Semantic Tableaux with Equality

Bernhard Beckert
Institute for Logic, Complexity and Deduction Systems
University of Karlsruhe
Am Fasanengarten 5, 76128 Karlsruhe, Germany
beckert@ira.uka.de, http://i12.www.ira.uka.de/~beckert

November 16, 1995

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.

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; 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 being paramodulation [24] and RUE-resolution [12]—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 [17, 22]. 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 [23, 13, 7] and the connection method [14, 21].

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. Finally, in Section 6 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 [13]).

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 formulae.

Since in the tableau proofs it will be necessary to introduce Skolem terms, we extend our first-order language $\mathcal{L}$ to a language $\mathcal{L}_{\text{Sko}}$ 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, unifier, most general unifier (MGU), sentence, model, logical consequence (denoted by $\models$), valuation, satisfiability and tautology. 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. $\sigma(x_i) = t_i$ $(1 \leq i \leq n)$. 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. $\mathcal{L}_{\text{Sko}}$) 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.

The definitions of tautology, satisfiability and logical consequence are different for first-order logic with and without equality. To avoid confusion, we use the following notions:

Definition 1 A formula ϕ (a set of formulae Φ) is *E-satisfiable* if there is a normal model satisfying ϕ (resp. Φ), else it is *E-unsatisfiable*.

A sentence ϕ is an *E-tautology* if it is satisfied by all normal models.

A formula ϕ is an *E-consequence* of a set of formulae Ψ, denoted $\Psi \models_{\approx} \phi$, if ϕ is satisfied by all normal models that satisfy Ψ.

In addition to the normal (weak) consequence and E-consequence relations we use the notion of strong (E-)consequence:

α	α_1	α_2
$\phi \wedge \psi$	ϕ	ψ
$\neg(\phi \vee \psi)$	$\neg\phi$	$\neg\psi$
$\neg(\phi \supset \psi)$	ϕ	$\neg\psi$
$\neg\neg\phi$	ϕ	ϕ

β	β_1	β_2
$\phi \vee \psi$	ϕ	ψ
$\neg(\phi \wedge \psi)$	$\neg\phi$	$\neg\psi$
$\phi \supset \psi$	$\neg\phi$	ψ

γ	$\gamma_1(t)$
$(\forall x)\phi(x)$	$\phi(t)$
$\neg(\exists x)\phi(x)$	$\neg\phi(t)$

δ	$\delta_1(t)$
$\neg(\forall x)\phi(x)$	$\neg\phi(t)$
$(\exists x)\phi(x)$	$\phi(t)$

Table 1: Formula classes.

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

where t is any ground term.

where c is a new (Skolem) constant.

Table 2: Tableau expansion rule schemata (ground version).

Definition 2 Let Ψ be a set of formulae and ϕ a formula; ϕ is a *strong consequence* (*strong E-consequence*) of Ψ, denoted $\Psi \models^{\circ} \phi$ (resp. $\Psi \models^{\circ}_{\approx} \phi$), if for all (normal) models $\langle \mathcal{D}, \mathcal{I} \rangle$ and for all variable assignments ν: If $val_{I,\nu}(\psi) = true$ for all $\psi \in \Psi$, then $val_{I,\beta}(\phi) = true$.

Example 3 $p(x) \models (\forall x)p(x)$, but $p(x) \not\models^{\circ} (\forall x)p(x)$.

2.2 The Ground Version of Semantic Tableaux

The ground version [27], 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.

For our purposes it suffices to visualize tableaux as binary trees. The branches of a tableau are implicitly disjunctively connected, and the formulae on a branch are implicitly conjunctively connected. We identify a branch with the set of formulae it contains.

To every formula that is not a literal exactly one tableau expansion rule can be applied. Following Smullyan, formulae are divided into four classes with corresponding rules, namely α (conjunctive propositional), β (disjunctive propositional), γ (universal quantification) and δ (existential quantification). The formula classes are summarized in Table 1, and Table 2 shows the expansion rule schemata for the ground version of tableaux.

A tableau T is expanded by choosing a branch B of T and a formula $\phi \in B$ and extending B by as many subbranches as the rule corresponding to ϕ has extensions; the new subbranches contain the formulae in the extensions.

To prove a sentence ϕ to be a tautology, we apply the expansion rules starting from the initial tableau that consists of the single node $\neg\phi$. A proof is found, if all branches of the constructed tableau are closed (contain complementary formulae):

$\dfrac{\alpha}{\alpha_1 \\ \alpha_2}$	$\dfrac{\beta}{\beta_1 \mid \beta_2}$	$\dfrac{\gamma}{\gamma_1(y)}$	$\dfrac{\delta}{\delta_1(f(x_1,\ldots,x_n))}$
		y is a free variable.	f is a new Skolem function symbol, and $x_1,\ldots,x_n$ are the free variables in δ.

Table 3: Tableau expansion rule schemata (free variable version).

Theorem 4 (Ground Version) A first-order sentence ϕ is a tautology if and only if there is a sequence $T_0,\ldots,T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by applying one of the tableau expansion rules from Table 2.
3. All branches of T_n are closed, i.e., contain complementary formulae ϕ and $\neg\phi$.

The construction of a closed tableau is a highly indeterministic process, because at each step one is free to choose a branch B of the tableau and a formula $\phi \in B$ for expansion. If ϕ is a γ-formula, in addition, a term has to be chosen that is substituted for the bound variable.

There are two ways for resolving the indeterminism (actual implementations usually employ a combination of both): (1) fair strategies can be used such that, for example, each formula will finally be used to expand each branch on which it occurs. (2) Backtracking can be used; if a branch cannot be closed (observing a limit on its length), other possibilities are tried; for example, other terms are used in γ-rule applications. If no proof is found, the limit has to be increased (iterative deepening).

2.3 Free Variable Semantic Tableaux

Using free variable quantifier rules [13] is crucial for efficient implementation—even more if equality has to be handled. They reduce the number of possibilities to proceed at each step in the construction of a tableau proof and thus the size of the search space. 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.

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 (the free variable rule schemata are shown in Table 3).[1]

It is often difficult to find a substitution σ that instantiates the variables in a tableau T such that *all* branches of T are closed when σ is applied. The problem of finding a closing substitution σ can be simplified (as is usually done in practice) by

[1] The δ-rules we use are more liberal than that proposed in [13] (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 [16]) makes it easier to close branches. Even more liberalized δ-rules have been investigated in [8, 1].

closing the branches of T one after the other: if a substitution is found that closes a single branch B, it is applied (to the whole tableau) to close B before other branches are handled. Thus, there is a second possibility besides expansion to derive a new tableau from an old one: applying a closure rule, i.e., applying a (most general) closing substitution to the tableau.

Theorem 5 (Free Variable Version) A first-order sentence ϕ is a tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by
   (a) applying one of the tableau expansion rules from Table 3; or
   (b) applying a substitution σ that closes a branch B of T_{i-1}, i.e., there are formulae $\phi, \neg\psi \in B$ such that σ is an MGU of ϕ and ψ ($\phi\sigma$ and $\neg\psi\sigma$ are complementary).[2]
   The branch $B\sigma$ of $T_i = T_{i-1}\sigma$ is marked as being closed in T_i.
3. All branches of T_n are marked as being closed.

Contrary to the ground version, the free variable γ-rule is deterministic; but one is still free (1) to choose a branch B of the tableau, (2) to expand or to close B, and to choose (3a) a formula $\phi \in B$ for expansion or (3b) a most general substitution that closes B.

2.4 Semantic Tableaux with Universal Formulae

Free variable semantic tableaux can be further improved by using the concept of universal formulae [7]: γ-formulae—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 (as is often done in implementations), 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 formulae will be added to the tableaux enlarging the search space for a proof.

These problems can at least partly be avoided by recognizing formulae (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):

[2] The restriction to *most general* unifiers is not necessary, but leads to a much smaller search space.

Definition 6 A formula ϕ on a tableau branch B is *universal on B with respect to the variable x* if $B \models^{\circ} (\forall x)\phi$ (resp. $B \models^{\circ}_{\approx} (\forall x)\phi$ in tableaux for first-order logic with equality).

A *method Υ for recognizing universal formulae* assigns to a tableau branch B and a 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 formulae,[3] it is—in general—undecidable whether a formula is universal. However, an important class of universal formulae can be recognized easily (and the method is easy to implement):

Theorem 7 Υ is a method for recognizing universal formulae 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 $\phi(x) \in B$ is recognized as being universal w.r.t. x by this method, if new instances $\phi(x'), \phi(x''), \ldots$ can be added to the branch without affecting other branches or generating new ones.

Once formulae 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 branch (or a whole tableau): instantiations of variables w.r.t. which the formulae used to close a branch are universal are not taken into consideration. Soundness is not affected if this notion of closed tableau is used (completeness is not affected anyway).

The closure rule is the only difference between tableaux with universal formulae and classical free variable tableaux. Neither the expansion rules nor the way tableau are constructed have to be changed:[4]

Theorem 8 (Universal Formula Version) Let Υ be a method for recognizing universal formulae. A first-order sentence ϕ is a tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by
   (a) applying one of the tableau expansion rules from Table 3; or
   (b) applying a substitution σ that closes a branch B of T_{i-1}, i.e., there are formulae $\phi, \neg\psi \in B$ such that ϕ and ψ are unifiable with an MGU σ' and $\sigma = \sigma'_{|(\mathcal{V}\setminus U)}$ is the restriction of σ' to the set $U = \Upsilon(B, \phi) \cap \Upsilon(B, \psi)$ of variables with respect to which both ϕ and ψ are universal.
   The branch $B\sigma$ of $T_i = T_{i-1}\sigma$ is marked as being closed in T_i.[5]
3. All branches of T_n are marked as being closed.

[3] A sentence ϕ is unsatisfiable iff the formula $\phi \wedge p(x) \wedge \neg p(a)$ is universal w.r.t. x on a tableau branch consisting only of that formula (x does not occur in ϕ).

[4] The additional expansion rules for handling equality are different, too; see Section 4.

[5] As in classical free variable tableaux, the restriction to *most general* unifiers is not necessary, but restricts the search space.

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 signed formulae, with different δ-rules [13, 8, 1], 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 formulae 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 formulae or sets of formulae that are complementary; these formulae close branches on which they occur.

For handling equality additional expansion and/or closure rules are needed, because in *canonical* models, on the one hand, additional formulae are valid and, thus, have to be added to a branch; on the other hand, there are additional inconsistencies.

Example 9 If a tableau branch B contains the formulae $p(a)$ and $a \approx b$, then $p(b)$ is true in all canonical models of B.

$\neg(a \approx a)$ is false in all canonical models; therefore all branches containing this formula can be closed.

Equality reasoning is an instance of *theory reasoning*, where problems from a certain domain (or theory) that is defined by a set of axioms, are handled separately by a *background reasoner*. The background reasoner applies special purpose techniques and makes use of knowledge about the theory.

Virtually all methods for handling equality can be regarded a special case of the general method for theory reasoning in semantic tableaux:[6] A set Φ of formulae (a *key*) that is a subset of the formulae on a tableau branch B is transferred to the background reasoner, which then searches for a *refuter* of Φ. A refuter consists of a substitution σ and a set $R = R\sigma$ of formulae (the *residue*) such that Φ and the negation $\overline{R} = \{\neg\phi : \phi \in R\}$ of R are complementary, i.e., all instances of $\Phi\sigma \cup \overline{R}$ are unsatisfiable in models of the theory (E-unsatisfiable in the case of equality reasoning).

From another point of view, the residue contains formulae that are valid in all models of $\Phi\sigma$, in particular in all models (partially) defined by the instance $B\sigma \supset \Phi\sigma$ of the branch B.

There are two ways a refuters can be made use of, depending on whether the residue R is empty or not:

[6] The only exception are methods based on pre-processing the input formulae, e.g. STE-modification [9]; they do not require the tableau calculus to be adopted to equality reasoning.

Partial theory reasoning ($R = \{\rho_1, \ldots, \rho_n\}$, $n \geq 1$): Using the background reasoner constitutes a tableau expansion rule: the branch B (resp. its instance $B\sigma$) may be extended by n subbranches where $B_i = B \cup \{\rho_i\}$ $(1 \leq i \leq n)$; the substitution σ has to be applied to the whole tableau.

Total theory reasoning ($R = \emptyset$): Using the background reasoner constitutes a closure rule: the branch B is closed under σ.

The general condition for expansion and closure rules for free variable tableaux to be sound is, that a tableau branch B (or all its subbranches) can only be closed if all ground instances of B are unsatisfiable. Thus, expansion rules have to preserve the satisfiability of ground instances of the branch B being expanded. The (partial theory reasoning) expansion rule is sound, because if a ground instance $B\sigma\tau$ is satisfiable then at least one of the subbranches $(B\sigma \cup \{\rho_i\})\tau$ is satisfiable; else $(B\sigma \cup \overline{R})\tau$ and, thus, $(\Phi\sigma \cup \overline{R})\tau$ were satisfiable—in contradiction to $\Phi\sigma$ and $\overline{R}$ being complementary. The (total theory reasoning) closure rule is sound, because $R = \emptyset$ implies that all instances of $\Phi\sigma \subset B\sigma$ are unsatisfiable.

To preserve completeness of the calculus the background reasoner has to be able to compute a sufficiently complete subset (though not all) of the possible refuters. There is a trade off between partial and total reasoning: On the one hand, if only empty residues are allowed, more complex methods have to be employed to find refuters; the background reasoner has to make more complex deductions that, using partial reasoning, could be divided into several expansion steps followed by a simple closure step. On the other hand, the restriction to total theory reasoning leads to a much smaller search space, because there are less refuters for each key and the search is more goal-directed.

Accordingly, there are two possibilities for handling equality in semantic tableaux: The first and more straightforward method—corresponding to partial theory reasoning—is to define additional tableau expansion rules; then very simple additional closure rules can be used. The second possibility—corresponding to total theory reasoning—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 (instances of) tableau branches are unsatisfiable in canonical models and, therefore, closed. Then, no additional expansion rules are needed.

4 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 [17, 22]. R. C. Jeffrey introduced the additional tableau expansion rule shown in Table 4 (i.e., a partial reasoning method): If a branch B contains a formula $\phi[t]$ and an equality $t \approx s$ or $s \approx t$, that can be "applied" to $\phi[t]$ to derive a formula $\phi[s]$,[7] then $\phi[s]$ may be added to B.

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 $\neg(t \approx t)$.

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

$$\frac{t \approx s \quad \phi[t]}{\phi[s]} \qquad \frac{s \approx t \quad \phi[t]}{\phi[s]}$$

Table 4: Jeffrey's additional expansion rules.

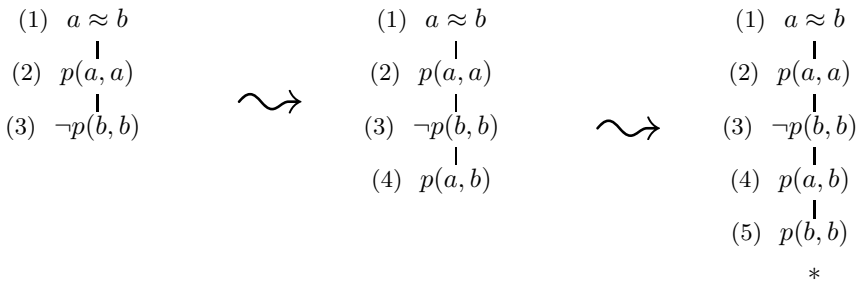

Figure 1: The application of Jeffrey's additional rules to expand and close a tableau branch (Example 11).

Theorem 10 (Jeffrey) A first-order sentence ϕ is an E-tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by applying one of the tableau expansion rules from Table 2 or one of the equality expansion rules from Table 4.
3. All branches of T_n are closed, i.e., contain an inequality of the form $\neg(t \approx t)$ or complementary formulae ϕ and $\neg\phi$.

Example 11 Figure 1 shows an example for the application of Jeffrey's additional tableau expansion and closure rules: 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 formulae (3) and (5). Note, that it is not possible to derive $p(b, b)$ in a single step.

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

The rules presented by S. Reeves [23] (see Table 5) generate a smaller search space. They are the tableau counterpart of RUE-resolution [12] and are more goal-directed than Jeffrey's expansion rules: only atomic formulae that potentially close a branch are used for expansion. Like RUE-resolution, the rules are based upon the following fact: If in a canonical model $\mathcal{M}$ the inequality $\neg(f(a_1, \ldots, a_n) \approx f(b_1, \ldots, b_n))$ is valid or the formulae $p(a_1, \ldots, a_n)$ and $\neg p(b_1, \ldots, b_n)$, then at least one of the inequalities $\neg(a_1 \approx b_1)$, $\ldots$, $\neg(a_n \approx b_n)$ has to be valid in $\mathcal{M}$. With these expansion rules it is sufficient to use the same closure rules as in Theorem 10:

$$\frac{p(a_1,\ldots,a_n) \quad \neg p(b_1,\ldots,b_n)}{\neg(a_1 \approx b_1) \mid \cdots \mid \neg(a_n \approx b_n)} \qquad \frac{\neg(f(a_1,\ldots,a_n) \approx f(b_1,\ldots,b_n))}{\neg(a_1 \approx b_1) \mid \cdots \mid \neg(a_n \approx b_n)}$$

Table 5: Reeves's additional expansion rules.

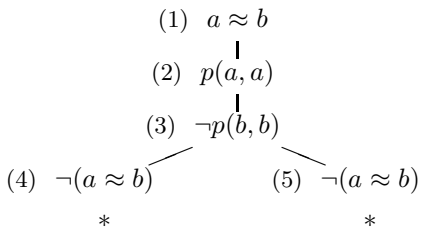

Figure 2: The application of Reeves's additional expansion rule (Example 13).

Theorem 12 (Reeves) A first-order sentence ϕ is an E-tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by applying one of the tableau expansion rules from Table 2 or one of the equality expansion rules from Table 5.
3. All branches of T_n are closed, i.e., contain an inequality of the form $\neg(t \approx t)$ or complementary formulae ϕ and $\neg\phi$.

Example 13 Figure 2 shows the application of Reeves's rule to expand and close the same tableau branch as in Figure 1: It is applied to the atomic formulae (2) and (3) to generate the inequalities (4) and (5). The branches are closed by the formulae (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 14 Figure 3 shows an example for the heavy branching that can occur using Reeves's expansion rules. 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).

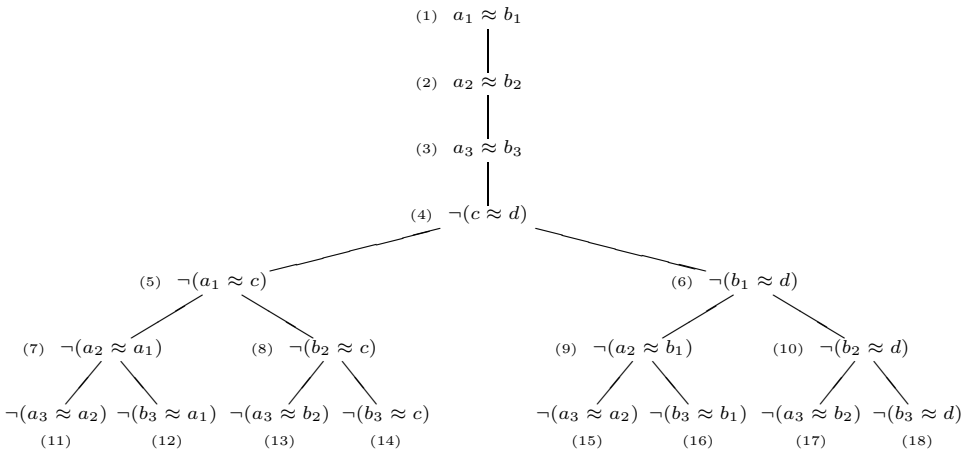

Figure 3: The disadvantage of Reeves's method (Example 14).

$\dfrac{t \approx s \quad \phi[t']}{(\phi[s])\mu}$	$\dfrac{s \approx t \quad \phi[t']}{(\phi[s])\mu}$
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.

Table 6: Fitting's additional expansion rules for free variable tableaux.

4.2 Additional Expansion Rules for Free Variable Tableaux

M. Fitting [13] extended Jeffrey's approach and adapted it to free variable tableaux. The main difference is that equality rule applications may require instantiating 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 $t \approx s$ is to be applied to a formula $\phi[t']$, the application of an 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 formulae involved but to the whole tableau (see Table 6).

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

Theorem 15 (Fitting) A first-order sentence ϕ is an E-tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by
   (a) applying one of the tableau expansion rules from Table 3 or one of the equality expansion rules from Table 6; or

(b) applying a substitution σ that closes a branch B of T_{i-1}, i.e., there is an inequality $\neg(t \approx t') \in B$ such that σ is an MGU of t and t' or there are formulae $\phi, \neg\psi \in B$ such that σ is an MGU of ϕ and ψ.
The branch $B\sigma$ of $T_i = T_{i-1}\sigma$ is marked as being closed in T_i.[8]

3. All branches of T_n are marked as being closed.

Example 16 Figure 4 shows a free variable tableau that proves the following set of formulae to be inconsistent:

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

By applying the standard free variable tableau rules, formula (6) is derived from (2), (7) from (1), and (8) and (9) from (7). The framed formulae are added to the left branch by applying Fitting's additional expansion rules for handling equality: Formula (10) is derived by applying equality (8) to (4) (the substitution $\{x_2/a\}$ has to be applied), formula (11) is derived by applying (6) to (10) (the substitution $\{x_1/a\}$ has to be applied), and formula (12) is derived by applying (3) to (11). Formulae (12) and (5) close the left branch. The right branch is closed by the inequality (9) (the substitution $\{x_2/a\}$ has already been applied).

The example demonstrates a difficulty involved in using additional expansion rules: If equality (8) is applied to (4) in the wrong way, i.e., if the formula (10') $p(f(g(a)), b)$ is derived instead of (10) $p(g(f(a)), b)$, then the term $g(a)$ is substituted for x_2 and the tableau cannot be closed. Either a new instance of (7), (8) and (9) 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 Formulae

Fitting's method can easily be extended to free variable tableau *with universal formulae*. When equalities are used to derive new formulae, universality of both the equality $t \approx s$ (resp. $s \approx t$) 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 $t \approx s$ (resp. $s \approx t$) and $\phi[t']$ are *not* universal, i.e., $\mu' = \mu_{|(\mathcal{V} \setminus U)}$ where $U = \Upsilon(B, (t \approx s)) \cap \Upsilon(B, \phi[t'])$. If an equality is universal with respect to a variable x, the variable x does not have to be instantiated to apply the equality. When branches are closed, the universality of formulae has to be taken into consideration as well:

Theorem 17 Let Υ be a method for recognizing universal formulae. A first-order sentence ϕ is an E-tautology if and only if there is a sequence $T_0, \ldots, T_n$ of tableaux ($n \geq 0$) such that

[8] Again, the restriction to *most general* unifiers is not necessary, but restricts the search space.

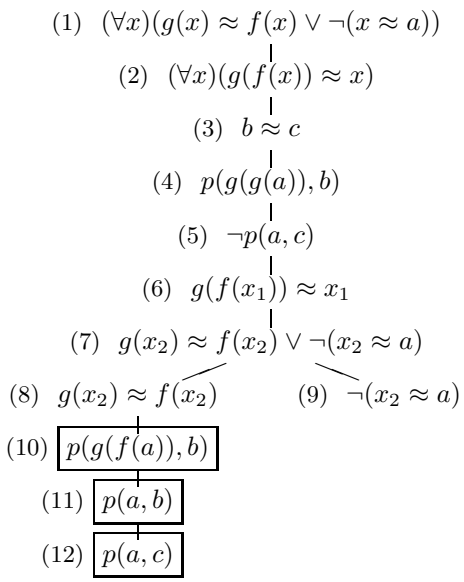

Figure 4: A free variable tableau using Fitting's expansion rules (Example 16).

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by
   (a) applying one of the tableau expansion rules from Table 3 to a branch B of T_{i-1} or one of the equality expansion rules from Table 6 where not the MGU μ of t and t' is applied to the tableau but only its restriction $\mu' = \mu_{|(\mathcal{V} \setminus U)}$ where $U = \Upsilon(B, (t \approx s)) \cap \Upsilon(B, \phi[t'])$; or
   (b) applying a substitution σ that closes a branch B of T_{i-1}, i.e., there is an inequality $\neg(t \approx t') \in B$ such that t and t' are unifiable with an MGU σ' or there are formulae $\phi, \neg\psi \in B$ such that ϕ and ψ are unifiable with an MGU σ', and $\sigma = \sigma'_{|(\mathcal{V} \setminus U)}$ where $U = \Upsilon(B, \phi) \cap \Upsilon(B, \psi)$.
   The branch $B\sigma$ of $T_i = T_{i-1}\sigma$ is marked as being closed in T_i.
3. All branches of T_n are marked as being closed.

Example 18 If the method from Theorem 7 for recognizing universal formulae is used, the tableau in Figure 4 (without the framed formulae) is closed using the substitution $\{x_2/a\}$. x_1 does not have to be instantiated, because equality (6) is recognized to be universal w.r.t. to x_1.

5 Handling Equality Using *E*-Unification

5.1 Motivation

The common problem of all the partial reasoning 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 transform more elaborate and efficient methods for handling equality, such as completion-based approaches, into (sufficiently) simple tableau expansion rules. A set of rules that implement a completion procedure for the ground version of tableaux has been described in [10]; however, these expansion rules are quite complicated, and the method cannot be extended to free variable tableaux.

If total equality reasoning is used, i.e., if no equality expansion rules are added, then the problem of finding substitutions that close a tableau branch is equivalent to solving E-unification problems. E-unification is much more difficult than just checking whether the terms of an inequality $\neg(t \approx t')$ are unifiable; on the other hand, no equality expansion steps are necessary, and arbitrary algorithms can be used to compute E-unifiers. Such algorithms do not have to take the tableau into concern.

In [7] it has been shown that an implementation based on E-unification is much more efficient than one based on additional rules—even if the comparatively inefficient algorithm from [7] is used to solve E-unification problems.

Depending on the version of semantic tableaux that equality is to be added to, different types of E-unification problems have to solved. These are introduced in the following section, and in Section 5.3 the extraction of E-unification problems from tableau branches is described.

5.2 Universal, Rigid and Mixed *E*-Unification

The different versions of E-unification that are important for handling equality in semantic tableaux are: the classical "universal" E-unification [26], "rigid" E-unification [14] and "mixed" E-unification which is a combination of both [5]. The different versions 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.[9] For tableaux with universal formulae both versions have to be combined [4]; 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.

[9] 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.

E	s	t	MGUs	Type
$\{f(x) \approx x\}$	$f(x)$	a	$\{x/a\}$	purely rigid
$\{f(a) \approx a\}$	$f(a)$	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

Table 7: Examples for the different versions of E-unification.

Definition 19 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

$$E\sigma \models^{\circ}_{\approx} (s\sigma \approx t\sigma)\ .^{10}$$

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).
- The strong consequence relation $\models^{\circ}_{\approx}$ is used in the definition instead of $\models_{\approx}$.
- 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 (if E is ground, the problem is both purely rigid and purely universal).

Example 20 Table 7 shows some simple examples for the different versions of E-unification. 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.

Since purely universal E-unification is already undecidable, *mixed* E-unification is—in general—undecidable as well. Is is, however, possible to enumerate a complete set of MGUs. *Purely rigid* E-unification is decidable [14].[11]

5.3 Closing Tableau Branches Using *E*-Unification

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:

[10] This is equivalent to $E\sigma \models_{\approx} (s\sigma \approx t\sigma)$ where the free variables in $E\sigma$ are "held rigid", i.e. treated as constants.

[11] Deciding purely rigid E-unification, i.e., deciding whether a solution to a given problem exists, is NP-complete [14].

Definition 21 Let B be a tableau branch and Υ a method for recognizing universal formulae (Def. 6). Then the *set $E(B)$ of equalities on B* consists of the equalities $(\forall x_1)\cdots(\forall x_n)(s \approx t)$ such that

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

Example 22 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 formulae from Theorem 7 is used, both $E(B_1)$ and $E(B_2)$ contain the equalities $b \approx c$ and $(\forall x)(g(f(x)) \approx x)$. $E(B_1)$ contains in addition the equality $g(x_2) \approx f(x_2)$.

Substitutions that close a branch B can be computed by extracting the set $P(B)$ of rigid E-unification problems from B according to the following definition and solving the problems in $P(B)$. Problems are only extracted from literals, which is sufficient to preserve completeness.

Definition 23 Let B be a tableau branch and Υ a method for recognizing universal formulae. Then the *set $P(B)$ of E-unification problems on B* consists exactly of:

$$\langle E(B), \langle s_1\sigma, \ldots, s_n\sigma\rangle, \langle t_1\sigma, \ldots, t_n\sigma\rangle\rangle$$

for each pair of literals $p(s_1, \ldots, s_n)$, $\neg p(t_1, \ldots, t_n)$ on B such that $p \neq \approx$, and

$$\langle E(B), s\sigma, t\sigma\rangle$$

for each inequality $\neg(s \approx t)$ on B.
The substitution $\sigma = \{x_1/y_1, \ldots, x_m/y_m\}$ (where $y_1, \ldots, y_m$ are new variables) renames all the variables w.r.t. which the literals are universal, i.e.,

$$\{x_1, \ldots, x_m\} = \Upsilon(B, p(s_1, \ldots, s_n)) \cap \Upsilon(B, \neg p(t_1, \ldots, t_n)) \ ,$$

resp.

$$\{x_1, \ldots, x_m\} = \Upsilon(A, \neg(s \approx t)) \ .$$

If one of the problems in the set $P(B)$ of unification problems of a branch B has a solution σ, all instances of $B\sigma$ are unsatisfiable in canonical models; therefore the branch B is closed under the substitution σ. The pair of literals corresponding to the solved unification problem has been proven to actually be complementary; or the corresponding inequality has been proven to be inconsistent (provided the unifier is applied to the tableau).

Example 24 If, again, B_1 denotes the left and B_2 the right branch of the tableau in Figure 4 (without the framed formulae), and Υ is the method from Theorem 7 for recognizing universal formulae, then both $P(B_1)$ and $P(B_2)$ contain the problem $\langle E(B_i), \langle g(g(a)), b\rangle, \langle a, c\rangle\rangle$. $P(B_2)$ contains in addition the problem $\langle E(B_2), x_2, a\rangle$.

Besides the version of E-unification problems that have to be solved, the way equality is handled is nearly the same for the different versions of semantic tableau. Therefore it is sufficient to only formulate one general soundness and completeness theorem:

Theorem 25 Let Υ be a method for recognizing universal formulae. A first-order sentence ϕ is an E-tautology if and only if there is a sequence $T_0, \ldots, T_n$ of (ground, free variable, or universal formula) tableaux ($n \geq 0$) such that

1. T_0 consists of the single node $\neg\phi$.
2. For $1 \leq i \leq n$ the tableau T_i is constructed from T_{i-1} by
   (a) applying a tableau expansion rule (a rule from Table 2 for the ground version of tableau, and a rule from Table 3 for the free variable version with and without universal formulae); or
   (b) applying a substitution σ that is a solution to one of the E-unification problems in $P(B)$ for a branch B of T_{i-1} and that, thus, closes B.
   The branch $B\sigma$ of $T_i = T_{i-1}\sigma$ is marked as being closed in T_i.
3. All branches of T_n are marked as being closed.

As before, it is sufficient to only apply most general closing substitutions. Additional restrictions are possible. For example, completeness is preserved if only such closing substitutions are applied that are minimal (w.r.t. a term ordering) in their equivalence class, where substitutions are equivalent if they are identical modulo the set of equalities $\{(l \approx r) : (l \approx r) \in E(B)$ for all branches B of the tableau$\}$.

Example 26 If the method from Theorem 7 for recognizing universal formulae is used, both branches of the tableau from Figure 4 can be closed (the framed formulae not taken into consideration): The substitution $\sigma = \{x_2/a\}$ is a solution both to the mixed E-unification problems

$$\begin{aligned} \langle E(B_1),\, \langle g(g(a)), b\rangle,\, \langle a, c\rangle\rangle &\in P(B_1) \\ \langle E(B_2),\, x_2,\, a\rangle &\in P(B_2) \end{aligned}$$

When σ is applied to close one of the branches, the other branch is then closed under the empty substitution.

Since purely rigid E-unification is decidable, it is decidable whether a given free variable tableau branch *without universal formulae* can be closed. However, if a branch cannot be closed it may, nevertheless, be unsatisfiable (and, thus, be expandable to a closed branch). It is undecidable whether a tableau branch with universal formulae is closed, because mixed E-unification is undecidable.

Instead of closing one branch after the other, one could search for a substitution that closes all branches simultaneously. However, this is much more difficult than closing a single branch. To find a substitution that closes *all* branches of a given tableau, *simultaneous* E-unification problems have to be solved:

Definition 27 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 $\langle E_k, s_k, t_k\rangle$ ($1 \leq k \leq n$).

Simultaneous mixed E-unification is much more difficult than non-simultaneous mixed E-unification; moreover, simultaneous rigid E-unification is already undecidable [11] (the non-simultaneous rigid problem is NP-complete [14]).

5.4 Solving *E*-Unification Problems

To solve the E-unification problems that are extracted from tableaux (Def. 23), 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) [25, 19].

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 formulae.

However, for solving non-ground problems, it is much better to use completion-based methods. Unfortunately, the Unfailing Knuth-Bendix-Algorithm [18, 2] with narrowing [20], 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. A completion-based method for rigid E-unification has been described in [14]; it is, however, indeterministic 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 [3] and for mixed E-unification [5]. 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 (contrary to the method described in [14]); this is important because the E-unification problems extracted from a branch B, that share the same set of equalities, can thus be solved using a single completion of $E(B)$. Secondly, E-unification problems of the form $\langle E, \langle s_1, \ldots, s_n\rangle, \langle t_1, \ldots, t_n\rangle\rangle$ can be solved by searching for common specializations of solutions to its components $\langle E, s_i, t_i\rangle$.

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.

Example 28 The following example requires only very little non-equality reasoning. A powerful equality handling technique is needed to find a closed tableau, and the universal formula version of tableaux has to be used to restrict the search space: If Γ consists of the axioms[12]

$$(\forall x)(i(tr, x) \approx x)$$
$$(\forall x)(\forall y)(\forall z)(i(i(x, y), i(i(y, z), i(x, z))) \approx tr)$$
$$(\forall x)(\forall y)(i(i(x, y), y) \approx i(i(y, x), x))$$

then

$$\Gamma \models_{\approx} (\forall x)(\forall y)(\forall z)(\exists w)(i(x, w) \approx tr \wedge w \approx i(y, i(z, y))) \; .$$

To prove this, the tableau shown in Figure 5 has to be closed. Formulae (2) to (4) are derived from the axioms by γ-rule applications. (5) is derived from the negated theorem (1) by three δ- and one γ-rule application; (6) and (7) are derived from (5).

[12] This is an axiomatization of propositional logic, $i(x, y)$ stands for "x implies y" and tr for "true".

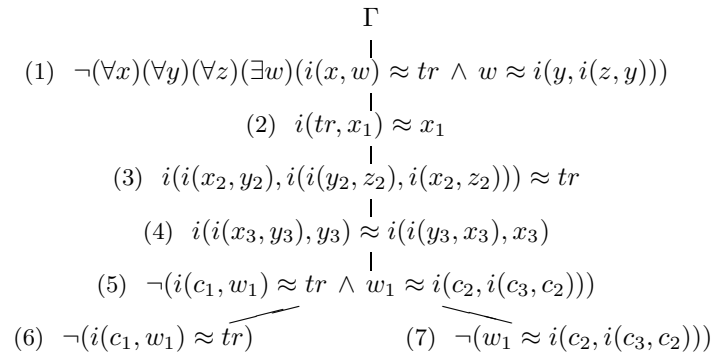

Figure 5: The tableau that has to be closed to prove the theorem from Example 28.

The free variable instances (2), (3) and (4) of the axioms can easily be recognized to be universal w.r.t. the variables they contain, Thus, to close the left branch the E-unification problem

$$P_l = \langle \Gamma,\ i(c_1, w_1),\ tr \rangle$$

has to be solved, and the problem

$$P_r = \langle \Gamma,\ w_1,\ i(c_2, i(c_3, c_2)) \rangle$$

has to be solved to close the right branch.

The search for solutions performed by the tableau-based theorem prover ${}_3\mathcal{TAP}$ [6, 15], that uses the completions-based method from [5] for solving E-unification problems, proceeds as follows: One of the first reduction rules that are deduced from Γ is $(\forall x)(i(x, x) \rightarrow tr)$. Using this rule the solution

$$\sigma = \{w_1/c_1\}$$

to the problem P_l is found and applied to the tableau. Then the Problem $P_r\sigma$ has to be solved to close the right branch; unfortunately no solution exists. Thus, after a futile try to close the right branch, backtracking is initiated. More reduction rules are computed, until finally the rule $(\forall x)(i(x, tr) \rightarrow tr)$ is applied to the problem P_l and the solution

$$\sigma' = \{w_1/tr\}$$

is found. Now, the problem $P_r\sigma'$ has to be solved to close the right branch. It takes the computation of 48 critical pairs to deduce the rule $(\forall x)(\forall y)(i(y, i(x, y)) \rightarrow tr)$ that can be applied to show that the empty substitution is a solution to $P_r\sigma'$, and that therefore the right branch is closed.

It takes ${}_3\mathcal{TAP}$ about ten seconds to find this proof (running on a SUN SPARC workstation); ${}_3\mathcal{TAP}$ reuses the partial completion computed to close a branch during backtracking.

6 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 formulae and elaborate methods for solving E-unification problems have to be used.

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; experiments show promising results. Further investigations are necessary to combine these methods and semantic tableau in a more efficient way.

Acknowledgements

I would like to thank Marcello D'Agostino and three anonymous referees for their constructive criticism and useful comments on earlier versions of this paper.

References

[1] Matthias Baaz and Christian G. Fermüller. Non-elementary speedups between different versions of tableaux. In *Proceedings, 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, St. Goar*, LNCS 918, pages 217–230. Springer, 1995.

[2] Leo Bachmair, Nachum Dershowitz, and David A. 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.

[3] Gerard Becher and Uwe Petermann. Rigid E-unification by completion and rigid paramodulation. Report 93-22, LAIAC, University of Caen, 1993.

[4] Bernhard Beckert. Adding equality to semantic tableaux. In K. Broda, M. D'Agostino, R. Goré, R. Johnson, and S. Reeves, editors, *Proceedings, 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Abingdon*, pages 29–41, Imperial College, London, TR-94/5, 1994.

[5] Bernhard Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France*, LNCS 814, pages 678–692. Springer, 1994.

[6] Bernhard Beckert, Stefan Gerberding, Reiner Hähnle, and Werner Kernig. The tableau-based theorem prover $_3$TAP for multiple-valued logics. In *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, LNCS 607, pages 758–760. Springer, 1992.

[7] Bernhard Beckert and Reiner Hähnle. An improved method for adding equality to free variable semantic tableaux. In Depak Kapur, editor, *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, LNCS 607, pages 507–521. Springer, 1992.

[8] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt. The even more liberalized δ-rule in free variable semantic tableaux. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic*, LNCS 713, pages 108–119. Springer, 1993.

[9] D. Brand. Proving theorems with the modification method. *SIAM Journal on Computing*, 4(4):412–430, 1975.

[10] Randall Jeffrey Browne. Ground term rewriting in semantic tableaux systems for first-order logic with equality. Technical Report UMIACS-TR-88-44, College Park, MD, 1988.

[11] Anatoli Degtyarev and Andrei Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, May 1995. Presented at: *Annual Conference of the European Association for Computer Science Logic (CSL'95), Paderborn.*

[12] Vincent J. Digricoli and Malcolm C. Harrison. Equality-based binary resolution. *Journal of the ACM*, 33(2):253–289, April 1986.

[13] Melvin C. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer, 1990.

[14] Jean H. Gallier, Paliath Narendran, Stan Raatz, and Wayne Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the ACM*, 39(2):377–429, April 1992.

[15] Reiner Hähnle, Bernhard Beckert, and Stefan Gerberding. The many-valued tableau-based theorem prover ${}_3T^AP$. TR 30/94, Universität Karlsruhe, Fakultät für Informatik, November 1994.

[16] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *Journal of Automated Reasoning,*, 13(2):211–222, 1994.

[17] Richard C. Jeffrey. *Formal Logic. Its Scope and Limits.* McGraw-Hill, New York, 1967.

[18] Donald E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras*, pages 263–297. Pergamon Press, Oxford, 1970.

[19] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, April 1980.

[20] Werner Nutt, P. Réty, and Gert Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7(3/4):295–318, 1989.

[21] Uwe Petermann. A framework for integrating equality reasoning into the extension procedure. In D. Basin, R. Hähnle, B. Fronhöfer, J. Posegga, and C. Schwind, editors, *Proceedings, 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Marseille/France*, Saarbrücken, MPI-I-92-213, March 1993. Max-Planck-Institut für Informatik.

[22] R. J. Popplestone. Beth-tree methods in automatic theorem proving. In *Machine Intelligence*, volume 1, pages 31–46. Oliver and Boyd, 1967.

[23] Steve V. Reeves. Adding equality to semantic tableau. *Journal of Automated Reasoning*, 3:225–246, 1987.

[24] J. A. Robinson and L. Wos. Paramodulation and theorem proving in first order theories with equality. In B. Meltzer and Mitchie, editors, *Machine Intelligence*. Edinburgh University Press, 1969.

[25] Robert E. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21(7):583–585, 1978.

[26] Jörg H. Siekmann. Universal unification. *Journal of Symbolic Computation*, 7(3/4):207–274, 1989. Earlier version in *Proceedings, 7th International Conference on Automated Deduction (CADE), Napa/FL*, LNCS 170, Springer, 1984.

[27] Raymond Smullyan. *First-Order Logic*. Springer, 1968.