MAX-PLANCK-INSTITUT FÜR INFORMATIK

Special Cases and Substitutes for Rigid E-Unification

David A. Plaisted

MPI–I–95–2–010 November 1995

INFORMATIK

Im Stadtwald
D 66123 Saarbrücken
Germany

Authors' Addresses

Department of Computer Science
CB# 3175, 352 Sitterson Hall
University of North Carolina at Chapel Hill
Chapel Hill, North Carolina 27599-3175
USA
plaisted@cs.unc.edu

Acknowledgements

I would like to thank A. Degtyarev and A. Voronkov for their comments on preliminary versions of this paper, especially the undecidability proof. I would also like to thank Sergei Vorobyov for his detailed comments on some of the early sections of the paper, and Uwe Waldmann for his detailed reading as well. Of course, remaining errors are solely my responsibility. This research was partially supported by the National Science Foundation under grant CCR-9108904. The author also gratefully acknowledges the support of Max-Planck Institute, Saarbrücken during the summer of 1995.

Abstract

The simultaneous rigid E-unification problem arises naturally in theorem proving with equality. This problem has recently been shown to be undecidable. This raises the question whether simultaneous rigid E-unification can usefully be applied to equality theorem proving. We give some evidence in the affirmative, by presenting a number of common special cases in which a decidable version of this problem suffices for theorem proving with equality. We also present some general decidable methods of a rigid nature that can be used for equality theorem proving and discuss their complexity. Finally, we give a new proof of undecidability of simultaneous rigid E-unification which is based on Post's Correspondence Problem, and has the interesting feature that all the positive equations used are ground equations (that is, contain no variables).

Contents

1 Introduction

Simultaneous rigid E-unification, which is related to theorem proving with equality, has recently been shown to be undecidable. However, we show that in many common cases of interest, this problem is decidable and of reasonable complexity. We also present some decidable rigid inference systems that can substitute for simultaneous rigid E-unification, in a certain sense. Although first-order logic is not decidable, it is possible to use decidable techniques as part of a method for theorem proving in first-order logic with equality. Finally, we give another undecidability proof for simultaneous rigid E-unification based on Post's Correspondence Problem.

Simultaneous rigid E-unification is related to the matings method of Andrews [1] as well as to the connection framework of Bibel [6]. In fact, [6] presents methods closely related to rigid E-unification. Rigid E-unification is also used as a way to incorporate equality into the semantic tableau framework [11] for theorem proving. For simplicity, we discuss this problem in the context of sets of clauses, although the semantic tableau framework is more general and the original paper of Andrews [1] considers a more general structure of formulas (negation normal form) which has certain advantages.

The organization of this paper is as follows. In section 2, we discuss the rigid approach to first-order theorem proving in the absence of equality. In section 3, we discuss rigid theorem proving with equality, and define the simple and simultaneous rigid E-unification problems. We also give the NP-completeness proof for simple rigid E-unification. The simultaneous problem, however, is of the most interest for first-order logic with equality. Since the general problem is undecidable, we give some decidable special cases of it in section 4. These involve unit equations and Horn clauses. We also give a fully general method based on case analysis, but it involves some explicit enumeration of terms, which is often inefficient. In section 5, we give a fully general path-based method for equality in the context of rigid E-unification. This method avoids the explicit enumeration of terms, and relies instead on unification. Some modifications of this method are also presented, and complexity analyses of some of the methods are given. Another paramodulation-based rigid approach to equality reasoning is given in section 6, and a set of clauses is also given that illustrates some of its properties. In section 7, we discuss some theoretical questions related to the sizes of the amplifications needed for various methods, and a number of open problems are presented. Finally, in section 8, we present a new proof of the undecidability of the general problem of simultaneous rigid E-unification. In fact, we show that the special case of positive ground equations is still undecidable, and that for undecidability one needs only six simple rigid E-unification problems. This proof makes use of a reduction from Post's Correspondence Problem.

2 Paths and Spanning Sets

Rigid approaches to theorem proving make use of amplifications, paths, and spanning sets, which we now define. We restrict our attention to first-order logic. In addition, although rigid methods are often used for non-clausal theorem proving, we restrict our attention to sets of clauses, both f¡or simplicity and also to bring out the underlying ideas. An *amplification* of a set S of clauses is a set T of clauses including one or more copies of each clause in S, with variables renamed so that each variable appears in at most one clause in T. Thus if S is $\{C[x,y], D[u,v]\}$, then $\{C[x,y], C[x',y'], D[u,v]\}$ is an amplification of S. Of course, there are also many other amplifications.

The following result is an immediate consequence of Herbrand's theorem:

Theorem 2.1 *A set S of clauses of first-order logic is unsatisfiable iff there exist an amplification T of S and a substitution Θ such that $T\Theta$ is ground and unsatisfiable.*

However, given an amplification T, it is not clear how one decides if such a substitution Θ exists, other than by enumerating all substitutions. The matings (connection graph, Prawitz [20]) methods have been developed for this purpose. These involve paths and spanning sets, which we now define.

If S is a set $\{C_1, C_2, \ldots, C_n\}$ of clauses where all C_i are distinct, then a *path* in S is a set $\{L_1, L_2, \ldots, L_n\}$ of literals such that $L_i \in C_i$ for all i, $1 \leq i \leq n$.

A *pair* is a set of two elements.

A set Sp of pairs of literals is *spanning* for a set T of clauses if for every path P in T, there is a pair p in Sp such that p is a subset of P. We call such a set Sp a *spanning set*.

A substitution Θ is a *simultaneous unifier* of a spanning set Sp if for all pairs $\{L, M\}$ in Sp, $L\Theta$ and $M\Theta$ are complementary literals.

As an example, suppose that T is $\{\{p(a)\}, \{\neg p(x), q(x)\}, \{\neg q(a)\}\}$. Then the paths are $\{p(a), \neg p(x), \neg q(a)\}$ and $\{p(a), q(x), \neg q(a)\}$. A spanning set is $\{\{p(a), \neg p(x)\}, \{q(x), \neg q(a)\}$, and a simultaneous unifier is $\{x \leftarrow a\}$.

Theorem 2.2 *If S is a set of clauses, then there is a Θ such that $S\Theta$ is ground and unsatisfiable iff there is a spanning set Sp for S and a simultaneous unifier Θ for Sp.*

Proof. Let us express $S = \{C_1, C_2, \ldots, C_n\}$ in disjunctive normal form as $D_1 \vee D_2 \vee \ldots \vee D_m$, where each D_i is a conjunction $L_1 \wedge L_2 \wedge \ldots \wedge L_n$ for literals $L_i \in C_i$. Then $S \equiv D_1 \vee D_2 \vee \ldots \vee D_m$, and so $S\Theta$ is unsatisfiable iff $D_i\Theta$ is unsatisfiable for all i. But $D_i\Theta$ can only be unsatisfiable iff it contains two complementary literals. Also, each D_i is the conjunction of the literals in some path in S. Therefore, $S\Theta$ is unsatisfiable iff for every path P in S, $P\Theta$ contains a pair of complementary literals. Let Sp be the set of pairs $p \subseteq P$ for paths P in S, such that $p\Theta$ is a pair of complementary literals. Then Sp is a spanning set for S, and Θ is a simultaneous unifier for Sp. Conversely, if such a simultaneous unifier Θ exists, then $P\Theta$ is unsatisfiable for all paths P of S, so $(D_1 \vee D_2 \vee \ldots \vee D_m)\Theta$ is unsatisfiable, so $S\Theta$ is unsatisfiable. □

Combining this with theorem 2.1, we obtain the following result, which gives the connection between amplifications, spanning sets, and satisfiability.

Theorem 2.3 *A set S of clauses is unsatisfiable iff there is an amplification T of S, a spanning set Sp for T, and a simultaneous unifier Θ of Sp.*

Therefore, in order to show that S is unsatisfiable, we have to choose T, find Sp, and test if Θ exist. Choosing T consists in deciding how many copies of each clause of S are needed for the proof; of course, one cannot know these numbers in advance, but must try increasing numbers of copies in the search for a proof. Finding Sp involves a search through a large number of possible sets of pairs of literals for a set that is spanning. The number of possible sets Sp one needs to examine is at most exponential in the size of S. To show this, we note that the number of pairs of literals is at most the square of the number of literals in S, so each set Sp is of size polynomial in the size of S. The search for a spanning set Sp can be time consuming, but at least it is possible to verify that a set Sp is a spanning set in a small amount of space, by looking through all paths in some order, one by one. Of course, this can be made somewhat more efficient. Formally, this problem is in co-NP. Testing if Θ exists is (for first-order logic) easy, since solving simultaneous unification problems is not more difficult than solving single problems, which can be done in linear time. For this, it suffices to note that Θ unifies the pairs $\{L_1, M_1\}, \ldots, \{L_n, M_n\}$ if Θ unifies $f(L_1, \ldots, L_n)$ and $f(M_1, \ldots, M_n)$, where f is a new function symbol.

This same approach can be applied to higher-order logic. The prover of Andrews [1] uses higher-order logic, which is much more expressive, but also has a harder unification problem. Here we are concerned with the extension to first-order logic with equality.

3 Critical Pairs and Rigid E-Unification

The foregoing approach was extended to equality reasoning in [12]. The motivation for this is that it is awkward to consider the equality axioms explicitly, so one would like to build in equality in some way. Also, it is difficult to know how many copies of the equality axioms to choose. This extension to equality is done as follows.

We say that a set p is Eq-unsatisfiable if $p \cup Eq$ is unsatisfiable, where Eq are the equality axioms (reflexivity, transitivity, symmetry, and substitutivity ($x = y \supset f(\ldots x \ldots) = f(\ldots y \ldots)$)). We similarly define Eq-satisfiability and Eq-validity.

Again, by the use of Herbrand's theorem, we obtain the following result.

Theorem 3.1 *A set S of clauses is Eq-unsatisfiable iff there is an amplification T of S and a Θ such that $T\Theta$ is ground and Eq-unsatisfiable.*

As before, we want to find computable criteria for the existence of such a Θ. To do this, we need to consider more closely the structure of minimal unsatisfiable sets of equations and inequations. This will entail a translation to eliminate non-equality predicates as well as a consideration of critical pairs and critical pair proofs.

We first translate the clause set so that the only predicate in S is the equality predicate. This simplifies the formalism to some extent. In some sections of this paper, we will assume that this equality translation has already been done, and in other places we will not assume this. This translation is done by replacing a positive literal $P(r_1, \ldots, r_n)$ by the equation $f_P(r_1, \ldots, r_n) = \mathbf{true}$ and replacing a negative literal $\neg P(r_1 \ldots r_n)$ by $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$, where f_P is a new function symbol depending only on P. Assuming that "**true**" is a new constant symbol, one can show that this translation is satisfiability preserving. For now, we generally assume that this translation has been done, although sometimes we refer to the equations $f_P(r_1, \ldots, r_n) = \mathbf{true}$ as positive literals and $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$ as negative literals. We also refer to these new equations and inequations involving f_P as *literal equations* and *literal inequations*, respectively. If this translation is not done, then it is necessary to consider non-equality literals and equality literals separately, leading to more inference rules.

Definition 3.2 An *equational set* for a set S of clauses is a set of equations and inequations from S containing exactly one inequation. We also require that if an equational set contains a literal inequation $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$, then it contains exactly one literal equation having the function symbol f_P, possibly with other non-literal equations. If the equational set contains no literal inequations, then it contains no literal equations.

We will show later that any minimal unsatisfiable set of equations and inequations is an equational set. The intuition for this is that an unsatisfiable set of Horn clauses needs only one negative clause. If we had not done the translation to equality, then we would also have to consider equational sets containing equations and a positive and negative non-equality literal.

Definition 3.3 A *simplification ordering* $>$ is a partial ordering on terms that is well-founded and satisfies the following three properties:

1. The *subterm property*, that is, for all terms s, $f(\ldots s \ldots) > s$,
2. the *replacement property*, that is, for all terms r and s, $r > s$ implies $f(\ldots r \ldots) > f(\ldots s \ldots)$, and
3. the full invariance property, that is, for all terms r and s and all substitutions Θ, $r > s$ implies $r\Theta > s\Theta$.

We say that a simplification ordering is *total on ground terms* if for all distinct ground terms s and t, either $s > t$ or $t > s$.

In the sequel, we will generally assume that $>$ (or $<$) is a simplification ordering that is total on ground terms.

Definition 3.4 Suppose $r_1[u] = s_1$ and $r_2 = s_2$ are two equations and $r_1[u]$ has a non-variable subterm u that unifies with r_2. Let α be a most general unifier of u and r_2. Then $(r_1[s_2]\alpha, s_1\alpha)$ is a *critical pair* between the equations $r_1[u] = s_1$ and $r_2 = s_2$. We sometimes also call the equation $(r_1[s_2] = s_1)\alpha$ a critical pair. We call α the *critical substitution* of this critical pair. A *critical pair operation* infers $R \cup \{(r_1[s_2] = s_1)\}\alpha$ from R, where R contains the equations $r_1[u] = s_1$ and $r_2 = s_2$. We also define the inequation $(r_1[s_2] \neq s_1)\alpha$ as a critical pair between $r_1[u] \neq s_1$ and $r_2 = s_2$ and define the associated critical pair operation that infers $R \cup \{(r_1[s_2] \neq s_1)\}\alpha$ from R if R contains $r_1[u] \neq s_1$ and $r_2 = s_2$. In this case, too, we call α a critical substitution. We say that this critical pair is with respect to the simplification ordering $>$ if it is not the case that $r_i\alpha < s_i\alpha$.

Definition 3.5 A *critical pair proof* from a set q of equations and inequations is a sequence $e_1, e_2, \ldots e_n$ of equations and inequations where each e_i is either an element of q or is a critical pair between two previous equations in the sequence. A *critical pair refutation* is a critical pair proof $e_1, \ldots, e_n$ where for some term t, e_n is of the form $t \neq t$.

It is known [15, 2] that if q is an Eq-unsatisfiable set of equations and inequations, then there is a critical pair refutation from q in which each critical pair is with respect to $>$.

Theorem 3.6 *Suppose that q is a set of equations and inequations from S. Then if q is Eq-unsatisfiable, q has a minimal Eq-unsatisfiable subset p such that p is an equational set for S.*

Proof. We first note that q may contain non-literal equations and inequations as well as literal equations and inequations. We know that if q is unsatisfiable, then there is a critical pair proof of an inequation of the form $t \neq t$ from q. For this we assume that $>$ is an ordering such that **true** is minimal in the ordering. We prove the theorem by observing what kinds of critical pair operations can contribute to this proof. We note that the following kinds of critical pair operations are possible:

1. A critical pair between a non-literal equation and a non-literal equation, yielding a non-literal equation.
2. A critical pair between a non-literal equation and a non-literal inequation, yielding a non-literal inequation.

3. A critical pair between a non-literal equation and a literal inequation, yielding a literal inequation.

4. A critical pair between a non-literal equation and a literal equation, yielding a literal equation.

5. A critical pair between a literal equation and a literal inequation, yielding $\mathbf{true} \neq \mathbf{true}$.

There are no other possibilities, because each critical pair operation must involve at least one equation, and critical pairs between two literal equations must involve literal equations with the same symbol f_P, yielding $\mathbf{true} = \mathbf{true}$, which cannot contribute to a proof. Also, the symbols f_P occur only at the top level, so that the result of step 4 always has a symbol f_P at the top level and is therefore a literal equation. For the same reason, there cannot be a critical pair between a literal equation and a non-literal inequation.

Now, we can only obtain an equation of the form $t \neq t$ from step 5 or 2. (The result of step 3 will have a **true** on one side.) Working backwards, to obtain an inequation from step 2, we only need a set of non-literal equations and one non-literal inequation. Working backwards from step 5, we only need a set of non-literal equations, a literal equation with a symbol f_P, and a literal inequation with the same symbol f_P. We can let the set q be the set of equations and inequations actually used in the proof, and we observe in all cases that q is an equational set. □

We again make use of paths. A *path* is defined as for the non-equational case. A set Sp of equational sets is *spanning* for S if for every path P for S there is an equational set p in Sp such that $p \subseteq P$. Such a set Sp is called an *equational spanning set* for S.

Theorem 3.7 *Suppose that T is a set of clauses and Θ is a substitution such that $T\Theta$ is ground. Then $T\Theta$ is unsatisfiable iff there is an equational spanning set Sp for T such that for all p in Sp, $p\Theta$ is Eq-unsatisfiable.*

Proof. Similar to that for the non-equational case, except that we note that for Eq-unsatisfiability, we need to consider sets of more than two equations and inequations. If $T\Theta$ is unsatisfiable, the every path of $T\Theta$ is Eq-unsatisfiable. By theorem 3.6, we can find an unsatisfiable equational subset p of the path. Taking all of these subsets together, we obtain an equational spanning set for T as specified in the theorem. For the reverse direction, if there is an equational spanning set Sp for T as specified, then every path of $T\Theta$ is Eq-unsatisfiable, so $T\Theta$ is Eq-unsatisfiable. □

Combining this with the previous result, we obtain the following.

Theorem 3.8 *A set S of clauses is Eq-unsatisfiable iff there is an amplification T of S, an equational spanning set Sp for T, and a Θ such that for all p in Sp, $p\Theta$ is Eq-unsatisfiable.*

Now, in contrast to the non-equality case, determining if such a Θ exists can be very difficult. In order to study this question, we introduce rigid E-unification.

Definition 3.9 The (simple) rigid E-unification problem (E, e), where E is a set of equations and e is a single equation, is to determine if there is a substitution Θ such that $E\Theta \models e\Theta$ (relative to equality). Such a substitution is called a *rigid E-unifier* of e relative to E.

It is easy to see that this problem is equivalent to determining whether an equational set $(E \cup \{\neg e\})\Theta$ is Eq-unsatisfiable, since $(E \wedge \neg e)\Theta$ is Eq-unsatisfiable iff $E\Theta \models e\Theta$ relative to equality.

Definition 3.10 The *simultaneous* rigid E-unification problem $(E_1, e_1), \ldots, (E_n, e_n)$, where the E_i are sets of equations and the e_i are equations, is to determine if there is a substitution Θ such that for all i, $E_i\Theta \models e_i\Theta$ relative to equality. Such a Θ is called a *simultaneous rigid E-unifier* of E_i relative to e_i, or a *solution* of the simultaneous rigid E-unification problem.

Suppose $(E \cup \{\neg e\})$ is an equational set. Then the *associated rigid E-unification problem* is to determine if there is a Θ such that $E\Theta \models e\Theta$ relative to equality. Suppose $Sp = \{p_1, \ldots, p_n\}$ is a set of equational sets. Then the *associated simultaneous rigid E-unification problem* is $(E_1, e_1), \ldots, (En, e_n)$, where (E_i, e_i) is the simple rigid E-unification problem associated with p_i.

Proposition 3.11 *Θ is a solution of the simultaneous rigid E-unification problem associated with Sp iff for all p in Sp, $p\Theta$ is Eq-unsatisfiable.*

We therefore say that a substitution Θ is a simultaneous rigid E-unifier of Sp if for every p in Sp, $p\Theta$ is Eq-unsatisfiable.

Theorem 3.12 *A set S of clauses is Eq-unsatisfiable iff there is an amplification T of S, an equational spanning set Sp for T, and a simultaneous rigid E-unifier Θ of Sp.*

As before, this gives us a way to prove theorems involving equality. To implement this, one must choose an amplification, find equational spanning sets, and then test if such a Θ exists. The first two may be done as in the non-equality case. The third part involves simultaneous rigid E-unification. This problem has a history of faulty attempts to develop algorithms, but was recently shown to be undecidable [10, 9]. However, the problem of performing a single rigid E-unification is NP-complete. This motivates the search for other special cases of simultaneous rigid E-unification that are decidable, or even NP-complete, for applications to deduction. We first review the NP-completeness proof, to fix terminology. Then we present some special cases in which simultaneous rigid E-unification is decidable. Next we present some complete (decidable) techniques for incorporating equality into theorem proving; these techniques have a rigid flavor. As mentioned before, the decidability of these approaches does not contradict the fact that first-order logic is not decidable. We also discuss some related questions. Finally, we give a new proof that the general simultaneous rigid E-unification problem is undecidable.

3.1 NP-Completeness of Rigid E-Unification

To fix notation, we here review the proof that the simple rigid E-unification problem is NP-complete. NP-hardness was shown in [16], so it suffices to show membership in NP. For this, it suffices to show that if there is a Θ such that $E\Theta \models e\Theta$, then there is a proof of this fact whose length is polynomial in (E, e).

From now on, for simplicity we assume that E is a set of equations and inequations, and we are concerned with the problem of whether there is a (ground) Θ such that $E\Theta$ is ground and Eq-unsatisfiable.

We first note that the assumption of groundness does not lead to a loss of generality, since if $E\Theta \models e\Theta$ then for all substitutions γ, $E\Theta\gamma \models e\Theta\gamma$. Thus we can choose γ to replace distinct variables in $E\Theta$ and $e\Theta$ by distinct new constant symbols.

Definition 3.13 If E is a set of equations, then the *subterm size* $St(E)$ of E is the number of distinct subterms that appear in E. Note that a subterm is only counted once, even if it appears many times. Also, subterms are only considered identical if their variables are identical. We say that a quantity is polynomial in E if it is polynomial in $St(E)$, and similarly for sets S of clauses.

Since a term may appear many times in E, the length of E, written out as a string, may be exponential in $St(E)$.

Definition 3.14 (Recall definition 3.4.) Suppose E is a set of equations and inequations. Suppose $e[u]$ is an equation or inequation of E and and $r_2 = s_2$ (or its symmetric version) is an equation of E. Let α be a most general unifier of u with r_2. Then we call α a *critical substitution* for E. We also call α a critical substitution between $e[u]$ and the rewrite rule $r_2 \rightarrow s_2$.

Theorem 3.15 *Suppose E is a set of equations and α is a critical substitution for E. Then $St(E\alpha) \leq St(E)$.*

Proof. See [19]. □

Definition 3.16 A *uniform rewriting* step on a term-rewriting system $R[r]$ containing a rule $r \rightarrow s$, infers the system $R[s]$ in which all occurrences of the subterm r in R have been replaced by s except for the occurrence on the left-hand side of the rule $r \rightarrow s$ itself.

The following result, also from [19], will be useful to us.

Theorem 3.17 *Suppose that R is a ground term-rewriting system and $>$ is an arbitrary simplification ordering that is total on ground terms. Then we can complete R in a polynomial number of uniform rewriting steps to obtain R' such that R and R' are equivalent (that is, the sentence $R' \equiv R$ is Eq-valid), R' is terminating and confluent, for all rules $r \rightarrow s$ in R', $r > s$, and $St(R') \leq St(R)$. For this, it suffices to choose at each step the rewrite rule $r \rightarrow s$ such that s is $>$-minimal subject to the condition that $R[r]$ and $R[s]$ are not identical.*

Proof. See [19]. □

Definition 3.18 If A is a formula, then $Gr(A)$ is A with the variables x systematically replaced by new constant symbols c_x. We similarly define $Gr(R)$ for a term-rewriting system R.

We now present a proof system in which polynomial length proofs of rigid E-unifiability can be constructed. Others have also studied systems for dealing with rigid E-unification. We are not sure of the relationships of these methods to our own. In [17], a complete method is given for building in equality reasoning in the connection calculus. In [5], a method is given to combine classical and rigid E-unification. In [13], a rule-based method is given for finding complete sets of rigid E-unifiers. This uses congruence closure. In [4], a method is given to add rigid E-unification to an ordered theory resolution calculus. An algorithm for simultaneous rigid E-unification from [14] is apparently incomplete. An approach to equality combining the matrix rule with equation solving is given in [8].

We represent inference rules in the format

$$\frac{A}{B}\ X,$$

meaning that B is derivable from A, and X is the name of the rule. We use $A \vdash D$ to indicate that D may be obtained from A by a sequence of zero or more applications of such inference rules. We represent a set of equations and inequations as a quadruple (R, N, C, α) where R is a set of rewrite rules, N is a set of inequations, C is a constraint, and α is a substitution. The constraints are conjunctions of inequalities of the form $r > s$ for terms r, s. The constraints are not needed for soundness or completeness, but help to prune the search space. We require that in every inference rule, for a quadruple used as the hypothesis of the inference rule, the constraint C be *satisfiable*. This means that there must be a simplification ordering satisfying the inequalities in the constraint. To test for this, we consider the associated term-rewriting system $R_C = \{r \to s : r > s \text{ is a conjunct of } C\}$. Then C is satisfiable only if $Gr(R_C)$ is terminating, and we note that ground termination can be tested in polynomial time [18]. This proof system has four inference rules, orientation, ground completion, critical substitution, and contradiction, as follows:

$$\frac{E}{(R, N, \wedge\{r > s : r \to s \in R\}, id)} \quad \textit{Orientation}$$

where R are the equations of E oriented into rewrite rules in an arbitrary manner and N are the inequations of E and id is the identity substitution.

$$\frac{(R, N, C, \alpha)}{(R', N', C \wedge C', \alpha)} \quad \textit{Ground Completion}$$

where $Gr(R')$ is canonical and Eq-equivalent to $Gr(R)$. Thus $(R \equiv R')$ is Eq-valid. Also, $Gr(N')$ are in normal form with respect to $Gr(R')$. Furthermore, $R \supset (N \equiv N')$ and $(R \cup N) \equiv (R' \cup N')$ (relative to Eq). Note that it is possible that the orientations of the rules of R' are not uniquely determined by those of R. The additional constraint C' represents the orientation decisions that were made during the ground completion of R, as explained below. The satisfiability of the constraint $C \wedge C'$ implies that $Gr(R \cup R')$ is terminating.

We now show in more detail how the ground completion step can be done in (nondeterministic) time polynomial in $St(R \cup N)$. For this, it suffices to use the congruence closure method of [21]. It is also possible to perform this step using the method of [19]. This will complete a ground system in a polynomial number of rewrites. At each step, one chooses the rule $r \to s$ with the smallest right-hand side s that can be applied somewhere, and applies it everywhere. When a rewrite rule $u \to v$ is rewritten to $u' \to v'$, then it may be re-oriented as $v' \to u'$. Since we are working with constraints, when a rule $r \to s$ is chosen, then we include in C' all constraints of the form $s' > s$ for all rules $r' \to s'$ that are also applicable. We also add the constraint $r > s$. In this way, we complete R in a polynomial number of steps and update the constraint C to $C \wedge C'$. We also rewrite N to normal form along with the rewrites done on R, as the method progresses. As in [19], we have that $St(R' \cup N') \leq St(R \cup N)$.

$$\frac{(R, N, C, \alpha)}{(R\beta, N\beta, C\beta, \alpha\beta)} \quad \textit{Critical Substitution}$$

where β is a (non-trivial) critical substitution for $(R \cup N)$. Thus we can create critical substitutions involving rules and inequations, as well as critical substitutions involving rules and rules. We require here too that $Gr(R\beta)$ be terminating. Note that the critical substitution rule eliminates at least one variable, since β is not the identity.

$$\frac{(R, N \cup \{s \neq t\}, C, \alpha)}{(R\beta, \mathbf{false}, C\beta, \alpha\beta)} \quad \textit{Contradiction}$$

where s and t are unifiable and β is a most general unifier of s and t.

From now on, for simplicity we often omit the constraint. We write $(R, N, \alpha) \vdash (R', N', \alpha')$ if (R', N', α') can be derived from (R, N, α) using these rules. We first remark that for each one-step derivation using these rules, if $(R, N, \alpha) \vdash (R', N', \alpha')$, then there exists β such that $\alpha' \equiv \alpha\beta$. By transitivity, this also follows for many-step derivations.

We now show that for each one-step derivation, if $(R, N, \alpha) \vdash (R', N', \alpha\beta)$, then $(R \cup N)\beta \equiv (R' \cup N')$ relative to Eq. This can be shown by examining the last three rules. In particular, we consider the contradiction rule. If N has an inequation $s \neq t$ such that s, t are unifiable with most general unifier β, then $N\beta$ contains an instance of $x \neq x$ and so is Eq-unsatisfiable. Thus $(R \cup N)\beta$ is Eq-unsatisfiable and equivalent to $(R \cup \mathbf{false})$ relative to Eq. The arguments for the other rules are straightforward. It follows by transitivity that, if $(R, N, \alpha) \vdash (R', N', \alpha\beta)$ for an n-step derivation, then $(R \cup N)\beta \equiv (R' \cup N')$ relative to Eq.

Also, if $E \vdash (R, N, \beta)$ by a one-step derivation then $E\beta \equiv (R \cup N)$ relative to Eq, since then β is the identity substitution. It follows by transitivity and an above result that if $E \vdash (R, N, \beta)$ by an n-step derivation, then $E\beta \equiv (R \cup N)$ relative to Eq. Thus if $E \vdash (R', \mathbf{false}, \beta)$, then $E\beta$ is Eq-unsatisfiable. Thus we have found a rigid E-unifier, namely, β. If we desire a β' such that $E\beta'$ is ground and Eq-unsatisfiable, then this can be found by modifying β to replace variables by distinct new constant symbols.

We now show that if there is a Θ such that $E\Theta$ is Eq-unsatisfiable, then $E \vdash (R', \mathbf{false}, \alpha')$ for some R' and α'. Furthermore, the length of this proof is polynomial in the size of E. Since the result is already known, we will try to be brief.

Definition 3.19 The *length* of a proof is the number of inference steps in it.

Theorem 3.20 *Suppose R is a set of equations, N is a set of inequations, and Θ is a substitution such that $(R \cup N)\Theta$ is ground and Eq-unsatisfiable. Then $(R, N, \alpha) \vdash (R', \mathbf{false}, \alpha\beta)$ for some β such that $(R \cup N)\beta$ is Eq-unsatisfiable, and there is a proof whose length is polynomial in $St(R \cup N)$.*

Proof. By induction. For this we order the triples (R, N, α) lexicographically, first by the number of variables in $R \cup N$, and then by the ordering of a minimal ground substitution β such that $(R \cup N)\beta$ is ground and Eq-unsatisfiable. Here we are ordering substitutions using the simplification ordering $<$ which is assumed to be total on ground terms. That is, $\beta \leq \beta'$ iff for all variables x, $x\beta \leq x\beta'$. We know that some such β exists at the beginning (namely Θ), and we prove that given a triple (R, N, α) such that for some β, $(R \cup N)\beta$ is Eq-unsatisfiable, there is a proof step that generates a smaller Eq-unsatisfiable triple.

For the proof, we note that $(R \cup N)\Theta$ is a ground system and can be completed by ground completion. The result of this ground completion will be a system containing an inequation of the form $t \neq t$. We lift these ground completion steps to R and N to obtain a proof as desired. This proof will generate something having $t \neq t$ as an instance. This must therefore be an inequation of the form $r \neq s$ where r and s are unifiable, in other words, an inequation that unifies with $x \neq x$.

We assume that the ground completion rule has already been applied to $R \cup N$. We can always choose an ordering consistent with Θ for ground completion, that is, we can order terms r and s by $r >_\Theta s$ (for ground completion) if $r\Theta > s\Theta$. If $R \cup N$ is the result of a ground completion step with such an ordering, then we know that if $r = s$ is an equation in R and $r\Theta > s\Theta$, then all occurrences of r in R (other than in this equation) will have been replaced by s. Now, if $(R \cup N)\Theta$ is Eq-unsatisfiable, then either N contains already an inequation that unifies with

$x \neq x$, or $(R \cup N)\Theta$ has a critical pair. We often write a rewrite rule $r \rightarrow s$ as the equation $r = s$ in the following. Suppose the critical pair involves the two equations $(r_1\Theta)[u] = s_1\Theta$ and $r_2\Theta = s_2\Theta$, where $(r_1\Theta)[u]$ has a subterm u that is identical to $r_2\Theta$. Since all terms (with Θ applied) are ground, $((r_1\Theta)[s_2\Theta], s_1\Theta)$ is a critical pair between the equations $r_1[u] = s_1$ and $r_2 = s_2$. Now, we consider the position of the subterm u in $r_1\Theta$. If u occurs inside a Θ term that replaces a variable y of r_1, then we can write $y\Theta$ as $(y\Theta)[u]$ and let Θ' be defined by $x\Theta \equiv x\Theta'$ if $x \not\equiv y$, and $y\Theta' = (y\Theta)[s_2\Theta]$. We note in passing here that $y\Theta$ has u as a subterm. Then $\Theta' < \Theta$ and $(R \cup N)\Theta'$ is also Eq-unsatisfiable, and we can proceed by induction. If the critical substitution is trivial (the identity), then this step would have been done during ground completion. The only other possibility is that u occurs at a non-variable position of r_1 and there is a critical pair with a non-trivial critical substitution. This substitution must bind a variable to a term not containing that variable, and therefore reduces the number of variables. So we can perform a critical substitution step and again argue by induction.

There is a technicality to consider here. That is, we need to know that the equation $r_2\Theta = s_2\Theta$ still occurs in $(R \cup N)\Theta'$ in order to know that $(R \cup N)\Theta'$ is Eq-unsatisfiable. This will be true unless r_2 or s_2 contains y, since Θ and Θ' are identical on terms not containing y. We noted above that $y\Theta$ has u as a subterm, and $u \equiv r_2\Theta$. Thus $r_2\Theta$ is a subterm of $y\Theta$. If r_2 contains y, then $r_2\Theta$ has $y\Theta$ as a subterm, so it must be that $y\Theta \equiv r_2\Theta$, and therefore $y \equiv r_2$. Also, y cannot occur in s_2 since we are assuming that $r_2\Theta > s_2\Theta$. Thus the equation $r_2 = s_2$ is of the form $y = s_2$ for some variable y that does not appear in s_2. We note in passing that such equations cannot be derived if R has a nontrivial model. If r_1 also has the variable y at the u position, then this replacement of y by s_2 in $r_1\Theta$ could have occurred already in the ground completion step, so we do not need to consider this possibility. Otherwise, we perform a critical substitution step by unifying r_2 with whatever term occurs at the u position of r_1. This will reduce the number of variables, so we can proceed by induction. Thus in all the cases we need to consider, the equation $r_2\Theta = s_2\Theta$ still occurs in $(R \cup N)\Theta'$, and so $(R \cup N)\Theta'$ is Eq-unsatisfiable

We now need to show that the length of the proof is polynomial in $St(R \cup N)$. The number of variable eliminating steps (applications of critical substitutions) is linear in the number of variables. We only have to check that the combined work to ground complete is polynomial in the size of the original $R \cup N$. This is not obvious, because it could be that $R \cup N$ grows in size, and so although each ground completion is polynomial, their combined time could be exponential in the size of the original $R \cup N$. The polynomial bound is obtained by noting that ground completion does not increase the subterm size, and the work to ground complete is polynomial in the subterm size (for example, by the method of [19] or [21]). $\square$

Corollary 3.21 *Suppose E is a set of equations and inequations and Θ is a substitution such that $E\Theta$ is ground and Eq-unsatisfiable. Then $E \vdash (R', \mathbf{false}, \beta)$ for some β such that $E\beta$ is Eq-unsatisfiable, and there is a proof of this that can be found in nondeterministic polynomial time (polynomial in $St(E)$). We can find β' such that $E\beta'$ is ground and Eq-unsatisfiable, too, by replacing variables by distinct new constant symbols.*

Proof. One application of the orientation rule yields $E \vdash (R, N, id)$. Combining this with the theorem, we obtain the desired result. $\square$

Corollary 3.22 *Rigid E-unification is in NP.*

Proof. If for some Θ, $E\Theta$ is ground and Eq-unsatisfiable, then in nondeterministic polynomial time (polynomial in $St(E)$) we can find a proof of this fact, using the preceding corollary. $\square$

Since the NP-hardness of rigid E-unification is known by other methods [16], we obtain that rigid E-unification is NP-complete.

4 Decidable Cases of Simultaneous Rigid E-Unification

We now present some special cases which are fairly common in which simultaneous rigid E-unification is decidable and of reasonable complexity. This shows that the rigid E-unification approach is still viable for theorem proving with equality, in many cases. For these cases, we are still assuming that S is translated to eliminate non-equality predicates.

4.1 Unit equations

We now assume that S is a set of clauses consisting of a set E of unit equations (that is, clauses of the form $\{r = s\}$ for some terms r and s) and a set S' of clauses not containing positive occurrences of the equality predicate. We say a spanning set Sp for T is *uniform* if for all (non-literal) equations e in E, for all p_1 and p_2 in Sp, $e \in p_1$ iff $e \in p_2$.

Theorem 4.1 *Suppose S is a set of clauses consisting of a set E of unit equations and a set S' of clauses not containing positive occurrences of the equality predicate (for translated S, not containing positive non-literal equations). Then S is Eq-unsatisfiable iff there is an amplification T of S, a uniform equational spanning set Sp' for T, and a Θ such that for all p' in Sp', $p'\Theta$ is Eq-unsatisfiable.*

Proof. If such a uniform equational spanning set Sp' exists, then S is Eq-unsatisfiable as before. We now show that if S is as stated, then such a uniform equational spanning set Sp' exists. We know from theorem 3.8 that if S is Eq-unsatisfiable then there is an amplification T of S, an equational spanning set Sp for T, and a Θ such that for all p in Sp, $p\Theta$ is Eq-unsatisfiable. Now, we modify Sp to obtain a set Sp' that is uniform. We define Sp' to be $\{p \cup E : p \in Sp\}$. Then if $p\Theta$ is Eq-unsatisfiable, $(p \cup E)\Theta$ is Eq-unsatisfiable. This completes the proof. $\square$

Now, the proof actually gives us more information than stated. In particular, we have that Sp' is a spanning set for the same amplification T as Sp. This means that in order to use uniform spanning sets in this case, we do not need to increase the size of the amplification.

We now show that simultaneous E-unification problems derived from uniform spanning sets are NP-complete.

Theorem 4.2 *Suppose Sp is a uniform spanning set. Let $(E_1, e_1), \ldots, (E_n, e_n)$ be the simultaneous rigid E-unification problem corresponding to Sp. Then there is a simple rigid E-unification problem (E, e) such that for all Θ, Θ is a solution of the problem (E, e) iff Θ is a solution of the simultaneous problem $(E_1, e_1), \ldots, (E_n, e_n)$. Also, (E, e) is obtainable from $(E_1, e_1), \ldots, (E_n, e_n)$ in polynomial time.*

Proof. Consider the simultaneous rigid E-unification problem corresponding to Sp. This will be a set $(E_1, e_1), \ldots, (E_n, e_n)$ of simple rigid E-unification problems in which all sets of non-literal equations are identical. That is, all E_i are identical except possibly for some literal equations. Let E be this set of non-literal equations. To obtain (E,e), we first eliminate these literal equations. The only kind of equational sets containing literal equations that we need to consider are those containing a single positive literal equation $f_P(u_1 \ldots u_n) = \mathbf{true}$ and a single negative literal equation $f_P(v_1 \ldots v_n) \neq \mathbf{true}$. The literal equation $f_P(u_1 \ldots u_n) = \mathbf{true}$ would then occur in some E_i and then e_i would be $f_P(v_1 \ldots v_n) = \mathbf{true}$. We can remove the literal equation from E_i and replace e_i by the equation $g(u_1 \ldots u_n) = g(v_1 \ldots v_n)$ where g is a new function symbol, since the only way to obtain a contradiction is to derive equality between (instances of) u_i and v_i for all i. In this way, we can replace (E_i, e_i) by $(E, g(u_1 \ldots u_n) = g(v_1 \ldots v_n))$. We then obtain an equivalent simultaneous problem $(E, e'_1), \ldots, (E, e'_n)$ by performing this translation on all (E_i, e_i). Also, we can assume that there are no literal equations or inequations among E or the e'_i. Let us express e'_i as $r_i = s_i$ and let e be the equation $f(r_1, \ldots, r_n) = f(s_1, \ldots, s_n)$, where f is a new function symbol. We then note that $\wedge_i(E\Theta \models e_i\Theta)$ iff $E\Theta \models \wedge_i e_i \Theta$ iff $E\Theta \models e$. □

Corollary 4.3 *Suppose Sp is a uniform spanning set. Then the problem of deciding if there is a Θ such that for all p in Sp, $p\Theta$ is Eq-unsatisfiable, is in NP.*

Proof. A substitution Θ satisfies (for all p in Sp, $p\Theta$ is Eq-unsatisfiable) iff Θ is a solution to the simultaneous rigid E-unification problem associated with Sp. However, if the simultaneous problem is uniform, then there is a simple rigid E-unification problem (E,e) such that Θ is a solution to (E,e) iff Θ is a solution to the simultaneous problem. Furthermore, (E,e) is obtainable in polynomial time, and the simple rigid E-unification problem is in NP. Putting all of these facts together, we obtain the above corollary. □

Corollary 4.4 *Suppose S is a set of clauses consisting of a set E of equations and a set S' of clauses containing no positive occurrences of the equality predicate (that is, no positive non-literal equations). Suppose that T is an amplification of S. Then the problem of determining whether there exists a ground Θ such that $T\Theta$ is ground and Eq-unsatisfiable is solvable in exponential time.*

Proof. We know that $T\Theta$ is Eq-unsatisfiable iff there is a uniform spanning set Sp such that for all p in Sp, $p\Theta$ is Eq-unsatisfiable. Each spanning set Sp will be of the form $E \cup p'$ where p' has one or two elements. Therefore the number of elements of Sp is polynomial in S, and the number of such Sp is exponential in S. To determine whether $T\Theta$ is Eq-unsatisfiable, we can enumerate all the uniform spanning sets Sp of T and test for each one whether such a Θ exists. There may be exponentially many such spanning sets Sp, and testing for the existence of Θ will take at most exponential time (since it is a problem in NP). Thus the total running time is exponential. (Actually, we can bound the position of this problem in the polynomial hierarchy better than this, if we like, but we still cannot improve the exponential time bound at present.) □

This shows us (essentially) that the simultaneous rigid E-unification problem is still NP-complete if all equations occur in unit clauses.

4.2 Horn clauses

Definition 4.5 A *Horn clause* is a clause containing at most one positive literal. Thus $\{\neg p(x), \neg q(x), r(x)\}$ and $\{\neg p(x), \neg q(x)\}$ are Horn clauses.

Definition 4.6 We say that a clause C is *all-negative* or just *negative* if every literal in C is a negative literal.

Definition 4.7 A *Horn equational set* h for a set S of clauses is a set such that for all elements y of h, either either y is a literal from S or y is a Horn clause from S. Also, h contains at most one all-negative clause.

Definition 4.8 A set h of Horn clauses *covers* a path P of S if P intersects every clause in h.

Definition 4.9 A set Sp of Horn equational sets is *spanning* for S if for every path P in S, there is a Horn equational set h in Sp such that h covers P. We call such a set Sp a *Horn spanning set* for S.

Definition 4.10 A substitution Θ is a *simultaneous unifier* of a Horn spanning set Sp if for all Horn equational sets h in Sp, $h\Theta$ is Eq-unsatisfiable.

Theorem 4.11 *If S is a set of clauses, then there is a Θ such that $S\Theta$ is ground and Eq-unsatisfiable iff there is a Horn spanning set Sp for S and a simultaneous unifier Θ for Sp.*

Proof. Similar to the proofs of theorems 2.2 and 3.7. If $S\Theta$ is Eq-unsatisfiable, then there is an equational spanning set, which we can take as Sp since it is also a Horn spanning set. The only new part is to show that if a simultaneous unifier Θ for Sp exists, then S is Eq-unsatisfiable. This follows because if $h\Theta$ is Eq-unsatisfiable and h covers a path P, then $P\Theta$ is also Eq-unsatisfiable. □

We note that this theorem doesn't say much yet, since if $S\Theta$ is Eq-unsatisfiable we can just take Sp as an equational spanning set (without Horn clauses). But the other direction of the theorem gives us some new information, and will be useful later.

We now develop some special cases that will be useful for obtaining another decidable subcase for rigid E-unification.

Definition 4.12 Suppose that H is a subset of S such that every clause C in H is a Horn clause. Then we say that H is *separable from* S if H satisfies the following conditions:

1. No clause D in $S - H$ contains a literal M such that M is a non-literal equation.
2. If there exists a clause C of H and a literal L of C such that L is of the form $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$, then no clause D in $S - H$ contains a literal M of the form $f_P(s_1, \ldots, s_n) = \mathbf{true}$.
3. H contains no all-negative clauses.

Briefly stated, 2. means that predicate symbols that appear negatively in H cannot appear positively outside of H.

Definition 4.13 A Horn spanning set Sp for S is *uniform* for S relative to H if H is separable from S and every Horn equational set h in Sp is of the form $H \cup h'$ where h' contains either

1. A non-literal inequation from a clause in $S - H$ or
2. A literal inequation from a clause in $S - H$ or

3. A literal inequation from a clause in $S - H$ and a literal equation with the same predicate, also from a clause in $S - H$.

We note that if H is separable from S and T is an amplification of S, then there is a H' that is separable from T; this is obtained by taking the copies of H that are in T.

Theorem 4.14 *Suppose S is a set of clauses and H is a set of Horn clauses in S that is separable from S. Then S is Eq-unsatisfiable iff there is an amplification T of S, a set H' of Horn clauses in T that is separable from T, a uniform (relative to H) Horn spanning set Sp' for T, and a Θ such that for all p' in Sp', $p'\Theta$ is Eq-unsatisfiable.*

Proof. If such a uniform equational spanning set Sp' exists, then S is Eq-unsatisfiable as before. We now show that if S is as stated, then such a uniform equational spanning set Sp' exists. We know from theorem 3.1 that if S is Eq-unsatisfiable, then there is an amplification T of S and a Θ such that $T\Theta$ is Eq-unsatisfiable. Let H' be the part of T consisting of copies of clauses from H and let $\mathcal{P}$ consist of *all* the paths in $T - H'$. Let Sp be $\{p \cup H' : p \in \mathcal{P}\}$. Then we know that for all p in Sp, $p\Theta$ is unsatisfiable. (This follows by simple propositional reasoning.) Also, every path in T is covered by some element of Sp. Now, we construct a spanning set Sp' for T which contains for each element $p \cup H'$ in Sp a path $q \cup H'$, where q is minimal such that $(q \cup H')\Theta$ is Eq-unsatisfiable. By properties of Horn sets and equality, it follows that q must contain exactly one all-negative clause C. Since H' has no all-negative clauses and q consists only of single literals, C must be a negative literal L. If L is a non-literal inequation, then by the definition of separable it follows that q contains no other literals, and condition 1. of the definition of uniformity is satisfied. If L is a literal inequation, then by the definition of separable it follows that q may contain no other literals or perhaps one other literal M which is a literal equation having the same symbol f_P as L. These cases correspond to conditions 2. and 3. of the definition of uniformity, respectively. Therefore Sp' is uniform and satisfies the conditions of the theorem.

We may see that this condition on q is satisfied by considering that H' and the equality axioms are Horn clauses, and so we can obtain a proof of unsatisfiability by Prolog-style backward chaining from the inequation L. By the definition of separable, it follows that when we back chain into the H' region we never get out of it again. The only way that the H' region can contribute to the proof is by deriving a set G of literal equations and non-literal equations that, together with q, are Eq-unsatisfiable. By theorem 3.6, we know that a minimal set $q \cup G$ of this form either contains (a) no literal equations or inequations, or (b) a literal inequation of the form $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$ and a literal equation of the form $f_P(s_1, \ldots, s_n) = \mathbf{true}$. The set $q \cup G$ may also contain non-literal equations, and since no non-literal equations occur in $T - H'$, these must be elements of G. If (a) holds, then condition 1 of the definition of uniformity is satisfied. If (b) holds, then L is $f_P(r_1, \ldots, r_n) \neq \mathbf{true}$. Depending on whether the literal $f_P(s_1, \ldots, s_n) = \mathbf{true}$ is a lemma derived from H' or in q, we obtain conditions 2 and 3 of the definition of uniformity, respectively.

This completes the proof. □

We note here as in theorem 4.1 that the proof actually gives us more information than stated. In particular, we have that Sp' is a spanning set for the same amplification T as Sp. This means that in order to use uniform spanning sets in this case, we do not need to increase the size of the amplification.

Definition 4.15 The (simple) rigid Horn unification problem (H, e), where H is a set of Horn clauses (possibly involving equality) and e is a single equation, is to determine if there is a substitution Θ such that $H\Theta \models e\Theta$ (relative to equality). Such a substitution is called a *rigid Horn unifier* of e relative to H.

It is easy to see that this problem is equivalent to determining whether a Horn set $(H \cup \{\neg e\})\Theta$ is Eq-unsatisfiable, since $(H \wedge \neg e)\Theta$ is Eq-unsatisfiable iff $H\Theta \models e\Theta$ relative to equality.

Definition 4.16 The *simultaneous* rigid Horn unification problem $(H_1, e_1), \ldots, (H_n, e_n)$, where the H_i are sets of Horn clauses (possibly involving equality) and the e_i are equations, is to determine if there is a substitution Θ such that for all i, $H_i\Theta \models e_i\Theta$. Such a Θ is called a *simultaneous rigid Horn unifier* of the H_i relative to e_i.

Suppose that e is an equation and $(H \cup \{\neg e\})$ is a Horn equational set. Then the *associated rigid Horn unification problem* is to determine if there is a Θ such that $H\Theta \models e\Theta$ relative to equality. Suppose $Sp = \{p_1, \ldots, p_n\}$ is a set of Horn equational sets. Then the *associated simultaneous rigid Horn unification problem* is $(H_1, e_1), \ldots, (H_n, e_n)$, where (H_i, e_i) is the simple rigid Horn unification problem associated with p_i.

Proposition 4.17 *Θ is a solution of the simultaneous rigid Horn unification problem associated with Sp iff for all p in Sp, $p\Theta$ is Eq-unsatisfiable.*

We now show that if Sp is uniform, the simultaneous rigid Horn unification problem associated with Sp can be converted to an equivalent simple rigid Horn unification problem. Then we show that this simple problem is in NP.

Theorem 4.18 *Suppose Sp is a uniform Horn spanning set relative to H. Let $(H_1, e_1), \ldots, (H_n, e_n)$ be the simultaneous rigid Horn unification problem corresponding to Sp. Then there is a simple rigid Horn unification problem (H, e) such that for all Θ, Θ is a solution of the problem (H, e) iff Θ is a solution of the simultaneous problem $(H_1, e_1), \ldots, (H_n, e_n)$. Also, (H, e) is obtainable from $(H_1, e_1), \ldots, (H_n, e_n)$ in polynomial time.*

Proof. Very similar to the proof of theorem 4.2. It is only necessary to modify (H_i, e_i) to (H, e_i') and then introduce the new function symbol f as before. The only difference is that we may have either 1) $H_i = H$ and e_i is a single non-literal equation, or 2) $H_i = H$ and e_i is a single literal equation, or 3) $H_i = H \cup L_i$ where L_i is a non-literal equation, and e_i is a non-literal equation with the same symbol f_P. This follows by the definition of uniformity of Sp relative to H. Cases 1) and 3) are handled as for the unit equational case. Thus in case 3), we introduce a new equation e_i' as in theorem 4.2 such that (H, e_i') is equivalent to (H_i, e_i). Case 2) is new, but this does not matter, since it is in any event a single equation and can be treated the same as case 1). We thus obtain an equivalent system $(H, e_1'), \ldots, (H, e_n')$. Then we introduce e equivalent to the conjunction $e_1' \wedge \ldots \wedge e_n'$ as in theorem 4.2, so that (H, e) is equivalent to $(H, e_1'), \ldots, (H, e_n')$. □

We now present a set of inference rules in order to show that the simple rigid Horn unification problem is in NP. We refer to a Horn clause, all of whose literals are equality or inequality literals, and which contains at least one positive literal, as a *Horn equational clause.* Note that this includes ordinary equations as a special case. *A non-positive clause* is a clause containing at least one negative literal, and in this context, refers to a Horn equational clause containing at least one negative literal.

Note that every Horn equational clause is either a unit equation or a non-positive clause.

We need to redefine the notion of a critical substitution, as follows:

Definition 4.19 Suppose H is a set of Horn equational clauses. Suppose $e[u]$ is a clause of H and and $r_2 = s_2$ (or its symmetric version) is a unit equation of H. If e is a non-positive clause, suppose that u is a term occurring in its leftmost inequation. Let α be a most general unifier of u with r_2 . Then we call α a *critical substitution* for E. We also call α a critical substitution between $e[u]$ and the rewrite rule $r_2 \to s_2$.

As before, we represent a set of Horn equations and Horn inequations as a quadruple (R, N, C, α) where R is a set of rewrite rules, N is a set of non-positive clauses, C is a constraint, and α is a substitution. The constraints are conjunctions of inequalities of the form $r > s$ for terms r, s. As before, the constraints are not needed for soundness, but help to prune the search space. We require that in every inference rule, for a quadruple used as the hypothesis of the inference rule, the constraint C be satisfiable. This proof system has four inference rules, orientation, ground completion, critical substitution, and inequation elimination, as follows:

$$\frac{H}{(R, N, \wedge\{r > s : r \to s \in R\}, id)} \quad \textit{Orientation}$$

where H is a set of Horn equational clauses and R are the (unit) equations of H oriented into rewrite rules in an arbitrary manner and N are the non-positive clauses of H and id is the identity substitution.

$$\frac{(R, N, C, \alpha)}{(R', N', C \wedge C', \alpha)} \quad \textit{Ground Completion}$$

where $Gr(R')$ is canonical and Eq-equivalent to $Gr(R)$. Thus $(R \equiv R')$ is Eq-valid. Also, $Gr(N')$ are in normal form with respect to $Gr(R')$. Furthermore, $R \supset (N \equiv N')$ and $(R \cup N) \equiv (R' \cup N')$ (relative to Eq). The additional constraint C' represents the orientation decisions that were made during the ground completion of R. The satisfiability of the constraint $C \wedge C'$ implies that $Gr(R \cup R')$ is terminating. Ground completion is done exactly as for the non-Horn equational case; it is only necessary to rewrite N as R is rewritten.

$$\frac{(R, N, C, \alpha)}{(R\beta, N\beta, C\beta, \alpha\beta)} \quad \textit{Critical Substitution}$$

where β is a (non-trivial) critical substitution for $(R \cup N)$. Note that in the Horn case which we are now considering, we can generate critical substitutions between rules and non-positive clauses. We require here too that $Gr(R\beta)$ be terminating. Note as before that the critical substitution rule eliminates at least one variable, since β is not the identity.

$$\frac{(R, N \cup \{\{s \neq t, L_2, \ldots, L_n\}\}, C, \alpha)}{(R\beta, N \cup \{\{L_2, \ldots, L_n\}\}\beta, C\beta, \alpha\beta)} \quad \textit{Inequation Elimination}$$

where s and t are unifiable and β is a most general unifier of s and t. We note that the resulting clause $\{L_2, \ldots, L_n\}\beta$ can be empty, and we consider an empty clause as equivalent to "**false**."

We also have a special case of inequation elimination:

$$\frac{(R, N \cup \{\{s \neq t, r_1 = r_2\}\}, C, \alpha)}{((R \cup \{r_1 \to r_2\})\beta, N\beta, (C \wedge (r_1 > r_2))\beta, \alpha\beta)} \quad \textit{Inequation Elimination}$$

Here β is a most general unifier of s and t, as before. The equation $r_1 = r_2$ (or $r_2 = r_1$) is converted to a rewrite rule and added to R, and the constraint is modified.

From now on, for simplicity we often omit the constraint. We write $(R, N, \alpha) \vdash (R', N', \alpha')$ if (R', N', α') can be derived from (R, N, α) using these rules. As before, for each one-step derivation using these rules, if $(R, N, \alpha) \vdash (R', N', \alpha')$, then there exists a substitution β such that $\alpha' \equiv \alpha\beta$. By transitivity, this also follows for many-step derivations. Also, as for the unit equational case, if $(R, N, \alpha) \vdash (R', N', \alpha\beta)$ for an n-step derivation, then $(R \cup N)\beta \equiv (R' \cup N')$ relative to Eq. That is, the rules are equivalence (hence satisfiability) preserving. Similarly, if $H \vdash (R, N, \beta)$ by an n-step derivation, then $H\beta \equiv (R \cup N)$ relative to Eq. Thus if $H \vdash (R', N' \cup \{\{\}\}, \beta)$, then $H\beta$ is Eq-unsatisfiable. This means that we have found a rigid Horn unifier, namely, β (suitably grounded, if desired).

We now show that if there is a Θ such that $H\Theta$ is Eq-unsatisfiable, then $H \vdash (R', N' \cup \{\{\}\}, \alpha')$ for some R', N', and α'. Furthermore, the length of this proof is polynomial in the size of H. Since the result is similar to that for the unit equational case, we will omit some details.

Theorem 4.20 *Suppose H is a set of equational Horn clauses and Θ is a substitution such that $H\Theta$ is ground and Eq-unsatisfiable. Then $H \vdash (R', N' \cup \{\mathbf{false}\}, \beta)$ for some R', N', and β such that $H\beta$ is Eq-unsatisfiable, and there is a proof whose length is polynomial in $St(H)$.*

Proof. The basic idea of the proof is to observe that we can ground complete the system $H\Theta$ by applying ground completion to the pure equational part of H. We also rewrite literals in inequations of H. If some inequation rewrites to the form $r \neq r$, then we can delete it. We note that when all the inequations in a Horn clause are deleted, we either obtain "**false**" or a new equation, which can then be added in to the unit equations, which in turn can be ground completed again. If no inequation rewrites to this form then we can do nothing more and unless $\{\}$ has been derived, $H\Theta$ is consistent. This ground proof from $H\Theta$ can then be lifted to a proof from H.

Formally, let H be $E \cup N$, where E are unit equations and N are non-positive clauses. We showed in theorem 3.20 that the ground completion of $E\Theta$ could be lifted to E, and the same proof applies here since all necessary critical pair operations can still be performed. Thus we obtain $H \vdash (R, N', \alpha)$, where $\Theta = \alpha\alpha'$, α' is a substitution such that $(R \cup N')\alpha'$ is ground, and $R\alpha'$ is a confluent ground term-rewriting system equivalent to $E\Theta$. Also, we can assume that N' is in normal form relative to R (by the ground completion inference rule).

Suppose that $N'\alpha'$ has some clause C with an inequation of the form $r \neq r$. Then there is an inequation elimination rule that can be applied to (R, N', α), simplifying the system.

Suppose that none of the inequations of $N'\alpha'$ are of the form $r \neq r$. Suppose also that **false** (i.e., $\{\}$) has not been derived. Then $(R \cup N')\alpha'$ is consistent, since we can interpret every term to its $R\alpha'$ - normal form. Also, we interpret "=" as equality. Let us call this interpretation I. Now, all remaining inequations $s \neq t$ have s, t with different $R\alpha$-normal forms. This means that these inequations are true (satisfied) in the interpretation I. Also, the unit equations are in R and thus are true in the interpretation I. Thus every clause in $(R \cup N')\alpha'$ has a true literal, so all clauses are satisfied by I. But we know as in theorem 3.20 that $(R \cup N')\alpha' \equiv H\Theta$, which is unsatisfiable.

The only other case is that $N'\alpha'$ and thus N' contains $\{\}$. In this case we have already derived a proof of a contradiction.

We now show that this proof is of polynomial length. This is straightforward since the ground completion rule is polynomial time, the orientation rule is only applied once at the beginning, the critical substitution rule eliminates a variable, and the inequation elimination rule eliminates an inequation. By polynomial time we mean polynomial in $St(H)$. This is valid since all steps can be done in time polynomial in $St(R \cup N)$ and none of the inference steps increase $St(R \cup N)$. □

Corollary 4.21 *Simple rigid Horn unification is NP-complete.*

Proof. If for some Θ, $H\Theta$ is ground and Eq-unsatisfiable, then in nondeterministic polynomial time (polynomial in $St(H)$) we can find a proof of this fact, using the preceding result. Also, rigid Horn unification has rigid E-unification as a special case, and the latter is known to be NP-hard. Hence simple rigid Horn unification is NP-complete. □

Corollary 4.22 *Suppose S is a set of clauses, H is a subset of S that is separable from S, and Sp is a Horn spanning set for S that is uniform relative to H. Then the problem of deciding if there is a Θ such that for all h in Sp, $h\Theta$ is Eq-unsatisfiable, is in NP.*

Proof. A substitution Θ satisfies (for all h in Sp, $h\Theta$ is Eq-unsatisfiable) iff Θ is a solution to the simultaneous rigid Horn unification problem associated with Sp. However, if the simultaneous problem is uniform, then by theorem 4.18 there is an equivalent simple rigid Horn unification problem (H, e). Furthermore, (H, e) is obtainable in polynomial time, and the simple rigid Horn unification problem is in NP. Putting all of these facts together, we obtain the corollary. □

Corollary 4.23 *Suppose S is a set of clauses consisting of a set H of equational Horn clauses that is separable from S. Suppose that T is an amplification of S. Then the problem of determining whether there exists a Θ such that $T\Theta$ is ground and Eq-unsatisfiable is solvable in exponential time.*

Proof. Let H' be the part of T containing copies of clauses in H. Then H' is separable from T. Also, we know by theorem 4.14 that $T\Theta$ is Eq-unsatisfiable iff there is a uniform (relative to H') spanning set Sp' such that for all h in Sp', $h\Theta$ is Eq-unsatisfiable. In addition. any such Sp' must be of size polynomial in T, since each set p in Sp' consists of H' together with one or two literals from $T - H'$, and there are only a polynomial number of such combinations to consider. So to determine whether $T\Theta$ is Eq-unsatisfiable, we can enumerate all the uniform spanning sets Sp' of T and test for each one whether such a Θ exists. There may be exponentially many such spanning sets Sp, and testing for the existence of Θ will take at most exponential time (since it is a problem in NP). Thus the total running time is exponential. □

4.3 Case analysis

We now present yet another special case in which the rigid E-unification problem can be solved in nondeterministic polynomial time (in a certain sense). From now on in this paper, we will not assume that the equality translation (introduction of f_P terms) has been done.

Definition 4.24 Suppose that S is a set of clauses. We say that S is *splittable* if for every clause C in S and for every literal L in C, if L is a (positive) equation then L has no variables in common with $C - \{L\}$.

Definition 4.25 Suppose S is a set of clauses. Then we say that $S_1, \ldots, S_n$ is a *splitting* of S iff the following properties hold:

1. S is unsatisfiable iff for all i, S_i is unsatisfiable.
2. For all S_i, if a clause C in S_i contains a positive equality literal, then C is a unit equation.

Theorem 4.26 *Suppose S is splittable. Then there is a splitting $S_1, \ldots, S_n$ of S such that n is at worst exponential in the number of literals in S. This splitting can be found in time exponential in S.*

Proof. One obtains the S_i by recursively performing the following operation: Given S, we find a clause C of S containing L such that L is a positive equation. We then consider the two sets $(S - \{C\}) \cup \{\{L\}\}$ and $(S - \{C\}) \cup \{C - L\}$. Since L shares no variables with the remainder of S, S is unsatisfiable iff both of these two sets are unsatisfiable. Also, each of these two clause sets has fewer literals than S. The same transformation can be applied to each of these clause sets, if they have non-unit equations. The theorem then follows by induction. □

We note that if S is splittable and $S_1 \ldots S_n$ is a splitting of S, then (essentially) by corollary 4.4, we can reduce simultaneous rigid E-unification for each S_i to simple rigid E-unification, which is decidable and NP-complete. Thus we can decide in exponential time whether for all i, there exists a substitution Θ_i such that $S_i\Theta_i$ is Eq-unsatisfiable.

Definition 4.27 We say that a set of clauses S is *rigidly Eq-unsatisfiable* if there is a substitution Θ such that $S\Theta$ is ground and Eq-unsatisfiable.

Now, if S is an amplification, we can determine if it is rigidly Eq-unsatisfiable by splitting it and then testing each S_i for rigid Eq-unsatisfiability. Testing S_i for rigid Eq-unsatisfiability is in exponential time, by corollary 4.4. However, even if all S_i are rigidly Eq-unsatisfiable, it does not follow that S is, because the S_i together have more copies of clauses. But if all S_i are rigidly Eq-unsatisfiable, then S is Eq-unsatisfiable, which is what we are really interested in anyway. Thus we have the following definition:

Definition 4.28 An *interpolation test* is a predicate P on clause sets such that if S is rigidly Eq-unsatisfiable, then $P(S)$ is true, and if $P(S)$ is true, then S is Eq-unsatisfiable.

We note that as far as theorem proving goes, interpolation tests are as good as rigid Eq-unsatisfiability tests. We can test if S is Eq-unsatisfiable by enumerating amplifications $T_1, T_2, \ldots, T_n$ of S and applying an interpolation test to each one. Formally,

Theorem 4.29 *Suppose P is an interpolation test. Then a set S of clauses is Eq-unsatisfiable iff there is an amplification T of S such that $P(T)$ is true.*

Proof. Suppose S is Eq-unsatisfiable. Then there is an amplification T of S such that T is rigidly Eq-unsatisfiable. Since P is an interpolation test, $P(T)$ is true. Now, suppose that there is an amplification T of S such that $P(T)$ is true. Then S is Eq-unsatisfiable, by the definition of an interpolation test. □

We have such a test for clause sets that are splittable.

Theorem 4.30 *Suppose S is a splittable clause set. The the following is an interpolation test for S:*

1. *Generate a splitting $S_1, \ldots, S_n$ of S.*
2. *Test each S_i for rigid Eq-unsatisfiablity; we note that this test is in exponential time.*
3. *Return "***true***" if all these tests succeed.*

Proof. By the definition of splitting, if 2. succeeds, then S is unsatisfiable. Also, if S is rigidly unsatisfiable, then so is each S_i. □

Thus in this case we have an interpolation test whose complexity is exponential. The question arises whether in the general case such an interpolation test exists. If so, it could take the place of simultaneous rigid E-unification, even though the latter is undecidable. The author has invested considerable effort to find such an interpolation test or to prove it does not exist, without success in either direction. However, the large number of special cases for which efficient methods exist suggests that some method might work in general.

We now indicate how the above splitting method can be extended to the general case, that is, to non-splittable clause sets. However, we cannot give any complexity bound in terms of S.

Definition 4.31 Suppose S is an arbitrary set of clauses. Then a substitution Θ is a *splitting substitution* for S if $S\Theta$ is splittable.

Definition 4.32 A *shared variable* for S is a variable x such that for some clause C in S and some literal L in C, L is a positive equality literal and x appears both in L and in $C - \{L\}$. Note that we are not assuming that the equality translation has been done on S.

Proposition 4.33 *If S is a set of clauses, and Θ is a substitution such that for all shared variables x, $x\Theta$ is a ground term, then Θ is a splitting substitution for S.*

We then obtain the following result.

Theorem 4.34 *An arbitrary set S of clauses is Eq-unsatisfiable iff there is an amplification T of S, a splitting substitution Θ for T, and a splitting $T_1, \ldots, T_n$ of $T\Theta$ such that each T_i is rigidly Eq-unsatisfiable.*

Proof. If S is Eq-unsatisfiable, then there is an amplification T of S and a ground substitution Θ such that $T\Theta$ is ground and Eq-unsatisfiable. Now, such a Θ is a splitting substitution for T, and for any splitting $T_1, \ldots, T_n$ of $T\Theta$, all T_i will be rigidly Eq-unsatisfiable. Conversely, if such a splitting substitution Θ exists, then $T\Theta$ is Eq-unsatisfiable, hence S is also Eq-unsatisfiable. □

This implicitly gives a fully general equality theorem proving method based on rigid E-unification. To test if S is Eq-unsatisfiable, we enumerate amplifications T of S, and for each T, we enumerate splitting substitutions Θ and splittings for $T\Theta$, testing the T_i for rigid Eq-unsatisfiability using the method of theorem 4.30. The drawback is that it is necessary to enumerate such substitutions Θ. However, one can easily see that it is only necessary to enumerate splitting substitutions Θ whose domain is the set of shared variables of S. Also, in many cases there are only a small number of shared variables, making the task easier. Finally, we can make this method somewhat more efficient by choosing a simplification ordering $>$ on terms and noting that if S itself contains an equation $r = s$, then we need not consider substitutions Θ containing subterms $r\alpha$ such that $r\alpha > s\alpha$. That is, we need only consider Θ that are irreducible with respect to ordered rewriting. The combination of these techniques yields a simple and fully general equality method that may be sufficient for many purposes.

The set of shared variables can be made smaller, too, by splitting T to obtain clauses sets T_i having Horn subsets H_i that are separable from T_i. For this, we can say that a variable of a clause C is shared if it appears

1. in two (positive) equations in C, or
2. in an equation of C and in a non-equality literal of C.

This suffices because any set T_i without such shared variables must consist of a set H_i of equational Horn clauses, together with clauses not containing positive occurrences of equality. It follows that H_i is separable from T_i, and so one can test in exponential time if a Θ exists making $T_i\Theta$ Eq-unsatisfiable, by corollary 4.23.

5 Path Paramodulation

We now give another fully general solution method for the problem of rigid Eq-unsatisfiability. We cannot say much in general about the complexity of this method, but it avoids the necessity to enumerate substitutions. This method, path paramodulation, is based on paramodulation. This path-based equality method is complete (in a sense to be described) and has a rigid flavor. Path paramodulation converts an amplification S to a set of paths, then performs rigid operations on the paths. If all paths can be contradicted, then we know that S is Eq-unsatisfiable. Also, because of its rigid flavor, the search space for a given amplification S is finite. For this inference system, we consider a path to be a multiset of literals, one literal from each clause. Path paramodulation has five inference rules, as follows.

$$\frac{S}{S \supset p_1, \ldots, p_n} \quad \textit{Path Creation}$$

where S is an amplification and $p_1 \ldots p_n$ is a listing of the set of paths in S. Note that for large S, just to list the set of all paths can be impractical. Therefore, it may be better for practical purposes to let $p_1, \ldots, p_n$ be an equational spanning set for S. We do not know, however, if this preserves completeness.

$$\frac{S \supset p_1, \ldots, T \cup \{L[u], r = s\}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, T\alpha \cup L\alpha[s\alpha], \ldots, p_n\alpha} \quad \textit{Paramodulation}$$

where α is a most general unifier of u and r. Here we do not distinguish $r = s$ from $s = r$. Note that L is either an equation, an inequation, or a non-equality literal and L has the non-variable term u as a subterm. (We are not assuming that the equality translation has been done on S.) Also, note that the two "parent" clauses of the paramodulation are deleted.

Often we will want to specify a termination ordering $<$ that is total on ground terms and restrict paramodulation so that the inference is not performed if $r\alpha < s\alpha$. We call this paramodulation *with respect to* the ordering $<$.

$$\frac{S \supset p_1, \ldots, p_{i-1}, T \cup \{L, \neg M\}, p_{i+1}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, p_{i-1}\alpha, p_{i+1}\alpha, \ldots, p_n\alpha} \quad \textit{Resolution}$$

where L and $\neg M$ are literals and α is a most general unifier of L and M. The path p_i has been deleted since a contradiction has been derived.

$$\frac{S \supset p_1, \ldots, p_{i-1}, T \cup T_1 \cup \{L\}, p_{i+1}, \ldots, T \cup T_2 \cup \{L'\}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, p_{i-1}\alpha, p_{i+1}\alpha, \ldots, (T \cup \{L'\})\alpha, \ldots, p_n\alpha} \quad \textit{Factoring}$$

where α is a most general unifier of L and L'. Note that the T parts of the paths $T \cup T_1 \cup \{L\}$ and $T \cup T_2 \cup \{L'\}$ are identical even before the application of α. As a special case, if two non-empty paths are identical, one of them can be deleted.

$$\frac{S \supset p_1, \ldots, T \cup \{r \neq s\}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, p_{i-1}\alpha, p_{i+1}\alpha, \ldots, p_n\alpha} \quad \textit{Inequation Removal}$$

where α is a most general unifier of r and s. Again, the path p_i has been deleted because a contradiction has been derived.

We consider an empty set of paths as equivalent to "**false**". An empty path, however, is equivalent to "**true**."

Theorem 5.1 (Soundness) *If we can derive $S \supset p_1, \ldots, p_n$ in this system, then $S \supset (\wedge p_1) \vee (\wedge p_2) \vee \ldots \vee (\wedge p_n)$ (relative to Eq), where $\wedge p_i$ is the conjunction of the literals in the path p_i. For this, we interpret an empty disjunction (when all paths have been contradicted) as "**false**."*

Proof. For the path creation rule, we note that $(\wedge p_1) \vee (\wedge p_2) \vee \ldots \vee (\wedge p_n)$ is essentially a disjunctive normal form of S, which is logically equivalent to S.

Now, for each inference rule

$$\frac{S \supset p_1, \ldots, p_n}{S\alpha \supset q_1, \ldots, q_p}$$

we show that if $S \supset (\wedge p_1) \vee (\wedge p_2) \vee \ldots \vee (\wedge p_n)$ (relative to Eq) then $S\alpha \supset (\wedge q_1) \vee (\wedge q_2) \vee \ldots \vee (\wedge q_n)$ (relative to Eq). By induction, this yields the desired result. We show this for each inference rule separately. For example, consider the paramodulation rule. By equality reasoning, it follows that from $C_1[u] \wedge r = s$ one can derive $C_1\alpha[s\alpha]$, where α is a most general unifier of u and r. Therefore, if $S \supset (\wedge p_1) \vee \ldots \vee (T \wedge C_1[u] \wedge r = s) \vee \ldots \vee (\wedge p_n)$, then $S\alpha \supset (\wedge p_1)\alpha \vee \ldots \vee (T\alpha \wedge C_1[u]\alpha \wedge (r = s)\alpha) \vee \ldots \vee (\wedge p_n)\alpha$, hence $S\alpha \supset (\wedge p_1)\alpha \vee \ldots \vee (T\alpha \wedge C_1\alpha[s\alpha]) \vee \ldots \vee (\wedge p_n)\alpha$. This corresponds to the conclusion of the paramodulation rule. The other rules can be handled similarly. □

Corollary 5.2 *If $S \vdash (S\beta \supset$ **false**$)$ in this system, then $S\beta$ is Eq-unsatisfiable, hence S is Eq-unsatisfiable.*

We now want to show the converse, that is, if S is Eq-unsatisfiable, then for some β, $S\beta \supset$ **false** can be derived in this system. That is, this system is complete. However, for this we have to place some restrictions on the amplification S. That is, we show that if S is Eq-unsatisfiable, then there is an amplification T of S and a substitution β such that $T\beta \supset$ **false** can be derived. In order to specify T, it is necessary to define resolution-paramodulation proofs and proof trees.

Definition 5.3 A *resolution-paramodulation proof* of C_n from S is a sequence $C_1C_2 \ldots C_n$ of clauses such that each C_i is either in S or is a resolvent or paramodulant of two previous clauses in the sequence. We also call such a proof a *linear proof,* since it is a linear sequence. This is not meant to imply anything about the theorem proving strategy used to obtain the proof.

Definition 5.4 A *resolution-paramodulation proof tree* of C from S is a binary tree in which the root node is labeled with the clause C, the leaves are labeled with clauses in S, and if the two children of a node N are labeled with clauses C_1 and C_2, respectively, then N is labeled with a resolvent or a paramodulant of C_1 and C_2. The distinctive feature of a proof tree is that each clause is used only once.

We use the greek letters Π and Π' to refer to resolution-paramodulation proofs (or proof trees). Also, $In(\Pi)$ is the set of input clauses of the proof Π, that is, the set of C_i such that $C_i \in S$ for a linear proof, or, the set of labels of leaf nodes, for a proof tree. Note that any linear proof can be converted to a proof tree by (possibly) duplicating some inference steps.

Definition 5.5 For a resolution-paramodulation proof tree Π of "**false**" from S, we define an amplification A_Π which has one clause C_N for each leaf node N in the tree Π, and such that for two leaf nodes M and N, C_M and C_N share no variables.

We note that A_Π will be an amplification of $In(\Pi)$ for all Π. Furthermore, every clause in $In(\Pi)$ will appear exactly once in A_Π (assuming that all the clauses in Π are distinct). Also, the number of clauses in A_Π will be one greater than the number of resolution-paramodulation steps in Π. This observation follows from a property of binary trees.

For the following proof, we need to consider paths in S as multisets of literals, one from each clause in S, and we need the following definition.

Definition 5.6 Suppose that q_j and p'_i are paths, considered as multisets of literals. Let $\{q_1 \ldots q_m\}$ and $\{p'_1 \ldots p'_n\}$ be multisets of paths. A *path correspondence* is a function $F : \{q_1 \ldots q_m\} \mapsto \{p'_1 \ldots p'_n\}$ from multi-sets of paths to multi-sets of paths which is onto, such that for all i and j, if $F(q_j) = p'_i$, then q_j is a super-multiset of p'_i.

Theorem 5.7 (Completeness) *If S is Eq-unsatisfiable, then there is an amplification T of S and a substitution β such that $T \vdash (T\beta \supset$ **false**$)$ in this system.*

Proof. Suppose S is Eq-unsatisfiable. Then there is a (non-rigid) resolution-paramodulation proof of the empty clause from $S \cup \{x = x\}$, since resolution and ordered paramodulation are complete relative to equality [15, 3]. It follows that there is a (rigid) resolution-paramodulation proof tree Π of the empty clause from S. In particular, we can choose Π so that all of the labels of the leaves have disjoint variables. We show that there is a substitution β such that $A_\Pi \vdash (A_\Pi\beta \supset \textbf{false})$, that is, we can let T be A_Π.

We prove the theorem by induction on the size (number of resolution-paramodulation steps) of the proof tree Π. We first do the base case, that is, Π consists of a single node, labeled by the empty clause. Thus S contains the empty clause. Then A_Π will contain one copy of the empty clause, and nothing else. Then A_Π has no paths, that is, the set of paths of A_Π is empty. Since we are considering an empty set of paths as equivalent to **false**, we obtain a proof simply by the path creation rule from A_Π, with β as the identity substitution.

Now we do the inductive step. Suppose that Π contains more than just a single node, and that one of the resolution-paramodulation steps of Π derives D from C_1 and C_2, where C_1 and C_2 are labels of leaves of the proof tree. Let Π' be the remaining proof from $(T - \{C_1, C_2\}) \cup \{D\}$. Since Π is a proof tree, so is Π'. (This is not always true for linear proofs, since C_1 and C_2 may be used elsewhere in a linear proof.) Then A'_Π is $(A_\Pi - \{C_1, C_2\}) \cup \{D\}$. Let T' be A'_Π for ease of notation. Since Π' has fewer inference steps than Π, we can assume by induction that for some β', $T' \vdash (T'\beta' \supset \textbf{false})$. We then show that for some β, $T \vdash (T\beta \supset \textbf{false})$.

Let $p'_1 \ldots p'_n$ be the paths of T'. Then we assume by induction that $T' \vdash (T'\beta' \supset$ **false**). Since this proof must begin with the path creation rule, we have that $(T' \supset p'_1 \ldots p'_n) \vdash (T'\beta' \supset$ **false**). Let $p_1 \ldots p_m$ be the paths of T. Then by the path creation rule we obtain $T \vdash (T \supset p_1 \ldots p_m)$. We show that there is a (partial) proof $(T \supset p_1 \ldots p_m) \vdash (T\alpha \supset q_1 \ldots q_m)$ such that every q_j is a superset (super-multiset) of some p'_i and every p'_i is a sub-multiset of some q_j. More precisely, there is a path correspondence $F : \{q_1 \ldots q_m\} \mapsto \{p'_1 \ldots p'_n\}$. We will show that such a partial proof exists below. For now, let us just assume this and see how the rest of the proof can be carried out.

Assuming that we have a proof $(T \supset p_1 \ldots p_m) \vdash (T\alpha \supset q_1 \ldots q_m)$ as specified, the steps in the proof $(T' \supset p'_1 \ldots p'_n) \vdash (T'\beta' \supset$ **false**) can be simulated on $(T\alpha \supset q_1 \ldots q_m)$, in the sense that a path correspondence exists. For this, every step on p'_i has to be performed on all the q_j such that $F(q_j) = p'_i$. Suppose that γ is the substitution generated by the application of this inference rule on p'_i. Let $s(q)$ for a path q denote the path obtained by applying an inference rule to q as in the simulating or simulated proof. Thus $s(p'_i)$ is $p'_i\gamma$ with some literals deleted or modified according to the inference rule used. Then if $F(q_j) = p'_i$, q_j has p'_i as a sub-multiset and we can apply the same inference rule on q_j, generating the substitution γ also. We may have other paths q_k such that $F(q_k) = p'_i$; these paths will now become $q_k\gamma$. These will have the path $p'_i\gamma$ as a sub-multiset. Thus we can apply the same inference rule to $q_k\gamma$ as we applied to q_j, but since γ has already been applied, and we are in a rigid framework, γ need not be applied again. We then update the path correspondence F so that $F(s(q_j)) = s(p'_i)$ and $F(s(q_k)) = s(p'_i)$. This will preserve the fact that F is a path correspondence.

The factoring rule is slightly different, since a pair of paths are involved. However, as before, one can simulate a factoring operation on paths p'_i and p'_j by factoring operations on paths q and r such that $F(q) = p'_i$ and $F(r) = p'_j$. Suppose p'_i is $T \cup T_1 \cup \{L\}$ and p'_j is $T \cup T_2 \cup \{L'\}$. Then the result of the factoring operation on p'_i and p'_j is the path $(T \cup \{L\})\gamma$ for some most general unifier γ of L and L'. Now, since q and r have p'_i and p'_j as sub-multisets, q is $T \cup T_q \cup \{L\}$ and r is $T \cup T_r \cup \{L'\}$ for some T_q and T_r. By a factoring operation on q and r, we obtain $(T \cup \{L\})\gamma$, which still preserves the path correspondence. Then we perform the same operation on other paths q and r such that $F(q) = p'_i$ and $F(r) = p'_j$. The effect is similar, except that the substitution γ will have already been applied and need not be applied again.

At the end, all the paths p'_i are removed, yielding that $(T' \supset p'_1 \ldots p'_n) \vdash (T'\beta' \supset$ **false**). Now, if we have a path correspondence $F : P_1 \to P_2$ for sets P_1 and P_2 of paths and P_2 is empty, P_1 is empty, also. Therefore, if the simulated proof is a proof of $(T'\beta' \supset$ **false**), by simulating it we also obtain a proof of $(T\alpha\beta' \supset$ **false**). That is, we have a proof $(T\alpha \supset q_1 \ldots q_m) \vdash (T\alpha\beta' \supset$ **false**). Together with the proof $(T \supset p_1 \ldots p_m) \vdash (T\alpha \supset q_1 \ldots q_m)$, this yields the result $(T \supset p_1 \ldots p_m) \vdash (T\alpha\beta' \supset$ **false**). Therefore, using the path creation rule, $T \vdash (T\alpha\beta' \supset$ **false**).

Now, it remains to do the step that we left out, namely, to show that there is a proof $(T \supset p_1 \ldots p_m) \vdash (T\alpha \supset q_1 \ldots q_m)$ such that there is a path correspondence between $q_1 \ldots q_m$ and $p'_1 \ldots p'_n$. This implies that every q_j is a super-multiset of some p'_i. We show that such a proof exists by case analysis on the inference steps. Suppose D is obtained by a paramodulation step between C_1 and C_2. Let us write C_1 as $B_1 \cup \{L[u]\}$ and C_2 as $B_2 \cup \{r = s\}$. We can then write D as $B_1\alpha \cup \{L\alpha[s\alpha]\} \cup B_2\alpha$, where α is a most general unifier of r and u. Let T_1 be such that $T = T_1 \cup \{C_1, C_2\}$. (Here we use $\cup$ for multiset union.) Every path p_i of T includes one of the literals L_1 from C_1 and one of the literals L_2 from C_2, together with a path in T_1. We now consider several cases, depending on whether the literal L_1 from C_1 is in B_1 or not, and depending on whether the literal L_2 in C_2 is in B_2 or not. Let p_i be a path in T that includes the literals $L[u]$ and $r =$ s, together with a path p_i^1 in

T_1. Performing the paramodulation rule on the path p_i, we remove the literals $L[u]$ and $r = s$ and add the literal $L\alpha[s\alpha]$. The result of this operation will be the path $(p_i - \{L[u], r = s\})\alpha \cup \{L\alpha[s\alpha]\}$. The same operation can then be done on all the other paths p_j of T which are of the form $p_j^1 \cup \{L[u]\alpha(r = s)\alpha\}$ for some path p_j^1 in $T_1\alpha$. All these paths will contain the literals $L[u]$ and $r = s$, which will have been instantiated to $L[u]\alpha$ and $(r = s)\alpha$ because we are in a rigid framework, and so will not require any further substitution. In all these other paths, the paramodulation rule will remove the literals $L[u]\alpha$ and $(r = s)\alpha$ and replace these literals by $L\alpha[s\alpha]$. In this way, we obtain the proof $(T \supset p_1 \ldots p_m) \vdash (T\alpha \supset q_1 \ldots q_m)$, by performing all of these paramodulation inference steps.

Now, every path p' in T' will include one of the literals of D, plus literals from other clauses in T'. We note that $T' = (T - \{C_1, C_2\})\alpha \cup D$. Consider a path q_j. It must include a literal of $B_1\alpha$ and a literal of $B_2\alpha$, or else it includes the literal $L\alpha[s\alpha]$ resulting from the paramodulation step. In addition, q_j must include a literal from each of the other clauses $T_1\alpha$ in $T\alpha$. We show that a path correspondence F exists. For this step, we need to have that paths are multisets of literals. If q_j includes this literal $L\alpha[s\alpha]$ then we can let $F(q_j)$ be q_j, since q_j will be a path in T', too. If q_j includes a literal from $B_1\alpha$ and an arbitrary literal L_2 from $C_2\alpha$, then we can let $F(q_j)$ be $q_j - \{L_2\}$, which will be a path in T' since $B_1\alpha$ is a subset of D. If q_j includes a literal from $B_2\alpha$ and the literal $L\alpha$ from $C_1\alpha$, then we can let $F(q_j)$ be $q_j - \{L_1\}$, which will be a path in T', since $B_2\alpha$ is a subset of D. If q_j does not include literals from $B_1\alpha$ or $B_2\alpha$, then it must include both $L[u]\alpha$ and $(r = s)\alpha$, but this cannot be, because we have already applied the paramodulation operation to all such paths.

The reasoning is similar for the other operations (resolution between C_1 and C_2, and resolution of C_1 with $x = x$). We use the factoring rule on paths for the factoring operations on clauses that may be performed as part of a resolution inference. The factoring rule on paths was designed not only to simulate clause factoring, but also to be compatible with path correspondences, which is needed for the rest of the argument. For clause resolution with $x = x$, we use inequation removal on paths, which has the same effect.

This completes the proof. □

We note that in an implementation, one might want to modify these inference rules so that the literals involved in paramodulation, resolution, or equation removal are not deleted from their paths. One might also want to allow additional rules, such as simplification (rewriting) relative to some ordering. However, here we are only interested in establishing a general framework, which others may refine.

One interesting feature of this method is that it solves the simultaneous rigid E-unification problem, in a sense. In particular, if S consists entirely of equational clauses (that is, all literals are equations or inequations), and T is an amplification of S, then every path in T will be a conjunction of equations and inequations. Therefore, if S is Eq-unsatisfiable, and T is an amplification of S, and $T \vdash (T\beta \supset$ **false**), then β is a simultaneous rigid E-unifier of all the paths in T. We have just shown that if S is Eq-unsatisfiable, then some such T and β exist such that $T \vdash (T\beta \supset$ **false**), that is, our inference system will find the simultaneous rigid E-unifier β. Even though the general simultaneous rigid E-unification problem is undecidable, this technique finds enough simultaneous rigid E-unifiers to make it sufficient for theorem proving purposes. This fact makes clear the mismatch between simultaneous rigid E-unification and theorem proving, in the sense that it is still possible that there are efficient complete applications of simultaneous rigid E-unification to theorem proving, even though simultaneous rigid E-unification is undecidable.

Theorem 5.8 *Suppose that T is an amplification and there is a proof $T \vdash (T\beta \supset$ **false**) *in this system. Then this proof has a number of steps equal to at most the number of paths of T times the number of clauses in T.*

Proof. Each inference rule except for path creation reduces the number of literals in at least one path. □

The implication of this is that it is decidable whether there is a proof $T \vdash (T\beta \supset$ **false**) in this system, and one need only examine proofs of length exponential in T (since the number of paths is at most exponential in T). This does not contradict the undecidability of rigid E-unification, since this system may need much larger amplifications.

We now consider sets S of ground clauses and amplifications T of S. Such an amplification T must be considered as a multiset, since all copies of a ground clause are identical. We would like to know how large an amplification T of S is needed to find a proof by path paramodulation if S is Eq-unsatisfiable. We have the following result.

Recall that if S is a set of clauses, $St(S)$ is the number of distinct subterms appearing in clauses of S, each subterm being counted only once, regardless of how many times it appears in S.

Theorem 5.9 *There is a fixed polynomial π such that for all m, n and for all Eq-unsatisfiable sets S of ground clauses with $St(S) = n$ having at most m literals per clause, there is an amplification T of S having $m * 2^{\pi(n)}$ copies of each clause of S, such that there is a path paramodulation proof $T \vdash (T\beta \supset$* **false**).

Proof. We use a counting argument. Let π be the polynomial from [19] such that if a set E of ground equations and inequations has at most n distinct subterms, then there is a critical pair proof of some inequation of the form $s \neq s$ from E, that has at most $\pi(n)$ steps. Consider a path p in T. For each clause C in S, let L be the literal in C that appears the greatest number of times in p. Let q be the path in S consisting of the set of such literals L. Note that every element of q appears at least $2^{\pi(n)}$ times in q, since C has at most m literals. Now, using a completion approach such as that in [19], we know that there is a completion proof of some such inequation $s \neq s$ from q involving $\pi(n)$ critical pair steps. Expanding this proof to a tree, we obtain a paramodulation proof tree of $s \neq s$ with at most $2^{\pi(n)} - 1$ steps. The path p has enough copies of each element of q to enable this proof to be accomplished in path paramodulation. One application of inequation removal on the inequation $s \neq s$ yields a proof of **false**, that is, the path is removed. This can be done for all paths, obtaining an empty set of paths, which represents **false**. In this way we obtain the desired proof $T \vdash (T\beta \supset$ **false**). □

Corollary 5.10 *Under the conditions of the theorem, if S has k clauses, there is a path paramodulation proof $T \vdash (T\beta \supset$* **false**) *having at most $m^k 2^{\pi(n)}$ steps.*

Proof. Each path requires $2^{\pi(n)}$ steps for a proof. The number of paths in S is at most m^k. □

We note (by the arguments given in [19]) that this still applies if we do path paramodulation with respect to an arbitrary termination ordering $<$ that is total on ground terms. That is, we never do paramodulations replacing a subterm $r\alpha$ by $s\alpha$ if $r\alpha < s\alpha$. This is better than the situation for clause paramodulation, as we shall see.

5.1 A modified system

We now consider path paramodulation with the same set of inference rules as above except that the paramodulation rule is replaced by the following:

$$\frac{S \supset p_1, \ldots, T \cup \{L[u], r = s\}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, T\alpha \cup L\alpha[s\alpha] \cup (r = s)\alpha, \ldots, p_n\alpha} \quad \textit{Paramodulation}$$

where α is a most general unifier of u and r. As above, we do not distinguish $r = s$ from $s = r$. Note that L is either an equation, an inequation, or a non-equality literal and L has the non-variable term u as a subterm. The difference between this rule and the corresponding rule for path paramodulation is that the equation $r = s$ is not deleted.

Theorem 5.11 *Suppose S is an Eq-unsatisfiable set of ground clauses. Then $S \vdash (S \supset$ **false**$)$ in this system. That is, for ground clause sets, we do not need larger amplifications.*

Proof. Each path is Eq-unsatisfiable, and therefore we can obtain a contradiction by equational completion and resolution. The substitution β generated will be the identity substitution, and thus does not appear explicitly. The proof can even be obtained by a sequence of rewriting operations, as in [19]. This means that the literal paramodulated into can be deleted, as in modified path paramodulation. Thus we can simulate this proof by a sequence of paramodulations using the above inference rule, followed by an inequation removal operation or a resolution operation on each path. Using ideas from [19], we can also show that there is a proof in which the number of operations is a polynomial in $St(S)$ times the number of paths in S. Also, for ground sets S, we do not need the factoring rule. □

We note that this theorem does not seem to be true for the original path paramodulation system, because it is difficult to bound the length of a resolution-paramodulation proof even from a set of ground clauses.

We now consider modified path paramodulation for non-ground clause sets. In general, each inference rule either binds a variable or else essentially treats the system as a ground system. The number of variable bindings is limited by the number of variables. We would expect a result such as the following:

Suppose S is an Eq-unsatisfiable set of clauses and T is an amplification of S such that for some substitution β, $T \vdash (T\beta \supset \textbf{false})$ in modified path paramodulation. Then there is a proof of $T \vdash (T\beta \supset \textbf{false})$ whose length is at most a polynomial in T times the number of paths in T.

We would expect this to be true, because the number of variable binding steps is linear in T. In between variable bindings, the set T is treated essentially as a ground system. Unfortunately, this result is not true. Later, in the discussion of our undecidability proof, we will construct sets S of clauses such that $S \vdash (S\beta \supset \textbf{false})$, but such that there is no recursive bound on the length of these proofs. If we perform paramodulation with respect to a size-respecting term ordering, we can probably limit the length of the proof to be recursive in T, but it might be a stack of exponentials in T since unification can increase term size by an exponential amount. Also, restricting paramodulation in this way would miss some proofs, because of the undecidability result.

5.2 Delayed path creation

We now modify the above two path paramodulation systems to avoid the need to explicitly list all of the paths in S. This is done by modifying the path creation rule so that the paths are only generated as needed, and not all at once at the beginning. In this delayed path creation system, paths can contain clauses as well as literals, and such paths can be split up into more paths by partitioning one of the clauses they contain. The new version of the path creation rule is simply the following:

$$\frac{S}{S \supset S} \quad \textit{Path Creation}$$

We also have a path splitting rule as follows:

$$\frac{S \supset p_1, \ldots, p_{i-1}, T \cup \{\{L\} \cup C\}, p_{i+1}, \ldots, p_n}{S \supset p_1, \ldots, p_{i-1}, T \cup \{\{L\}\}, T \cup \{C\}, p_{i+1}, \ldots, p_n} \quad \textit{Path Splitting}$$

where L is a literal and C is a non-empty clause. This splits the path $T \cup \{\{L\} \cup C\}$ containing the clause $\{\{L\} \cup C\}$ into two paths, one containing just $\{L\}$ and one containing C. The other rules are unchanged, except that instead of literals L in paths, we now have to consider unit clauses $\{L\}$. Thus the paramodulation rule for the original path paramodulation inference system would look like this:

$$\frac{S \supset p_1, \ldots, T \cup \{\{L[u]\}, \{r = s\}\}, \ldots, p_n}{S\alpha \supset p_1\alpha, \ldots, T\alpha \cup \{L\alpha[s\alpha]\}, \ldots, p_n\alpha} \quad \textit{Paramodulation}$$

Soundness and completeness follow easily from the soundness and completeness of the original systems. However, delayed path creation is more practical for large sets of clauses, since one does not have to explicitly create all the paths, but only those that are needed for subsequent inference rules.

6 Rigid Clause Paramodulation

Another approach to theorem proving with equality is rigid clause paramodulation, which is related to path paramodulation. We distinguish this from ordinary non-rigid clause paramodulation. Rigid clause paramodulation is essentially paramodulation and resolution in which all the variables in a set of clauses are treated rigidly. This system has the following five inference rules, where T is a set of clauses:

$$\frac{T \cup \{C_1[u], \{r = s\} \cup D\}}{T\alpha \cup \{C_1[u], \{r = s\} \cup D\}\alpha \cup \{C_1\alpha[s\alpha] \cup D\alpha\}} \quad \textit{Paramodulation}$$

where α is a most general unifier of u and r. Here we do not distinguish $r = s$ from $s = r$. Note that C_1 is an arbitrary clause having the non-variable term u as a subterm. Also, note that neither of the two "parent" clauses of the paramodulation are deleted.

$$\frac{T \cup \{C \cup \{L\}, \{\neg M\} \cup D\}}{T\alpha \cup \{C \cup \{L\}, \{\neg M\} \cup D\}\alpha \cup \{C \cup D\}\alpha} \quad \textit{Resolution}$$

where L and $\neg M$ are literals and α is a most general unifier of L and M.

$$\frac{T \cup \{C \cup \{L, M\}\}}{T\alpha \cup \{C \cup \{L\}\}\alpha} \quad \textit{Factoring}$$

where α is a most general unifier of L and M.

$$\frac{T \cup \{C \cup \{r \neq s\}\}}{T\alpha \cup \{C\}\alpha} \quad \textit{Inequation Removal}$$

where α is a most general unifier of r and s.

$$\frac{T \cup \{\{\}\}}{\mathbf{false}} \quad \textit{Contradiction}$$

The idea is that we do resolution and paramodulation on T, but all substitutions are applied to the whole set of clauses, not just to the one or two clauses involved in the inference.

Theorem 6.1 *Rigid paramodulation is sound, that is, if there is a proof of $\{\}$ from T, then T is Eq-unsatisfiable.*

Proof. Except for inequation removal, rigid paramodulation is a restriction of ordinary non-rigid resolution and paramodulation, which is sound. Also, the inference rule for inequation removal is sound, by properties of equality. □

Theorem 6.2 *Rigid paramodulation is complete, that is, if S is an Eq-unsatisfiable set of clauses, then there is an amplification T of S such that there is a rigid paramodulation proof of* **false** *(that is, $\{\}$) from T.*

Proof. Suppose S is Eq-unsatisfiable. Then, since ordinary resolution-paramodulation is complete, there is a resolution-paramodulation proof of $\{\}$ from $S \cup \{x = x\}$. Therefore there is also a resolution-paramodulation proof tree Π of $\{\}$ from $S\cup\{x = x\}$. Recall that A_Π is an amplification having one clause for each use of an element of S in this proof. Suppose that we rename variables in A_Π so that all clauses have disjoint sets of variables. Then the proof from S can be simulated by a rigid proof from A_Π. The reason for this is that Π itself, with variables possibly renamed, is a rigid proof, since each clause is only used once. Now, the (rigid) proof Π may use instances of the clause $x = x$. These can all be eliminated by applications of the inequation removal rule. Thus we can let T be A_Π with the instances of $x = x$ removed. □

We consider whether the proof length for rigid paramodulation-resolution can be recursively bounded in terms of the amplification T. Since there are only a linear number of variable binding steps, we only need to consider portions of the search in which no new variables are bound. If there are a bounded number of literals, then the search space is bounded, because resolution can only generate an exponential number of clauses from a given set of literals. Therefore, if the minimal proof length cannot be recursively bounded, neither can be the number of literals generated. These new literals can only be generated by paramodulation steps (without binding variables), and so there must be a non-recursive upper bound on the number of paramodulation steps. Even if we do ordered paramodulation, we might have a non-recursive number of paramodulation steps. In fact, we cannot even show finiteness of the search space. We could conceivably generate a sequence $e_1, e_2, e_3 \ldots$ of equations and using these to paramodulate into a given literal L, generate a sequence $L_1, L_2, L_3 \ldots$ of literals with $L_1 < L_2 < L_3 \ldots$. It is possible that by using ideas analogous to those of [19], we might get a better bound or at least show finiteness. We consider it an interesting question whether one can do paramodulation even on ground clause sets in a more efficient manner.

We note that it is possible to get a better result for path paramodulation, as remarked earlier, since we can bound the size of a ground amplification needed in order to be able to obtain a proof. Also, non-rigid clause paramodulation and rigid paramodulation are identical for sets of ground clauses, since there are no variables. However, for clause paramodulation from sets S of ground clauses, if we choose a termination ordering that respects term size, then there are only an exponential number of new literals that can be generated.

Definition 6.3 An ordering $<$ on terms *respects term size* iff for all ground terms s, t, if $s < t$ then s (as a character string) is no longer than t.

Definition 6.4 The *length* of a literal is the number of characters in it, written out as a character string. We can similarly speak of the length of a term.

Definition 6.5 A clause C is a *tautology* if for some literal L, C contains both L and $\neg L$.

Theorem 6.6 *Suppose that S is a set of ground clauses. Suppose we perform resolution and paramodulation on S, with respect to an ordering that respects term size. Then the set of non-tautologous clauses C that can be generated from S by resolution and paramodulation has at most 3^{a^n} elements, where a is the number of function, constant, and predicate symbols in S and n is the length of the longest literal in S.*

Proof. The only literals that can be generated are those whose length is at most n. There are at most a^n such literals (not counting negations). Each such literal can appear either positively, negatively, or not at all in a non-tautologous clause. Assuming tautologous clauses are deleted, we obtain 3^{a^n} possibilities in all. □

Corollary 6.7 *Under the conditions of the theorem, if S is Eq-unsatisfiable, then there is a resolution-paramodulation proof whose length is at most 3^{a^n}.*

This is worse than the situation for path paramodulation, where we can use an arbitrary termination ordering and still obtain an exponential bound on proof length, by the remark after corollary 5.10.

6.1 Non-Horn equality problems

Let us consider the following simultaneous rigid E-unification problem, which illustrates some properties of ordered paramodulation:

$$
\begin{array}{ll}
f(d_0) = d_0 \models x = d_0 & (x \text{ is } f^i(d_0) \text{ for some } i) \\
f(d_0) = d_1 \wedge f(d_1) = d_0 \models x = d_1 & (i \text{ is odd}) \\
f(d_0) = d_1 \wedge f(d_1) = d_2 \wedge f(d_2) = d_0 \models x = d_2 & (i = 3k-1, \text{ some } k) \\
f(d_0) = d_1 \wedge f(d_1) = d_2 \wedge f(d_2) = d_3 \wedge f(d_3) = d_0 & \\
\quad \models x = d_3 & (i = 4k-1, \text{ some } k) \\
& \\
\ldots & \\
& \\
f(d_0) = d_1 \wedge f(d_1) = d_2 \wedge \ldots \wedge f(d_{n-2}) = d_{n-1} & \\
\quad \wedge f(d_{n-1}) = d_0 \models x = d_{n-1} & (i = nk-1, \text{ some } k)
\end{array}
$$

This formulation is a modification by Harald Ganzinger of a similar one by the author. Any solution substitution binds x to $f^i(d_0)$, where i is congruent to -1 modulo $2, 3, 4, \ldots$, and n. Such an i will be of the form $km - 1$ for some $k \geq 0$, where m is the least common multiple of $1, 2, 3, \ldots, n$. We note that since there are about $n/(\log n)$ primes less than n, and each one is at least 2, m is asymptotically at least $2^{n/(\log n)}$, and therefore exponential in n. We consider whether this solution $f^i(d_0)$ can be found by rigid paramodulation. We obtain the following set of clauses whose simultaneous rigid E-unification problem is that given above:

$$
\begin{array}{l}
P_1(x) \supset x \neq d_0 \\
P_1(x) \supset f(d_0) = d_0 \\
\\
P_2(x) \supset x \neq d_1 \\
P_2(x) \supset f(d_0) = d_1 \\
P_2(x) \supset f(d_1) = d_0 \\
\\
P_3(x) \supset x \neq d_2 \\
P_3(x) \supset f(d_0) = d_1 \\
P_3(x) \supset f(d_1) = d_2 \\
P_3(x) \supset f(d_2) = d_0 \\
\\
P_4(x) \supset x \neq d_3 \\
P_4(x) \supset f(d_0) = d_1 \\
P_4(x) \supset f(d_1) = d_2 \\
P_4(x) \supset f(d_2) = d_3 \\
P_4(x) \supset f(d_3) = d_0 \\
\ldots \\
\\
P_n(x) \supset x \neq d_{n-1} \\
P_n(x) \supset f(d_0) = d_1 \\
P_n(x) \supset f(d_1) = d_2 \\
P_n(x) \supset f(d_2) = d_3 \\
P_n(x) \supset f(d_3) = d_4 \\
\ldots \\
P_n(x) \supset f(d_{n-2}) = d_{n-1} \\
P_n(x) \supset f(d_{n-1}) = d_0
\end{array}
$$

$$P_1(x) \vee P_2(x) \vee P_3(x) \vee \ldots \vee P_n(x)$$

Such clause sets are unsatisfiable for every n, since we can let x be $f^i(d_0)$ where i is $m-1$. However, such clause sets are difficult for some good theorem provers. We feel that these clause sets are interesting, because non-Horn equality problems are not common.

Suppose that we do ordered rigid paramodulation using an ordering that respects term size. The equations in this set of clauses are all ground equations, and the only other terms in these clauses are occurrences of the variable x. The only possible paramodulations are therefore between two ground equations or between a ground equation and x. After a paramodulation between a ground equation and x, x will be bound to a constant symbol. Also, by paramodulating between two equations of the form $f(d_i) = d_k$, we may obtain equations of the form $s = t$ where both s and t are constant symbols. More equations of this type can then be obtained by further paramodulations. However, these are the only kinds of equations and bindings for x that may be obtained using ordered rigid paramodulation on the given clause set.

For this proof, we need to consider a larger amplification in order to get a proof. Without a larger amplification, only the instance $x = f^j(c)$ or something larger will lead to a contradiction, and this instance can never be constructed by ordered rigid paramodulation with the ordering we have chosen (which respects term size).

The conclusion is that even if there is a Θ such that $T\Theta$ is ground and unsatisfiable, it may be that ordered rigid paramodulation from T cannot find a proof. This may not be true for unrestricted rigid paramodulation, since we can always paramodulate backwards to derive $p_1(x) \supset x \neq f^j(d_0)$, $p_2(x) \supset x \neq f_j(d_0)$, and so on, and then obtain the proof using the axiom $x = x$. Of course, ordered non-rigid paramodulation can find a proof from this set of clauses, because it implicitly generates a larger amplification. We see then that ordered paramodulation trades larger terms for a larger amplification.

The question then arises, how many more instances in general will ordered rigid paramodulation with a size-respecting ordering, need to get a proof? At present, we cannot say much about this problem.

7 Sizes of Amplifications

We now consider the general question of how large an amplification one needs for rigid theorem proving using various methods. For example, using the above paramodulation-based approach, the size of the amplification A_Π depends on the length of a resolution-paramodulation proof Π. It is not clear how this depends on the size of a minimal amplification T of S such that T is rigidly unsatisfiable.

In general, given a rigid theorem proving method M and a set S of clauses, let $Amp_M(S)$ be the number of clauses in the smallest amplification T of S such that the unsatisfiability of T can be demonstrated by method M. The question we would like to propose is how the quantities $Amp_M(S)$ relate to one another for various methods M. For example, we can let M_1 be unrestricted simultaneous rigid E-unification; thus $Amp_{M_1}(S)$ is the number of clauses in the smallest T such that T is rigidly Eq-unsatisfiable, that is, there is a Θ such that $T\Theta$ is Eq-unsatisfiable. We can let M_2 be path paramodulation. Note that this reduces to rigid completion in the pure equational case. We can let M_3 be resolution on the equality transformation of Brand [7]. (Note that this is not the same as the equality translation used earlier, which involves the new function symbols f_P.) This method converts a set S of clauses to a set S' such that $S' \cup \{x = x\}$ is unsatisfiable (without the equality axioms) iff S is Eq-unsatisfiable. Then $Amp_{M_3}(S)$ is the size of the minimal T such that T is an amplification of $S' \cup \{x = x\}$ and T is rigidly unsatisfiable. Also, we can let M_4 be rigid clause paramodulation, and M_5 be modified path paramodulation (in which parents of paramodulations are not deleted).

The question we would like to study is the relationships between $Amp_{M_i}(S)$ for $i = 1, 2, 3, 4, 5$. In particular, is there a recursive function f_{ij} such that $Amp_{M_i}(S) \leq f_{ij}(St(S), Amp_{M_j}(S))$ for all S? Can one give a better bound on f_{ij}, such as exponential? Let us call a function f an *ij-bounding function* if $Amp_{M_i}(S) \leq f_{ij}(St(S), Amp_{M_j}(S))$ for all S. We note that if no recursive ij-bounding function exists, then for some sets S of clauses, method M_i will require much larger amplifications than method M_j. If such a recursive ij-bounding function exists, and especially if it is small, this means that method M_i never requires many more copies of clauses than method M_j. Either conclusion would be very interesting, we feel.

For some i and j, this question is easily settled. For example, the function $f(x, y) = y$ is an ii-bounding function for all i. It is also fairly straightforward to show that $f(x, y) = y$ is an $1i$-bounding function, since M_1 gives the smallest amplifications of any method. Since Brand's transformation simulates paramodulation, and path paramodulation uses an amplification whose number of clauses is bounded by the size of a paramodulation proof, it may be possible without much work to find a recursive 23-bounding function. We have worked some on the 31 and 21-bounding functions, but the 32-case is also unknown. Results of Voda [22] seem to suggest that there are no recursive $i1$-bounding functions for $i \neq 1$.

The systems M_2 and M_4 are closely related. We note that using proof trees, we can get an exponential relation between amplifications for path paramodulation and rigid clause paramodulation. Path paramodulation will never need more clauses than are needed for a (rigid) resolution-paramodulation proof tree. Also, any rigid resolution-paramodulation proof can be converted to a proof tree with at most an exponential increase in the number of inferences. Thus we obtain that there is a recursive 24-bounding function, in fact, there is an exponential 24-bounding function. We do not know if there is a recursive 42-bounding function.

We can also study this question for specialized S, for example, when S is a set of unit equations and unit inequations. In this case, M_5 is similar to rigid completion, that is, M_4. Rigid completion can be regarded as a sequence of ground completions, and in between there are critical substitutions (unifications) applied to S that eliminate a variable. The ground completions do not generate new instances of the input clauses, and the critical pair operations merely apply a substitution to the entire clause set. Therefore, we never increase the size of the amplification. It follows by theorem 3.20 that if $S\Theta$ is a set of equations and inequations that is ground and Eq-unsatisfiable, then M_4 and M_5 can generate a proof of unsatisfiability from S. Thus one only needs to consider amplifications having a number of clauses equal to the number of clauses in S. Thus $f_{41}(x, y) = y$ and $f_{51}(x, y) = y$ in this special case.

Despite a lot of work trying to resolve some of the remaining cases one way or the other, we have not been able to. We invite others to attempt the remaining cases. This is not only of interest theoretically, but has implications for the comparative efficiencies of various approaches to theorem proving with equality. One might think that the recent discovery that simultaneous rigid E-unification is undecidable would have implications for this question concerning the functions f_{ij}, but it does not seem to directly answer the questions. Another related question is how the existence of interpolation tests influences bounds on the sizes of the f_{ij}.

8 A New Undecidability Proof

A number of undecidability proofs for simultaneous rigid E-unification have recently been given, including [9]. We present another one, which we feel is interesting because it is based on Post's Correspondence Problem and also because it has a bearing on the existence of interpolation tests. In addition, this test has the feature that all of the (positive) equations used are ground equations. The construction can easily be modified (and possibly simplified) to give a direct reduction from Turing machines, as well. We consider sets of clauses that correspond to such hard instances of the simultaneous rigid E-unification problem, and find that for these clause sets, small Θ always exist when a larger amplification is taken.

8.1 Expressing Finite Automata and Regular Sets

We begin by giving some simple rigid E-unification problems to illustrate the techniques involved. For these constructions, we represent lists $[a, b, c, a]$ using "cons" by $cons(a, cons(b, cons(c, cons(a, NIL))))$, as is standard. Here NIL represents the empty list, and from now on we use $[]$ for this instead of NIL. In this section, we use $s, s_0, \ldots, s_n$, and $t, t_0, \ldots, t_n$ for unspecified ground terms or constant symbols. Consider this rigid E-unification problem:

$$cons(c, []) = [] \wedge cons(d, []) = [] \models x = []$$

The solutions are substitutions Θ binding x to lists of the constants c and d, that is, lists such as $cons(c, cons(d, cons(d, [])))$. We can also give a simultaneous rigid E-unification probem whose solutions Θ bind x to non-empty lists of the constants c and d. This consists of the above rigid E-unification problem together with the following:

$$cons(c, []) = s \wedge cons(d, []) = s \wedge cons(c, s) = s$$
$$\wedge cons(d, s) = s \models x = s$$

Definition 8.1 If X is a language over some finite alphabet T, where T is a finite set of ground terms, then $List_X$ is the set of lists $[a_1, a_2, \ldots, a_n]$ such that the string $a_1 a_2 \ldots a_n \in X$.

We recall also that T^* is the set of strings $a_1 a_2 \ldots a_n$ such that $a_i \in T$ for all i, and T^+ is the set of non-empty strings in T^*.

Theorem 8.2 *Suppose T is an arbitrary finite set of ground terms. Then there is a set eq^T of ground equations such that Θ is a solution to the rigid E-unification problem*

$$eq^T \models x = []$$

iff $x\Theta$ is in $List_{T^*}$.

Proof. Let $T = \{t_1, t_2, \ldots, t_n\}$, where the t_i are ground terms distinct from $[]$ and not containing the symbol "cons." Consider the following rigid E-unification problem:

$$cons(t_1, []) = [] \wedge cons(t_2, []) = [] \wedge \ldots$$
$$\wedge cons(t_n, []) = [] \models x = []$$

This problem then has solutions that bind x to lists of elements of T. □

Theorem 8.3 *Suppose T is an arbitrary finite set of ground terms. Then there is a constant symbol s (not in T) and a set $eq^{T,s}$ of ground equations such that a substitution Θ is a solution to the simultaneous rigid E-unification problem*

$$eq^T \models x = []$$
$$eq^{T,s} \models x = s$$

iff $x\Theta$ is in $List_{T^+}$.

Proof. Let $T = \{t_1, t_2, \ldots, t_n\}$ where the t_i are ground terms. Consider the following rigid E-unification problem $eq^{T,s} \models x = s$:

$$cons(t_1, []) = s \wedge cons(t_2, []) = s \wedge \ldots \wedge cons(t_n, []) = s \wedge$$
$$cons(t_1, s) = s \wedge cons(t_2, s) = s \wedge \ldots \wedge cons(t_n, s) = s \models x = s$$

This problem, together with $eq^T \models x = []$, has solutions that bind x to non-empty lists of elements of T. □

We can express relations that must hold between adjacent elements in a list. For example, to express that the constants c and d must alternate in a list, we have the simultaneous rigid E-unification problem

$$cons(c, []) = [] \wedge cons(d, []) = [] \models x = []$$
$$cons(c, []) = s_1 \wedge cons(d, []) = s_2 \wedge cons(c, s_2) = s_1$$
$$\wedge cons(d, s_1) = s_2 \wedge g(s_1) = c \wedge g(s_2) = c \models g(x) = c$$

We can think of s_1 and s_2 as states of a finite automaton traversing the list. In general, we can express regular sets in this way. In this example, both s_1 and s_2 would be accepting states. In general, since there may be more than one accepting state $s_1 \ldots s_n$, we need to express disjunctions of equalities such as $x = s_1 \vee \ldots \vee x = s_n$ where the s_i are constants (or terms). We can express that x is either s_1 or $\ldots$ or s_n by the rigid E-unification problem

$$g(s_1) = c \wedge \ldots \wedge g(s_n) = c \models g(x) = c$$

where g is a new function symbol.

Definition 8.4 If A is a finite automaton with input alphabet T, where T is a finite set of ground terms, then $\mathcal{L}(A)$ is the set of strings $a_1 \ldots a_n$ in T^* such that A accepts $a_1 \ldots a_n$.

Theorem 8.5 *Suppose that A is a deterministic finite automaton. Then there is a function symbol g, a constant symbol c, and a set $eq_{A,g,c}$ of ground equations such that Θ is a solution to the simultaneous rigid E-unification problem*

$$eq^T \models x = []$$

$$eq_{A,g,c} \models g(x) = c$$

iff $x\Theta$ is an element of $List_{\mathcal{L}(A)}$.

Proof. Suppose that A is a deterministic finite automaton with start state s_0, set of states $\{s_0, s_1 \ldots s_n\}$, input alphabet T, and accepting states $\{s_1 \ldots s_p\}$. Let $F_A(a, s)$ for s in $\{s_0, \ldots, s_n\}$ and a in T be the next-state function for A when reading a in state s. Consider the following simultaneous rigid E-unification problem:

$$eq^T \models x = []$$

$$[] = s_0 \wedge (\wedge_{s \in \{s_0 \ldots s_n\}, a \in T}(cons(a, s) = F_A(a, s)))$$

$$\wedge(\wedge_{s \in s_1 \ldots s_p} g(s) = c) \models g(x) = c$$

The solutions will bind x to lists $[a_1, a_2, \ldots, a_n]$ such that T accepts the string $a_1 a_2 \ldots a_n$. $\square$

We note that if A has only one accepting state, then there is a simultaneous problem of the form

$$eq^T \models x = []$$

$$eq_{A,c} \models x = c$$

that has solutions that bind x to elements of $List_{\mathcal{L}(A)}$.

8.2 Lists of Lists

For our construction, we will need two levels of lists, that is, simple lists and lists of lists. A list of lists is of the form $[list_1, list_2, \ldots, list_n]$ where each $list_i$ is a simple list, that is, a list of elements from an arbitrary finite alphabet T. Simple lists will be terminated by $[]$, but we will use $[]'$ to signify the end of a list of lists. Thus a list of lists is represented by a term of the form $cons(list_1, cons(list_2, \ldots, cons(list_n, []') \ldots))$. The reasons for this will become clearer later.

Definition 8.6 If X is a set of strings over the finite alphabet T, where T is a finite set of ground terms, then $LList_X$ is the set of lists of lists of the form $[list_1, list_2, \ldots, list_n]$ where each $list_i$ is in $List_X$.

We can give a rigid E-unification problem whose solutions are in $LList_{T^+}$, that is, the solutions bind x to lists of the form $[list_1, list_2, \ldots, list_n]$ where each $list_i$ is a non-empty list of elements of an arbitrary finite set T of ground terms. This can be done by the simultaneous rigid E-unification problem

$$eq^T \wedge cons([], []') = []' \models x = []'$$

$$eq^{T,s} \wedge cons(s, []') = []' \models x = []'$$

where eq^T and $eq^{T,s}$ are as defined above. We can also express membership in $LList_{\mathcal{L}(A)}$ for a finite automaton A.

Theorem 8.7 *Suppose A is a finite automata whose alphabet is a (finite) set T of ground terms and such that A does not accept the empty word. Then there is a set eq of ground equations and a constant symbol c' such that a substitution Θ is a solution to the simultaneous rigid E-unification problem*

$$eq^T \wedge cons([], []') = []' \models L = []'$$

$$eq \models L = c'$$

iff $L\Theta$ is a non-empty element of $LList_{\mathcal{L}(A)}$.

Proof. Let S_A be the states of A. Let S'_A be the accepting states of A. Suppose s_0 is the start state of A. Then we have the following simultaneous problem:

$$(\wedge_{a \in T}(cons(a, []) = []) \wedge cons([], []') = []' \models L = []'$$

$$[] = s_0 \wedge (\wedge_{s \in S_A, a \in T}(cons(a, s) = F_A(a, s))) \wedge$$
$$(\wedge_{s \in S'_A} cons(s, []') = c') \wedge (\wedge_{s \in S'_A} cons(s, c') = c') \models L = c'$$

The first problem constrains L to be a list of elements $list_i$ in which each $list_i$ is a list of elements of T. Consider a list of lists of the form $[list_1, list_2, list_3]$ where $list_i$ are in $List_{\mathcal{L}(A)}$. Then using the equations $[] = s_0 \wedge$ $(\wedge_{s \in S_A, a \in T}$ $(cons(a, s) = F_A(a, s)))$, we equate this list to $[s_1, s_2, s_3]$, where s_i are accepting states of S_A. Then using the equations $\wedge_{s \in S_A} cons(s, []') = c'$, this equals $cons(s_1, cons(s_2, c'))$. Finally, two applications of the equations $\wedge_{s \in S'_A} cons(s,\, c') = c'$ equate the entire list to c'. The same argument works in general for a list $[list_1, \ldots, list_n]$ in $LList_{\mathcal{L}(A_1)}$. Conversely, one shows that any solution to the above rigid E-unification problem must be of this form. □

We can express the fact that Z is the first element of a list (of lists) L by the equation $cons(Z, L_1) = L$. We can also express the fact that the last element of a list of lists L is in $List_{\mathcal{L}(A_2)}$ and the remaining elements of L are in $List_{\mathcal{L}(A_1)}$, where A_i are finite automata.

Definition 8.8 If X and Y are two sets of strings over finite alphabets of ground terms, then $LList_{X,Y}$ is the set of lists of the form $[list_1, list_2, \ldots, list_m]$ where $list_m \in List_Y$ and $list_i \in List_X$ for $1 \leq i < m$.

Theorem 8.9 *Suppose A_1 and A_2 are two finite automata whose alphabets are (finite) sets of ground terms and such that neither A_i accepts the empty word. Then there is a set eq of ground equations and a constant symbol c' such that a substitution Θ is a solution to the simultaneous rigid E-unification problem*

$$eq^T \wedge cons([], []') = []' \models L = []'$$

$$eq \models L = c'$$

iff $L\Theta$ is an element of $LList_{\mathcal{L}(A_1),\mathcal{L}(A_2)}$.

Proof. For this, we let A be a product automaton, that is, A simulates A_1 and A_2 simultaneously, and the states of A are pairs of states from A_1 and A_2. Let T be the union of the alphabets of A_1 and A_2 and let S_A be the states of A. Let $S_{A,1}$ be the states of A that correspond to acceptance by A_1, and let $S_{A,2}$ be the states of A that correspond to acceptance by A_2. That is, if we make $S_{A,1}$ the accepting states of A, A is equivalent to A_1, and if we make $S_{A,2}$ the accepting states of A, A

is equivalent to A_2. Suppose s_0 is the start state of A. Then we have the following simultaneous problem:

$$(\wedge_{a\in T}(cons(a,[]) = []) \wedge cons([],[]') = []' \models L = []'$$

$$[] = s_0 \wedge (\wedge_{s\in S_A, a\in T}(cons(a,s) = F_A(a,s)))\wedge$$
$$(\wedge_{s\in S_{A,2}} cons(s,[]') = c') \wedge \wedge_{s\in S_{A,1}} cons(s,c') = c') \models L = c'$$

The first problem constrains L to be a list of elements $list_i$ in which each $list_i$ is a list of elements of T. Consider a list of lists of the form $[list_1, list_2, list_3]$ where $list_1$ and $list_2$ are in $List_{\mathcal{L}(A_2)}$ and $list_3$ is in $List_{\mathcal{L}(A_1)}$. Then using the equations $[] = s_0 \wedge (\wedge_{s\in S_A, a\in T}\ (cons(a,s) = F_A(a,s)))$, we equate this list to $[s_1, s_2, s_3]$, where s_1 and s_2 are states of $S_{A,1}$ and s_3 is an accepting state of $S_{A,2}$. Then using the equations $\wedge_{s\in S_{A,2}} cons(s,[]') = c'$, this equals $cons(s_1, cons(s_2, c'))$. Finally, two applications of the equations $\wedge_{s\in S_{A,1}} cons(s,\ c') = c'$ equate the entire list to c'. The same argument works in general for a list $[list_1, \ldots, list_n]$ in $LList_{\mathcal{L}(A_1),\mathcal{L}(A_2)}$. Conversely, one shows that any solution to the above rigid E-unification problem must be of this form. □

We can also extend this to three automata, as follows:

Definition 8.10 If X, Y, and Z are three sets of strings over finite alphabets of ground terms, then $LList_{X,Y,Z}$ is the set of lists of the form $[list_1, list_2, \ldots, list_m]$ where $m \geq 2$, $list_m \in List_Z$, $list_i \in List_Y$ for $1 < i < m$, and $list_1 \in List_X$.

Theorem 8.11 *Suppose A_1, A_2, and A_3 are three finite automata whose alphabets are (finite) sets of ground terms and such that none of the A_i accepts the empty word. Then there is a set eq of ground equations and a constant symbol c' such that a substitution Θ is a solution to the simultaneous rigid E-unification problem*

$$eq^T \wedge cons([],[]') = []' \models L = []'$$

$$eq \models L = c'$$

iff $L\Theta$ is an element of $LList_{\mathcal{L}(A_1),\mathcal{L}(A_2),\mathcal{L}(A_3)}$.

Proof. A fairly straightforward extension of the construction in theorem 8.9. □

8.3 Pairing Lists

We also need a way to pair lists up to specify how a list can be modified by a sequence of operations. For this we need some definitions. Recall that $[t_1, \ldots, t_n]$ is an abbreviation for the term $cons(t_1, cons(t_2, \ldots, cons(t_n, []) \ldots))$.

Definition 8.12 Let $u_1 \ldots u_n$ be ground terms. Suppose that M is a list of the form $[f(t_1,u_1), \ldots, f(t_n,u_n)]$ and L_1 is the list $[t_1, \ldots, t_n]$. Then we call L_1 the *1-projection* of M. We call the list $L_2 = [u_1 \ldots u_n]$ the *2-projection* of M. We call M a *pairing* of L_1 and L_2. Thus $[a,b,a]$ is the 1-projection of $[f(a,c), f(b,d), f(a,d)]$ and $[c,d,d]$ is the 2-projection of this list. Also, the list $[f(a,c), f(b,d), f(a,d)]$ is a pairing of $[a,b,a]$ and $[c,d,d]$.

If T_1 and T_2 are arbitrary finite sets of ground terms and f is a binary function symbol, let $f(T_1,T_2)$ be the set of terms $f(x,y)$ for $x \in T_1$ and $y \in T_2$. We can express the fact that M is a list of elements of $f(T_1,T_2)$ and L is the 1-projection of M by the simultaneous problem

$$\begin{array}{rcl} eq^{T_1} & \models & L = [] \\ eq^{f(T_1,T_2)} & \models & M = [] \\ \wedge_{x\in T_1, y\in T_2} f(x,y) = x & \models & M = L \end{array}$$

We can express the fact that L is the 2-projection of M by the simultaneous problem

$$\begin{array}{rcl} eq^{T_1} & \models & L = [] \\ eq^{f(T_1,T_2)} & \models & M = [] \\ \wedge_{x\in T_1, y\in T_2} f(x,y) = y & \models & M = L \end{array}$$

Definition 8.13 We say that the list of lists Lp is a *shifted pairing* of the list of lists L if Lp is of the form $[M_1, M_2, \ldots, M_m]$ where the M_i are lists of elements of $f(T,T)$, the list of lists L is of the form $[list_1, \ldots, list_m]$, the lists $list_i$ are lists of elements of T, and M_1 is a pairing of $list_1$ and $list_2$, M_2 is a pairing of $list_2$ and $list_3$, $\ldots$, and M_m is a pairing of $list_m$ and some list of the form $[\#, \#, \ldots, \#]$.

As an example, the list of lists $Lp = [[f(a,b), f(b,c)], [f(b,a), f(c,d)], [f(a,\#), f(d,\#)]]$ is a shifted pairing of the list of lists $L = [[a,b],[b,c],[a,d]]$. Thus the elements of Lp are a pairing of $[a,b]$ and $[b,c]$, a pairing of $[b,c]$ and $[a,d]$, and a pairing of $[a,d]$ and $[\#,\#]$, respectively.

Theorem 8.14 *Suppose that T is a set of ground terms and $\#$ and is a constant symbol not in T. Then there is a simultaneous rigid E-unification problem such that Θ is a solution to the problem iff $L\Theta$ is a list of lists of elements of T and $Lp\Theta$ is a shifted pairing of $L\Theta$.*

Proof. Consider the following simultaneous problem:

$$\begin{array}{rcl} eq^{T} \wedge cons([],[]') = []' & \models & L = []' \\ eq^{f(T,T\cup\{\#\})} \wedge cons([],[]') = []' & \models & Lp = []' \\ \wedge_{x\in T, y\in T\cup\{\#\}} f(x,y) = x & \models & Lp = L \\ cons([\#],[]') = []' \wedge cons(\#, cons(\#,[])) = cons(\#,[]) & & \\ \wedge(\wedge_{x\in T, y\in T\cup\{\#\}} f(x,y) = y) & \models & L = cons(Z, Lp) \end{array}$$

The equation $cons([\#],[]') = []'$ means that any list of lists can be viewed as having an arbitrary number of lists of the form $[\#]$ on the end. Thus $[list_1, \ldots, list_m] = [list_1, \ldots, list_m, [\#], \ldots, [\#]]$. The equation $cons(\#, cons(\#,[])) = cons(\#,[])$ means that any one of these lists $[\#]$ can be expanded to include an arbitrary number of occurrences of $\#$, that is, can be expanded to a list of the form $[\#, \#, \ldots, \#]$. That is, $[\#] = [\#, \#, \ldots, \#]$.

The first (simple) rigid E-unification problem given expresses the fact that L is a list of lists of elements of T. The second problem expresses the fact that Lp is a list of lists of elements of $f(T, T\cup\{\#\})$. The third problem expresses the fact that elements of L are 1-projections of corresponding elements of Lp. The fourth problem expresses the fact that elements of L are 2-projections of preceding elements of Lp. Together this implies that Lp is a shifted pairing of L. We can also show that any solution of this simultaneous problem will be of this form. □

8.4 Expressing PCP as a Regular Set

With this machinery, we can express Post's Correspondence Problem (PCP). An instance of PCP is two sequences $\alpha_1 \ldots \alpha_n$ and $\beta_1 \ldots \beta_n$ of strings over a finite alphabet Σ. A solution to this instance of PCP is a sequence $i_1 \ldots i_k$ of integers such that $\alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k}$. It is known to be undecidable whether an instance of PCP has a solution.

We construct a regular set R from a PCP instance such that this PCP instance is solvable iff there is a string y in R having a certain property, which we will call *index consistency.* Using this regular set R, we express the Post Correspondence Problem as a simultaneous rigid E-unification problem that has a solution iff the instance of PCP is solvable. We represent a solution to this instance of PCP as a list of lists $L = [list_1, \ldots, list_m]$, where $list_m$ is a string in R and the index consistency of $list_m$ is checked using the other lists $list_i$. We represent a string as before by a list of constants, so the string $abca$ would be represented by the list $[a, b, c, a]$, that is, $cons(a, cons(b, cons(c, cons(a, []))))$.

Definition 8.15 Consider an instance of the PCP as given above. Let Γ_1 be the set of indices a'_j for $1 \le j \le n$, and let Γ_2 be the set of indices b'_j for $1 \le j \le n$. Let Γ be $\Gamma_1 \cup \Gamma_2$. Let α'_i be the regular set consisting of strings obtained from α_i by inserting arbitrary occurrences of the indices b'_j at arbitrary places. Thus if α_i is $c_1c_2\ldots c_m$, then α'_i is $\Gamma_2^* c_1 \Gamma_2^* c_2 \ldots \Gamma_2^* c_m \Gamma_2^*$. Let β'_i be the regular set consisting of strings obtained from the β_i by inserting arbitrary occurrences of the indices a'_j at arbitrary places. The regular set R is the intersection of the two regular sets $(\alpha'_1 a'_1 + \ldots + \alpha'_n a'_n)^* \Gamma_2^*$ and $(\beta'_1 b'_1 + \ldots + \beta'_n b'_n)^* \Gamma_1^*$.

Since the intersection of two regular sets is a regular set, R is a regular set. Note that R is a subset of $(\Sigma \cup \Gamma)^*$. Recall that a solution to this instance of PCP is a string that can be written either as $\alpha_{i_1} \ldots \alpha_{i_k}$ or as $\beta_{i_1} \ldots \beta_{i_k}$. The intuitive idea of R is to represent the solution as the string $\alpha_{i_1} \ldots \alpha_{i_k}$ with the new constants $a'_{i_1} \ldots a'_{i_k}$ inserted after the strings $\alpha_{i_1} \ldots \alpha_{i_k}$, respectively, and with the new constants $b'_{i_1} \ldots b'_{i_k}$ inserted after the strings $\beta_{i_1} \ldots \beta_{i_k}$, respectively.

Definition 8.16 The *α-index trace* of a string $c_1c_2\ldots c_n$ is the string $\phi(c_1)\phi(c_2)\ldots\phi(c_n)$, where ϕ maps the indices a'_j onto j and other elements of $\Sigma \cup \Gamma$ are mapped onto the empty string. Thus the α-index trace of a string is a list of integers. The *β-index trace* of a string $c_1c_2\ldots c_n$ is the string $\phi(c_1)\phi(c_2)\ldots\phi(c_n)$, where ϕ maps the indices b'_j onto j and other elements of $\Sigma \cup \Gamma$ are mapped onto the empty string. A string γ in $(\Sigma \cup \Gamma)^*$ is *index consistent* if its α-index trace is identical to its β-index trace.

Thus the string $a'_1 a'_2 c_3 b'_1 c_4 b'_2$ is index consistent, because its α-index trace is $1, 2$ and its β-index trace is also $1, 2$. However, $a'_1 a'_2 c_3 b'_1 c_4 b'_1$ is not index consistent, because its α-index trace is $1, 2$ and its β-index trace is $1, 1$.

Theorem 8.17 *The given instance of PCP has a solution iff there is a string y in R such that y is index consistent.*

Proof. If such an y exists, then we can determine the strings α_i of a PCP solution using the a-indices and we can determine the strings β_i of a PCP solution using the b-indices. Thus we can obtain a solution to the PCP instance. Conversely, if the PCP instance is solvable, we can take a solution $\alpha_{i_1} \ldots \alpha_{i_k} = \beta_{i_1} \ldots \beta_{i_k}$ and insert a and b-indices to obtain an element of R that is index consistent. □

8.5 Expressing PCP as a List of Lists

Let A_1 be a finite automaton accepting $(\Sigma \cup \Gamma)^*$. Since R is a regular set, it can be expressed as the set of strings accepted by a finite automaton A_2. Therefore, it is possible to express $List_{\mathcal{L}(A_2)}$ using a rigid E-unification problem as mentioned earlier. We can then express the fact that L is in $LList_{\mathcal{L}(A_1),\mathcal{L}(A_2)}$ by a simultaneous rigid E-unification problem, by theorem 8.9 above. What cannot easily be expressed is the fact that $list_m$ is index consistent. To express this, we need to constrain the other elements $list_1, \ldots, list_{m-1}$ of the list L in a manner which we now explain.

Definition 8.18 We say that a string γ_2 is a *legal transposition* of γ_1 if γ_2 is obtained from γ_1 by exchanging two adjacent symbols, at least one of which is an element of Γ. However, we do not permit two indices of the same type to be permuted. That is, we do not permute two elements of Γ_1 or two elements of Γ_2. Thus the string $c_1a'_1c_2b'_2$ is a legal transposition of the string $c_1c_2a'_1b'_2$, but the string $c_1a'_1a'_2c_2$ is not a legal transposition of the string $c_1a'_2a'_1c_2$, since that involves an exchange two indices of the same type (a-indices).

Note that if γ_2 is a legal transposition of γ_1, then γ_1 and γ_2 have the same α-index trace and the same β-index trace.

Definition 8.19 A string γ is a *simple correspondence* if every occurrence of a'_i is immediately followed by b'_i and every occurrence of b'_i is immediately preceded by a'_i.

Note that the set of simple correspondences is a regular set. Also, all simple correspondences are index consistent.

Definition 8.20 A *transposition history* is a list Lp of the form $[list_1, list_2, \ldots, list_m]$ where $list_m$ is an element of R, each $list_i$ is a legal transposition of $list_{i+1}$, and $list_1$ is a simple correspondence.

Note that if Lp is a transposition history, then all the lists $list_i$ have the same α-index trace and the same β-index trace.

Theorem 8.21 *The given PCP instance is solvable iff a legal transposition history Lp exists.*

Proof. If the PCP instance is solvable, then we can choose $list_m$ as an element of R that is index consistent. By a sequence of legal transpositions, we can obtain $list_1$ that is a simple correspondence. Conversely, if there is a transposition history, then $list_m$ is an element of R that is index consistent, hence the PCP instance is solvable. □

Theorem 8.22 *There is a finite automata A such that for any lists $list_1$ and $list_2$ of elements of $\Sigma \cup \Gamma$, lists $list_1$ and $list_2$ are legal transpositions of each other iff A accepts $list_3$, where $list_3$ is a pairing of $list_1$ and $list_2$.*

Proof. The list $list_3$ can only be of the form

$$[f(a_1, a_1), f(a_2, a_2), \ldots, f(a_i, a_{i+1}), f(a_{i+1}, a_i), \ldots, f(a_n, a_n)]$$

where a_i and a_{i+1} are elements of $\Sigma \cup \Gamma$ that are not both in Γ_1 or both in Γ_2. This can easily be checked by a finite automaton (with one accepting state). □

We recall here by theorem 8.14 above that there is a simultaneous rigid E-unification problem whose solutions are lists L and Lp such that Lp is a shifted pairing of L. In fact, we can find such a problem of the form

$$\begin{array}{lll} E_1 & \models & L = []' \\ E_2 & \models & Lp = []' \\ E_3 & \models & Lp = L \\ E_4 & \models & L = cons(Z, Lp), \end{array}$$

by theorem 8.14.

What remains is to find a way to constrain the lists $list_i$ to be obtained by permutations in this manner from $list_{i+1}$. For this we use the list pairing mechanism developed above.

Theorem 8.23 *There is a rigid E-unification problem whose solutions consist of the lists L and Lp such that L is a transposition history for the given PCP instance and Lp is a shifted pairing of L.*

Proof. We just showed (recalled) how to express the fact that Lp is a shifted pairing of L. What remains is to express the fact that adjacent elements of L are obtained by legal transpositions. By theorem 8.22, there is a finite automaton A to check if each element of Lp is a pairing of two lists that are legal transpositions of each other. We have to modify this automaton A to also accept lists of the form $[f(a_1, \#), f(a_2, \#), \ldots, f(a_n, \#)]$ for a_i in $\Sigma \cup \Gamma$, since the last element of Lp will be of this form. By theorem 8.7, there is a rigid E-unification problem

$$\begin{array}{lll} E_2 & \models & Lp = []' \\ E_5 & \models & Lp = c' \end{array}$$

expressing membership in $LList_{\mathcal{L}(A)}$, where E_2 is as above. Therefore we obtain a simultaneous problem of the form

$$\begin{array}{lll} E_1 & \models & L = []' \\ E_2 & \models & Lp = []' \\ E_3 & \models & Lp = L \\ E_4 & \models & L = cons(Z, Lp) \\ E_5 & \models & Lp = c' \end{array}$$

which expresses the fact that Lp is a shifted pairing of L and that adjacent elements of L are legal transpositions. The only parts that remain are to check that the first element of L is a simple correspondence and that the last element of L is an element of R. This is equivalent to checking that L is in $LList_{\mathcal{L}(A_1),\mathcal{L}(A_2),\mathcal{L}(A_3)}$ where A_i are finite automata and A_1 checks for simple correspondences, A_2 accepts anything, and A_3 checks for membership in R. By theorem 8.11, this can be checked by the problem $E_1 \models L = []'$ above together with a problem of the form

$$E_6 \models L = c''.$$

These six problems together then satisfy the conditions of the theorem. □

Corollary 8.24 *The given simultaneous rigid E-unification problem is solvable iff the PCP instance is solvable. Therefore, simultaneous rigid E-unification is undecidable.*

We note here that we could just as easily specify that $list_i$ of L be obtained from $list_{i+1}$ by one step of a Turing computation, since this can also be checked by a finite automaton, and thus obtain a simple reduction directly from Turing machines. This Turing construction might be interesting, because it might lead to a smaller undecidable subclass of simultaneous rigid E-unification.

8.6 The Final Simultaneous Problem

Putting this all together, we obtain a simultaneous problem of the following form:

$$
\begin{array}{llcll}
(\exists L, Z, Lp) & [E_1 & \models & L = []' & \text{(alphabet of } L\text{)} \\
 & \wedge E_2 & \models & Lp = []' & \text{(alphabet of } Lp\text{)} \\
 & \wedge E_3 & \models & Lp = L & \text{(first element of pairs in } Lp\text{)} \\
 & \wedge E_4 & \models & L = cons(Z, Lp) & \text{(second element of pairs in } Lp\text{)} \\
 & \wedge E_5 & \models & Lp = c' & \text{(check } Lp \text{ that successive elements of } L \\
 & & & & \text{are obtained by legal transpositions)} \\
 & \wedge E_6 & \models & L = c''] & \text{(first element of } L \text{ has indices matching,} \\
 & & & & \text{last element of } L \text{ is in } R\text{)}
\end{array}
$$

This corresponds to the following set of clauses:

$$
\begin{array}{c}
\neg p_1(L, Z, Lp) \vee L \neq []' \\
\neg p_1(L, Z, Lp) \vee E_1 \\
\neg p_2(L, Z, Lp) \vee Lp \neq []' \\
\neg p_2(L, Z, Lp) \vee E_2 \\
\neg p_3(L, Z, Lp) \vee Lp \neq L \\
\neg p_3(L, Z, Lp) \vee E_3 \\
\neg p_4(L, Z, Lp) \vee L \neq cons(Z, Lp) \\
\neg p_4(L, Z, Lp) \vee E_4 \\
\neg p_5(L, Z, Lp) \vee Lp \neq c' \\
\neg p_5(L, Z, Lp) \vee E_5 \\
\neg p_6(L, Z, Lp) \vee L \neq c'' \\
\neg p_6(L, Z, Lp) \vee E_6 \\
p_1(L, Z, Lp) \vee p_2(L, Z, Lp) \vee p_3(L, Z, Lp) \vee p_4(L, Z, Lp) \vee p_5(L, Z, Lp) \vee p_6(L, Z, Lp)
\end{array}
$$

We use $\neg p_1(L, Z, Lp) \vee E_1$ to abbreviate the set of clauses of the form $\neg p_1(L, Z, Lp) \vee e$ for e in E_1, and similarly for the other sets of equations. For a spanning set of these clauses, we can take the equational sets $(L \neq []', E_1), (Lp \neq []', E_2), \ldots, (L \neq c'', E_6)$ together with the pairs of literals $\{\neg p_i(L, Z, Lp),\ p_i(L,\ Z, Lp)\}$. This is a spanning set because if a path does not include any of the equational sets, then it must include all the $\neg p_i$ literals (at least one of each) and it also must contain some p_j literal from the bottom clause, leading to a contradiction. Unifying on the sets p_i and $\neg p_i$ binds all the variables, whence we obtain the simultaneous problem above.

This implies that the following problem is undecidable: Given a set S of clauses, is there a substitution Θ such that $S\Theta$ is Eq-unsatisfiable. For a proof, consider the above set of clauses. Then $S\Theta$ is Eq-unsatisfiable iff the PCP instance has a solution. We note that the equations appearing in each separate rigid unification problem are ground equations.

A problem is that there don't seem to be many ordered paramodulations possible here, since the only available function symbols to use for ordered paramodulation (except for ground equations) is "*cons*." It would be interesting to check this construction by deriving an ordered paramodulation refutation from the above clause set for a particular PCP instance; this would also help to check the completeness of ordered paramodulation.

We note that this set of clauses is splittable. This implies that there is an interpolation test of fairly low complexity which can substitute for simultaneous rigid E-unification in this case. Thus the relevance of the undecidability of simultaneous rigid E-unification to theorem proving is still unclear. However, some results of Voda [22] may be relevant here. He shows that by considering larger amplifications, one cannot always remove the undecidability property.

9 Conclusion

Despite the undecidability of simultaneous rigid E-unification, we have shown that in many common special cases of interest, decidable alternatives exist, and some of these have a complexity essentially the same as that of simple rigid E-unification. We have also presented some fully general alternatives that can be substituted for simultaneous rigid E-unification, and have analyzed their complexity to some extent. Finally, we presented a new proof of undecidability for the simultaneous problem. This proof is based on a reduction from the Post Correspondence Problem, and has the interesting feature that all of the positive equations used are ground equations in the instances constructed. This shows that simultaneous rigid E-unification is still undecidable when all the positive equations used are ground equations, a fairly restrictive special case. A number of open problems and directions for future research are also given. In general, we feel that the analysis of the complexity of theorem proving procedures is a challenging research area, besides giving insight into the behavior and utility of various approaches to theorem proving.

References

[1] P. B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28:193–214, 1981.

[2] Leo Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In Hassan Aït-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures 2: Rewriting Techniques*, pages 1–30, New York, 1989. Academic Press.

[3] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation. *Information and Computation*, 121(2):172–192, September 1995. To appear.

[4] Peter Baumgartner. An ordered theory resolution calculus. In A. Voronkov, editor, Logic Programming and Automated Reasoning (LPAR'92), pages 119–130, 1992. Volume 624 of *Lecture Notes in Computer Science*.

[5] B. Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, Automated Deduction — CADE-12. 12th International Conference on Automated Deduction., pages 678–692, Nancy, France, June/July 1994. Volume 814 of *Lecture Notes in Artificial Intelligence*.

[6] W. Bibel. *Automated Theorem Proving*. Vieweg, Braunschweig/Wiesbaden, 1987. second edition.

[7] D. Brand. Proving theorems with the modification method. *SIAM J. Comput.*, 4:412–430, 1975.

[8] A. Degtyarev and A. Voronkov. General connections via equality elimination. UPMAIL Technical Report 93, Uppsala University, Computing Science Department, January 1995.

[9] A. Degtyarev and A. Voronkov. Reduction of second-order unification to simultaneous rigid E-unification. UPMAIL Technical Report 109, Uppsala University, Computing Science Department, June 1995. To appear in Proc. CSL'95.

[10] A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, Computing Science Department, May 1995.

[11] M. Fitting. First-order modal tableaux. Journal of Automated Reasoning, 4:191–213, 1988.

[12] J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *J. ACM*, 39(2):377–429, 1992.

[13] Jean Goubault. A rule-based algorithm for rigid E-unification. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, Computational Logic and Proof Theory. Proceedings of the Third Kurt Gödel Colloquium, KGC'93, pages 202–210. Brno, August 1993. Volume 713 of *Lecture Notes in Computer Science*.

[14] Jean Goubault. Rigid $\vec{E}$-unifiability is DEXPTIME-complete. In *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science*, pages 498–506, Paris, France, July 1994.

[15] J. Hsiang and M Rusinowitch. Proving refutational completeness of theorem-proving strategies: the transfinite semantic tree method. *J. Assoc. Comput. Mach.*, 38(3):559–587, July 1991.

[16] D. Kozen. Positive first-order logic is NP-complete. *IBM Journal of Research and Development*, 25:4:327–332, 1981.

[17] U. Petermann. A complete connection calculus with rigid E-unification. In JELIA'94, pages 152–166, 1994. Volume 838 of *Lecture Notes in Computer Science*.

[18] D. Plaisted. Polynomial time termination and constraint satisfaction tests. In Claude Kirchner, editor, *Fifth International Conference on Rewriting Techniques and Applications*, pages 405–420, June 1993.

[19] D. Plaisted and Andrea Sattler-Klein. Proof lengths for equational completion. Technical Report SEKI Report SR-95-06, University of Kaiserslautern, Kaiserslautern, Germany, 1995.

[20] D. Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[21] W. Snyder. Efficient ground completion: an $O(nlogn)$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E. In *Proceedings of the 3rd International Conference on rewriting techniques and applications*, pages 419–433, 1989. Lecture Notes in Computer Science, Vol. 355.

[22] Voda. Personal communication from A. Voronkov, 1995.

INFORMATIK

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. Regina Kraemer
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `kraemer@mpi-sb.mpg.de`

MPI-I-95-2-009	L. Bachmair, H. Ganzinger	Ordered Chaining Calculi for First-Order Theories of Binary Relations
MPI-I-95-2-008	H. J. Ohlbach, R. A.Schmidt, U. Hustadt	Translating Graded Modalities into Predicate Logic
MPI-I-95-2-007	A. Nonnengart, A. Szalas	A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory
MPI-I-95-2-005	F. Baader, H. J. Ohlbach	A Multi-Dimensional Terminological Knowledge Representation Language
MPI-I-95-2-003	P. Barth	A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization
MPI-I-95-2-002	H. J. Ohlbach, R. A. Schmidt	Functional Translation and Second-Order Frame Properties of Modal Logics
MPI-I-95-2-001	S. Vorobyov	Proof normalization and subject reduction in extensions of Fsub
MPI-I-94-261	P. Barth, A. Bockmayr	Finite Domain and Cutting Plane Techniques in CLP($\mathcal{PB}$)
MPI-I-94-257	S. Vorobyov	Structural Decidable Extensions of Bounded Quantification
MPI-I-94-254	P. Madden	Report and abstract not published
MPI-I-94-252	P. Madden	A Survey of Program Transformation With Special Reference to *Unfold/Fold* Style Program Development
MPI-I-94-251	P. Graf	Substitution Tree Indexing
MPI-I-94-246	M. Hanus	On Extra Variables in (Equational) Logic Programming
MPI-I-94-241	J. Hopf	Genetic Algorithms within the Framework of Evolutionary Computation: Proceedings of the KI-94 Workshop
MPI-I-94-240	P. Madden	Recursive Program Optimization Through Inductive Synthesis Proof Transformation
MPI-I-94-239	P. Madden, I. Green	A General Technique for Automatically Optimizing Programs Through the Use of Proof Plans
MPI-I-94-238	P. Madden	Formal Methods for Automated Program Improvement
MPI-I-94-235	D. A. Plaisted	Ordered Semantic Hyper-Linking