What You Always Wanted to Know About Rigid E-Unification

Anatoli Degtyarev* and Andrei Voronkov**

Computing Science Department, Uppsala University
Box 311, S-751 05 Uppsala, Sweden

Abstract. This paper solves an open problem posed by a number of researchers: the construction of a complete calculus for matrix-based methods with rigid E-unification. The use of rigid E-unification and simultaneous rigid E-unification for such methods has been proposed by Gallier, Raatz and Snyder [35]. After our proof of the undecidability of simultaneous rigid E-unification [22] it has become clear that one should look for more refined techniques to deal with equality in matrix-based methods. In this article, we define a complete proof procedure for first-order logic with equality based on an incomplete but terminating procedure for rigid E-unification. Our approach is applicable to the connection method and the tableau method and illustrated on the tableau method.

1 Introduction

Algorithms for theorem-proving based on matings or tableaux in first-order logic without equality comprise two kinds of rules. Rules of the first kind construct matrices or tableaux from a given formula using a suitable amplification. Rules of the second kind try to close paths or branches using substitutions making the paths or branches inconsistent. These substitutions are unifiers of some atoms laying on a path or branch. Until recently, most approaches to introducing equality in such *matrix-based* methods tried to generalize such algorithms by a suitable modification of the notion of a unifier.

Such a modification using *simultaneous rigid E-unification* was introduced by Gallier, Raatz and Snyder [35] for the method of matings due to Andrews [1] or the connection method due to Bibel [12]. It can easily be represented in the tableau formalism. The method of matings interleaves two steps: amplification by quantifier duplication and search for mating for a given amplification. For formulas in disjunctive normal form this method was formulated earlier by Prawitz [56]. In this case amplification is represented by a matrix and mating is represented by a set of simultaneously satisfiable substitution conditions (mated pairs). Prawitz proposed a procedure for constructing substitution conditions one by one, closing the corresponding paths in the matrix through search with

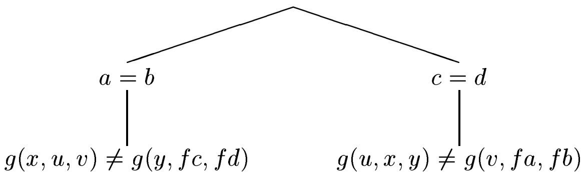
* Supported by a grant from the Swedish Royal Academy of Sciences.
** Supported by a TFR grant.

backtracking. Procedures of this kind were used in later formalizations and implementations of the tableau method, the method of matings or the connection method for formulas without equality. For example, Fitting [31] formulates this via an inference rule called MGU atomic closure rule.

Gallier, Raatz and Snyder [35] tried to describe a similar procedure for logic with equality. For example, Gallier et.al. [33] describe such a procedure in which a substitution condition is formalized via rigid E-unification, and the set of substitution conditions via simultaneous rigid E-unification.

Simultaneous rigid E-unification can be formulated as follows. Given equations $s_i = t_i$ and finite sets of equations E_i, $i \in \{1, \ldots, n\}$, find a substitution σ such that $\vdash (\bigwedge_{e \in E_i} e\sigma) \supset s_i\sigma = t_i\sigma$, for all i (here $\vdash$ means provability in first-order logic with equality). The corresponding instance of the simultaneous rigid E-unification problem is denoted by the system of *rigid equations* $E_i \vdash_\forall s_i = t_i$.

Example 1 In this and further examples we shall often omit parentheses in terms with unary function symbols, for example we write ffb instead of $f(f(b))$. Assume that we want to prove the formula $\exists xyzu((a = b \supset g(x, u, v) = g(y, fc, fd)) \wedge (c = d \supset g(u, x, y) = g(v, fa, fb)))$. After several applications of tableau expansion rules (α-, β- and γ-rules in the terminology of Smullyan [60] or Fitting [31]) we obtain the following tableau (we only consider the part of the tableau containing literals, omitting non-literal formulas):



Collecting formulas lying on the two branches in this tableau we obtain the following two *rigid equations* expressing inconsistency of this tableau:

$$\begin{array}{l} a = b \vdash_\forall g(x, u, v) = g(y, fc, fd) \\ c = d \vdash_\forall g(u, x, y) = g(v, fa, fb) \end{array}$$

This system of rigid equations has one solution $\{fa/x, fb/y, fc/u, fd/v\}$. This substitution can be found by applying the functional reflexivity rule and MGU replacement rule of [31] (in fact, the reformulation of the paramodulation rule for tableaux). Obviously, this tableau cannot be closed without the use of the functional reflexivity.[3]

[3] The system of clauses corresponding to this example improves a result proved in Plaisted [55] by using a more complicated example. In this system of clauses

$$\begin{array}{l} a = b \vee c = d \\ a = b \vee g(u, x, y) \neq g(v, fa, fb) \\ c = d \vee g(x, u, v) \neq g(y, fc, fd) \\ g(x, u, v) \neq g(y, fc, fd) \vee g(u, x, y) \neq g(v, fa, fb) \end{array}$$

Since the invention of simultaneous rigid E-unification by Gallier, Raatz and Snyder [35], there were a number of publications on simultaneous rigid E-unification itself and its use in theorem proving, for example Gallier et.al. [34, 36, 32, 33], Baumgartner [5], Beckert and Hähnle [11], Becher and Petermann [8], Beckert [9], Goubault [38] and Petermann [54]. Some of these articles were based on the conjecture that simultaneous rigid E-unification is decidable. There were several faulty proofs of the decidability of this problem (e.g. [34, 32, 38]).

The refutation procedure for first-order logic with equality using simultaneous rigid E-unification (e.g. Gallier et.al. [33]) was based on a faulty assumption that solutions to simultaneous rigid E-unification can be found by consecutive combination of finite *complete* sets of solutions for (non-simultaneous) rigid E-unification [32, 34]. Later we [21, 22] proved that simultaneous rigid E-unification is undecidable, which implied that Gallier et.al.'s procedure cannot, in general, find solutions to simultaneous rigid E-unification. However, it is not clear whether this implies incompleteness of this procedure for first-order logic with equality: there are examples when their procedure cannot find a solution for a given amplification in spite that such a solution exists, but can find a solution for a bigger amplification. Completeness of Gallier et.al.'s procedure or existence of a procedure complete for first-order logic with equality based on some set of solutions to rigid E-unification was an open problem (see e.g. Petermann [54] and Beckert [10]). Our paper gives a positive solution to this problem.

An advantage of Gallier et.al.'s procedure is that it allows one to extend the proof-search technology developed for tableaux without equality to the case with equality, using solutions to rigid E-unification instead of most general unifiers. In particular, for a given amplification Gallier et.al.'s procedure always terminates. A procedure of this kind is used in the theorem prover ${}_3T^AP$ [39] (R.Hähnle, private communication).

In this paper we define a procedure extending the tableau method to logic with equality based on an incomplete procedure for rigid E-unification. Nevertheless, our procedure is complete for first-order logic with equality. Hence, we rehabilitate Gallier et.al.'s program for adding equality to semantic tableaux. Moreover, our procedure solves rigid equations laying on different tableau branches independently. This strongly improves Gallier et.al.'s procedure which uses solutions of some rigid equations to solve rigid equations on other branches.

A similar approach has already been defined by Kanger [40] based on a more straightforward way of variable instantiation. As a method for finding a closing substitution, Kanger proposed an algorithm which can now be characterized as an incomplete (but terminating) algorithm for simultaneous rigid E-unifiability. Variables in a matrix (or a tableau) could be consecutively substituted by ground terms already occurring in the matrix. This procedure does not solve simultaneous rigid E-unifiability, but it gives a procedure complete for first-order logic

there is no refutation even by *unrestricted* rigid paramodulation (i.e. using non-ordered rigid paramodulation and paramodulation into variables), while Plaisted [55] gives an example showing incompleteness of *ordered* rigid paramodulation only.

with equality. In the terminology of Fitting [31] it means that a closing substitution can be found after a sufficiently high (but not necessarily minimal) number of applications of the γ-rule. The approach to substitutions based on this idea has been characterized as minus-normalization in Matulis [47] and Maslov [46].

However, for a language with function symbols minus-normalization is interesting mostly theoretically. Even in simplest cases, minus-normalization requires a huge number of instantiations. For example, in the tableau of Example 1, we have to consider 8^4 possible instantiations of variables x, y, u, v by terms in the set $\{a, b, c, d, fa, fb, fc, fd\}$. Moreover, it was proved that the use of minus-normalization can lead to considerable growth of derivations. Some results on minus-normalization are proved by Norgela [53].

In this paper we describe a logical calculus $\mathcal{BPE}$ for rigid E-unification based on the *rigid basic superposition* rule that is an adaptation of basic superposition of Degtyarev [16, 17], Bachmair et.al. [3], Nieuwenhuis and Rubio [50], for "rigid" variables. For a given rigid E-unification problem (called *rigid equation* in this paper), there is only a finite number of $\mathcal{BPE}$-derivations for this problem. Thus, $\mathcal{BPE}$ gives us an algorithm returning a finite set of solutions to this rigid equation. We use these solutions to close a tableau branch in the same way as most general unifiers are used to close a branch in the MGU atomic closure rule of Fitting [31].

We only give sketches of proofs due to the lack of space. Complete proofs will be included in our technical report [27].

2 Preliminaries

We present here a brief overview of notions and preliminary definitions necessary for understanding the paper. We assume basic knowledge of substitutions and unification.

Let Σ be a signature, and X be a set of variables. $T(\Sigma, X)$ denotes the set of all terms in the signature Σ with variables from X. The set of all *ground* terms in the signature Σ is denoted by $T(\Sigma)$.

A *literal* is either an atomic formula or a negation of an atomic formula. An *equation* is a literal $s = t$, where $s, t \in T(\Sigma, X)$. *We do not distinguish equations* $s = t$ *and* $t = s$. Literals of the form $\neg(s = t)$ are denoted by $s \neq t$ and called *disequations*. For simplicity, we assume that $=$ is the only predicate symbol of our first-order language. As usual, arbitrary first-order languages can be represented in such language by introducing a sort `bool` and replacing any non-equational atom A by $A = \texttt{true}$ (for details see e.g. [4]).

By a *ground expression* (i.e. term or literal) we mean an expression containing no variables. For any expression E, $var(E)$ denotes the set of all variables occurring in E. We write $A[s]$ to indicate that an expression A contains s as a subexpression and denote by $A[t]$ the result of replacing this occurrence of s in A by t. By $A\sigma$ we denote the result of applying the substitution σ to A. A substitution θ with the domain $x_1, \ldots, x_n$ is denoted by $\{x_1\theta/x_1, \ldots, x_n\theta/x_n\}$.

A substitution σ is called *grounding for a set of variables* V iff for every variable $v \in V$ the term $v\sigma$ is ground.

From now on we assume that $\succ$ *is a lexicographic path ordering on* $T(\Sigma, X)$ *extending a total precedence relation on* Σ *[52].* Nevertheless, our results are valid for *any* reduction ordering total on the set of ground terms $T(\Sigma)$.

A *tableau* is a tree whose nodes are formulas. We shall refer to the standard free variable tableau calculus of Fitting [31].

A formula is in the *Skolem negation normal form* iff it is constructed from literals using the connectives $\wedge, \vee$ and the quantifier $\exists$. There is a provability-preserving structure-preserving translation of formulas without equivalences into formulas in Skolem negation normal consisting of the standard skolemization and a translation into negation normal form used e.g. in Andrews [1]. In this article, we consider the provability problem only for such formulas.

For an equation $s = t$ and a set of equations E we write $E \vdash s = t$ to denote that the formula $(\bigwedge_{e \in E} e) \supset s = t$ is provable in first-order logic with equality. For such formulas provability can be tested by the congruence closure algorithm [59].

Definition 1 A *rigid equation* is an expression of the form $E \vdash_{\forall} s = t$, where E is a finite set of equations and s, t are terms. Its *solution* is any substitution θ such that $E\theta \vdash s\theta = t\theta$.[4].

3 Rigid basic superposition

The term "rigid paramodulation" has already been used by Becher and Petermann [8] and Plaisted [55] for systems of inference rules in which all variables are treated as "rigid". For example, rigid clause paramodulation of Plaisted [55] is essentially paramodulation and resolution over a set of clauses, where all substitutions are applied to the whole set of clauses. A similar system for resolution has been proposed earlier by Chang [15] as V-resolution and by Chang and Lee [42] for resolution with paramodulation as V-resolution and V-paramodulation. We shall use the term "rigid basic superposition" to denote a "rigid" version of basic superposition. We formalize rigid basic superposition using constraints that is close to the presentation of Nieuwenhuis and Rubio [52].

Definition 2 (Constraints) By an (ordering) *constraint* we mean a set of expressions which can be of two kinds: an *equality constraint* $s = t$, or an *inequality constraint* $s \succ t$, where s, t are terms. A substitution θ is a *solution* to a constraint $s = t$ (respectively, a constraint $s \succ t$) iff θ is grounding for $var(s) \cup var(t)$ and $s\theta$ coincides with $t\theta$ (respectively, $s\theta \succ t\theta$).

A substitution θ is a solution to a constraint $\mathcal{C}$ iff θ is a solution to every equality or inequality constraint in $\mathcal{C}$. A constraint $\mathcal{C}$ is *satisfiable* iff it has a solution. Constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ are called *equivalent* iff they have the same sets of solutions.

[4] The term "rigid equation" could be more adequately expressed as "instance of a (non-simultaneous) rigid E-unification problem", but this would be too lengthy.

There are efficient methods for solving ordering constraints for lexicographic path orderings given e.g. by Nieuwenhuis [51] and Nieuwenhuis and Rubio [52].

Below we shall introduce a system $\mathcal{BPE}$ for solving rigid equations. The provable objects of $\mathcal{BPE}$ are *constrained rigid equations:*

Definition 3 (Constrained rigid equation) A *constrained rigid equation* is a pair consisting of a rigid equation R and a constraint $\mathcal{C}$. Such a constrained rigid equation will be denoted $R \cdot \mathcal{C}$.

Definition 4 (Calculus $\mathcal{BPE}$) The *calculus* $\mathcal{BPE}$ of constrained rigid equations consists of the following inference rules:

Left rigid basic superposition:

$$\frac{E \cup \{l = r, s[p] = t\} \vdash_\forall e \cdot \mathcal{C}}{E \cup \{l = r, s[p] = t, s[r] = t\} \vdash_\forall e \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}}\ (lrbs)$$

Right rigid basic superposition:

$$\frac{E \cup \{l = r\} \vdash_\forall s[p] = t \cdot \mathcal{C}}{E \cup \{l = r\} \vdash_\forall s[r] = t \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}}\ (rrbs)$$

Equality resolution:

$$\frac{E \vdash_\forall s = t \cdot \mathcal{C}}{\vdash_\forall s = s \cdot \mathcal{C} \cup \{s = t\}}\ (er)$$

Application of all the rules is restricted to the following conditions:

1. The constraint at the conclusion of the rule is satisfiable;
2. In the basic superposition rules, the rigid equation in the conclusion of any inference rule is different from the rigid equation in the premise of this rule;
3. In the basic superposition rules, the term p is not a variable.

The basic restriction in $\mathcal{BPE}$ is formalized by representing most general unifiers through equality constraints. Condition 1 has two purposes. The satisfiability of equations in constraints is needed to preserve correctness of the method. The satisfiability of inequality constraints is needed to ensure termination (Theorem 7 below). Condition 2 is also needed for termination. In fact, it prohibits repeated applications of the same rules. Condition 3 is not essential, it is added as a standard optimization used in paramodulation-based methods.

We denote by $R \cdot \mathcal{C} \rightsquigarrow R' \cdot \mathcal{C}'$ the fact that $R' \cdot \mathcal{C}'$ is obtained from $R \cdot \mathcal{C}$ by an application of one of the inference rules of $\mathcal{BPE}$. The symbol $\rightsquigarrow^*$ denotes the reflexive and transitive closure of $\rightsquigarrow$.

Example 2 Consider the rigid equation $hx = a, hb = fy \vdash_\forall y = gfy$. The ordering $\succ$ is based on the precedence relation $f > h > b > a$. Under this ordering we have $hb \succ a$ and $fy \succ hb$. The following is a $\mathcal{BPE}$-derivation for this rigid equation:

$$\dfrac{\dfrac{\dfrac{hx = a, hb = fy \vdash_\forall y = gfy \cdot \emptyset}{hx = a, hb = fy \vdash_\forall y = ghb \cdot \{fy \succ hb, gfy \succ y, fy = fy\}}\,(rrbs)}{\begin{array}{l} hx = a, hb = fy \vdash_\forall y = ga \\ \quad \cdot\{fy \succ hb, gfy \succ y, fy = fy, hx \succ a, ghb \succ y, hx = hb\}\end{array}}\,(rrbs)}{\vdash_\forall y = y \cdot \{fy \succ hb, gfy \succ y, fy = fy, hx \succ a, ghb \succ y, hx = hb, y = ga\}}\,(er)$$

By using *constraint simplification,* i.e. replacement of constraints by equivalent "more simple" constraints we can rewrite this derivation as

$$\dfrac{\dfrac{\dfrac{hx = a, hb = fy \vdash_\forall y = gfy \cdot \emptyset}{hx = a, hb = fy \vdash_\forall y = ghb \cdot \emptyset}\,(rrbs)}{hx = a, hb = fy \vdash_\forall y = ga \cdot \{ghb \succ y, x = b\}}\,(rrbs)}{\vdash_\forall y = y \cdot \{x = b, y = ga\}}\,(er)$$

Theorem 5 (Soundness of $\mathcal{BPE}$) *Let $R \cdot \emptyset \rightsquigarrow^* E \vdash_\forall t = t \cdot \mathcal{C}$. Then any substitution satisfying $\mathcal{C}$ is a solution to R. In particular, R is solvable.*

Proof. (Sketch) For any constraint $\mathcal{C}$, denote by $\mathcal{C}^=$ the constraint obtained from $\mathcal{C}$ be removing all inequality constraints. First we prove that for every application of an inference rule of $\mathcal{BPE}$ of the form $E_1 \vdash_\forall e_1 \cdot \mathcal{C}_1 \rightsquigarrow E_2 \vdash_\forall e_2 \cdot \mathcal{C}_2$ we have $E_1, \neg e_1, \mathcal{C}_2^= \vdash E_2, \neg e_2, \mathcal{C}_1^=$. By induction on the number of inference steps, we prove the same statement for multi-step derivations $E_1 \vdash_\forall e_1 \cdot \mathcal{C}_1 \rightsquigarrow^* E_2 \vdash_\forall e_2 \cdot \mathcal{C}_2$.

Let R have the form $E_0 \vdash_\forall r = s$. Applying the obtained statement to multi-step derivations, we get $E_0, r \neq s, \mathcal{C}^= \vdash E, t \neq t$. Hence, $E_0, \mathcal{C}^= \vdash r = s$. Let θ be any solution to $\mathcal{C}$. We have $E_0\theta, \mathcal{C}^=\theta \vdash r\theta = s\theta$. Any constraint in $\mathcal{C}^=\theta$ has the form $u = u$. Hence, $E_0\theta \vdash r\theta = s\theta$, i.e. θ is a solution to $E_0 \vdash_\forall r = s$. □

This theorem leads to the following definition:

Definition 6 (Answer constraint) A constraint $\mathcal{C}$ is called an *answer constraint for a rigid equation* R iff for some rigid equation $E \vdash_\forall t = t$ we have $R \cdot \emptyset \rightsquigarrow^* E \vdash_\forall t = t \cdot \mathcal{C}$.

We note that $\mathcal{BPE}$ is an incomplete calculus for solving rigid equations. It means that there are solvable rigid equations R that have no answer constraint. For instance, consider the rigid equation[5] $x = a \vdash_\forall gx = x$. It has one solution $\{ga/x\}$. However, the only derivation in $\mathcal{BPE}$ applicable to $x = a \vdash_\forall gx = x \cdot \emptyset$ is[6]

[5] Suggested by G.Becher (private communication).

[6] We can easily get rid of such rule applications resulting in an equation of the form $t = t$ by putting additional restrictions on rigid basic superposition, but such *optimizations* are not our aim.

$$\frac{x = a \vdash_\forall gx = x \cdot \emptyset}{x = a, x = x \vdash_\forall gx = x \cdot \{a \succ x, a = a\}}\ (lrbs)$$

Theorem 7 (Termination of $\mathcal{BPE}$) *For any constrained rigid equation $R \cdot \mathcal{C}$, there exists a finite number of derivations from $R \cdot \mathcal{C}$.*

Proof. (Sketch)
The main idea is the following. We note that if there is an infinite derivation, we have a solvable infinite set of constraints containing $\{t_0 \succ t_1, t_1 \succ t_2, \ldots\}$, such that $t_i \in T(\Sigma, X)$ for finite Σ and X. This contradicts to Kruskal's theorem [41]. □

Inequality constraints are not needed for soundness or completeness of our method. The pragmatics behind inequality constraints is to ensure that the search for solutions of a rigid equation is finite. In addition, the use of ordering constraints prunes the search space.

To illustrate this theorem, we consider Example 2. The rigid equation of this example has an infinite number of solutions including $\{a/x, gh^n a/y\}$, for every natural number n. However, all possible $\mathcal{BPE}$-derivations starting with $hx = a, hb = fy \vdash_\forall y = gfy \cdot \emptyset$ give only two answer constraints, one is

$$\{fy \succ hb, gfy \succ y, fy = fy, hx \succ a, ghb \succ y, hx = hb, y = ga\}$$

shown in Example 2, another is $\{fy \succ hb, gfy \succ y, fy = fy, y = ghb\}$ obtained from the following derivation:

$$\cfrac{\cfrac{hx = a, hb = fy \vdash_\forall y = gfy \cdot \emptyset}{hx = a, hb = fy \vdash_\forall y = ghb \cdot \{fy \succ hb, gfy \succ y, fy = fy\}}\ (rrbs)}{\vdash_\forall y = y \cdot \{fy \succ hb, gfy \succ y, fy = fy, y = ghb\}}\ (er)$$

This answer constraint can be simplified to $\{y = ghb\}$. There are some other derivations with the same answer constraints, but only a finite number.

Theorem 7 yields

Theorem 8 *Any rigid equation has a finite number of answer constraints. There is an algorithm giving by any rigid equation R the set of all answer constraints for R.*

4 Answer constraints and the tableau method

In this section we consider how to use the system $\mathcal{BPE}$ for theorem proving by the tableau method. Since we only consider skolemized formulas, we have no δ-rules in tableau calculi.

Definition 9 Let S be a branch of a tableau, considered as a set of formulas. The set of *rigid equations on* S is defined in the following way. A rigid equation $E \vdash_{\forall} s = t$ is on S iff E is the set of all equations in S and $(s \neq t) \in S$.

This definition is equivalent to the one given by Beckert and Häehnle [11].

We extend the notion of answer constraints to tableau branches:

Definition 10 A constraint $\mathcal{C}$ is an *answer constraint for a tableau branch* S iff $\mathcal{C}$ is an answer constraint for some rigid equation on S.

By Theorem 8, we obtain

Theorem 11 *Any tableau branch has a finite number of answer constraints. There is an algorithm giving by any tableau branch S the set of all answer constraints for S.*

The following theorem states soundness and completeness of the tableau method with answer constraints:

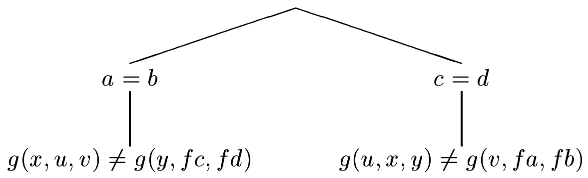
Theorem 12 (Soundness and completeness) *Let φ be a sentence. Then φ is provable in first-order logic with equality iff there is a tableau T obtained from $\neg\varphi$ by tableau expansion rules with the following property. Let $S_1, \ldots, S_n$ be all branches of T. Then there exist answer constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ for $S_1, \ldots, S_n$, respectively, such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$ is satisfiable.*

Proof. (Sketch) Soundness follows from soundness of $\mathcal{BPE}$.

The proof of completeness is quite lengthy, and we can only give its very general explanation. The proof consists of two stages. On the first stage, for a set of clauses S_φ representing the formula $\neg\varphi$ we construct a tree-like refutation of a special form using a modification of the basic superposition calculus of Bachmair et.al. ([3], calculus $\mathcal{S}$). The main feature of our modification is *regularity:* above superpositions from non-boolean equations there can only be superpositions from non-boolean equations (we also ignore maximality conditions of calculus $\mathcal{S}$ for equalities involved in superpositions). The existence of such a refutation is based on techniques and results of Degtyarev [17], Bachmair et.al. [4], Nieuwenhuis and Rubio [52] and some original techniques.

The second stage of the proof uses the techniques of *equality elimination* [19, 20] to transform this refutation into the required tableau. $\Box$

To illustrate this theorem, consider the formula of Example 1. Assume that we want to prove the formula $\exists xyzu((a = b \supset g(x,u,v) = g(y,fc,fd)) \wedge (c = d \supset g(u,x,y) = g(v,fa,fb)))$. The ordering $\succ$ is based on the precedence $g > f > a > b > c > d$. After one quantifier duplication (application of a γ-rule) and some other tableau expansion rules we obtain the following tableau:



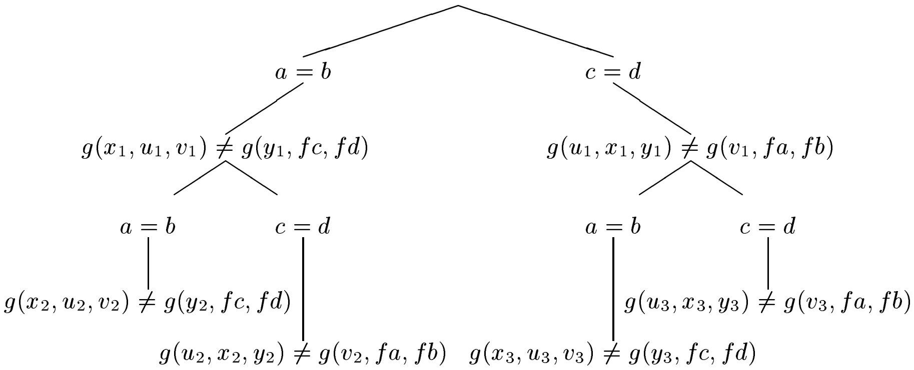
There is one rigid equation on each branch of the tableau:

$$a = b \vdash_{\forall} g(x, u, v) = g(y, fc, fd) \tag{1}$$
$$c = d \vdash_{\forall} g(u, x, y) = g(v, fa, fb) \tag{2}$$

Rigid basic superposition is applicable to none of this rigid equations. Rigid equation (1) has one answer constraint $\{g(x, u, v) = g(y, fc, fd)\}$ obtained by an application of the equality resolution rule:

$$\frac{a = b \vdash_{\forall} g(x, u, v) = g(y, fc, fd) \cdot \emptyset}{\vdash_{\forall} g(x, u, v) = g(x, u, v) \cdot \{g(x, u, v) = g(y, fc, fd)\}}\ (er)$$

Similarly, rigid equation (2) has one answer constraint $\{g(u, x, y) = g(v, fa, fb)\}$. The union of these constraints $\{g(x, u, v) = g(y, fc, fd), g(u, x, y) = g(v, fa, fb)\}$ is unsatisfiable. Thus, our method does not find solution after one quantifier duplication. After three quantifier duplications and some other tableau expansion steps we obtain the following tableau:



It has four branches:

$$\begin{array}{ll}
S_1: & \{a = b, g(x_1, u_1, v_1) \neq g(y_1, fc, fd), g(x_2, u_2, v_2) \neq g(y_2, fc, fd)\} \\
S_2: & \{a = b, c = d, g(x_1, u_1, v_1) \neq g(y_1, fc, fd), g(u_2, x_2, y_2) \neq g(v_2, fa, fb)\} \\
S_3: & \{a = b, c = d, g(u_1, x_1, y_1) \neq g(v_1, fa, fb), g(x_3, u_3, v_3) \neq g(y_3, fc, fd)\} \\
S_4: & \{c = d, g(u_1, x_1, y_1) \neq g(v_1, fa, fb), g(u_3, x_3, y_3) \neq g(v_3, fa, fb)\}
\end{array}$$

Consider the following rigid equations R_1–R_4 on the branches S_1–S_4, respectively:

$$\begin{array}{ll}
R_1: & a = b \vdash_{\forall} g(x_2, u_2, v_2) = g(y_2, fc, fd) \\
R_2: & a = b, c = d \vdash_{\forall} g(x_1, u_1, v_1) = g(y_1, fc, fd) \\
R_3: & a = b, c = d \vdash_{\forall} g(u_1, x_1, y_1) = g(v_1, fa, fb) \\
R_4: & c = d \vdash_{\forall} g(u_3, x_3, y_3) = g(v_3, fa, fb)
\end{array}$$

We can apply the following $\mathcal{BPE}$-derivations to R_1–R_4:

$$\dfrac{a=b \vdash_\forall g(x_2,u_2,v_2)=g(y_2,fc,fd)\cdot\emptyset}{\vdash_\forall g(x_2,u_2,v_2)=g(x_2,u_2,v_2)\cdot\{g(x_2,u_2,v_2)=g(y_2,fc,fd)\}}\ (er)$$

$$\dfrac{\dfrac{a=b,c=d\vdash_\forall g(x_1,u_1,v_1)=g(y_1,fc,fd)\cdot\emptyset}{\begin{array}{l}a=b,c=d\vdash_\forall g(x_1,u_1,v_1)=g(y_1,fd,fd)\\ \quad\cdot\{c\succ d, g(y_1,fc,fd)\succ g(x_1,u_1,v_1), c=c\}\end{array}}\ (rrbs)}{\begin{array}{l}\vdash_\forall g(x_1,u_1,v_1)=g(x_1,u_1,v_1)\\ \quad\cdot\{c\succ d, g(y_1,fc,fd)\succ g(x_1,u_1,v_1), c=c, g(x_1,u_1,v_1)=g(y_1,fd,fd)\}\end{array}}\ (er)$$

$$\dfrac{\dfrac{a=b,c=d\vdash_\forall g(u_1,x_1,y_1)=g(v_1,fa,fb)\cdot\emptyset}{\begin{array}{l}a=b,c=d\vdash_\forall g(u_1,x_1,y_1)=g(v_1,fb,fb)\\ \quad\cdot\{a\succ b, g(v_1,fa,fb)\succ g(u_1,x_1,y_1), a=a\}\end{array}}\ (rrbs)}{\begin{array}{l}\vdash_\forall g(u_1,x_1,y_1)=g(u_1,x_1,y_1)\\ \quad\cdot\{a\succ b, g(v_1,fa,fb)\succ g(u_1,x_1,y_1), a=a, g(u_1,x_1,y_1)=g(v_1,fb,fb)\}\end{array}}\ (er)$$

$$\dfrac{c=d \vdash_\forall g(u_3,x_3,y_3)=g(v_3,fa,fb)\cdot\emptyset}{\vdash_\forall g(u_3,x_3,y_3)=g(u_3,x_3,y_3)\cdot\{g(u_3,x_3,y_3)=g(v_3,fa,fb)\}}\ (er)$$

The union of the answer constraints of these derivations is

$$\begin{array}{l}\{g(x_2,u_2,v_2)=g(y_2,fc,fd),\\ \ c\succ d, g(y_1,fc,fd)\succ g(x_1,u_1,v_1), c=c, g(x_1,u_1,v_1)=g(y_1,fd,fd),\\ \ a\succ b, a=a, g(v_1,fa,fb)\succ g(u_1,x_1,y_1), g(u_1,x_1,y_1)=g(v_1,fb,fb),\\ \ g(u_3,x_3,y_3)=g(v_3,fa,fb)\ \}\end{array}$$

This constraint is satisfiable. To check this, we can consider the following substitution:

$$\{fb/x_1, fb/y_1, fd/u_1, fd/v_1, b/x_2, b/y_2, fc/u_2, fd/v_2, d/u_3, d/v_3, fa/x_3, fb/y_3\}.$$

5 Tableau basic superposition

As a simple consequence of our results, we prove a completeness result for a paramodulation rule working on tableaux. A paramodulation rule working directly on tableaux was proposed by Loveland [44] in the context of model elimination and later by Fitting [31]. However, their formulations have all disadvantages of the early paramodulation rule of Robinson and Wos [57]:

1. Functional reflexivity rule is used;
2. Paramodulation into variables is allowed;

3. Increasing applications of paramodulation are allowed (for example, x can be rewritten to $f(x)$.

As a consequence, for a given tableau expansion there may be an infinite sequence of paramodulations, either due to the use of functional reflexivity or due to the use of increasing applications of paramodulation. Since the publication of Loveland's book [44], no improvements of the paramodulation-based tableau calculi have been described except for Plaisted [55] who has shown how to transform derivations with resolution and paramodulation to tableaux by introducing a tableau factoring rule.

Here we show that paramodulation is complete under the following restrictions:

1. No functional reflexivity is needed;
2. Paramodulation into variables is not allowed;
3. Orderings are used so that there are no increasing applications of paramodulation;
4. Basic restriction on paramodulation that allows us to prohibit paramodulation into non-variables terms introduced by unification.

All these refinements are a consequence of our main result (Theorem 12).

To conform our results to Fitting's terminology, we also consider in this section predicates different from equality.

First, we formulate the tableau basic superposition rule. In order to formalize the basic strategy, we keep the substitution condition as a set of constraints, as before. Thus, we work with *constrained tableaux:*

Definition 13 (Constrained tableau) A *constrained tableau* is a pair consisting of a tableau T and a constraint $\mathcal{C}$, denoted $T \cdot \mathcal{C}$.

Now we adapt the tableau rules of [31] to the case of constrained tableaux. When we prove a formula φ, we start with the constrained tableau $\neg\varphi \cdot \emptyset$. All *tableau expansion rules* are formulated in the same way as in [31], they do not change the constraint associated with the tableau. The other rules are changed as follows. Let $T \cdot \mathcal{C}$ be a constrained tableau.

The MGU atomic closure rule. Suppose that two nodes lying on the same non-closed branch in T have the form $P(s_1, \ldots, s_n)$ and $\neg P(t_1, \ldots, t_n)$. Then we can derive the tableau $T' \cdot \mathcal{C} \cup \{P(s_1, \ldots, s_n) = P(t_1, \ldots, t_n)\}$, where T' is obtained from T by marking as closed all branches containing these two nodes, provided that the constraint $\mathcal{C} \cup \{P(s_1, \ldots, s_n) = P(t_1, \ldots, t_n)\}$ is satisfiable;

The tableau reflexivity rule. Suppose that there is a node $s \neq t$ lying on a non-closed branch in T. Then we can derive the tableau $T' \cdot \mathcal{C} \cup \{s = t\}$, where T' is obtained from T by marking as closed all branches containing this node, provided that the constraint $\mathcal{C} \cup \{s = t\}$ is satisfiable;

Rigid basic superposition. Suppose that a non-closed branch in T contains two nodes $l = r$ and $s[p] = t$ (respectively, $s[p] \neq t$; respectively, $L[p]$, where L is a literal whose function symbol is not $=$). Then we can derive the tableau $T' \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}$ (respectively, $T' \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}$; respectively $T' \cdot \mathcal{C} \cup \{l \succ r, l = p\}$), where T' is obtained from T by adding to this branch the literal $s[r] = t$ (respectively, $s[r] \neq t$; respectively, $L[r]$), provided that

1. The constraint $\mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}$ (respectively, $\mathcal{C} \cup \{l \succ r, s[p] \succ t, l = p\}$; respectively $\mathcal{C} \cup \{l \succ r, l = p\}$) is satisfiable;
2. The literal $s[r] = t$ (respectively, $s[r] \neq t$; respectively, $L[r]$) does not belong to this branch;
3. The term p is not a variable.

This logical system has the required completeness property:

Theorem 14 (Soundness and completeness) *Let φ be a formula. It is provable in first-order logic with equality iff there is a derivation from the constrained tableau $\neg\varphi \cdot \emptyset$ of a tableau $T \cdot \mathcal{C}$ all whose branches are closed, using the tableau expansion rules and the above rules.*

Proof. Straightforward from Theorem 12 by noting that the rules of $\mathcal{BPE}$ can be simulated by the corresponding tableau rules. □

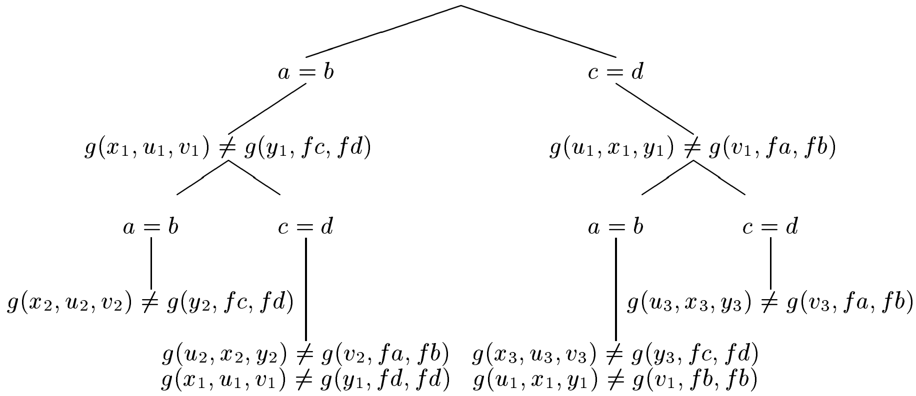
This logical system has one more pleasant property:

Theorem 15 (Termination) *For any constrained tableau $T \cdot \mathcal{C}$ there is only a finite number of derivations from $T \cdot \mathcal{C}$ not using tableau expansion rules.*

Proof. Similar to that of Theorem 7. □

This means, that for a given amplification, we cannot have infinite search. Infinite search without any expansion steps is possible in the Fitting's system.

To illustrate the connection between the tableau rigid basic superposition rule and rules of $\mathcal{BPE}$, we reconsider the example of Section 4. On the branch containing the literal $g(x_1, u_1, v_1) \neq g(y_1, fc, fd)$ and the equation $c = d$, we can apply rigid basic superposition that adds $g(x_1, u_1, v_1) \neq g(y_1, fd, fd)$ to the branch. Similarly, we can apply rigid basic superposition to the branch containing $g(u_1, x_1, y_1) \neq g(v_1, fa, fb)$ and $a = b$, obtaining $g(u_1, x_1, y_1) \neq g(v_1, fb, fb)$. This results in the following tableau:



After four application of the tableau reflexivity rules all branches of this tableau become closed. The resulting constraint of this derivation is the same as the union of the answer constraints shown at the end of Section 4.

6 Related work

The problem of extending tableaux with equality rules is crucial for enriching the deductive capabilities of the tableau method. Despite the fact that this problem is attacked by a growing number of researchers during the last years, known solutions are not yet convincing. At the same time tableau methods of automated deduction play an important role in various areas of artificial intelligence and computer science — see e.g. special issues of the Journal of Automated Reasoning, v. 13, no. 2,3, 1994. These issues contain a survey by Schumann [58] of implementations of tableau-based provers. Among 24 systems mentioned in the survey only two are able to handle equality.

The system PROTEIN [7] (and also KoMeT [13]) implement the modification method of Brand [14]. This method transforms a set of clauses with equality into a set of clauses without equality. This transformation usually yields a considerably larger set of clauses. In particular, the symmetry and the transitivity axioms must be explicitly applied to all positive occurrences of the equality predicate. Recently, we proposed a new translation method based on the so-called *basic folding* demonstrated for Horn clauses in [25].

According to Schumann [58], the system ${}_3T^AP$ uses the method of Beckert and Hähnle [11]. Paper [11] claims the completeness of the method, but this claim is not true. The method expands the tableau using the standard tableau rules, including γ-rules. For finding a closing substitution, an analog of linear paramodulation without function reflexivity has been proposed. As it is well known, linear paramodulation is incomplete without function reflexivity. The same is true for the method of Beckert and Hähnle [11], as the following example shows. Suppose that we prove the formula $\exists x(a = b \wedge g(fa, fb) = h(fa, fb) \supset g(x, x) = h(x, x))$. In order to prove it using paramodulation, we need to paramodulate $a = b$ into $g(fa, fb) = h(fa, fb)$. The method of Beckert and Hähnle [11] only allows for paramodulation into copies of $g(x, x) = h(x, x)$ obtained by the application of γ-rules. Thus, this (provable) formula cannot be proved using the method of Beckert and Hähnle [11].

Consider now approaches based on the simultaneous rigid E-unifiability by Gallier et.al. [35, 33] and related methods. We do not consider numerous works dedicated to the non-simultaneous rigid E-unifiability. This problem is NP-complete and there exist a number of complete algorithms for its solution (Gallier et.al. [34, 32], Goubault [37], Becher and Petermann [8], De Kogel [28] and Plaisted [55]). Since simultaneous rigid E-unification is undecidable (Degtyarev and Voronkov [21, 22]), their completeness is useless from the viewpoint of general purpose theorem proving as proposed by Gallier et.al. [35, 33]. Our system $\mathcal{BPE}$ can easily be extended to a calculus complete for rigid E-unifiability, but such completeness was not our aim. We tried to restrict the number of possible $\mathcal{BPE}$-derivations preserving completeness of the general-purpose method of Section 4.

It is not known whether the procedure described in Gallier et.al. [33] is complete for theorem proving. Even if it is complete, our procedure based on $\mathcal{BPE}$ has some advantages over the Gallier et.al.'s procedure. For example, for every tableau branch with p equations and q disequations, we consider q rigid equations, while Gallier et.al.'s procedure checks $q \cdot 2^p$ rigid equations.

Gallier et.al. [34, 32] introduced the notion of a complete set of solutions for rigid E-unification, proved finiteness of such sets and gave an algorithm computing finite complete set of solutions. According to this result, Goubault [38] proposed to solve simultaneous rigid E-unifiability by using finite complete sets of solutions to the components of the simultaneous problem. Paper [38] contained faulty results. The undecidability of simultaneous rigid E-unification shows that finite complete sets of solutions do not give a solution to the simultaneous problem. The reason for this is that substitutions belonging to complete sets of solutions for different rigid equations are minimal modulo *different* congruences.

Petermann [54] introduces a "complete connection calculus with rigid E-unification". Here the completeness is achieved by changing the notion of a complete set of unifiers so that solutions to all subproblems are compared modulo the same congruence (generated by the empty equality theory). In this case, a non-simultaneous problem can have an infinite number of solutions and no finite complete set of solutions. For example, for the rigid E-unification problem $f(a) = a \vdash_\forall x = a$ the complete set of solutions in the sense of Gallier et.al. [33] consists of one substitution $\{a/x\}$ (and there is only one answer constraint $\{x = a\}$ obtained by our method), but the complete set of solutions in the sense of Petermann [54] is infinite and consists of substitutions $\{f^n(a)/x\}$, for all $n \in \{0, 1, \ldots\}$. This implies that the proof-search by the method of Petermann [54] can be non-terminating even for a limited number of applications of γ-rule (i.e. for a particular tableau), unlike algorithms based on the finite complete sets of unifiers in the sense of Gallier et.al. [33] or based on minus-normalization (Kanger [40]). The implementation of the method of [54] uses a completion-based procedure by Beckert [9] of generation of complete sets of rigid E-unifiers. This procedure is developed with the aim of solving a more general problem — so-called *mixed E-unification* and has been implemented as part of the tableau-based theorem prover ${}_3T^AP$. Complete sets of unifiers both in the sense of Gallier et.al. [33] and in the sense of Petermann [54] can be computed by this procedure in the case when all variables are treated as rigid. However, the termination is not guaranteed even for complete sets of rigid E-unifiers in the sense of Gallier et.al. [33].

Plaisted [55] gives "techniques for incorporating equality into theorem proving; these techniques have a rigid flavor". His method called *path paramodulation* guarantees termination for a given amplification and, in the case of success "solves the simultaneous rigid E-unification problem", in a sense. However, this does not solve the problem attacked by a number of researchers: extend the method of matings to languages with equality by rigid E-unification. First, unlike [33] the search for solutions for a given amplification is not incremental (the method does not allow "branch-wise" computation of solutions to rigid E-unification for separate branches). Second, within a given amplification Plaisted uses factoring rules which involves two branches (paths). As a consequence, even when the original formula contains no equality, his method results in the standard tableau calculus *plus* the factoring rule.

In fact, path paramodulation of Plaisted [55] simulates resolution-paramodulation inference in a connection-like calculus. Although it is not noted in [55], but this technique has been demonstrated for resolution in many papers, for example by Bibel [12], Eder [29, 30], Mints [48], Baumgartner and Furbach [6] and Avron [2]. The generalization of this simulation to paramodulation is straightforward.

However, this simulation technique is insufficient for proving results of our paper since, in particular, it gives no insight on how to avoid factoring in tableaux with equality. The use of factoring prevents not only from the independent search for solutions for tableau branches, but even from the incremental solving of rigid equations on tableau branches as proposed by Gallier et.al.

Our *equality elimination method* [19, 23, 24] is based on extending a tableau prover by a bottom-up equation solver using basic superposition. Solutions to equations are generated by this solver and used to close branches of a tableau. Thus, the method combines (non-local) tableau proof search with the (local) equation solving. Only completely solved equations are used in the tableau part of the proof, thus reducing non-determinism created by applications of MGU replacement rule of Fitting [31]. The equation solution is even more restricted by the use of orderings, basic simplification and subsumption.

A similar idea: combination of proof-search in tableaux with a bottom-up equality saturation of the original formula, is used in [49] for constructing a goal-directed version of model elimination and paramodulation.

One of advantages of the tableau method is its applicability to non-classical logics. However, handling equality in non-classical logics seems to be much more difficult problem than that in classical logic. For example, it is shown by Voronkov [61] that procedures for intuitionistic logic with equality must handle simultaneous rigid E-unification. This implies that our method based on $\mathcal{BPE}$ *does not* give a complete procedure for intuitionistic logic with equality. Other results on relations between simultaneous rigid E-unification and intuitionistic logic are considered by Degtyarev and Voronkov [26], Degtyarev, Matiyasevich and Voronkov [18], but we do not have enough space to describe them here.

References

1. P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.
2. A. Avron. Gentzen-type systems, resolution and tableaux. *Journal of Automated Reasoning*, 10:256–281, 1993.
3. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 462–476, Saratoga Springs, NY, USA, June 1992. Springer Verlag.
4. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basis paramodulation. *Information and Computation*, 121:172–192, 1995.
5. Peter Baumgartner. An ordered theory resolution calculus. In A. Voronkov, editor, *Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 119–130, 1992.
6. P. Baumgartner and U. Furbach. Consolution as a framework for comparing calculi. *Journal of Symbolic Computations*, 16:445–477, 1993.
7. P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *IN*terface. In A. Bundy, editor, *Automated Deduction — CADE-12. 12th International Conference on Automated Deduction.*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 769–773, Nancy, France, June/July 1994.
8. G. Becher and U. Petermann. Rigid unification by completion and rigid paramodulation. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence. 18th German Annual Conference on Artificial Intelligence*, volume 861 of *Lecture Notes in Artificial Intelligence*, pages 319–330, Saarbrücken, Germany, September 1994. Springer Verlag.
9. 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.*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Nancy, France, June/July 1994.
10. B. Beckert. Are minimal solutions to simultaneous rigid E-unification sufficient for adding equality to semantic tableaux? Privately circulated manuscript, University of Karlsruhe, 1995.
11. B. Beckert and R. Hähnle. An improved method for adding equality to free variable semantic tableaux. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Saratoga Springs, NY, USA, June 1992. Springer Verlag.
12. W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, 1981.
13. W. Bibel. Issues in theorem proving based on the connection method. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 1–16, Schloß Rheinfels, St. Goar, Germany, May 1995.

14. D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4:412–430, 1975.
15. C.L. Chang. Theorem proving with variable-constrained resolution. *Information Sciences*, 4:217–231, 1972.
16. A. Degtyarev. The strategy of monotone paramodulation (in Russian). In *Fifth Soviet All-Union Conference on Mathematical Logic*, page 39, Novosibirsk, 1979.
17. A. Degtyarev. On the forms of inference in calculi with equality and paramodulation (in Russian). In Yu.V. Kapitonova, editor, *Automation of Research in Mathematics*, pages 14–26. Institute of Cybernetics, Kiev, 1982.
18. A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification and related algorithmic problems. To appear in *LICS'96*, 9 pages, 1996.
19. A. Degtyarev and A. Voronkov. Equality elimination for semantic tableaux. UPMAIL Technical Report 90, Uppsala University, Computing Science Department, December 1994. To appear in *DISCO'96*.
20. A. Degtyarev and A. Voronkov. General connections via equality elimination. UPMAIL Technical Report 93, Uppsala University, Computing Science Department, January 1995.
21. A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, Computing Science Department, May 1995. To appear in Theoretical Computer Science, v.166, 1996.
22. 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.
23. A. Degtyarev and A. Voronkov. General connections via equality elimination. In M. De Glas and Z. Pawlak, editors, *Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI-95)*, pages 109–120, Paris, July 1995. Angkor.
24. A. Degtyarev and A. Voronkov. Equality elimination for the inverse method and extension procedures. In C.S. Mellish, editor, *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 342–347, Montréal, August 1995.
25. A. Degtyarev and A. Voronkov. Handling equality in logic programs via basic folding. In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Extensions of Logic Programming (5th International Workshop, ELP'96)*, volume 1050 of *Lecture Notes in Computer Science*, pages 119–136, Leipzig, Germany, March 1996.
26. A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. To appear in *LICS'96*, 10 pages, 1996.
27. A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. UPMAIL Technical Report, Uppsala University, Computing Science Department, to appear.
28. E. De Kogel. Rigid E-unification simplified. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 17–30, Schloß Rheinfels, St. Goar, Germany, May 1995.
29. E. Eder. A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In E. Börger, G. Jäger, H. Kleine Büning, and M.M. Richter, editors, *CSL'88 (Proc. 2nd Workshop on Computer Science Logic)*, volume 385 of *Lecture Notes in Computer Science*, pages 80–98. Springer Verlag, 1988.
30. E. Eder. Consolution and its relation with resolution. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 132–136, 1991.

31. M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer Verlag, New York, 1990.
32. J. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification: NP-completeness and applications to equational matings. *Information and Computation*, 87(1/2):129–195, 1990.
33. J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the Association for Computing Machinery*, 39(2):377–429, 1992.
34. J.H. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, July 1988.
35. J.H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid E-unification: Equational matings. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, 1987.
36. J.H. Gallier, S. Raatz, and W. Snyder. Rigid E-unification and its applications to equational matings. In H. Aït Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 1, pages 151–216. Academic Press, 1989.
37. J. 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*, volume 713 of *Lecture Notes in Computer Science*, pages 202–210, Brno, August 1993.
38. J. Goubault. Rigid $\bar{E}$-unifiability is DEXPTIME-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS).* IEEE Computer Society Press, 1994.
39. R. Hähnle, B. Beckert, and S. Gerberding. The many-valued tableau-based theorem prover ${}_3T^AP$. Technical Report 30/94, Universität Karlsruhe, Fakultät für Informatik, November 1994.
40. S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 364–371. Springer Verlag, 1983. Originally appeared in 1963.
41. J. Kruskal. Well quasi ordering, the tree problem and Vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95:210–225, 1960.
42. R.C.T. Lee and C.L. Chang. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.
43. D.W. Loveland. Mechanical theorem proving by model elimination. *Journal of the Association for Computing Machinery*, 15:236–251, 1968.
44. D.W. Loveland. *Automated Theorem Proving: a Logical Basis.* North Holland, 1978.
45. S.Yu. Maslov. The inverse method of establishing deducibility in the classical predicate calculus. *Soviet Mathematical Doklady*, 5:1420–1424, 1964.
46. S.Yu. Maslov. An invertible sequential variant of constructive predicate calculus (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 4, 1967. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 4, Consultants Bureau, NY-London, 1969, p.36–42.
47. V.A. Matulis. On variants of classical predicate calculus with the unique deduction tree (in Russian). *Soviet Mathematical Doklady*, 148:768–770, 1963.
48. G. Mints. Gentzen-type systems and resolution rules. part I. propositional logic. In P. Martin-Löf and G. Mints, editors, *COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer Verlag, 1990.

49. M. Moser, C. Lynch, and J. Steinbach. Model elimination with basic ordered paramodulation. Technical Report AR-95-11, Fakultät für Informatik, Technische Universität München, München, 1995.
50. R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In *ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer Verlag, 1992.
51. R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.
52. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computations*, 19:321–351, 1995.
53. S.A. Norgela. On the size of derivations under minus-normalization (in Russian). In V.A. Smirnov, editor, *The Theory of Logical Inference*. Institute of Philosophy, Moscow, 1974.
54. U. Petermann. A complete connection calculus with rigid E-unification. In *JELIA'94*, volume 838 of *Lecture Notes in Computer Science*, pages 152–166, 1994.
55. D.A. Plaisted. Special cases and substitutes for rigid E-unification. Technical Report MPI-I-95-2-010, Max-Planck-Institut für Informatik, November 1995.
56. D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 162–201. Springer Verlag, 1983. Originally appeared in 1960.
57. G. Robinson and L.T. Wos. Paramodulation and theorem-proving in first order theories with equality. In Meltzer and Michie, editors, *Machine Intelligence*, volume 4, pages 135–150. Edinburgh University Press, Edinburgh, 1969.
58. J. Schumann. Tableau-based theorem provers: Systems and implementations. *Journal of Automated Reasoning*, 13(3):409–421, 1994.
59. R. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21:583–585, July 1978.
60. R.M. Smullyan. *First-Order Logic*. Springer Verlag, 1968.
61. A. Voronkov. On proof search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. To appear in *CADE'96*, 15 pages, 1996.