Adding Equality to Semantic Tableaux

Bernhard Beckert

University of Karlsruhe
Institute for Logic, Complexity und Deduction Systems
76128 Karlsruhe, Germany
beckert@ira.uka.de

Abstract. This paper tries to identify the basic problems encountered in handling equality in the semantic tableau framework, and to describe the state of the art in solving these problems. The two main paradigms for handling equality are compared: adding new tableau expansion rules and using E-unification algorithms; their efficient implementation is discussed.

1 Introduction

One of the main goals of Automated Deduction is to efficiently handle first-order logic *with equality*. Just adding the equality axioms to the data base leads to a huge search space (e.g. [10]); even very simple theorems cannot be proven. The only solution is to make the handling of equality part of the inference rules. Then, still, equality typically allows a lot of different derivations. Methods have to be used for further restricting the search space.

For resolution-based provers such methods—the most important is *paramodulation*—have been known since the 1960s and have often been implemented (although the problem of preventing the derivation of redundant information remains to be solved).

At the same time methods for adding equality to Gentzen-type calculi, such as semantic tableaux and the connection method, have been developed [14, 19]. These, have not been used as often; in comparison to resolution with paramodulation they are quite inefficient. But, recently much more efficient methods have been developed, and over the last years there has been a growing interest in handling equality in semantic tableaux [20, 10, 7] and the connection method [11, 18].

This paper tries to identify the basic problems encountered in handling equality in the semantic tableau framework, and to describe the state of the art in solving these problems.

Which method is appropriate for handling equality depends heavily on the version of semantic tableaux used, namely on the type of variables occurring in the tableaux. Therefore, after giving some basic definitions, we describe the versions of semantic tableaux that are important to distinguish in the next section. In Section 3 the two main paradigms for handling equality in semantic tableaux are presented: (i) adding new tableau expansion rules and (ii) using E-unification algorithms; they are described in detail in Sections 4 and 5. In Section 6 we compare the different methods and discuss how they can be implemented efficiently. Finally, in Section 7 we draw some conclusions. The reader should be familiar with the ground and the free variable versions of semantic tableaux (an excellent introduction can be found in [10]).

2 Syntax and Semantics

2.1 Basic Definitions and Notations

Let us fix a first-order language $\mathcal{L}$ which is built up from countable sets $\mathcal{P}$ of predicate symbols, $\mathcal{F}$ of function symbols, $\mathcal{C}$ of constant symbols and $\mathcal{V}$ of object variables in the usual manner (for each arity there are countably many function and predicate symbols). We use the logical connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication) and $\neg$ (negation), and the quantifier symbols $\forall$ and $\exists$. The binary predicate symbol $\approx\, \in \mathcal{P}$ denotes equality such that no confusion with the meta-level equality predicate $=$ can arise. There is no restriction on where equalities can occur in formulæ. We use the signed version of semantic tableaux, i.e., the formulæ in tableaux are prefixed with one of the signs T (true) and F (false). Signed formulæ $\mathsf{T}\,G$ and $\mathsf{F}\,G$ are called **complementary**.

Since in the tableau proofs it will be necessary to introduce Skolem terms, we extend our first-order language $\mathcal{L}$ to a language $\mathcal{L}_{\mathrm{Sko}}$ by adding countably many constant symbols and function symbols for each arity which do not appear already in $\mathcal{L}$.

We use the standard notions of free and bound variable, (grounding) substitution, sentence, model, logical consequence (denoted by $\models$), valuation, satisfiability and tautology.[1] All occurring substitutions have a finite domain; thus, a substitution σ with domain $\{x_1, \ldots, x_n\}$ can be denoted by $\{x_1/t_1, \ldots, x_n/t_n\}$, i.e. $\sigma(x_i) = t_i$ $(1 \le i \le n)$. The restriction of σ to a set V of variables is denoted by $\sigma_{|V}$.

A model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ (with domain $\mathcal{D}$ and interpretation $\mathcal{I}$) is called **normal** iff $\mathcal{I}(\approx)$ is the identity relation on $\mathcal{D}$. A model is called **canonical** iff, moreover, for every $d \in \mathcal{D}$ there is a term t in $\mathcal{L}$ (resp. $\mathcal{L}_{\mathrm{Sko}}$) such that $\mathcal{I}(t) = d$. In the sequel, we restrict all considerations to canonical models. For first-order logic *with equality* they are the analogue of Herbrand models: A sentence is satisfied by a *normal* model iff it is satisfied by a *canonical* model.

2.2 The Ground Version of Semantic Tableaux

The ground version [23], i.e. the version without free variables, is the simplest and basic version of semantic tableaux. Unfortunately, it is (with and without equality) the least efficient as well. But, since all other versions are based upon it, we present it first.

There is a tableau rule for each combination of sign and logical connective (resp. quantifier); thus, to every signed formula that is not a literal exactly one rule can be applied. We do not list all the rules but only the schemata: α-rules (conjunctive type rules), β-rules (disjunctive), γ-rules (universally quantified), and δ-rules (existentially quantified):[2]

$$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}} \qquad \frac{\beta}{\beta_1 \mid \beta_2} \qquad \frac{\gamma}{\gamma_1(t)} \qquad \frac{\delta}{\delta_1(c)}$$

t is any ground term. c is a new Skolem constant.

To prove a formula G to be a tautology, we apply the above rules starting from the initial tableau that consists of the single formula $\mathsf{F}\,G$. A proof is found, if all branches of the constructed tableau are closed simultaneously. We identify a branch with the set of the formulæ it contains.

Definition 1. A tableau T with branches $B_1, \ldots, B_k$ is **closed** iff there are complementary formulæ $\mathsf{T}\,G_i, \mathsf{F}\,G_i \in B_i$ $(1 \le i \le k)$.

2.3 Free Variable Semantic Tableaux

Using free variable quantifier rules [10] is crucial for efficient implementation—even more if equality has to be handled. When γ-rules are applied, a new free variable is substituted for the quantified variable, instead of replacing it by a ground term, that has to be "guessed". Free variables can later be instantiated "on demand", when a tableau branch is closed or an equality is applied to expand a branch.

[1] If in doubt, the reader should consult [10] for the precise definitions.

[2] For example, if $\alpha = \mathsf{T}\,(F \wedge G)$ then $\alpha_1 = \mathsf{T}\,F$ and $\alpha_2 = \mathsf{T}\,G$; if $\beta = \mathsf{F}\,(F \wedge G)$ then $\beta_1 = \mathsf{F}\,F$ and $\beta_2 = \mathsf{F}\,G$; if $\gamma = \mathsf{T}\,(\forall x)G(x)$ then $\gamma_1(t) = \mathsf{T}\,G(t)$; if $\delta = \mathsf{F}\,(\forall x)G(x)$ then $\delta_1(t) = \mathsf{F}\,G(t)$.

To preserve correctness, the schema for δ-rules has to be changed as well: the Skolem terms introduced now contain the free variables occurring in the δ-formula.[3]

$$\frac{\gamma}{\gamma_1(y)} \qquad\qquad \frac{\delta}{\delta_1(f(x_1,\ldots,x_n))}$$

y is a free variable.

f is a new Skolem function symbol, and $x_1,\ldots,x_n$ are the free variables in δ.

Definition 2. A free variable tableau T with branches $B_1,\ldots,B_k$ is **closed** iff there are

1. a substitution σ,
2. formulæ $\phi_i, \psi_i \in B_i$ $(1 \leq i \leq k)$

such that $\phi_i\sigma$ and $\psi_i\sigma$ are complementary.

2.4 Semantic Tableaux with Universal Formulae

Free variable semantic tableaux can be further improved by using the concept of universal formulæ [3, 7]: γ-formulæ—in particular equalities—have often to be used multiply in a tableau proof, with different instantiations for the free variables they contain. A typical example is the associativity axiom $(\forall x)(\forall y)(\forall z)((x \cdot y) \cdot z \approx x \cdot (y \cdot z))$ from group theory. Usually, it has to be applied several times with different substitutions for x, y and z to prove even very simple theorems from group theory. Therefore, in semantic tableaux the γ-rule has to be applied repeatedly to generate several instances of the axiom each with different free variables substituted for x, y and z. This, however, has disadvantages: Firstly, if the number of γ-rule applications is limited (this is often done in implementations, see Sec. 6), the limit has to be chosen high enough to generate a sufficient number of instances. Secondly, since it is difficult to decide how many instances will be needed, unnecessary formulæ will be added to the tableaux enlarging the search space for a proof.

These problems can at least partly be avoided by recognizing formulæ (including equalities) that are "universal", i.e. that can be used multiply in a tableau proof with different substitutions for the variables they contain (without affecting soundness):

Definition 3. Let ϕ be a signed formula on some tableau branch B and F_ϕ the "unsigned version" of ϕ, i.e., if $\phi = \mathsf{T}\, G$ for some G then $F_\phi = G$, else if $\phi = \mathsf{F}\, G$ then $F_\phi = \neg G$.

ϕ is **universal** with respect to the variable x iff the following holds for every normal model $\mathcal{M}$ and every grounding substitution σ:

$$\text{If } \mathcal{M} \models B\sigma, \text{ then } \mathcal{M} \models ((\forall x)F_\phi))\sigma \ .$$

A **method Υ for recognizing universal formulæ** assigns to a tableau branch B and a signed formula ϕ a set $\Upsilon(B,\phi)$ of variables such that: if $x \in \Upsilon(B,\phi)$ then $\phi \in B$ and ϕ is universal w.r.t. x.

Since the satisfiability problem of first-order logic can be reduced to the problem of recognizing universal formulæ,[4] it is—in general—undecidable whether a formula is universal. However, an important class of universal formulæ can be recognized easily (and the method is easy to implement):

[3] The δ-rules we use are more liberal than that proposed in [10] (there, all the free variables on the *branch* have to be included in the Skolem term). Using the liberalized rules (they have been proven to be sound in [13]) makes it easier to close branches. There is an even more liberalized version, where the same Skolem function symbol is used to Skolemize δ-formulæ that are identical up to variable renaming [8].

[4] A sentence G is unsatisfiable iff $\mathsf{T}\ (G \wedge p(x) \wedge \neg p(a))$ is universal w.r.t. x on a tableau branch consisting only of that signed formula (x does not occur in G).

Theorem 4. *Υ is a method for recognizing universal formulæ where $\Upsilon(B,\phi)$ contains exactly the variables x such that the formula $\phi \in B$ has been added to B*

1. *by applying a γ-rule, and x is the free variable that has been introduced; or*
2. *by applying an α-, δ- or γ-rule to a formula that is universal w.r.t. x.*

A formula $G(x)$ is recognized as being universal w.r.t. x by this method, if new instances $G(x'), G(x''), \ldots$ can be added to the branch without affecting other branches or generating new ones.

Once formulæ are recognized as being universal, this knowledge can be taken advantage of to make it easier to find a substitution σ that closes a tableau: instantiations of variables w.r.t. which the formulæ used to close a branch are universal are not taken into consideration. Soundness is not affected if this notion of closed tableau is used [3]; completeness is not affected anyway.

Definition 5. Let Υ be a method for recognizing universal formulæ. A free variable tableau T with branches $B_1, \ldots, B_k$ is **closed** iff there are

1. a grounding substitution σ,
2. for $1 \le i \le k$
   (a) formulæ $\phi_i, \psi_i \in B_i$,
   (b) grounding substitutions σ_i

such that

1. $\phi_i\sigma_i$ and $\psi_i\sigma_i$ are complementary;
2. σ_i differs from σ only on the set U of variables with respect to which both ϕ_i and ψ_i are universal, i.e. $\sigma_{i|(\mathcal{V}\setminus U)} = \sigma_{|(\mathcal{V}\setminus U)}$ where $U = \Upsilon(B_i,\phi_i) \cap \Upsilon(B_i,\psi_i)$.

One can take advantage of the universality of *equalities* in a similar way: If an equality $\mathsf{T}\,(t \approx s)$ is universal with respect to a variable x, the variable x does not have to be instantiated to apply the equality (see the following sections).

The theorems and methods presented in the sequel do not depend on the method for recognizing universal formulæ actually used.

2.5 Other Versions of Semantic Tableaux

All results and methods described from here on can as well be adapted to other versions of semantic tableaux for first-order logic (with equality); all of them can be considered a variant or special case of the versions described above: calculi with unsigned formulæ, with different δ-rules [10, 8], with methods for restricting the search space such as links or ordering restrictions, with lemma generation, etc. Difficulties can arise with adaptations to tableau calculi for other logics, in particular if the notion of equality itself is affected (e.g. if sorted terms are used).

3 The Two Paradigms for Adding Equality

Constructing a tableau for a formula ϕ can be considered a search for a model of ϕ. Therefore, methods have to be employed for:

1. adding formulæ that are valid in a model $\mathcal{M}$ of ϕ to the tableau branch that corresponds to $\mathcal{M}$ (i.e., that is a partial definition of $\mathcal{M}$);
2. recognizing formulæ or sets of formulæ that are unsatisfiable; these formulæ close branches on which they occur.

In *canonical* models, on the one hand, additional formulæ are valid and, thus, have to be added to a branch: If $P(a)$ and $a \approx b$ are true in a canonical model $\mathcal{M}$, then $\mathcal{M}$ is a model of $P(b)$, too. On the other hand, there are additional inconsistencies: $\neg(a \approx a)$ is false in all *canonical* models.

Accordingly, there are two possibilities for handling equality in semantic tableaux: The first and more straightforward method is to define additional tableau rules for expanding branches by all the formulæ valid in the canonical models they (partially) define; then very simple additional closure rules can be used. The second possibility is to use a more complicated notion of closed branch: Different versions of *E-unification* (depending on the version of semantic tableaux) are used to decide whether a tableau branch is unsatisfiable in canonical models and, therefore, closed. Then, no additional expansion rules are needed.

4 Handling Equality Using Additional Expansion Rules

4.1 Additional Expansion Rules for Ground Tableaux

The first methods for adding equality to the ground version of semantic tableaux have been developed in the 1960s [14, 19]. R. C. Jeffrey introduced the following additional tableau expansion rules: If a branch B contains a signed formula $\phi[t]$ and an equality $\mathsf{T}\,(t \approx s)$ or $\mathsf{T}\,(s \approx t)$, that can be "applied" to $\phi[t]$ to derive a formula $\phi[s]$,[5] then $\phi[s]$ may be added to B.

$$\frac{\begin{array}{c}\mathsf{T}\,(t \approx s)\\ \phi[t]\end{array}}{\phi[s]} \qquad\qquad \frac{\begin{array}{c}\mathsf{T}\,(s \approx t)\\ \phi[t]\end{array}}{\phi[s]}$$

In addition to the new expansion rules there is a new closure rule: A branch is closed if it contains a formula of the form $\mathsf{F}\,(t \approx t)$:

Definition 6. A tableau T with branches $B_1, \dots, B_k$ is **closed** iff for $1 \le i \le k$

- there are complementary formulæ ϕ_i, ψ_i on B_i, or
- there is an inequality $\mathsf{F}\,(t_i \approx t_i)$ on B_i

Example 1. Figure 1 shows an example for the application of Jeffrey's additional tableau expansion and closure rules. Note, that it is not possible to derive $\mathsf{T}\,P(b,b)$ in a single step.

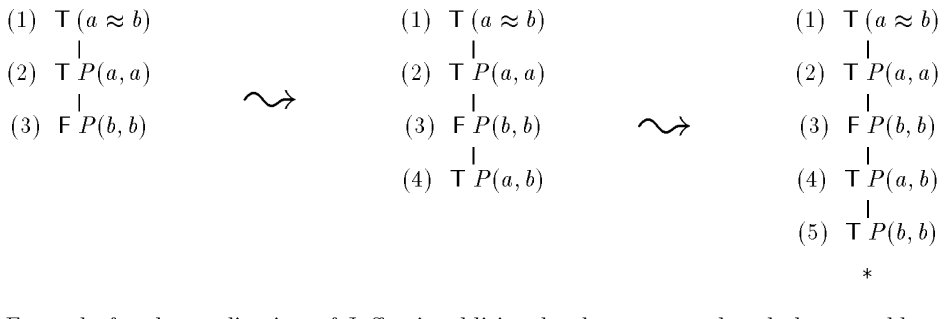


Fig. 1. Example for the application of Jeffrey's additional rules to expand and close a tableau branch. The equality (1) is applied to the formula (2) to derive formula (4) and to (4) to derive (5). The branch is closed by the complementary formulæ (2) and (5).

Besides being based on the ground version of tableaux, the new expansion rules have a major disadvantage: they are symmetrical and their application is completely unrestricted. This leads

[5] $\phi[s]$ is constructed by substituting one occurrence of t in $\phi[t]$ by s.

to much indeterminism and a huge search space; an enormous number of irrelevant formulæ can be added. If, for example, a branch B contains the formulæ $\mathsf{T}\,(f(a) \approx a)$ and $\mathsf{T}\,P(a)$ then all the formulæ $\mathsf{T}\,P(f(a))$, $\mathsf{T}\,P(f(f(a)))$, $\mathsf{T}\,P(f(f(f(a))))$, $\ldots$ can be added to B.

The rules presented by S. Reeves [20] generate a smaller search space, because only atomic formulæ that potentially close a branch are used for expansion. The rules are based upon the following fact: If in a canonical model $\mathcal{M}$ the inequality $\mathsf{F}\,(f(a_1, \ldots, a_n) \approx f(b_1, \ldots, b_n))$ is valid or the formulæ $\mathsf{T}\,P(a_1, \ldots, a_n)$ and $\mathsf{F}\,P(b_1, \ldots, b_n)$, then at least one of the inequalities $\mathsf{F}\,(a_1 \approx b_1), \ldots, \mathsf{F}\,(a_n \approx b_n)$ has to be valid in $\mathcal{M}$. With these rules, too, it is sufficient to use the notion of closed tableau from Def. 6.

$$\frac{\begin{array}{c}\mathsf{T}\,P(a_1, \ldots, a_n)\\ \mathsf{F}\,P(b_1, \ldots, b_n)\end{array}}{\mathsf{F}\,(a_1 \approx b_1 \wedge \ldots \wedge a_n \approx b_n)} \qquad \frac{\mathsf{F}\,(f(a_1, \ldots, a_n) \approx f(b_1, \ldots, b_n))}{\mathsf{F}\,(a_1 \approx b_1 \wedge \ldots \wedge a_n \approx b_n)}$$

Example 2. Figure 2 shows the application of Reeves's additional rules to expand and close the same tableau branch as in Figure 1.

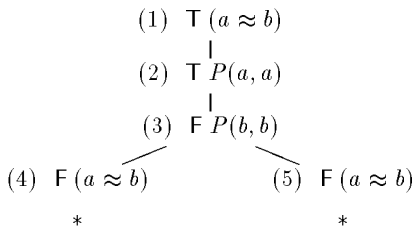


Fig. 2. Example for the application of Reeves's additional expansion rule. It is applied to the atomic formulæ (2) and (3) to generate the inequalities (4) and (5). The branches are closed by the formulæ (1) and (4) and (1) and (5) respectively.

Reeves's approach, however, can lead to heavy branching, because the new expansion rules can as well be applied to pairs of equalities and inequalities. In the worst case the number of branches generated is exponential in the number of equalities on the branch.

Example 3. Figure 3 shows an example for the heavy branching that can occur using Reeves's expansion rules.

4.2 Additional Expansion Rules for Free Variable Tableaux

M. Fitting [10] extended Jeffrey's approach and adapted it to free variable tableaux. The main difference is that equality rule applications may require substituting free variables. These substitutions can be obtained in a similar way as those needed to close a branch in free variable tableaux: If an equality $\mathsf{T}\,(t \approx s)$ is to be applied to a formula $\phi(t')$, the application of a most general unifier (MGU) μ of t and t' to the tableau is sufficient to derive $(\phi[s])\mu$. However, the unifier μ has to be applied not only to the formulæ involved but to the whole tableau:

$$\frac{\begin{array}{c}\mathsf{T}\,(t \approx s)\\ \phi[t']\end{array}}{(\phi[s])\mu} \qquad \frac{\begin{array}{c}\mathsf{T}\,(s \approx t)\\ \phi[t']\end{array}}{(\phi[s])\mu}$$

μ is a MGU of t and t' that is applied to the whole tableau. μ is a MGU of t and t' that is applied to the whole tableau.

Unification can as well become necessary if a branch is to be closed using equality; for example, a branch that contains the inequality $\mathsf{F}\,(f(x) \approx f(a))$ is closed if the substitution $\{x/a\}$ is applied (to the whole tableau):

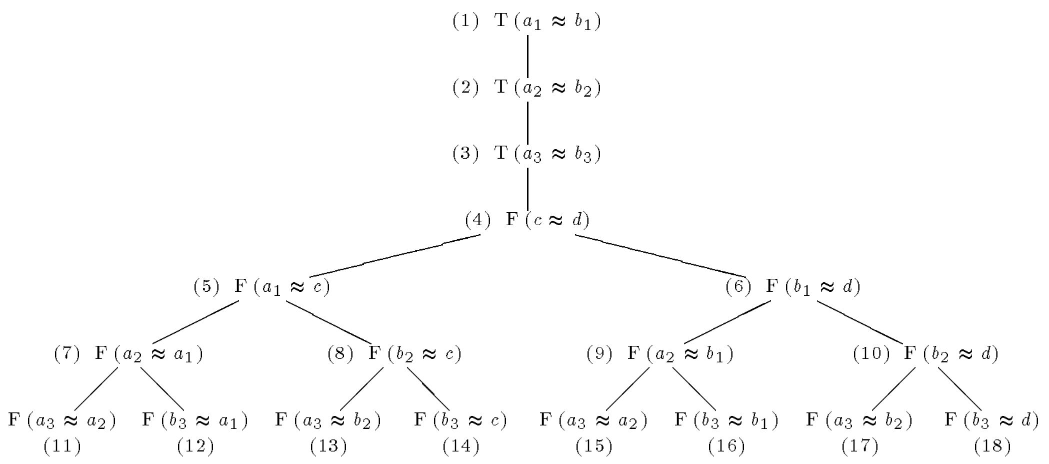

(1) T $(a_1 \approx b_1)$
(2) T $(a_2 \approx b_2)$
(3) T $(a_3 \approx b_3)$
(4) F $(c \approx d)$
(5) F $(a_1 \approx c)$
(6) F $(b_1 \approx d)$
(7) F $(a_2 \approx a_1)$
(8) F $(b_2 \approx c)$
(9) F $(a_2 \approx b_1)$
(10) F $(b_2 \approx d)$
F $(a_3 \approx a_2)$ (11)
F $(b_3 \approx a_1)$ (12)
F $(a_3 \approx b_2)$ (13)
F $(b_3 \approx c)$ (14)
F $(a_3 \approx a_2)$ (15)
F $(b_3 \approx b_1)$ (16)
F $(a_3 \approx b_2)$ (17)
F $(b_3 \approx d)$ (18)

Fig. 3. The disadvantage of Reeves's method: The three equalities (1), (2), (3) and the inequality (4) result in eight branches; and even more branches could be added to the tableau. By applying the expansion rule, the inequalities (5) and (6) are derived from (1) and (4), (7) and (8) from (2) and (5), (9) and (10) from (2) and (6), (11) and (12) from (3) and (7), (13) and (14) from (3) and (8), (15) and (16) from (3) and (9), (17) and (18) from (3) and (10).

Definition 7. A free variable tableau T with branches $B_1, \ldots, B_k$ is **closed** iff there are

1. a grounding substitution σ,
2. for $1 \leq i \leq k$
   (a) signed formulæ $\phi_i, \psi_i \in B_i$ such that $\phi_i\sigma$ and $\psi_i\sigma$ are complementary, or
   (b) an inequality $\mathsf{F}\,(t_i \approx t'_i) \in B_i$ such that $t_i\sigma = t'_i\sigma$.

Example 4. Figure 4 shows a free variable tableau for the set of formulæ

$$\begin{array}{cc}
(1) & (\forall x)((g(x) \approx f(x)) \vee \neg(x \approx a)) \\
(2) & (\forall x)(g(f(x)) \approx x) \\
(3) & b \approx c \\
(4) & P(g(g(a)), b) \\
(5) & \neg P(a, c)
\end{array}$$

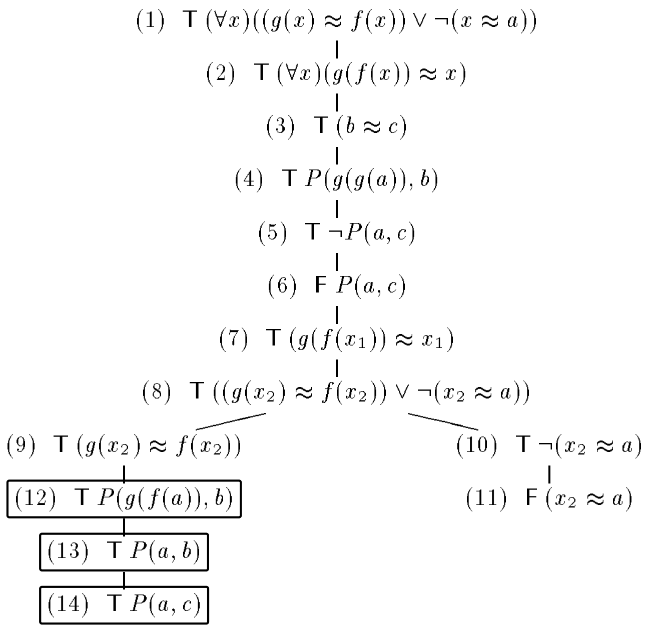
The framed formulæ have been added using Fitting's additional tableau rules. The tableau is closed with the substitution $\{x_1/a,\ x_2/a\}$ (Def. 7).

The example demonstrates a difficulty involved in using additional expansion rules: If Equality (9) is applied to (4) in the wrong way, i.e., if the formula (12') $\mathsf{T}\ P(f(g(a)), b)$ is derived instead of (12) $\mathsf{T}\ P(g(f(a)), b)$, then the term $g(a)$ is substituted for x_2 and the tableau cannot be closed. Either a new instance of (8) has to be generated by applying the γ-rule to (1), or backtracking has to be initiated.

4.3 Additional Rules for Tableaux with Universal Formulæ

Fitting's method can easily be extended to free variable tableau *with universal formulæ*. When equalities are used to derive new formulæ, universality of both the equality $\mathsf{T}\,(t \approx s)$ (resp. $\mathsf{T}\,(s \approx t)$) and the formula $\phi[t']$ it is applied to has to be taken into consideration. The difference to the additional equality expansion rules from Section 4.2 is, that instead of the MGU μ of t and t' only its restriction μ' to those variables is applied w.r.t. which $\mathsf{T}\,(t \approx s)$ (resp. $\mathsf{T}\,(s \approx t)$) and $\phi[t']$ are *not* universal, i.e., $\mu' = \mu_{|(\mathcal{V} \setminus U)}$ where $U = \Upsilon(B, \mathsf{T}\,(t \approx s)) \cap \Upsilon(B, \phi[t'])$.

When branches are closed, the universality of formulæ has to be taken into consideration as well. The following notion of closed tableau is a combination of that given in Definitions 5 and 7:



(1) $\mathsf{T}\,(\forall x)((g(x) \approx f(x)) \vee \neg(x \approx a))$
(2) $\mathsf{T}\,(\forall x)(g(f(x)) \approx x)$
(3) $\mathsf{T}\,(b \approx c)$
(4) $\mathsf{T}\,P(g(g(a)), b)$
(5) $\mathsf{T}\,\neg P(a, c)$
(6) $\mathsf{F}\,P(a, c)$
(7) $\mathsf{T}\,(g(f(x_1)) \approx x_1)$
(8) $\mathsf{T}\,((g(x_2) \approx f(x_2)) \vee \neg(x_2 \approx a))$
(9) $\mathsf{T}\,(g(x_2) \approx f(x_2))$
(10) $\mathsf{T}\,\neg(x_2 \approx a)$
(12) $\mathsf{T}\,P(g(f(a)), b)$
(11) $\mathsf{F}\,(x_2 \approx a)$
(13) $\mathsf{T}\,P(a, b)$
(14) $\mathsf{T}\,P(a, c)$

Fig. 4. A free variable tableau for the formulæ from Example 4 (Formulæ (1) to (5)). By applying the standard free variable tableau rules, Formula (6) is derived from (5), (7) from (2), (8) from (1), (9) and (10) from (8), and (11) from (9); The framed formulæ are added to the left branch by applying Fitting's additional expansion rules for handling equality: Formula (12) is derived by applying Equality (9) to (4) (the substitution $\{x_2/a\}$ has to be applied), Formula (13) by applying (7) to (12) (the substitution $\{x_1/a\}$ has to be applied), and formula (14) by applying (3) to (13). Formulæ (14) and (6) close the left branch. The right branch is closed by Formula (11). The whole tableau is closed (Def. 7) using the substitution $\{x_1/a,\, x_2/a\}$. The substitution of free variables is not shown in the figure.

Definition 8. Let Υ be a method for recognizing universal formulæ. A free variable tableau T with branches $B_1, \ldots, B_k$ is **closed** iff there are

1. a grounding substitution σ,
2. for $1 \leq i \leq k$
   (a) formulæ $\phi_i, \psi_i \in B_i$ or $\mathsf{F}\,(t_i \approx t'_i) \in B_i$,
   (b) grounding substitutions σ_i

such that

1. $\phi_i\sigma_i$ and $\psi_i\sigma_i$ are complementary or $t_i\sigma_i = t'_i\sigma_i$;
2. σ_i differs from σ only on variables with respect to which both ϕ_i and ψ_i are universal, i.e. $\sigma_{i|(\mathcal{V}\setminus U)} = \sigma_{|(\mathcal{V}\setminus U)}$ where $U = \Upsilon(B_i, \phi_i) \cap \Upsilon(B_i, \psi_i)$.

Example 5. If the method from Theorem 4 for recognizing universal formulæ is used, the tableau in Figure 4 (without the framed formulæ) is closed using the substitution $\{x_2/a\}$. x_1 does not have to be instantiated, because Equality (7) is recognized to be universal w.r.t. to x_1.

5 Handling Equality Using *E*-Unification

5.1 Universal, Rigid and Mixed *E*-Unification

There are different versions of E-unification that are important for handling equality in semantic tableaux: the classical "universal" E-unification [22], "rigid" E-unification [11] and "mixed"

E-unification which is a combination of both [5]. The different versions of E-unification allow equalities to be used differently in an equational proof: in the universal case the equalities can be applied several times with different instantiations for the variables they contain; in the rigid case they can be applied more than once but with only one instantiation for each variable; in the mixed case there are both types of variables.

Which type of E-unification problems has to be solved to decide whether a tableau is closed, depends on the version of semantic tableaux that equality is to be added to: Universal E-unification can only be used in the ground case. For handling equality in free variable tableaux, rigid E-unification problems have to be solved.[6] For tableaux with universal formulæ both versions have to be combined [5]; then, equalities contain two types of variables, namely universal and rigid ones. To distinguish them syntactically, equalities $(\forall x_1)\cdots(\forall x_n)(l \approx r)$ are used that can be explicitly quantified w.r.t. variables they contain.

Definition 9. A **mixed *E*-unification problem** $\langle E, s, t\rangle$ consists of a finite set E of equalities of the form $(\forall x_1)\cdots(\forall x_n)(l \approx r)$ and terms s and t.

A substitution σ is a **solution** to the problem, iff $E\sigma \models (s\sigma \approx t\sigma)$ where the free variables in $E\sigma$ are "held rigid", i.e. treated as constants.[7]

The major differences between this definition and that generally given in the literature on (universal) E-unification are:

- The equalities in E are *explicitly* quantified (instead of considering all the variables in E to be *implicitly* universally quantified).
- In contrast to the "normal" notion of logical consequence, free variables in $E\sigma$ are "held rigid".
- The substitution σ is applied not only to the terms s und t but as well to the set E.

We call a mixed E-unification problem $\langle E, s, t\rangle$ **purely universal** if there are no free variables in E, and **purely rigid** if there are no bound variables in E;[8] Table 1 shows some simple examples.

Table 1. Examples for the different versions of E-unification. Note, that the fourth problem has no solution, since the free variable x would have to instantiated with both a and b. Contrary to that, the empty substitution id is a solution to the third problem, where the variable x is universally quantified.

E	s	t	MGUs	Type
$\{f(x) \approx x\}$	$f(a)$	a	$\{x/a\}$	purely rigid
$\{f(a) \approx a\}$	$f(x)$	a	$\{x/a\}$	ground
$\{(\forall x)(f(x) \approx x)\}$	$g(f(a), f(b))$	$g(a, b)$	id	purely universal
$\{f(x) \approx x\}$	$g(f(a), f(b))$	$g(a, b)$	—	purely rigid
$\{(\forall x)(f(x, y) \approx f(y, x))\}$	$f(a, b)$	$f(b, a)$	$\{y/b\}$	mixed

For handling equality in semantic tableaux, several E-unification problems—one for each branch—have to be solved simultaneously:

Definition 10. A finite set $\{\langle E_1, s_1, t_1\rangle, \ldots, \langle E_n, s_n, t_n\rangle\}$ $(n \geq 1)$ of mixed E-unification problems is called **simultaneous** E-unification problem.

A substitution σ is a solution to the simultaneous problem iff it is a solution to every component $\langle E_k, s_k, t_k\rangle$ $(1 \leq k \leq n)$.

[6] Using universal E-unification corresponds to *not* applying the substitutions necessary for applying equalities (Sec. 4.2) to the whole tableau—correctness would be destroyed.

[7] Here, $\models$ is the logical consequence in *normal* models, i.e., $(s\sigma \approx t\sigma)$ is true in all *normal* models of $E\sigma$.

[8] If E is ground, the problem is both purely rigid and purely universal.

Since purely universal E-unification is already undecidable, (simultaneous) *mixed* E-unification is—in general—undecidable as well. Is is, however, possible to enumerate a complete set of MGUs. (Simultaneous) *purely rigid* E-unification is decidable [11, 12].[9]

5.2 Closing Semantic Tableaux with E-Unification

The common problem of all the methods described in Section 4, that are based on additional tableau expansion rules, is that there are virtually no restrictions on the application of equalities. Because of their symmetry this leads to a very large search space; even very simple problems cannot be solved in reasonable time.

It is difficult to employ more elaborate and efficient methods for handling equality in semantic tableaux, such as completion-based approaches, because it is nearly impossible to transform these methods into (sufficiently) simple tableau expansion rules.[10]

Contrary to that, arbitrary algorithms can be used, if the handling of equality is reduced to solving E-unification problems. Then, no additional expansion rules are needed; once the E-unification problems have been extracted, the tableau does not have to be taken into consideration any more. In [7] it has been shown that methods based on E-unification are much more efficient than that based on additional rules—even if the comparatively inefficient algorithm from [7] is used to solve E-unification problems.

The equality theory defined by a tableau branch B consists of the equalities on B; they are (explicitly) quantified w.r.t. to the variables w.r.t. which they can be recognized as being universal:

Definition 11. Let B be a tableau branch and Υ a method for recognizing universal formulæ (Def. 3). Then the **set $\boldsymbol{E(B)}$ of equalities** consists of the equalities $(\forall x_1)\cdots(\forall x_n)(s \approx t)$ such that

1. $\mathsf{T}\,(s \approx t)$ is formula on B,
2. $\{x_1, \ldots, x_n\} = \Upsilon(B, \mathsf{T}\,(s \approx t))$.

Example 6. As an example we use the tableau from Figure 4. Its left branch is denoted by B_1 and its right branch by B_2. If the method for recognizing universal formulæ from Theorem 4 is used, $E(B_2)$ contains the equalities $b \approx c$ and $(\forall x)(g(f(x)) \approx x)$. $E(B_1)$ contains in addition the equality $g(x_2) \approx f(x_2)$.

Definition 12. Atomic formulæ $\mathsf{T}\,P(s_1, \ldots, s_n)$ and $\mathsf{F}\,P(t_1, \ldots, t_n)$ with the same predicate symbol and complementary signs are called a pair of **potentially closing atoms**.

Definition 13. Let B be a tableau branch and Υ a method for recognizing universal formulæ. Then the **set $\boldsymbol{\mathcal{P}(B)}$ of unification problems** consists exactly of the sets of term pairs:

$$\{\langle s_1\sigma, t_1\sigma\rangle, \ldots, \langle s_n\sigma, t_n\sigma\rangle\}$$

for each pair $\mathsf{T}\,P(s_1, \ldots, s_n)$, $\mathsf{F}\,P(t_1, \ldots, t_n)$ of potentially closing atoms on B such that $P \not\approx \approx$, and

$$\{\langle s\sigma, t\sigma\rangle\}$$

for each inequality $\mathsf{F}\,(s \approx t)$ on B.
The substitution σ renames all the variables

$$x_1, \ldots, x_m \in \Upsilon(B, \mathsf{T}\,P(s_1, \ldots, s_n)) \cap \Upsilon(B, \mathsf{F}\,P(t_1, \ldots, t_n)) \enspace ,$$

[9] Purely rigid E-unification is NP-complete [11]; simultaneous purely rigid E-unification is NEXPTIME-complete [12].

[10] R. J. Browne [9] describes a completion-based method for handling equality, that uses additional expansion rules. It is, however, only applicable to the ground version of tableaux and cannot be extended to free variable tableaux.

or

$$x_1, \ldots, x_m \in \Upsilon(A, \mathsf{F}\,(s \approx t)) \ ,$$

respectively, i.e., $\sigma = \{x_1/y_1, \ldots, x_m/y_m\}$ and $y_1, \ldots, y_m$ are new variables.

If one of the problems in the set $\mathcal{P}(B)$ of unification problems of a branch B has a solution σ (w.r.t. the equalities $E(B)$), $B\sigma$ is unsatisfiable in canonical models; therefore the branch B is closed under the substitution σ. The pair of potentially closing atoms corresponding to the solved unification problem has been proven to actually be complementary; or the the corresponding inequality has been proven to be inconsistent (provided the unifier is applied to the tableau).

Example 7. If, again, B_1 denotes the left and B_2 the right branch of the tableau in Figure 4 (without the framed formulæ), and Υ is the method from Theorem 4 for recognizing universal formulæ, then both $\mathcal{P}(B_1)$ and $\mathcal{P}(B_2)$ contain the set $\{\langle g(g(a)), a\rangle,\ \langle b, c\rangle\}$. $\mathcal{P}(B_2)$ contains in addition the set $\{\langle x_2, a\rangle\}$.

The following is a formal definition of the simultaneous mixed E-unification problems that have to be solved to close a tableau. The soundness of this notion of closed tableau has been proven in [3]:

Definition 14. A tableau T with branches $B_1, \ldots, B_k$ is **closed** iff in the sets of unification problems $\mathcal{P}(B_i)$ $(1 \leq i \leq k)$ (Def. 13) there are elements $\{\langle s_{i1}, t_{i1}\rangle, \ldots, \langle s_{in_i}, t_{in_i}\rangle\} \in \mathcal{P}(B_i)$ such that there is a solution to the simultaneous mixed E-unification problem (Def. 10)

$$\begin{array}{c} \{\ \langle E(B_1), s_{11}, t_{11}\rangle,\ \ldots,\ \langle E(B_1), s_{1n_1}, t_{1n_1}\rangle, \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \langle E(B_k), s_{k1}, t_{k1}\rangle,\ \ldots,\ \langle E(B_k), s_{kn_k}, t_{kn_k}\rangle\ \} \end{array}$$

Example 8. If the method from Theorem 4 for recognizing universal formulæ is used, the tableau from Figure 4 is closed w.r.t. Def. 14 (the framed formulæ not taken into consideration): The substitution $\sigma = \{x_2/a\}$ is a solution to the simultaneous mixed E-unification problem $\{\langle\langle E(B_1), g(g(a)), a\rangle,\ \langle E(B_1), b, c\rangle,\ E(B_2), x_2, a\rangle\}$.

Actually, it is not necessary to split pairs $\mathsf{T}\,P(s_1, \ldots, s_n)$ and $\mathsf{F}\,P(t_1, \ldots, t_n)$ of potentially complementary atoms into n term pairs $\langle s_1, t_1\rangle, \ldots, \langle s_n, t_n\rangle$ that have to be unified. Instead the problem $\langle P(s_1, \ldots, s_n), P(t_1, \ldots, t_n)\rangle$ could be used. That, however, is inefficient, because the n simpler problems can be solved independently.

Since purely rigid E-unification is decidable [12], it is decidable whether a given free variable tableau *without universal formulæ* is closed (Def. 14). However, if a tableau is not closed it may, nevertheless, be unsatisfiable (and, thus, be expandable to a closed tableau). It is undecidable whether a tableau with universal formulæ is closed, because simultaneous mixed E-unification is undecidable.

5.3 Solving E-Unification Problems

To solve the E-unification problems that are extracted from tableaux (Def. 14), arbitrary algorithms can be used.

The problems extracted from ground tableaux consist solely of *ground* terms. There are very efficient methods for solving these ground E-unification problems, that are based on computing the equivalence classes of the terms to be unified (w.r.t. to the relation defined by the equalities on the branch) [21, 16].

Algorithms based on computing equivalence classes can be used as well to solve rigid and mixed E-unification problems, i.e., to add equality to free variable tableaux with universal formulæ [7].

However, for solving non-ground problems, it is much better to use completion-based methods. Unfortunately, the Unfailing Knuth-Bendix-Algorithm [15, 1] with narrowing [17], that is generally considered to be the best algorithm for universal E-unification and has often been implemented, cannot be used to solve rigid or mixed problems. Completion-based methods for rigid E-unification have been described in [11, 12]; these, however, are non-deterministic and unsuited for implementation, since the "guess" that is part of the algorithm is highly complex.

Recently, deterministic completion-based methods have been introduced, both for purely rigid E-unification [2] and for mixed E-unification [4, 5].[11] Besides being completion-based, there are several reasons why these methods are well suited for adding equality to free variable semantic tableaux: Firstly, the terms to be unified do not become part of the completion (in contrary to the method in [11]); this is important because the E-unification problems in Definition 14 that share the same set of equalities can, thus, be solved using a single completion. Secondly, simultaneous E-unification problems are solved by searching for common specializations of solutions to its components; this is of advantage, because the different E-unification problems consist of the same components.

6 Efficient Implementation

Using algorithms based on solving *simultaneous* rigid E-unification problems, all branches of a tableau are closed simultaneously. Another possibility, that is easier to implement, is to close the branches one after the other; the first substitution found to close a branch B_i is applied to the whole tableau. If, later on, it is not possible to close a branch B_j $(j > i)$, backtracking is initiated to compute further closing substitutions for B_i. In [3] it has been proven that this method leads to a correct and complete calculus, provided the search for further substitutions closing a branch is limited. To preserve completeness the limit has to be incremented if no proof is found.

It is, however, more efficient to handle all (or several) branches of a tableau in parallel. Then, the information contained in the branches can be used simultaneously. Backtracking can be avoided and the search space can be restricted. For example, it is often possible to recognize branches for which only one closing substitution exists; these substitutions can be applied immediately, before other branches are closed.

If a completion-based method is used to solve the E-unification problems that are extracted from a tableau, it is advantageous to combine the completion process and the expansion of the tableau. Thus, if a β-rule is applied, the (partial) completion that has been computed up to that point can be shared by the new subbranches and has only to be computed once.

On the other hand, all described methods are much easier to implement if the handling of equality is separated from the expansion of tableaux [7, 10]: First, the (classical) tableau expansion rules are applied until the tableau is exhausted (observing a limit for γ-rule applications). Then, the E-unification problems are extracted and solved (resp. the equality expansion rules are applied).

7 Conclusion

Although it is easier to add equality to the ground version, to prove even simple theorems, free variable tableaux have to be used. These are sufficient as long as each branch is relatively easy to close, even if there is a large number of branches. If however, complex equational theories have to be applied to close the branches, universal formulæ and elaborate methods for solving E-unification problems have to be used.

[11] The method described in [4, 5] has been implemented as part of the tableau-based theorem prover $_3T^AP$ [6]. The implementation is written in Quintus Prolog. Besides the possibility to prove theorems from predicate logic with equality, the E-unification module can be used "stand alone" to solve simultaneous mixed E-unification problems. Upon request, the source code is available from the author.

After the handling of equality has been reduced to solving rigid and mixed E-unification problems, the search for efficient methods has not come to an end. However, the difficulties that have to be overcome have been identified.

Completion-based methods for solving mixed E-unification have just recently been developed; first experiments show promising results. Further investigations are necessary to combine these methods and semantic tableau in a more efficient way.

References

1. L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In H. Aït-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2*, chapter 1. Academic Press, 1989.
2. G. Becher and U. Petermann. Rigid E-unification by completion and rigid paramodulation. Report 93-22, LAIAC, University of Caen, 1993.
3. B. Beckert. Konzeption und Implementierung von Gleichheit für einen tableau-basierten Theorembeweiser. IKBS Report 208, Science Center, IBM Germany, 1991.
4. B. Beckert. Ein vervollständigungsbasiertes Verfahren zur Behandlung von Gleichheit im Tableaukalkül mit freien Variablen. Diploma thesis, Fakultät für Informatik, Universität Karlsruhe, 1993.
5. B. Beckert. A completion based method for mixed universal and rigid E-unification. In *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy*, LNCS. Springer, 1994.
6. B. Beckert, S. Gerberding, R. Hähnle, and W. Kernig. The tableau-based theorem prover $_3T^AP$ for multiple-valued logics. In *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, LNCS 607. Springer, 1992.
7. B. Beckert and R. Hähnle. An improved method for adding equality to free variable semantic tableaux. In D. Kapur, editor, *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, LNCS 607, pages 507–521. Springer, 1992.
8. B. Beckert, R. Hähnle, and P. H. Schmitt. The even more liberalized δ-rule in free variable semantic tableaux. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic*, LNCS 713, pages 108–119. Springer, August 1993.
9. R. J. Browne. Ground term rewriting in semantic tableaux systems for first-order logic with equality. Technical Report UMIACS-TR-88-44, College Park, MD, 1988.
10. M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, 1990.
11. J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the ACM*, 39(2):377–429, April 1992.
12. J. Goubault. Simultaneous rigid E-unifiability is NEXPTIME-complete. Technical report, Bull Corporate Research Center, 1993.
13. R. Hähnle and P. H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning*, 1993. To appear.
14. R. C. Jeffrey. *Formal Logic: Its Scope and Limits.* McGraw Hill, 1967.
15. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras*, pages 263–297. Pergamon Press, 1970.
16. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, April 1980.
17. W. Nutt, P. Réty, and G. Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7(3/4):295–318, 1989.
18. U. Petermann. Completeness of the pool calculus with an open built-in theory. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic*, LNCS 713, pages 264–277. Springer, August 1993. Extended version as Preprint Nr. 24, Naturwissenschaftlich-Theoretisches Zentrum, Universität Leipzig, 1994.
19. R. J. Popplestone. Beth-tree methods in automatic theorem proving. In *Machine Intelligence*, volume 1, pages 31–46. Oliver and Boyd, 1967.
20. S. Reeves. Adding equality to semantic tableau. *Journal of Automated Reasoning*, 3:225–246, 1987.
21. R. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21(7):583–585, July 1978.
22. J. Siekmann. Universal unification. *Journal of Symbolic Computation*, 7(3/4):207–274, 1989.
23. R. Smullyan. *First-Order Logic.* Springer, 1968.