Equality Elimination for the Tableau Method

Anatoli Degtyarev* and Andrei Voronkov**

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

Abstract. We apply the *equality elimination method* to semantic tableaux with equality. The resulting logical system is a combination of a goal-directed tableau calculus with a basic superposition calculus. Unlike most other known methods of adding equality to semantic tableaux, equality elimination does not use simultaneous rigid E-unification or its modifications. For controlling redundancy, we can use powerful strategies of subsumption and simplification. We also make an extensive comparison with related works in the area.[1]

1 Semantics tableaux and equality

The problem of extending tableaux with equality rules is crucial for enriching the deductive capabilities of the tableau method. This problem is attacked by a growing number of researchers during the last years.

When we studied literature on tableau-based automated reasoning techniques, we found two tendencies in this area. The first tendency is to search for a proof by methods bringing us back to classical papers in proof theory, mainly to Prawitz [36] and Kanger [22] and also to results on normalization of proofs in sequential calculi (Orevkov [34], Maslov, Mints and Orevkov [29]). Since many notions have been (for good or for bad) reinvented in papers on tableau methods, we would like to recall fundamental ideas of earlier papers.

Prawitz and Kanger proposed the method of applications of proof rules $(\forall \rightarrow)$ and $(\rightarrow \exists)$ (later called γ-rules in Smullyan [39] or Fitting [18])[2]. The idea was to introduce a new kind of variables (called "dummies" in both Prawitz [36] and Kanger [22]), and to delay instantiation of these variables until necessary information for it is obtained. Comparing this approach to an earlier work of Beth [5], Prawitz [36] noted: "the solution proposed here is quite different but well-suited for mechanical use". This method was independently proposed in Russia by N. Shanin and has been characterized as the "metavariable method" in Maslov, Mints and Orevkov [29]. Information for instantiation is provided by

* Supported by a grant from the Swedish Royal Academy of Sciences
** Supported by a TFR grant
[1] Due to the space restrictions, we omit proofs in this paper. Proofs may be found in the technical report [10].
[2] There are differences in *representation* of inferences between Kanger [22] and Fitting [18] — Kanger uses sequent proofs. Sequents in Kanger [22] correspond to branches in tableaux Fitting [18].

constructing an uninstantiated proof, and checking from time to time whether one can find values for dummies which make it a valid proof. In Kanger [22] this check is reduced to verifying that the top sequents are "directly demonstrable", i.e. can be obtained from axioms by applications of equality rules. The possibility of applying all equality rules before all other rules has also been demonstrated in Orevkov [34] and used in Maslov [28].

Dummies or metavariables have later been called "free variables" in Fitting [18]. Free variables are substituted for existentially quantified variables in the goal, when γ-rules are applied. Directly demonstrable sequents become closed branches of a tableau, the procedure for a "direct demonstration" becomes the check for the rigid E-unifiability (Gallier, Raatz and Snyder [19]). The simultaneous rigid E-unifiability for closing all branches of a tableau has recently been proved undecidable (Degtyarev and Voronkov [13]). Thus, the problem arises to find methods not based on the simultaneous rigid E-unifiability[3].

The second tendency in tableaux-based theorem proving with equality is that of "globality". The main idea of using metavariables — to provide a smallest instantiation — comes out through the notion of a most general unifier. But another advantage of post-Kanger procedures is that the idea of metavariables is combined with the principle of *locality* (Maslov, Mints and Orevkov [29]). Among these procedures are resolution, hyperresolution, the inverse method and their numerous modifications (see e.g. surveys Maslov, Mints and Orevkov [29], Degtyarev and Voronkov [9, 8]).

The tableau method achieves goal-orientation by sacrificing the principle of locality. A substitution in tableau methods is applied to the whole tableau (see e.g. the "MGU replacement rule" from Fitting [18]), which corresponds to a set of sequents, while in the inverse method (see Voronkov [42], Degtyarev and Voronkov [11]) a substitution is applied to a sequent. In Section 4 we show that the use of the functional reflexivity in Fitting [18] is a consequence of the global substitution application. In order to reduce high non-determinism caused by applications of functional reflexivity and MGU replacement rule, we have to search for methods using local rules.

In this paper we propose a method of handling equality in tableaux which tries to solve the above mentioned problems. Our method 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 in Fitting [18]. 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 and a bottom-up equality saturation of the original formula, is used in Moser, Lynch, and Steinbach [32] for constructing a goal-directed version of model elimination with re-

[3] Recently, we have found that an incomplete procedure for rigid E-unification can be used to give a proof procedure complete for first-order logic [15].

fined paramodulation.

2 Preliminaries

We present here a brief overview of notions and preliminary definitions necessary for understanding the paper. We assume basic knowledge of resolution, rewriting and the tableau method (e.g. Lee and Chang [23], Dershowitz and Jouannaud [17] and Fitting [18]).

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 $T(\Sigma, \emptyset)$ 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. A *clause* is a finite set of literals $\{L_1, \ldots, L_n\}$, denoted $L_1, \ldots, L_n$. If L is a literal and C a clause, then L, C denotes the clause $\{L\} \cup C$. The *empty clause* is denoted by $\Box$. An *equation* is a literal $s = t$, where $s, t \in T(\Sigma, X)$. Literals of the form $\neg(s = t)$ are denoted by $s \neq t$ and called *disequations.* By a *ground expression* (i.e. term, equation, clause etc.) we mean an expression containing no variables. We write $A[s]$ to indicate that an expression A contains s as a subexpression and denote by $A[t]$ the result of replacing a particular occurrence of s in A by t. By $A\sigma$ we denote the result of applying the substitution σ to A. $\sigma \leq \theta$ means that there is a substitution τ such that $\sigma\tau = \theta$. Substitutions θ with the domain $x_1, \ldots, x_n$ are denoted by $\{x_1\theta/x_1, \ldots, x_n\theta/x_n\}$.

From now on we assume that $\succ$ is a lexicographic path ordering on $T(\Sigma, X)$ *extending a total precedence relation on* Σ (e.g. Dershowitz and Jouannaud [17]). Nevertheless, our results are valid for *any* reduction ordering total on the set of ground terms $T(\Sigma)$.

Following Bachmair et.al. [2] we distinguish terms occurring in the original formula from terms introduced by substitution by using *closures,* i.e. pairs $C \cdot \sigma$, where C is a clause, σ is a substitution. Such a closure semantically corresponds to the clause $C\sigma$. For any expression E, the set $Var(E)$ is defined as the set of all variables occurring in E. Two closures $C_1 \cdot \sigma_1$ and $C_2 \cdot \sigma_2$ are *variants* iff C_1 is a variant of C_2 and $C_1\sigma_1$ is a variant of $C_2\sigma_2$.

The symbol $\vdash$ stands for the provability in classical first order logic with equality.

A *tableau* is a set $\{C_1, \ldots, C_n\}$ of clauses, denoted $\mid C_1 \mid \ldots \mid C_n \mid$, where $n \neq 0$. The tableau with $n = 0$ is called *the empty tableau* and denoted by $\#$.

There is a correspondence between the traditional representation of tableaux and our representation. For each branch B_i of a tableau represented as a tree (see e.g. Fitting [18]), take the set of all formulas $\varphi_1, \ldots, \varphi_n$ on this branch and replace these formulas on their *names* (literals, defined below), then we obtain the clause C_i.

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 translation of formulas without equivalences into formulas in Skolem negation normal form consisting of the standard skolemization and a translation into negation normal form used e.g. in Andrews [1].

Our algorithm uses *subset unification*[4]. A substitution σ is called a *subset unifier* of the clause D_1 against D_2 iff $D_1\sigma \subseteq D_2\sigma$. It is a *minimal subset unifier* of D_1 against D_2 iff it is minimal w.r.t. $\leq$ among all subset unifiers of D_1 against D_2. Subset-unifiability is NP-complete. Some other properties of subset unification are discussed in Degtyarev and Voronkov [10].

3 The equality elimination method

For the rest of this section we assume that ξ denotes a closed formula in the Skolem negation normal form to be proved (the "goal"). We assume that all different occurrences of quantifiers in ξ bind different variables. For example, ξ cannot have the form $\exists xA \vee \exists xB$. All formulas in this section are assumed to be subformulas of the goal formula ξ. We shall identify subformulas of ξ and their superformulas with their *occurrences* in ξ. For example, in the formula ξ of the form $A \wedge (A \vee B)$ the second occurrence of A is considered a subformula of $(A \vee B)$, but the first occurrence of A is not.

In order to define a more efficient tableau calculus, we shall only deal with some subformulas of ξ called conjunctive subformulas. The occurrence of a subformula φ of ξ is called *conjunctive* iff it is an occurrence in a subformula $\varphi \wedge \psi$ or in $\psi \wedge \varphi$. A *conjunctive superformula* of φ is a superformula[5] ψ of φ that is conjunctive. The *least conjunctive superformula* of φ is the conjunctive superformula ψ of φ such that any other conjunctive superformula of φ is a superformula of ψ. Let us note that conjunctive superformulas do not necessarily exist. For example, ξ has no conjunctive superformula. Any formula having a conjunctive superformula has the unique least conjunctive superformula.

We can enumerate all conjunctive subformulas $\xi_1, \ldots, \xi_n$ of ξ, for example in the order of their occurrences in ξ. Thus we can unambiguously use "the kth conjunctive (sub)formula" ξ_k of ξ.

Let $A_1, \ldots, A_n$ be predicate symbols not occurring in ξ. We say that the atomic formula $A_k(x_1, \ldots, x_m)$ is *the ξ-name of a subformula* φ of ξ iff

1. The least conjunctive superformula of φ is ξ_k;
2. $x_1, \ldots, x_m$ are all free variables of ξ_k in the order of their occurrences in ξ_k.

If a ξ-name of a formula φ exists, then it is unique. Note that different formulas may have the same ξ-names. Also note that some subformulas of ξ do not have ξ-names. We can use *the set of ξ-names* of a subformula. The set of ξ-names of a formula φ is either $\emptyset$ or a singleton $\{A_k(x_1, \ldots, x_m)\}$.

Figure 1 illustrates least conjunctive superformulas and ξ-names:

Lemma 1. *Let φ be a subformula of ξ. Then*

[4] A variant of subset unification has been called θ-subsumption in Minker, Rajasekar and Lobo [31].

[5] φ is a superformula of ψ iff ψ is a subformula of φ (not necessarily proper).

Subformula	least conjunctive superformula	
$(\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y))) \wedge \exists z D(z)) \vee E$	no	$\emptyset$
$\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y))) \wedge \exists z D(z)$	no	$\emptyset$
$\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y)))$	$\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y)))$	$\{A_1\}$
$F(x) \wedge (B(x) \vee \exists y C(x,y))$	$\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y)))$	$\{A_1\}$
$F(x)$	$F(x)$	$\{A_2(x)\}$
$B(x) \vee \exists y C(x,y)$	$B(x) \vee \exists y C(x,y)$	$\{A_3(x)\}$
$B(x)$	$B(x) \vee \exists y C(x,y)$	$\{A_3(x)\}$
$\exists y C(x,y)$	$B(x) \vee \exists y C(x,y)$	$\{A_3(x)\}$
$C(x,y)$	$B(x) \vee \exists y C(x,y)$	$\{A_3(x)\}$
$\exists z D(z)$	$\exists z D(z)$	$\{A_4\}$
$D(z)$	$\exists z D(z)$	$\{A_4\}$
E	no	$\emptyset$

Fig. 1. Least conjunctive superformulas and sets of ξ-names of subformulas of the formula $(\exists x(F(x) \wedge (B(x) \vee \exists y C(x,y))) \wedge \exists z D(z)) \vee E$

1. *If ψ is the least conjunctive superformula of φ then* $\vdash \forall(\varphi \supset \psi)$.
2. *If there is no conjunctive superformula of φ then* $\vdash \forall(\varphi \supset \xi)$.

This lemma partially explains the need for introducing least conjunctive superformulas. There are deterministic chains of inferences in sequent systems, where formulas with $\vee$ or $\exists$ occur. For example,

$$\cfrac{\cfrac{\Gamma \to \varphi}{\Gamma \to \varphi \vee \psi}}{\Gamma \to \exists x(\varphi \vee \psi)}$$

By restricting ourselves to conjunctive superformulas only, we eliminate these deterministic chains, making them in one step[6].

Now we can formulate the tableau deductive system we are going to use. The system depends on the goal formula ξ and is denoted T_ξ. The provable objects of the system are of two kinds: closures and tableaux. There are inference rules allowing to derive closures from closures and tableaux from closures and tableaux. The axioms of T_ξ are *initial closures* and the *initial tableau* defined below.

[6] This simple but powerful idea of restricting to conjunctive superformulas (see Voronkov [42]) has been successfully implemented in the theorem prover described in Voronkov [40] and in a theorem prover for intuitionistic logic implemented by T. Tammet (private communications). In the framework of tableau theorem proving, several papers used permutabilities of inference rules in sequent calculi (see e.g. Shankar [38]). In fact, the least conjunctive superformula of φ is a superformula ψ of φ provable from φ and such that all inference rules applied in the proof of ψ from φ are permutable with all other rules. The use of conjunctive superformulas allows us to get rid of non-conjunctive subformulas before the proof-search, unlike the dynamic use of permutabilities as in Shankar [38].

Initial closures of T_ξ are generated according to one of the three rules:

1. Whenever a literal $s \neq t$ occurs in ξ and C is the set of ξ-names of this occurrence of $s \neq t$, the closure $s = t, C \cdot \varepsilon$ is an initial closure.
2. Whenever $s = t$ occurs in ξ and C is the set of ξ-names of this occurrence of $s = t$, the closure $s \neq t, C \cdot \varepsilon$ is an initial closure.
3. Let literals $P(s_1, \ldots, s_n)$ and $\neg P(t_1, \ldots, t_n)$ occur in ξ and C_1, C_2 are their sets of ξ-names. Let the substitution σ rename variables such that variables of $P(s_1, \ldots, s_n)\sigma$ and $P(t_1, \ldots, t_n)$ are disjoint. Then the closure $s_1\sigma \neq t_1, \ldots, s_n\sigma \neq t_n, C_1\sigma, C_2 \cdot \varepsilon$ is an initial closure.

The **initial tableau** of T_ξ is the tableau $\mid \Box \mid$ consisting of the empty clause.

The calculus T_ξ consists of five inference rules: two basic superposition rules, the equality solution rule, the tableau expansion rule and the branch closure rule. The first three inference rules are adapted from basic superposition-based inference systems (Bachmair et.al. [2], Nieuwenhuis and Rubio[33]). As usual, we assume that premises of rules have disjoint variables which can be achieved by renaming variables.

Basic (right and left) superposition

$$\frac{(s = t, C) \cdot \sigma_1 \quad (u[s'] = v, D) \cdot \sigma_2}{(u[t] = v, C, D) \cdot \sigma_1\sigma_2\rho} \qquad \frac{(s = t, C) \cdot \sigma_1 \quad (u[s'] \neq v, D) \cdot \sigma_2}{(u[t] \neq v, C, D) \cdot \sigma_1\sigma_2\rho}$$

where

1. ρ is a most general unifier of $s\sigma_1$ and $s'\sigma_2$;
2. $t\sigma_1\rho \not\succeq s\sigma_1\rho$ and $v\sigma_2\rho \not\succeq u[s']\sigma_2\rho$;
3. s' is not a variable.
4. (for left superposition only) $u[s'] \neq v$ is the leftmost disequation in the second premise.[7]

Equality solution

$$\frac{(s \neq t, C) \cdot \sigma}{C \cdot \sigma\rho}$$

where ρ is a most general unifier of $s\sigma, t\sigma$ and $s \neq t$ is the leftmost disequation in the premise.

Tableau expansion Let D_1, D_2 and D be the sets of ξ-names of φ, ψ and $\varphi \wedge \psi$, where $\varphi \wedge \psi$ is a subformula of ξ. Then the following is a tableau expansion rule:

$$\frac{\mid C_1 \mid C_2 \mid \ldots \mid C_m \mid}{(\mid D_1, C_1 \mid D_2, C_1 \mid C_2 \mid \ldots \mid C_m \mid)\sigma}$$

where σ is a minimal subset unifier of D against C_1 (we assume that the variables of the premise are disjoint from the variables of D, D_1, D_2).

[7] According to our definitions, a clause is a set of literals, so the use of the leftmost disequation is not quite correct, but this restriction can easily be formalized using the selection mechanism (Bachmair et.al. [2]).

Branch closure

$$\frac{\mid C_1 \mid C_2 \mid \ldots \mid C_m \mid \quad C \cdot \rho}{(\mid C_2 \mid \ldots \mid C_m \mid)\sigma}$$

where σ is a minimal subset-unifier of $C\rho$ against C_1. (Note that the closure $C \cdot \rho$ used in this rule cannot contain equality because there is no rule in T_ξ introducing equality in a tableau.)

Let us make some comments on this logical system. Intuitively, tableau expansion rules encode the conjunction rule of sequent calculi or β-rules in the terminology of Smullyan [39] or Fitting [18]:

$$\frac{\Gamma \to \varphi \quad \Gamma \to \psi}{\Gamma \to \varphi \wedge \psi}$$

but turned upside down. The main difference is that instead of formulas φ, ψ and $\varphi \wedge \psi$ we use their ξ-names.

As we can see from the general description of our rules, the branch closure rule is non-local: the substitution produced by it changes the whole tableau, all other rules are local. (The substitution σ used in the tableau expansion rule only renames variables in the premise.)

Theorem 2 (Soundness and completeness). *The formula ξ is provable in first-order logic iff there is a derivation of the empty tableau # in T_ξ.*

Let us consider an example. We shall use a unary function symbol f. In order to avoid formulas having cluttered with parentheses, we shall write ft instead of $f(t)$. The formula to be proven is

$$\exists x((a \neq x \vee \neg G(b) \vee G(ffx)) \wedge (\neg G(fx) \vee x = b))$$

There are two conjunctive subformulas with the following sets of ξ-names:

Formula	Set of ξ-names
$a \neq x \vee \neg G(b) \vee G(ffx)$	$\{A_1(x)\}$
$\neg G(fx) \vee x = b$	$\{A_2(x)\}$

Thus, all tableau expansion rules have the form

$$\frac{\mid C_1 \mid C_2 \mid \ldots \mid C_m \mid}{\mid A_1(y), C_1 \mid A_2(y), C_1 \mid C_2 \mid \ldots \mid C_m \mid}$$

where y does not occur in the premise of this rule.

As the order $\succ$ we consider the lexicographic path ordering induced by the precedence relation $f > a > b$. The proof is as follows:

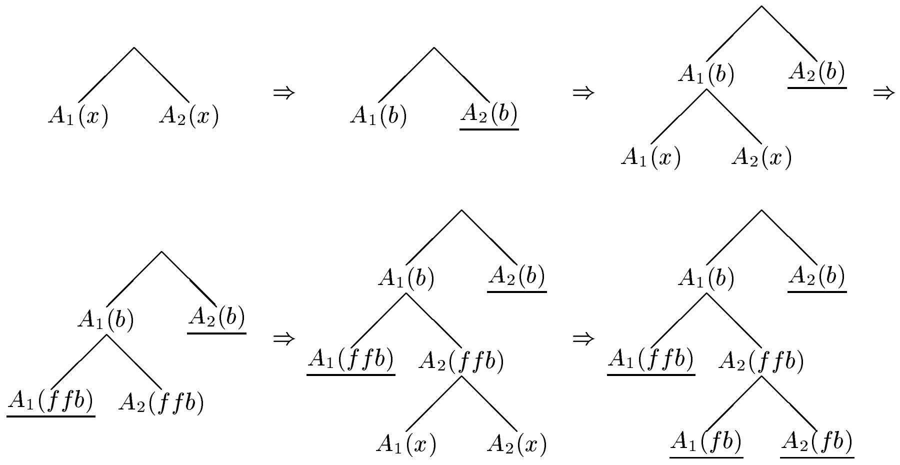

Fig. 2. A tableau proof

1. $\mid \Box \mid$ (the initial tableau)
2. $\mid A_1(x) \mid A_2(x) \mid$ (tableau expansion from 1)
3. $x \neq b, A_2(x) \cdot \varepsilon$ (initial closure)
4. $A_2(x) \cdot \{b/x\}$ (equality solution from 3)
5. $\mid A_1(b) \mid$ (branch closure from 2,4)
6. $\mid A_1(x), A_1(b) \mid A_2(x), A_1(b) \mid$ (tableau expansion from 5)
7. $a = x, A_1(x) \cdot \varepsilon$ (initial closure)
8. $b \neq ffx, A_1(x), A_1(y) \cdot \varepsilon$ (initial closure)
9. $b \neq a, A_1(x), A_1(y), A_1(z) \cdot \{ffy/x\}$ (left superposition from 7,8)
10. $b \neq u, A_1(u), A_1(x), A_1(y), A_1(z) \cdot \{ffy/x\}$ (left superposition from 7,9)
11. $A_1(u), A_1(x), A_1(y), A_1(z) \cdot \{ffy/x, b/u\}$ (equality solution from 10)
12. $\mid A_2(ffb), A_1(b) \mid$ (branch closure from 6,11)
13. $\mid A_1(x), A_2(ffb), A_1(b) \mid A_2(x), A_2(ffb), A_1(b) \mid$ (tableau expansion from 12)
14. $fx \neq ffy, A_2(x), A_1(y) \cdot \varepsilon$ (initial closure)
15. $A_2(x), A_1(y)) \cdot \{fy/x\}$ (equality solution from 14)
16. $\mid A_2(fb), A_2(ffb), A_1(b) \mid$ (branch closure from 13,15)
17. $\#$ (branch closure from 16,15)

In order to illustrate this proof, we display tableaux in more conventional tree-like form in Figure 2. We omit superposition steps in this proof. In the closed branches, the atom at the bottom of the branch is underlined. The last step here corresponds to two applications of the branch closure rule.

There are several ways to automate proofs in T_ξ. One possibility is to derive closures using a saturation (bottom-up) algorithm in the style described in Lusk [25] and to derive tableaux using the usual top-down search with backtracking. In this case we can apply without losing completeness the following powerful strategies reducing search space:

1. Simplification of tableaux by derived unit closures $s = t \cdot \sigma$;

2. Basic simplification and subsumption for closures (similar to that of Bachmair et.al. [2]).

Another possibility is to use bottom-up (saturation) algorithms for the whole calculus T_ξ. In this case, we shall need powerful strategies for keeping the generated tableaux in a reasonable space. For example, the following subsumption notion for tableaux can be used in the same way as subsumption in resolution, i.e. subsumed tableaux can be removed from the search space.

Definition 3 (Tableau subsumption). A tableau $T = \mid C_1 \mid \ldots \mid C_n \mid$ *subsumes* a tableau $T' = \mid C'_1 \mid \ldots \mid C'_m \mid$, denoted $T \leq T'$, iff there is a substitution θ such that for every $i \in \{1, \ldots, n\}$ there is $j \in \{1, \ldots, m\}$ such that $C'_j \subseteq C_i\theta$.

The ordering $\leq$ on tableaux satisfies all standard properties of subsumption orderings of Voronkov [42] and hence we can remove tableaux subsumed by other tableaux. This ordering has some other properties:

1. The goal tableau $\#$ is the least element w.r.t. $\leq$;
2. The initial tableau $\mid \Box \mid$ is the greatest element w.r.t. $\leq$;
3. If the tableau T is obtained from the tableau T' by a tableau expansion, then $T \leq T'$;
4. The tableau $T = \mid C_2 \mid \ldots \mid C_n \mid$ subsumes the tableau $T' = \mid C_1 \mid \ldots \mid C_n \mid$. In particular, it means that for an application of a branch closure rule
$$\frac{\mid C_1 \mid C_2 \mid \ldots \mid C_m \mid \quad C \cdot \rho}{(\mid C_2 \mid \ldots \mid C_m \mid)\sigma}$$
where the subset-unifier σ does not change variables of $C_2, \ldots, C_n$, the conclusion of the rule subsumes the premise of the rule.

When we prove an equation from the set of universally quantified equations, i.e. when the goal formula ξ has the form $\forall s_1 = t_1 \wedge \ldots \wedge \forall s_n = t_n \supset \exists s = t$, there are no conjunctive subformulas in ξ. Hence, the calculus T_ξ becomes the calculus for refutational basic superposition-based (unfailing) completion.

One of the main problems in the tableau method is to decrease the number of applications of γ-rules. A technique for decreasing this number has been proposed in Beckert and Hähnle [4]. This paper introduced an optimization based on the (undecidable) notion of a formula in a tableau branch universal with respect to a variable and proved a sufficient (decidable) condition for universality. A more powerful optimization based on the so-called *usable subformulas* has been used earlier in Voronkov [41]. This optimization is also built-in in T_ξ due to its way of assigning ξ-names to the subformulas of ξ. If a formula φ and a variable x satisfy the sufficient condition of Beckert and Hähnle [4], then x does not occur in the set of ξ-names of φ (the inverse is not true in general). In order to take full advantage of this optimization we have to miniscope existential quantifiers as in Wang [43].

Let us consider an example. The goal is $\forall z \forall v \forall y (v = fy \supset g(v, z) = fv) \supset \exists x (g(g(x, fx), x) = g(fx, g(fx, x)))$. After translation to negation normal form and miniscoping we obtain the goal ξ of the form $\exists v(\exists y(v = fy) \wedge \exists z(g(v,z) \neq fv)) \vee \exists x(g(g(x,fx),x) = g(fx,g(fx,x)))$. We introduce two ξ-names: $A_1(v)$ for $\exists y(v = fy)$ and $A_2(v)$ for $\exists z(g(v,z) \neq fv)$. The tableau expansion rules are the same as in the previous example. As the order $\succ$ we consider the lexicographic path ordering induced by the precedence relation $g > f$. The proof in T_ξ is as follows:

1. $\mid \Box \mid$ (the initial tableau)
2. $\mid A_1(x) \mid A_2(x) \mid$ (tableau expansion from 1)
3. $v \neq fy, A_1(v) \cdot \varepsilon$ (initial closure)
4. $A_1(v) \cdot \{fy/v\}$ (equality solution from 3)
5. $\mid A_2(fy) \mid$ (branch closure from 2,4)
6. $\mid A_1(x), A_2(fy) \mid A_2(x), A_2(fy) \mid$ (tableau expansion from 5)
7. $\mid A_2(fy_1), A_2(fy_2) \mid$ (branch closure from 4,6)
8. $g(v,z) = fv, A_2(v) \cdot \varepsilon$ (initial closure)
9. $g(g(x,fx),x) \neq g(fx,g(fx,x)) \cdot \varepsilon$ (initial closure)
10. $g(fv_1,x) \neq g(fx,g(fx,x)), A_2(v_1) \cdot \{x/v_1\}$ (left superposition from 8,9)
11. $g(fv_1,x) \neq fv_2, A_2(v_1), A_2(v_2) \cdot \{x/v_1, fx/v_2\}$ (left superposition from 8,10)
12. $fv_3 \neq fv_2, A_2(v_1), A_2(v_2), A_2(v_3) \cdot \{x/v_1, fx/v_2, fx/v_3\}$ (left superposition from 8,11)
13. $A_2(v_1), A_2(v_2), A_2(v_3) \cdot \{x/v_1, fx/v_2, fx/v_3\}$ (equality solution from 12)
14. # (branch closure from 13,7)

In the last branch closure rule, we applied subset-unification of $A_2(x), A_2(fx), A_2(fx)$ against $A_2(fy_1), A_2(fy_2)$ using the subset-unifier $\{fy_1/x, fy_1/y_2\}$. If we include the variable z into the ξ-name of $g(v,z) \neq fv$, using $A_2(v,z)$ instead of $A_2(v)$, this subset-unification would not be possible, since $A_2(v_2,z_2)$ and $A_2(v_3,z_3)$ would use different substitutions for z_2 and z_3. In that case, we would have to apply a branch expansion rule once more, that corresponds to additional applications of γ-rules and β-rules.

4 Related work

Semantic tableaux belong to automatic proof methods based on Gentzen sequent calculi. Other methods based on sequent calculi are the inverse method (Maslov [26]), model elimination (Loveland [24]) and the method of matings (Andrews [1]), or the connection method (Bibel [6]). Proof-search in these methods uses the Herbrand theorem and can be considered as the problem of checking (by different methods) that every path through a matrix obtained from the goal formula is inconsistent. For this reason, these methods are sometimes characterized as matrix methods. This idea was originally justified by Prawitz [36] and Kanger [22]. Instead of generating instances of a quantifier-free formula $M(\bar{x})$, they proposed to search for a substitution σ for $\bar{x}_1, \ldots, \bar{x}_n$ such that every path in $(M(\bar{x}_1) \wedge \ldots \wedge M(\bar{x}_n))\sigma$ becomes complementary. In the case without equality, the search for such a *closing* substitution can be carried out using ordinary (simultaneous) unification. Thus, for a given matrix $M(\bar{x}_1) \wedge \ldots \wedge M(\bar{x}_n)$, the problem of the existence of an appropriate substitution is decidable and can be performed by any known unification algorithm. The same idea works for logic with equality, but the closing substitution should make any path E-inconsistent, or E-complementary. The problem of finding such a closing substitution is equivalent to the *simultaneous rigid E-unifiability problem* (Gallier, Raatz and Snyder [19]). Simultaneous rigid E-unifiability has recently proved to be undecidable in Degtyarev and Voronkov [13].

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 that is complete for first-order logic with equality. In the terminology of Fitting [18] 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 [30] and Maslov [27]. Another way of using incomplete but terminating algorithm for rigid E-unification has recently been proposed in Degtyarev and Voronkov [15].

Resolution-like procedures usually handle equality using variants of the paramodulation rule. It is well-known Brand [7] that paramodulation allows one to get rid of all equality axioms (except for $x = x$), and especially of the functional reflexivity axioms $f(\bar{x}) = f(\bar{x})$ for every function symbol f. The functional reflexivity axioms allow one to substitute *arbitrary terms* for variables. An analog of paramodulation was introduced in tableaux (Fitting [18]), called MGU replacement rule, *with* the functional reflexivity rule. We give an example that functional reflexivity is unavoidable in the method of proving completeness used in Fitting [18]. Consider the formula $\exists xyvu((a = b \supset g(x,u,v) = g(y,fc,fd)) \wedge (c = d \supset g(u,x,y) = g(v,fa,fb)))$. After one quantifier duplication (application of a γ-rule) we obtain the following tableau.

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

The substitution $\{fc/u, fd/v, fa/x, fb/y\}$ closes this tableau. This substitution can be found by applying the functional reflexivity rule and MGU replacement rule.

The use of the functional reflexivity in the completeness proof of Fitting [18] is dictated by the fact that the substitutions in the tableau method are applied to the whole tableau. In the terminology of Maslov, Mints and Orevkov [29], they are *global* unlike the *local substitutions* used in resolution. In intuitionistic logic, for which no local methods exist, there are no complete systems without functional reflexivity. In addition, minus-normalization does not hold for the intuitionistic sequent calculus (Maslov [27]).

Consider now approaches based on the simultaneous rigid E-unifiability (Gallier, Raatz and Snyder [19] Gallier et.al. [20]) and related methods. Since simultaneous rigid E-unification is undecidable (Degtyarev and Voronkov [13]), 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.

In Gallier et.al. [20] it was proved that any (non-simultaneous) rigid E-unification problem has a finite complete set of solutions. According to this result, Goubault [21] proposed to solve simultaneous rigid E-unifiability by using finite complete sets of solutions to the components of the simultaneous problem. The undecidability result of Degtyarev and Voronkov [13] shows that finite complete sets of solutions do not give a solution to the simultaneous problem. The reason for this is that the comparison of solutions to different subproblems is made modulo different congruences.

Petermann [35] 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. [20] consists of one substitution $\{a/x\}$, but the complete set of solutions in the sense of Petermann [35] 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 [35] 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. [20] or based on minus-normalization (Kanger [22], Matulis [30]). The implementation of the method of Petermann [35] uses a completion-based procedure (Beckert [3]) 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. [20] and in the sense of Petermann [35] can be computed by this procedure. However, the termination is not guaranteed even for complete sets of rigid E-unifiers in the sense of Gallier et.al. [20].

A method not using rigid E-unification is proposed in Beckert and Hähnle [4]. This paper 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 [4], 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 [4] 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 [4].

In several tableau-based theorem provers (see the survey of Schumann [37]) equality is implemented using the modification method of Brand [7]. This method reduces a set of clauses with equality to a set of clauses without equality via a *flattening transformation.* Recently, we proposed a new reduction method based on the so called *basic folding*, demonstrated for Horn clauses in [14].

The equality elimination method was originally introduced in Degtyarev and Voronkov [12] as a method of handling equality in logic programs. Later, it has been applied to the inverse method in Degtyarev and Voronkov [11]. Applications of equality elimination to the tableau method and to the inverse method are based on a common matrix characterization of provability in terms of solution clauses (Degtyarev and Voronkov [11]). This characterization can be considered as a computationally improved version of the Herbrand's theorem. However, the resulting calculi for the inverse method and for the tableau method are quite different because the inverse method is local while the tableau method is global (Maslov, Mints and Orevkov [29]). Among other inference rules, the inverse method uses the *factoring rule* that is absent in the tableau method. The tableau method uses the *branch closure rule* based on subset-unification that has no analogue in the inverse method. Subset-unification has been earlier used in P-deduction (Demolombe [16]) and SLO-resolution (Minker, Rajasekar and Lobo [31]).

References

1. P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.
2. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation. *Information and Computation*, 121:172–192, 1995.
3. 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.
4. 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.
5. E.W. Beth. *The Foundations of Mathematics.* North Holland, 1959.
6. W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, 1981.
7. D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4:412–430, 1975.
8. A. Degtyarev and A. Voronkov. Automated theorem proving I and II. *Cybernetics*, 22(3):290–297, 1986 and 23(4):547–556, 1987.

9. A. Degtyarev and A. Voronkov. Equality control methods in machine theorem proving. *Cybernetics*, 22(3):298–307, 1986.
10. A. Degtyarev and A. Voronkov. Equality elimination for semantic tableaux. UPMAIL Technical Report 90, Uppsala University, Computing Science Department, December 1994.
11. 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.
12. A. Degtyarev and A. Voronkov. A new procedural interpretation of Horn clauses with equality. In Leon Sterling, editor, *Proceedings of the Twelfth International Conference on Logic Programming*, pages 565–579. The MIT Press, 1995.
13. 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.
14. 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.
15. A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. In *Submitted to JELIA'96*, page 20, 1996.
16. R. Demolombe. An efficient strategy for non-Horn deductive databases. In G.X. Ritter, editor, *Information Processing 89*, pages 325–330. Elsevier Science, 1989.
17. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 243–309. North Holland, Amsterdam, 1990.
18. M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer Verlag, New York, 1990.
19. 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.
20. 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.
21. J. Goubault. Rigid $\overline{E}$-unifiability is DEXPTIME-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1994.
22. 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.
23. R.C.T. Lee and C.L. Chang. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
24. D.W. Loveland. Mechanical theorem proving by model elimination. *Journal of the Association for Computing Machinery*, 15:236–251, 1968.
25. E.L. Lusk. Controlling redundancy in large search spaces: Argonne-style theorem proving through the years. In A. Voronkov, editor, *Logic Programming and Automated Reasoning. International Conference LPAR'92.*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 96–106, St.Petersburg, Russia, July 1992.
26. S.Yu. Maslov. The inverse method of establishing deducibility in the classical predicate calculus. *Soviet Mathematical Doklady*, 5:1420–1424, 1964.

27. 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.
28. S.Yu. Maslov. The generalization of the inverse method to predicate calculus with equality (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 20:80–96, 1971. English translation in: Journal of Soviet Mathematics 1, no. 1.
29. S.Yu. Maslov, G.E. Mints, and V.P. Orevkov. Mechanical proof-search and the theory of logical deduction in the USSR. In J.Siekmann and G.Wrightson, editors, *Automation of Reasoning (Classical papers on Computational Logic)*, volume 1, pages 29–38. Springer Verlag, 1983.
30. V.A. Matulis. On variants of classical predicate calculus with the unique deduction tree (in Russian). *Soviet Mathematical Doklady*, 148:768–770, 1963.
31. J. Minker, A. Rajasekar, and J. Lobo. Theory of disjunctive logic programs. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic. Essays in Honor of Alan Robinson.*, pages 613–639. The MIT Press, Cambridge, MA, 1991.
32. 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.
33. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computations*, 19:321–351, 1995.
34. V.P. Orevkov. On nonlengthening rule applications for equality (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 16:152–156, 1969. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 16, Consultants Bureau, NY-London, 1971, p.77–79.
35. 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.
36. 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.
37. J. Schumann. Tableau-based theorem provers: Systems and implementations. *Journal of Automated Reasoning*, 13(3):409–421, 1994.
38. N. Shankar. Proof search in the intuitionistic sequent calculus. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 522–536, Saratoga Springs, NY, USA, June 1992. Springer Verlag.
39. R.M. Smullyan. *First-Order Logic*. Springer Verlag, 1968.
40. A. Voronkov. LISS - the logic inference search system. In Mark Stickel, editor, *Proc. 10th Int. Conf. on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 677–678, Kaiserslautern, Germany, 1990. Springer Verlag.
41. A. Voronkov. A proof-search method for the first order logic. In P. Martin-Löf and G. Mintz, editors, *COLOG'88*, volume 417 of *Lecture Notes in Computer Science*, pages 327–340. Springer Verlag, 1990.
42. A. Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 648–662, Saratoga Springs, NY, USA, June 1992. Springer Verlag.
43. H. Wang. Towards mechanical mathematics. *IBM J. of Research and Development*, 4:2–22, 1960.