $i(x)$ and

$x * x$, which are eliminated in two steps to yield a transformed clause

$$x * x \not\approx x_1 \vee i(x) \not\approx x_2 \vee x_2 * x_2 \approx i(x_1).$$

A (constrained) clause is flat if, and only if, it is in M-normal form. The M-normal forms of a clause are unique up to renaming of the newly introduced variables (and hence we will speak of *the* M-normal form). Our interest in flat clauses stems from the following result:

Proposition 3.1 (Brand 1975) Let N be a set of equational clauses and N' be obtained from N by replacing each clause by its M-normal form. Then N has an equality model if, and only if, N' has a model in which the predicate $\approx$ is interpreted as an equivalence (but not necessarily a congruence) relation.

In other words, the monotonicity axioms are not needed for testing satisfiability of flat equational clauses.

Note. We need not abstract *all* occurrences of a subterm at once. For instance, the multiple occurrences of the nested term $g(x)$ in

$$f(g(x)) \not\approx h(x) \vee h(g(x)) \approx x$$

are eliminated all at once to yield the M-normal form

$$g(x) \not\approx z \vee f(z) \not\approx h(x) \vee h(z) \approx x.$$

We may instead abstract the different occurrences separately to obtain a different flat clause,

$$g(x) \not\approx z_1 \vee g(x) \not\approx z_2 \vee f(z_1) \not\approx h(x) \vee h(z_2) \approx x.$$

3.2 Partial Elimination of Reflexivity

We may use equality constraints to get rid of certain undesirable negative equality literals:

$$(x \not\approx y \vee C) \cdot \gamma \quad \Rightarrow \quad C \cdot (\gamma \wedge x = y)$$

where x and y are variables. This transformation is called *reflexivity resolution* as it represents an instance of resolution with the reflexivity axiom. We denote the corresponding set of transformation rules by R.

3.3 Elimination of Symmetry

Next we replace the equality predicate $\approx$ by the predicate $\simeq$ and eliminate the need for the symmetry axioms. Positive equality literals are eliminated by *positive symmetry elimination* rules:

$$\begin{array}{lcl} (C \vee s \approx t) \cdot \gamma & \Rightarrow & (C \vee s \simeq t) \cdot \gamma \\ (C \vee s \approx t) \cdot \gamma & \Rightarrow & (C \vee t \simeq s) \cdot \gamma \end{array}$$

If a clause C contains n positive equality literals, then clearly n transformation steps will eliminate all positive occurrences of equality. There are 2^n different normal forms, *all* of which need to be retained to eliminate symmetry. For example, from the clause

$$g(x) \not\approx z \vee f(z) \not\approx h(x) \vee h(z) \approx x$$

we obtain both

$$g(x) \not\approx z \vee f(z) \not\approx h(x) \vee h(z) \simeq x$$

and

$$g(x) \not\approx z \vee f(z) \not\approx h(x) \vee x \simeq h(z).$$

Negative occurrences of $\approx$ can in principle be simply replaced by $\simeq$, but we prefer a slightly refined transformation that moves variables to the right-hand side.[5] The following *negative symmetry elimination* rules achieve this purpose:

$$\begin{array}{lcll} (s \not\approx t \vee C) \cdot \gamma & \Rightarrow & (s \not\simeq t \vee C) \cdot \gamma & \text{if } s \text{ is not a variable} \\ (s \not\approx t \vee C) \cdot \gamma & \Rightarrow & (t \not\simeq s \vee C) \cdot \gamma & \text{if } s \text{ is a variable, but } t \text{ is not} \end{array}$$

The normal forms produced by these additional transformation rules are unique, as at most one rule can be applied to any negative equality literal.[6]

We denote by S the set of all positive and negative symmetry elimination rules. If a clause contains n positive equality literals, then 2^n different S-normal forms can be derived from it. Two S-normal forms that can be derived from the same clause are said to be *symmetric variants* of each other.

3.4 Elimination of Transitivity

The transitivity axioms are eliminated by splitting positive and negative equality literals via introduction of so-called "link variables." The idea is the same as in Brand's method, but we also introduce constraints on variables, which necessitates slightly different transformations from Brand's, as will be explained below.

We have both *positive* and *negative splitting* rules of the form:

$$\begin{array}{lcl} (C \vee s \simeq t) \cdot \gamma & \Rightarrow & (C \vee t \not\simeq z \vee s \simeq z) \cdot (\gamma \wedge t \succeq z \wedge s \succ z) \\ (C \vee s \not\simeq t) \cdot \gamma & \Rightarrow & (C \vee t \not\simeq z \vee s \not\simeq z) \cdot (\gamma \wedge t \succeq z \wedge s \succeq z) \end{array}$$

where t is *not* a variable and z is a variable not occurring in C, s or t. The variable z is called a *link variable* (between s and t) and the corresponding constraints are called *link constraints*.

We emphasize that equality literals are *not* split if the right-hand side is already a variable. This is different from Brand's method, where literals are split regardless of whether the right-hand side is a variable or not.

We do not split equality literals with a variable on the right-hand side, but still may add corresponding ordering constraints, as expressed by the following *positive* and *negative link constraint* rules:

$$\begin{array}{lcl} (C \vee s \simeq x) \cdot \gamma & \Rightarrow & (C \vee s \simeq x) \cdot (\gamma \wedge s \succ x) \\ (C \vee s \not\simeq x) \cdot \gamma & \Rightarrow & (C \vee s \not\simeq x) \cdot (\gamma \wedge s \succeq x) \end{array}$$

where the constraints $s \succ x$ and $s \succeq x$, respectively, must not be contained in γ already.[7]

By T we denote the set of all splitting and link constraint rules. The T-normal form of a clause is unique up to renaming of link variables.

The flat clause (with empty constraint)

$$i(x) \not\simeq x_1 \vee x_1 * x \simeq e$$

[5]The advantage is that fewer splitting rules (described below) will be applicable.

[6]Negative literals $x \not\approx y$, with variables x and y, are not eliminated by symmetry elimination, but by reflexivity resolution.

[7]There is no point in introducing the same constraint repeatedly.

is transformed by T to the constrained clause

$$(i(x) \not\simeq x_1 \vee e \not\simeq y \vee x_1 * x \simeq y) \cdot (i(x) \succeq x_1 \wedge e \succeq y \wedge x_1 * x \succ y),$$

whereas its symmetric variant

$$i(x) \not\simeq x_1 \vee e \simeq x_1 * x$$

is transformed to

$$(i(x) \not\simeq x_1 \vee x_1 * x \not\simeq y \vee e \simeq y) \cdot (i(x) \succeq x_1 \wedge x_1 * x \succeq y \wedge e \succ y).$$

Observe that the constraint of the last clause is unsatisfiable if e is a minimal ground term with respect to the given ordering $\succ$. In other words, the clause is void in that case, and the constraint $e \succeq y$ in the other clause can be simplified to $e = y$.

Note. The example indicates that it is not necessary to apply subterm abstraction to a *minimal* constant c, as the corresponding constraint $c \succeq x$ associated with the abstraction of c can be simplified to $x = c$. Also, Skolem constants that occur only negatively need not be abstracted.

4 Preservation of Satisfiablity

The sets M, R, S, and T contain all the transformation rules we need. They eliminate all equality axioms, except reflexivity. Thus, for any set of clauses N, let $\mathsf{CEE}(N)$ be the clause set $\mathsf{T}(\mathsf{S}(\mathsf{R}(\mathsf{M}(N))) \cup \{x \simeq x\}$. Our main result can then be stated as follows:

Theorem 4.1 A set N of unconstrained equational clauses is *equationally* unsatisfiable if and only if the transformed set $\mathsf{CEE}(N)$ is unsatisfiable.

It is not difficult to prove that if N is equationally satisfiable, then the transformed set $\mathsf{CEE}(N)$ is satisfiable. (In other words, the transformations are all sound.) The difficult part is to show that $\mathsf{CEE}(N)$ is unsatisfiable, whenever N is equationally unsatisfiable.

It suffices to establish this property for $\mathsf{M}(N)$ or, generally, for sets of flat (unconstrained) clauses. For that purpose we introduce a refutationally complete calculus for flat equational clauses (the "flat basic superposition calculus") and show that all inferences in this calculus are reflected by logical consequences on the transformed clauses. This will imply, in particular, that a transformed set of clauses is unsatisfiable whenever a contradiction can be derived from the original clauses by flat basic superposition.

The inference rules of the *flat basic superposition* calculus are depicted in Figure 1. We should point out that in the presentation of superposition calculi, one usually identifies (as we have done here) a literal $s \approx t$ with $t \approx s$ (and similarly for negative literals $s \not\approx t$).

This calculus is a slimmed-down version of a strict basic superposition calculus restricted to flat clauses, and the following theorem is a direct consequence of the results in (Bachmair & Ganzinger 1997*a*); see the appendix.

Theorem 4.2 Let N be a set of flat unconstrained equational clauses. The following statements are equivalent:
1. N is equationally unsatisfiable;
2. $\mathsf{FBS}^*(N)$ contains a contradiction;
3. $(\mathsf{R} \circ \mathsf{FBS})^*(\mathsf{R}(N))$ contains a contradiction.

Moreover, if N is a set of flat clauses, then so are the sets $\mathsf{FBS}^*(N)$ and $(\mathsf{R} \circ \mathsf{FBS})^*(\mathsf{R}(N))$.

Positive flat basic superposition

$$\frac{(C \vee s \approx t) \cdot \gamma \quad (D \vee u \approx v) \cdot \delta}{(C \vee D \vee t \approx v) \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ v \succ t)} ,$$

where neither s nor u is a variable.

Negative flat basic superposition

$$\frac{(C \vee s \approx t) \cdot \gamma \quad (D \vee u \not\approx v) \cdot \delta}{(C \vee D \vee t \not\approx v) \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ t \wedge s \succ v)} ,$$

where u is not a variable.

Reflexivity resolution

$$\frac{(C \vee s \not\approx t) \cdot \gamma}{C \cdot (\gamma \wedge s = t)} .$$

Factoring

$$\frac{(C \vee s \approx t \vee u \approx v) \cdot \gamma}{(C \vee s \approx t) \cdot (\gamma \wedge s = u \wedge t = v)} ,$$

where $s \equiv u$ and $t \equiv v$.

Figure 1: Flat Basic Superposition FBS

The crucial advantage over previous formulations of basic superposition is that FBS has no equality factoring inferences and no positive (top-level) superposition inferences *from* variables. Positive top-level superposition inferences from variables can be avoided if the reduction ordering on terms is lifted to literals in a way such that unshielded, maximal variables make a literal larger than literals in which the same maximal term appears, but does not occur entirely within the substitution part of the clause.[8] Also note that factoring is applied only when the terms which are unified by the inference have the same skeleton. In particular, if s is a variable, so is u, and vice versa, and the same is true for the pair t and v. Since transitivity elimination is sensitive to variables this uniformity will be helpful.

In the Lemma below we state a direct connection between flat basic superposition and the transformation system CEE, forming the core of our completeness proof.

Lemma 4.3 Let N be a set of flat constrained equational clauses simplified with respect to reflexivity resolution (so that $\mathsf{R}(N) = N$). If $\mathcal{D}$ is a clause in $\mathsf{R} \circ \mathsf{FBS}(N)$, then any $\mathsf{T} \circ \mathsf{S}$-normal form of $\mathcal{D}$ is a logical consequence of $\mathsf{T} \circ \mathsf{S}(N) \cup \{x \simeq x\}$ (see the diagram).[9]

[8] With this specific class of literal orderings, it should be possible to also close the main gap in the proof in (Moser & Steinbach 1997). There are some more details that need to be checked as the latter paper assumes a slightly different calculus and analyses superposition inferences on clauses as they are obtained after transitivity elimination. This causes additional problems which we avoid.

[9] We use the symbol $\circ$ to denote composition of operators. Thus, $\mathsf{T} \circ \mathsf{S}(N) = \mathsf{T}(\mathsf{S}(N))$.

$$
\begin{tikzcd}
\cdot \arrow[r, "\mathsf{T}\circ\mathsf{S}"] \arrow[d, "\mathsf{R}\circ\mathsf{FBS}"'] & \cdot \arrow[d, "\models"] \\
\cdot \arrow[r, "\mathsf{T}\circ\mathsf{S}"] & \cdot
\end{tikzcd}
$$

Proof. Let $\mathcal{D}$ be the simplified (by R) conclusion of an inference in FBS from premises in N and let $\mathcal{C}$ be in $\mathsf{T} \circ \mathsf{S}(\mathcal{D})$. For demonstrating that $\mathcal{C}$ is logically implied by $\mathsf{T} \circ \mathsf{S}(N) \cup \{x \simeq x\}$ we will usually apply resolution-based reasoning, followed by some strengthening of the constraint.

We prove the assertion by a case analysis over the inferences in FBS. Let

$$\frac{(C \vee s \approx t) \cdot \gamma \quad (D \vee u \approx v) \cdot \delta}{(C \vee D \vee t \approx v) \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ t \wedge u \succ v \succ t)}$$

be an inference by *positive flat basic superposition* from premises in N. Then neither s nor u is a variable. Also, the conclusion $\mathcal{D}$ is already simplified by R as any clause in N has this property by assumption. Any $\mathsf{T} \circ \mathsf{S}$-normalform of $\mathcal{D}$ has the form

$$\mathcal{C} = (C' \vee D' \vee E) \cdot (\gamma \wedge \delta \wedge \lambda_{C'} \wedge \lambda_{D'} \wedge s = u \wedge s \succ t \wedge u \succ v \succ t \wedge \varepsilon).$$

where (i) $C' \cdot (\gamma \wedge \lambda_{C'})$ and $D' \cdot (\delta \wedge \lambda_{D'})$ are $\mathsf{T} \circ \mathsf{S}$-normalforms of $C \cdot \gamma$ and $D \cdot \delta$, respectively; (ii) the subclause E and the link constraints ε for the literals in E depend on (a) whether the the new equation $t \approx v$ has been oriented into $t \simeq v$ or $v \simeq t$ during S normalization; and (b) on the result of T normalization, depending on whether or not t or v are variables. In this structuring of the constraint, we assume matching modulo associativity, commutativity, *and* idempotence of conjunction. That is, we allow for an implicit duplication of atomic constraints to bring the constraint into the form $(\gamma \wedge \delta \wedge \lambda_{C'} \wedge \lambda_{D'} \wedge s = u \wedge s \succ t \wedge u \succ v \succ t \wedge \varepsilon)$ with conjuncts $\lambda_{C'}$ and $\lambda_{D'}$ as they would be introduced during a separate $\mathsf{T} \circ \mathsf{S}$ normalization of the subclauses $C \cdot \gamma$ and $D \cdot \delta$. We proceed with an analysis of the variants of E.

(i) *Variant $t \simeq v$, and v is a variable.* Then $\mathcal{C}$ has the form

$$(C' \vee D' \vee t \simeq v) \cdot (\gamma \wedge \delta \wedge \lambda_{C'} \wedge \lambda_{D'} \wedge s = u \wedge s \succ t \wedge u \succ v \succ t \wedge t \succ v)$$

Evidently, the constraint part of $\mathcal{C}$ is unsatisfiable, that is, $\mathcal{C}$ is void, hence trivially follows from $\mathsf{T} \circ \mathsf{S}(N)$.

From now on, to simplify notation, we shall omit the "side-literals" C' and D' as well as the respective "standard constraints" $\gamma \wedge \delta \wedge \lambda_{C'} \wedge \lambda_{D'}$ which are inherited from the C and D subclauses of the respective premises and their $\mathsf{T} \circ \mathsf{S}$ normalforms.

(ii) *Variant $t \simeq v$, and v is not a variable.* Here, $\mathcal{C}$ has the form

$$(v \not\simeq x \vee t \simeq x) \cdot (s = u \wedge s \succ t \wedge u \succ v \succ t \wedge v \succeq x \wedge t \succ x),$$

or, equivalently,

$$(v \not\simeq x \vee t \simeq x) \cdot (s = u \wedge u \succ v \succ t \succ x) \tag{1}$$

with x a fresh link variable. As neither s nor u is a variable, $\mathsf{T} \circ \mathsf{S}(N)$ contains the clauses $(u \not\simeq x \vee v \simeq x) \cdot (u \succeq x \wedge v \succ x)$ and $(s \not\simeq y \vee t \simeq y) \cdot (s \succeq y \wedge t \succ y)$, with link variables x and y. Consider the resolution inference

$$\frac{(u \not\simeq x \vee v \simeq x) \cdot (u \succeq x \wedge v \succ x) \quad (s \not\simeq y \vee t \simeq y) \cdot (s \succeq y \wedge t \succ y)}{(v \not\simeq y \vee t \simeq y) \cdot (s \succeq y \wedge t \succ y \wedge u \succeq x \wedge v \succ x \wedge s = u \wedge y = x)}\ .$$

Since x and y are variables not occuring in s, t, u, v, the conclusion of this inference is equivalent to

$$(v \not\simeq x \vee t \simeq x) \cdot (t, v \succ x \wedge s, u \succeq x \wedge s = u) \tag{2}$$

The clause (2) is more general than (1) since the constraint $s = u \wedge u \succ v \succ t \succ x$ implies the constraint $t, v \succ x \wedge s, u \succeq x$. We have shown, as was required, that (1) is a logical consequence of $\mathsf{T} \circ \mathsf{S}(N)$.

(iii) *Variant $v \simeq t$, t is a variable.* After simplifying the constraint, $\mathcal{C}$ has the form

$$(v \simeq t) \cdot (s = u \wedge s \succ t \wedge u \succ v \succ t). \tag{3}$$

In this case, consider the resolution inference

$$\frac{(s \simeq t) \cdot (s \succ t) \quad (u \not\simeq x \vee v \simeq x) \cdot (u \succeq x \wedge v \succ x)}{(v \simeq x) \cdot (s \succ t \wedge u \succeq x \wedge v \succ x \wedge s = u \wedge t = x)}$$

from premises in $\mathsf{T} \circ \mathsf{S}(N)$. Since x does not occur in s, t, u, v, the conclusion of this inference is equivalent to

$$(v \simeq t) \cdot (u \succeq t \wedge s, v \succ t \wedge s = u) \tag{4}$$

which is more general than (3).

(iv) *Variant $v \simeq t$, t is not a variable.* In this case, $\mathcal{C}$ is equivalent to

$$(t \not\simeq x \vee v \simeq x) \cdot (s = u \wedge s \succ t \wedge u \succ v \succ t \succeq x), \tag{5}$$

with a fresh variable x. $\mathcal{C}$ can be derived from $\mathsf{T} \circ \mathsf{S}(N)$ via the inference

$$\frac{(t \not\simeq y \vee s \simeq y) \cdot (t \succeq y \wedge s \succ y) \quad (u \not\simeq x \vee v \simeq x) \cdot (u \succeq x \wedge v \succ x)}{(t \not\simeq y \vee v \simeq x) \cdot (t \succeq y \wedge s \succ y \wedge u \succeq x \wedge v \succ x \wedge s = u \wedge y = x)}\ .$$

Since x and y are variables not occuring in s, t, u, v, the conclusion of this inference is equivalent to

$$(t \not\simeq x \vee v \simeq x) \cdot (t, u \succeq x \wedge s, v \succ x \wedge s = u) \tag{6}$$

which is more general than (5).

Next we prove the assertion for inferences by *negative flat basic superposition*

$$\frac{(C \vee s \approx t) \cdot \gamma \quad (D \vee u \not\approx v) \cdot \delta}{(C \vee D \vee t \not\approx v) \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ t \wedge u \succ v)}$$

from premises in N. Let $\mathcal{D}$ denote the conclusion of the inference. Then u is not a variable. Again we need to conduct a case analysis according to whether or not one of the v and t are variables. Here the various cases also determine as to whether the conclusion of the inference can be further simplified by reflexivity resolution. Without loss of generality we assume that $t \geq v$ such that symmetry elimination transforms $t \not\approx v$ into $t \not\simeq v$. Under this assumption, if t is a variable, so must be v. We will again simplify the technical presentation by disregarding the side literals in C and D and their associated constraints.

(i) *Suppose v is a variable, but t is not.* Then $\mathcal{D}$ is in normalform with respect to R and the (unique) $\mathsf{T} \circ \mathsf{S}$-normalform of $\mathcal{D}$ has the form

$$(t \not\succeq v) \cdot (s = u \wedge s \succ t \wedge u \succ v \wedge t \succeq v). \tag{7}$$

Consider the resolution inference

$$\frac{(t \not\succeq x \vee s \simeq x) \cdot (t \succeq x \wedge s \succ x) \quad (u \not\succeq v) \cdot (u \succeq v)}{(t \not\succeq x) \cdot (t \succeq x \wedge s \succ x \wedge u \succeq v \wedge s = u \wedge x = v).}$$

Both premises are in $\mathsf{T} \circ \mathsf{S}(N)$. Since x is a variable not occuring in s, t, u, v, the conclusion of this inference is equivalent to

$$(t \not\succeq v) \cdot (t \succeq v \wedge s \succ v \wedge u \succeq v \wedge s = u)$$

which is more general than (7).

(ii) *If both v and t are variables* then $\mathcal{D}$ is normalized by the R-step in $\mathsf{R} \circ \mathsf{FBS}$ to

$$\square \cdot (s = u \wedge t = v \wedge s \succ t \wedge u \succ v) \tag{8}$$

which is in $\mathsf{T} \circ \mathsf{S}$-normalforms. A more general clause can also be derived from $\mathsf{T} \circ \mathsf{S}(N)$ by

$$\frac{(s \simeq t) \cdot (s \succ t) \quad (u \not\succeq v) \cdot (u \succeq v)}{\square \cdot (s \succ t \wedge u \succeq v \wedge s = u \wedge t = v).}$$

(iii) *Suppose that neither v nor t is a variable.* Then $\mathcal{D}$ is in normalform with respect to R and the (unique) $\mathsf{T} \circ \mathsf{S}$-normalform of $\mathcal{D}$ has the form

$$(t \not\succeq x \vee v \not\succeq x) \cdot (s = u \wedge s \succ t \wedge u \succ v \wedge t, v \succeq x), \tag{9}$$

where x is a fresh variable. Consider the resolution inference

$$\frac{(t \not\succeq x \vee s \simeq x) \cdot (t \succeq x \wedge s \succ x) \quad (u \not\succeq y \vee v \not\succeq y) \cdot (u, v \succeq y)}{(t \not\succeq x \vee v \not\succeq y) \cdot (t \succeq x \wedge s \succ x \wedge u, v \succeq y \wedge s = u \wedge x = y)}$$

from premises in $\mathsf{T} \circ \mathsf{S}(N)$. Since x is a variable not occuring in s, t, u, v, the conclusion of this inference is equivalent to

$$(t \not\succeq x \vee v \not\succeq x) \cdot (t, u, v \succeq x \wedge s \succ x \wedge s = u)$$

which is more general than (9).

Next we treat *reflexivity resolution*

$$\frac{(C \vee s \not\simeq t) \cdot \gamma}{C \cdot (\gamma \wedge s = t)}$$

from a clause in N. Its conclusion $\mathcal{C}$ is in normalform with respect to R. As N is simplified with respect to R not both s and t can be variables. Let us assume, without loss of generality, that s is not a variable. Any $\mathsf{T} \circ \mathsf{S}$-normalform $\mathcal{D}$ of $\mathcal{C}$ has the form

$$C' \cdot (\gamma \wedge \lambda_{C'} \wedge s = t), \tag{10}$$

where $C' \cdot (\gamma \wedge \lambda_{C'})$ is a $\mathsf{T} \circ \mathsf{S}$-normalform of $C \cdot \gamma$. We need to distinguish two cases:

(i) *t is not a variable*: Consider the following derivation in $\mathsf{T} \circ \mathsf{S}(N)$

$$\dfrac{z \simeq z \quad \dfrac{y \simeq y \quad (C' \vee t \not\simeq x \vee s \not\simeq x) \cdot (\gamma \wedge \lambda_{C'} \wedge s, t \succeq x)}{(C' \vee s \not\simeq x) \cdot (\gamma \wedge \lambda_{C'} \wedge s, t \succeq x \wedge y = t \wedge y = x)}}{C' \cdot (\gamma \wedge \lambda_{C'} \wedge s, t \succeq x \wedge y = t \wedge y = x \wedge z = s \wedge z = x)}$$

by two resolution steps with (variants of) $x \simeq x$. Since x, y and z are variables not appearing elsewhere, the conclusion of this inference is equivalent to

$$C' \cdot (\gamma \wedge \lambda_{C'} \wedge s \not\simeq t)$$

which is precisely (10).

(ii) *t is a variable*: This case is similar, considering the following single-step derivation from $\mathsf{T} \circ \mathsf{S}(N)$ by resolution with a variant of $x \simeq x$

$$\dfrac{y \simeq y \quad (C' \vee s \not\simeq t) \cdot (\gamma \wedge \lambda_{C'} \wedge s \succeq t)}{C' \cdot (\gamma \wedge \lambda_{C'} \wedge s \succeq t \wedge y = s \wedge y = t)}$$

yielding a clause which is equivalent to (10).

Factoring inferences take the form

$$\dfrac{(C \vee s \approx t \vee u \approx v) \cdot \gamma}{(C \vee s \approx t) \cdot (\gamma \wedge s = u \wedge t = v)}$$

where $s \equiv u$ and $t \equiv v$. Let again $\mathcal{D}$ denote the conclusion of the inference. We have to consider four possible cases, depending on whether or not s and t (and, hence, u and v) are variables. In the analysis we again omit side-literals and their constraints.

(i) *Neither s nor t is a variable.* Then none of the four terms s, t, u, v is a variable. Therefore, any $\mathsf{T} \circ \mathsf{S}$ normalform of $\mathcal{D}$ has the form

$$(v \not\simeq x \vee u \simeq x) \cdot (s = u \wedge t = v \wedge v \succeq x \wedge u \succ x), \tag{11}$$

where x is a fresh variable. By applying positive and negative factoring to the $\mathsf{T} \circ \mathsf{S}$ normalform

$$(v \not\simeq x \vee t \not\simeq y \vee u \simeq x \vee s \simeq y) \cdot (t \succeq y \wedge s \succ y \wedge v \succeq x \wedge u \succ x)$$

of the premise of the factoring inference, we obtain the clause

$$(v \not\simeq x \vee u \simeq x) \cdot (v = t \wedge x = y \wedge u = s \wedge t \succeq y \wedge s \succ y \wedge v \succeq x \wedge u \succ x)$$

which (y does not occur elsewhere) is equivalent to

$$(v \not\simeq x \vee u \simeq x) \cdot (v = t \wedge u = s \wedge v \succeq x \wedge u \succ x)$$

which is precisely (11). For the variant $u \not\simeq x \vee v \simeq x$ we may construct a similar derivation.

(ii) *s is a variable, and t is not a variable.* Then also v is not a variable. We need to derive clauses of the form $t \not\simeq x \vee s \simeq x$ and $t \simeq s$ with the appropriate constraints. This can be achieved by applying appropriate factoring inferences to $t \not\simeq x \vee v \not\simeq y \vee s \simeq x \vee u \simeq y$ and $t \simeq s \vee v \simeq u$, respectively, where the latter, since u is not a variable, can be obtained by a resolution step with reflexivity from $t \simeq s \vee u \not\simeq x \vee v \simeq x$. The case in which s is not a variable, but t is, is similar.

(iii) *Both s and t are variables.* Then we may derive clauses of the form $s \simeq t$ and $t \simeq s$ with the required constraints by factoring from $s \simeq t \vee u \simeq v$ and $t \simeq s \vee v \simeq u$, respectively. These clauses, in turn, are either T-modifications of symmetric variants of the premise of the factoring inference, or else may be obtained from these by reflexivity resolution.

□

By inductive application of this lemma we obtain the desired property for flat clauses:

Theorem 4.4 Let N be a set of flat equational clauses without constraints. Then N is equationally satisfiable if and only if $\mathsf{T} \circ \mathsf{S} \circ \mathsf{R}(N) \cup \{x \simeq x\}$ is satisfiable.

Proof. It can easily be shown that $\mathsf{T} \circ \mathsf{S} \circ \mathsf{R}(N) \cup \{x \simeq x\}$ is satisfiable whenever N is equationally satisfiable. Suppose that N is equationally unsatisfiable, and let N' denote $\mathsf{R}(N)$. By the completeness of flat basic superposition, we may infer that $(\mathsf{R} \circ \mathsf{FBS})^*(N')$ contains a contradiction. The set N', and all sets $(\mathsf{R} \circ \mathsf{FBS})^k(N')$ are simplified (with respect to R) flat equational clauses to which we may (inductively) apply the above lemma. Therefore, all clauses in $\mathsf{T} \circ \mathsf{S}((\mathsf{R} \circ \mathsf{FBS})^*(N'))$ are logical consequences of $\mathsf{T} \circ \mathsf{S} \circ \mathsf{R}(N) \cup \{x \simeq x\}$. As the $\mathsf{T} \circ \mathsf{S}$ normal form of a contradiction is also a contradiction, $\mathsf{T} \circ \mathsf{S} \circ \mathsf{R}(N) \cup \{x \simeq x\}$ must be unsatisfiable. □

5 Variations

Let us now briefly discuss the connection of our method to other transformation methods. Brand's original method is not directly comparable to our method. The main difference (aside from the fact that we use constraints) is that Brand uses only a positive splitting rule,

$$(C \vee u \simeq v) \quad \Rightarrow \quad (C \vee v \not\simeq z \vee u \simeq z),$$

but no negative splitting rule. However, the positive splitting rule is applied even if the right-hand side v of an equality literal is a variable. With Brand's method the clause

$$f(g(x)) \not\approx h(x) \vee h(g(x)) \approx x$$

is transformed into two clauses

$$g(x) \not\simeq z \vee f(z) \not\simeq h(x) \vee x \not\simeq y \vee h(z) \simeq y$$

and

$$g(x) \not\simeq z \vee f(z) \not\simeq h(x) \vee h(z) \not\simeq y \vee x \simeq y,$$

whereas our transformation would result in different (constrained) clauses

$$(g(x) \not\simeq z \vee f(z) \not\simeq z_1 \vee h(x) \not\simeq z_1 \vee h(z) \simeq x) \cdot (g(x) \succeq z \wedge f(z), h(x) \succeq z_1 \wedge h(z) \succ x)$$

and

$$(g(x) \not\simeq z \vee f(z) \not\simeq z_1 \vee h(x) \not\simeq z_1 \vee h(z) \not\simeq y \vee x \simeq y) \cdot \ (g(x) \succeq z \wedge f(z), h(x) \succeq z_1 \wedge h(z) \succeq y \wedge x \succ y).$$

It is not possible, though, to simply add link constraints to Brand's original transitivity elimination rule.

For example, Brand's transitivity elimination with link constraints, when applied to the *unsatisfiable* set of unit clauses $a \approx b$, $a \approx c$ and $b \not\approx c$ (with an ordering in which $c \succ b \succ a$) yields a set of constrained clauses

$$\begin{array}{l} (b \not\simeq x \vee a \simeq x) \cdot (b \succeq x \wedge a \succ x) \\ (a \not\simeq x \vee b \simeq x) \cdot (a \succeq x \wedge b \succ x) \\ (c \not\simeq x \vee a \simeq x) \cdot (c \succeq x \wedge a \succ x) \\ (a \not\simeq x \vee c \simeq x) \cdot (a \succeq x \wedge c \succ x) \\ b \not\simeq c \end{array}$$

that is *satisfiable* (in combination with the reflexivity axiom $x \simeq x$)! (The first and third clause contain the unsatisfiable constraint $a \succ x$ and hence are void. The remaining clauses, along with $x \simeq x$, are satisfiable even without the constraints.) In short, ordering constraints are not compatible with Brand's original transformations.

The method implemented in the Setheo prover (Moser & Steinbach 1997) can be described with our transformation rules, except that no link constraints are introduced. Positive equations with a variable on the right-hand side are not split, and hence negative equations with a non-variable right-hand side must be split also.

For example, the three unit clauses $f(x) \approx x$, $g(x) \approx x$ and $f(x) \not\approx g(x)$ are unsatisfiable. However, if negative equality literals are not split, we obtain a satisfiable set of clauses $f(x) \simeq x$, $f(x) \not\simeq y \vee x \simeq y$, $g(x) \simeq x$, $g(x) \not\simeq y \vee x \simeq y$, $f(x) \not\simeq g(x)$, and $x \simeq x$.

Let us conclude this section with an example. The presentation of group theory by three equations, $x * e \approx x$, $x * i(x) \approx e$, and $(x * y) * z \approx x * (y * z)$, is transformed with our method into the following set of constrained clauses:

$$\begin{array}{rl} x * e \simeq x & \\ x * e \not\simeq u \vee x \simeq u & \cdot\ x \succ u \\ i(x) \not\simeq u \vee x * u \simeq e & \cdot\ i(x) \succeq u \\ x * y \not\simeq u \vee y * z \not\simeq v \vee u * z \not\simeq w \vee x * v \simeq w & \cdot\ u * z \succeq w \wedge x * v \succ w \wedge x * y \succeq u \wedge y * z \succeq v \\ x * y \not\simeq u \vee y * z \not\simeq v \vee x * v \not\simeq w \vee u * z \simeq w & \cdot\ x * v \succeq w \wedge u * z \succ w \wedge x * y \succeq u \wedge y * z \succeq v \end{array}$$

where $\succ$ refers to a lexicographic path ordering induced by the precedence relation $i > * > e$ and constraints have been simplified accordingly. Note that with Brand's modification or with equality elimination as used in Setheo one gets an additional clause,

$$i(x) \not\simeq z \vee x * z \not\simeq w \vee e \simeq w.$$

This clause can be omitted, as its associated constraint $e \succ w$ is unsatisfiable in the given ordering.

6 Experiments

In Figure 6 we show some experimental results with the Protein prover (Baumgartner & Furbach 1994) on certain simple problems in group theory. "L" means that the goal was attempted in the presence a previously proved lemma. In the table we list runtimes and number of computed inferences ("K" denotes kilo, and "M" denotes mega inferences) for four kinds of transformation. The "B" column depicts the results for Brand's original modification. "S" refers to the method that is implemented in Setheo with splitting of both positive and negative equations that have no variable right-hand side, without attaching ordering constraints. "Ss" is like "S" except that Skolem constants in the goals have not been abstracted. Compared with Brand's method, the Setheo

Problem	ord	number of inferences B	S	Ss	C	time [sec] B	S	Ss	C
$x * e \approx x$	–	1.3M	21.4K	1.5K	**578**	123	2.1	0.16	**0.1**
$i(i(x)) \approx x$	1	4.7M	∞	83.6M	**15.5K**	508	∞	10191	**4.8**
$i(i(x)) \approx x$	2	∞	∞	∞	**178K**	∞	∞	∞	**84**
$i(i(x)) \approx x$	3	4.7M	∞	83.6M	**15.5K**	502	∞	10191	**4.8**
$x * i(x) \approx e$	1	4.1K	288K	288K	**461**	0.4	30	30.4	**0.1**
$x * i(x) \approx e$	2	2.4M	17.5M	17.6M	**671**	272	2204	2204	**0.2**
$i(x) * (x * y) \approx y$ L	1	19.5K	267K	267M	**3.8K**	1.9	28	28.3	**0.9**
$i(x) * (x * y) \approx y$ L	2	∞	∞	∞	**12.2M**	∞	∞	∞	**3235**
$i(x) * (x * y) \approx y$	1	**9.1M**	∞	∞	24M	**950**	∞	∞	10574
$i(x) * (x * y) \approx y$	2	∞	∞	∞	∞	∞	∞	∞	∞

Figure 2: Experimental results on a 167MHz UltraSparc for simple problems in group theory

method avoids more of those inferences which correspond to superposition into or from variables. However, it comes at the expense of also splitting negative equations. The experiments show that the price to pay is indeed very high so that in some of our experiments, "S" performs much worse than Brand's original method. However, if disequations $s \simeq t$ in which t is a Skolem constant of the goal are not split we obtain a uniform and more significant improvement. This variant is denoted by "Ss" below. Finally, "C" is CEE transformation, using the presentation of group theory as presented in Section 5. We implemented constraint inheritance and checking in a straightforward manner. Ordering constraints are collected through additional arguments of predicate symbols. (As to what extent the additional predicates affect Protein's proof strategy, we do not know.) The first subgoal of any (non-unit) clause first calls upon a satisfiability check (implemented in Prolog) for the accumulated constraint at this point. Constraint solving was implemented incompletely by simply checking independent satisfiability of each inequality in any conjunction of inequalities. A complete constraint solving which is available for large classes of lexicographic path orderings, is very expensive and does not seem to reduce the number of inferences by another order of magnitude.

Protein is extremely sensitive to how the clauses and the literals in a clause, respectively, are ordered. In the examples we have experimented with three different orderings of the subgoals in the goal clause. In ordering 1 the variable definitions for inner subterm positions precede those of the outer positions. This ordering seems to work better with Protein most of the time. Ordering 2 is the inverse of ordering 1. Ordering 3 is some mixture of orderings 1 and 2. Orderings 2 and 3 coincide for the CEE transformation. For ordering 2, the speedups obtained from the optimization are much more dramatic. This seems to indicate that with the constraints the performance of model elimination is somewhat less dependent on subgoal selection strategies. In particular upon backtracking, ordering constraints prevent one from searching redundant alternative proofs of subgoals.

Although these experiments are far from being conclusive, it appears as if the CEE transformation can have a dramatic effect on proof search. Except for one case, proofs using CEE transformation were found much faster, usually by several orders of magnitude. With the rather incomplete method of constraint satisfiability checking, the price paid on each single inference seems affordable.

As said before, Protein proof search is too much dependent on the ordering of clauses and of subgoals within clauses. It would be interesting to see the effect of our improvements on Setheo where dynamic goal selection strategies result in a more predictable behavior and find proofs more often, also for less trivial problems than the ones studied in our experiments (Ibens & Letz 1997).

7 Conclusions

We have described a refined variant of Brand's modification method via ordering constraints that also improves equality elimination as implemented in the prover Setheo. Our theoretical results imply that equality handling in Setheo is indeed refutationally complete (which was an open problem). The completeness proof draws on recent results about basic superposition and thus establishes a connection between the theory underlying local saturation-based methods, such as paramodulation and superposition, and optimizations of equality handling in global theorem proving methods, such as model elimination and semantic tableau-methods.

Our experiments seem to indicate that with the ordering constraints the search space is indeed drastically reduced. However, our results in this regards are not conclusive yet. An implementation of the method in a more advanced model elimination prover, like Setheo, should allow one to draw more definite conclusions in this regard.

References

Bachmair, L. & Ganzinger, H. (1997*a*), Strict basic superposition and chaining, Research report MPI-I-97-2-011, Max-Planck Institut für Informatik, Saarbrücken, Germany. *www.mpi-sb.mpg.de/~hg/papers/reports/MPI-I-97-2-011.ps.gz

Bachmair, L. & Ganzinger, H. (1997*b*), A theory of resolution, Research report MPI-I-97-2-005, Max-Planck Institut für Informatik, Saarbrücken, Germany. To appear in Handbook of Automated Reasoning, edited by A. Robinson and A. Voronkov.

Baumgartner, P. & Furbach, U. (1994), PROTEIN: A *PRO*ver with a *T*heory *E*xtension *IN*terface, *in* A. Bundy, ed., 'Automated Deduction — CADE-12. 12th International Conference on Automated Deduction.', Vol. 814 of *Lecture Notes in Artificial Intelligence*, Nancy, France, pp. 769–773.

Brand, D. (1975), 'Proving theorems with the modification method', *SIAM Journal of Computing* **4**, 412–430.

Degtyarev, A. & Voronkov, A. (1996*a*), Equality elimination for the tableau method, *in* J. Calmet & C. Limongelli, eds, 'Design and Implementation of Symbolic Computation Systems. International Symposium, DISCO'96', Vol. 1128 of *Lecture Notes in Computer Science*, Karlsruhe, Germany, pp. 46–60.

Degtyarev, A. & Voronkov, A. (1996*b*), What you always wanted to know about rigid E-unification, *in* J. Alferes, L. Pereira & E. Orlowska, eds, 'Logics in Artificial Intelligence. European Workshop, JELIA'96', Vol. 1126 of *Lecture Notes in Artificial Intelligence*, Évora, Portugal, pp. 50–69.

Ibens, O. & Letz, R. (1997), Subgoal alternation in model elimination, *in* D. Galmiche, ed., 'Automated Reasoning with Analytic Tableaux and Related Methods', Vol. 1227 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 201–215.

Moser, M., Lynch, C. & Steinbach, J. (1995), Model elimination with basic ordered paramodulation, Technical Report AR-95-11, Fakultät für Informatik, Technische Universität München, München.

Moser, M. & Steinbach, J. (1997), STE-modification revisited, Technical Report AR-97-03, Fakultät für Informatik, Technische Universität München, München.

Schumann, J. (1994), 'Tableau-based theorem provers: Systems and implementations', *Journal of Automated Reasoning* **13**(3), 409–421.

Positive basic superposition

$$\frac{(C \vee s \approx t) \cdot \gamma \qquad (D \vee w[u] \approx v) \cdot \delta}{(C \vee D \vee w[t] \approx v) \ \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ t \wedge w \succ v \wedge (w,v) \succeq_{\mathsf{lex}} (s,t) \wedge \zeta)}$$

where (i) u is not a variable and (ii) either s is not a variable or else u is a proper subterm of w.

Negative basic superposition

$$\frac{(C \vee s \approx t) \cdot \gamma \qquad (D \vee w[u] \not\approx v) \cdot \delta}{(C \vee D \vee w[t] \not\approx v) \cdot (\gamma \wedge \delta \wedge s = u \wedge s \succ t \wedge w \succ v \wedge \zeta)}$$

where u is not a variable.

Reflexivity resolution

$$\frac{(C \vee s \not\approx t) \cdot \gamma}{C \cdot (\gamma \wedge s = t \wedge \zeta)}$$

Factoring

$$\frac{(C \vee s \approx t \vee u \approx v) \cdot \gamma}{(C \vee s' \approx t') \cdot (\gamma \wedge s = u \wedge t = v \wedge \zeta)}$$

where $s \equiv u$ and $t \equiv v$.

Figure 3: Strict Basic Superposition BS

A Basic Superposition

The flat basic superposition calculus is derived from one of the strict basic superposition calculi described in (Bachmair & Ganzinger 1997*a*), see Figure 3.

In all cases, ζ refers to inference-specific ordering constraints that are irrelevant for our purposes and hence have been deleted in the flat versions of the inference rules. (Such constraints may always be deleted without impairing the soundness or refutational completeness of the calculus.)

Moreover, the remaining constraints in the flat inference rules have been simplified. For instance, if the second premise of a (positive or negative) basic superposition rule is a flat clause, then the two terms w and u must be identical, as any proper subterm in a flat clause is a variable, yet u is required to be a non-variable subterm of w. Using the facts that $w = u$ and $s = u$, one may then simplify the ordering constraints as indicated. In particular, the constraint $(w, v) \succeq_{\mathsf{lex}} (s, t)$ in positive superpositions reduces to $s \succ v \succeq t$ in the flat case. The instance $v = t$ can also be ignored. In that case the conclusion contains $t \approx t$ positively, hence, is a direct tautology which is redundant. Direct tautologies are a strict subclass of all tautologies, cf. (Bachmair & Ganzinger 1997*a*) for the exact definition.

Basic superposition calculi are also compatible with other various mechanisms for eliminating redundancies and simplifying clauses. For the present paper it suffices to point out that, in addition to direct tautologies, void clauses are redundant, as are inferences with a void conclusion. The former can be deleted, while the latter may be ignored. Furthermore, the special case of reflexivity resolution as represented by the transformation R is a simplification rule. That is, a clause $(x \not\approx y \vee C) \cdot \gamma$ with variables x and y, is redundant in the presence of a clause $C \cdot (\gamma \wedge x = y)$ and, hence, can be deleted. Summarizing these observations, we obtain the Theorem 4.2 stated above.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-001	S. Vorobyov	Context Unification = Word Equations = Linear Integer Constraints
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda\rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics
MPI-I-97-2-001	D.A. Basin, S. Matthews, L. Viganò	Labelled modal logics: quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D.A. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification
MPI-I-96-2-008	S. Vorobyov	On the decision complexity of the bounded theories of trees
MPI-I-96-2-007	A. Herzig	SCAN and Systems of Conditional Logic
MPI-I-96-2-006	D.A. Basin, S. Matthews, L. Viganò	Natural Deduction for Non-Classical Logics
MPI-I-96-2-005	A. Nonnengart	Auxiliary Modal Operators and the Characterization of Modal Frames
MPI-I-96-2-004	G. Struth	Non-Symmetric Rewriting
MPI-I-96-2-003	H. Baumeister	Using Algebraic Specification Languages for Model-Oriented Specifications