An Improved Method for Adding Equality to Free Variable Semantic Tableaux*

Bernhard Beckert Reiner Hähnle

Institute for Logic, Complexity and Deduction Systems
University of Karlsruhe, 7500 Karlsruhe, Germany
{beckert,haehnle}@ira.uka.de

Abstract. Tableau–Based theorem provers can be extended to cover many of the nonclassical logics currently used in AI research. For both, classical and nonclassical first–order logic, equality is a crucial feature to increase expressivity of the object language. Unfortunately, all so far existing attempts of adding equality to semantic tableaux have been more or less experimental and turn out to be useless in practice. In the present work we introduce an approach that leads much further and sets the stage for more advanced developments. We identify the problems that stem specifically from choosing semantic tableaux as a framework and state soundness and completeness results for our method.

Introduction

In this paper we present a theoretical basis for, as well as an actual implementation of equality handling in tableau–based theorem provers. We do not claim that it can compete with state–of–the–art equality reasoning systems; it is, however, to our best knowledge, the first equality extension of *semantic tableaux* that can grasp beyond textbook examples. Using tableaux as a logical basis for mechanical theorem proving has three major merits: First, they do not commit one to the usage of normal forms; second, they can be extended to cover many of the nonclassical logics (as has been done for example in [3, 6]) currently used in AI research; and third, counterexamples can be generated for non–tautologies. Moreover, while tableau–based systems may not be the most powerful provers being around, they have shown to be sophisticated enough to be interesting in real applications [9].

For both classical and nonclassical first–order logic equality is a crucial feature to increase expressivity of the object language. Unfortunately, all so far existing attempts of adding equality to semantic tableaux have been more or less experimental and turn out to be useless in practice. In the present work we introduce an approach that leads further and sets the stage for more advanced developments.

We assume that the reader is familiar with semantic tableaux and first–order logic with equality (if not, an excellent introduction can be found in [4]). To enhance the readability of our paper we give a short account of the version of the tableau system used by us in Section 1, together with some technical definitions that will be used later. In Section 2 we carry out a careful analysis of the shortcomings in previous approaches. As a result we can clearly identify the problems that stem specifically from choosing semantic tableaux as a framework. This analysis becomes the basis for our own treatment of equality in Section 3 which avoids the aforementioned drawbacks. We state soundness and completeness results for our method. In Section 4 the method is illustrated with an example. Our approach has been implemented as part of the many–valued theorem proving system $_3T^AP$ [5, 6] and we provide some empirical data from test runs of this implementation. Finally we

*This work has been partly supported by IBM Germany.

explain the limitations of the present work and suggest some directions where future research seems promising. Due to space constraints we give no proofs in this paper. All proofs may be found in [1].

1 Preliminaries

Syntax and Semantics

Let us fix a first–order language $\mathbf{L}$ which is built up from countable sets $\mathbf{R}$ of predicate symbols, $\mathbf{F}$ of function symbols, $\mathbf{C}$ of constant symbols and $\mathbf{Var}$ of object variables in the usual manner.[1] The quantifier symbols are $\forall, \exists$ and the binary logical connectives consist of $\wedge$ (conjunction), $\vee$ (disjunction) and $\supset$ (material implication), the only unary connective is $\neg$ (negation). Furthermore let us assume that $\mathbf{R}$ contains a binary predicate symbol for equality which we denote by $\approx$ such that no confusion with the meta–level equality predicate $=$ can arise. We stress that there is no restriction where equalities can occur in formulæ.
We are using the standard notions of free/bound variable, sentence, model, valuation, satisfiability and tautology.[2] Substitutions are mappings from variables to terms and are extended to formulæas usual. Since we will only be concerned with substitutions that are the identity mapping up to a finite number of variables, we will denote a substitution σ by $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$, where $\{x_1, \ldots, x_n\}$ are the variables that are changed by σ.
A model $\mathbf{M} = \langle \mathbf{D}, \mathbf{I} \rangle$ (with domain $\mathbf{D}$ and interpretation $\mathbf{I}$) is called **normal** iff $\approx^{\mathbf{I}}$ is the identity relation on $\mathbf{D}$. A model is called **canonical** if, moreover, for every $d \in \mathbf{D}$ there is a term t in $\mathbf{L}$ such that $t^{\mathbf{I}} = d$. The following theorem shows that canonical models are analogous to Herbrand models.

Theorem 1.1 *If a set S of sentences is satisfied by a normal model, then there is also a canonical model that satisfies S.*

Since in the tableau proofs it will be necessary to introduce Skolem terms, we extend our first order language to a language $\mathbf{L}^{\mathbf{Sko}}$ by adding countably many constant symbols and function symbols for each arity which do not appear already in $\mathbf{L}$. From now on we are working in $\mathbf{L}^{\mathbf{Sko}}$ and we consider only canonical models.

Semantic Tableaux

Semantic (or analytic) tableaux have been introduced in the 1950s by E. W. Beth and K. J. J. Hintikka, its ancestors being Gentzen systems. R. Smullyan [13] gave a particularly elegant version of tableaux which increased their popularity largely and most tableau systems used today are based on his formulation. Tableau systems are available in two versions, namely signed and unsigned, from which we will be using the former.
For our purposes it is sufficient to visualize a tableau proof as a finite labelled binary tree. The node labels are first–order formulæwhich are prefixed with a sign, i.e. an element from $\{\mathsf{T}, \mathsf{F}\}$. To prove tautologyhood of a formula ϕ we begin with a tree whose single node is labelled by $\mathsf{F}\,\phi$, i.e. we assume that ϕ is false in some model. A tableau proof represents a systematic search for such a model. For every combination of sign/leading connective (resp. sign/leading quantifier) there exists a tableau expansion rule which reflects its semantics. We call a maximal path in a tableau proof tree *branch* and say that a branch is *closed* if it contains a pair of unifiable atomic formulæwith complementary signs. A tableau proof tree represents a proof of the root formula when all branches in the tree can be closed simultaneously

[1] For each arity greater than 0 there are countably many function and predicate symbols.
[2] If in doubt, the reader should consult [4] for the precise definitions.

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

Table 1: Tableau rule schemes for different formula types.

(i.e. using the same unifier); in other words: when every attempt to construct a model that makes the root formula false leads to a contradiction.
Following Smullyan we divide the tableau expansion rules for signed formulæinto four classes: α–rules for propositional formulæof conjunctive type (e.g. $\mathsf{F}\, X \vee Y$), β–rules for propositional formulæof disjunctive type (e.g. $\mathsf{T}\, X \vee Y$), γ–rules for quantified formulæof universal type (e.g. $\mathsf{F}\, (\exists x)\phi(x)$), and finally δ–rules for quantified formulæof existential type (e.g. $\mathsf{T}\, (\exists x)\phi(x)$). The rule patterns are summarized in Table 1. It should be obvious how the α_i and β_i are computed from the semantical definitions. The quantifier rules have traditionally been working with ground terms, i.e. from a universal type formula like $\mathsf{T}\, (\forall x)\phi(x)$ the formula $\mathsf{T}\, \phi(t)$ may be inferred, where t is any ground term; and from an existential type formula like $\mathsf{T}\, (\exists x)\phi(x)$ the formula $\mathsf{T}\, \phi(c)$ may be inferred, where c is a Skolem constant not occurring on the current branch. Since a proof is only found when the right ground terms are "guessed", these rules are a source of much indeterminism, which in turn inevitably leads to expensive backtracking. Recent versions of tableau systems [4] therefore work with free variables that are instantiated *on demand*, i.e. when a branch is closed. We remark that the δ–rule we are using here is more liberal than that used in [4] and has recently been proposed and proved sound by Hähnle and Schmitt [7]. We will see in Section 3 that the availability of a liberal free version of tableaux is crucial for efficient equality handling.
For an example of a proof tree see Section 4. To achieve completeness some additional mechanism for handling equality must be provided. In the next section we review the most important approaches.

2 Analysis of Previous Approaches

Jeffrey's Approach

Jeffrey's approach is a very natural and straightforward way of adding equality to semantic tableaux. It is described in [8]; a summary can be found in [12]. The method is based on semantic tableaux without free variables. Therefore the ground versions of the quantifier rules are being used. As noted above it must be possible to add all the additional formulæ to a branch that are valid in canonical models. For this purpose Jeffrey introduced the following additional tableau expansion rules:
If a branch B has a formula $\phi(t)$ on it and an equality $\mathsf{T}\, (t \approx s)$ or $\mathsf{T}\, (s \approx t)$, then $\phi(s)$ may be added to B, where $\phi(s)$ is constructed by substituting one of the occurrences of t in $\phi(t)$ by s.

$$\frac{\mathsf{T}\, (t \approx s) \quad \phi(t)}{\phi(s)} \qquad \frac{\mathsf{T}\, (s \approx t) \quad \phi(t)}{\phi(s)}$$

Example 2.1 *Supposed the formulæ* $\mathsf{T}\, (a \approx b)$ *and* $\mathsf{T}\, P(a,a)$ *are on a branch* B*, by the application of Jeffrey's new expansion rule the new formulæ* $\mathsf{T}\, P(a,b)$*,* $\mathsf{T}\, P(b,a)$ *and* $\mathsf{T}\, P(b,b)$ *can be added to* B*. Note that it is not possible to derive* $\mathsf{T}\, P(b,b)$ *in a single step.*

Besides the expansion rules there is an additional closure rule. As before a branch B is closed if it contains complementary formulæ $\mathsf{T}\, G$ and $\mathsf{F}\, G$. But additionally it is also closed if it contains an inequality $\mathsf{F}\, (t \approx t)$, where t is any ground term.
The new expansion rules are symmetrical and their application is completely unrestricted. This leads to an enormous number of irrelevant formulæthat can be

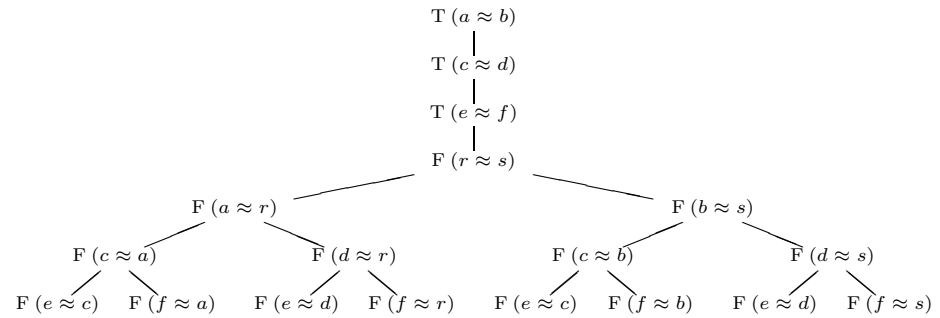

Figure 1: The disadvantage of Reeves's method: Three equalities and one inequality result in eight branches; and even more branches could be added to the tableau.

added. For example, if a branch contains formulæ$\mathsf{T}\,(f(a) \approx a)$ and $\mathsf{T}\,P(a)$, then it can subsequently be expanded by all the formulæin $\{\mathsf{T}\,P(f(a)),\ \mathsf{T}\,P(f(f(a))),\ \mathsf{T}\,P(f(f(f(a)))), \ldots\}$. According to Reeves [12] Jeffrey's method is therefore "useless in practice".

Reeves's Approach

Reeves uses an expansion rule that is based on the fact that in a canonical model where $\mathsf{T}\,P(a_1, \ldots, a_n)$ and $\mathsf{F}\,P(b_1, \ldots, b_n)$ or $\mathsf{F}\,(f(a_1, \ldots, a_n) \approx f(b_1, \ldots, b_n))$ is valid at least one of the formulæ $\mathsf{F}\,(a_1 \approx b_1)$, ..., $\mathsf{F}\,(b_1 \approx b_n)$ has to be valid, yielding the following equality expansion rules:

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

Expanding the formula $\mathsf{F}\,(a_1 \approx b_1 \wedge \ldots \wedge a_n \approx b_n)$ results in n new subbranches each containing one of the inequalities $\mathsf{F}\,(a_i \approx b_i)$ $(1 \leq i \leq n)$.
Without doubt Reeves's approach has some advantages. The search for the closure of a branch is more directed. Only atomic formulæthat potentially close a branch are used for expansion.
But the method also has a big disadvantage: If a branch contains several equalities and an inequality, the problem shown in Figure 1 occurs. As the new expansion rule can as well be applied to pairs of equalities and inequalities, a large number of new branches is added to the tableau. It grows exponentially with the number of equalities on the branch. As a consequence, Reeves's approach is not suitable for implementation.
Nevertheless we adopted the idea of transforming pairs of potentially closing atoms into disjunctions of inequalities. In our approach, however, building disjunctions from pairs of equalities and inequalities is avoided.

Popplestone's Approach

Popplestone's approach [11] is based on Jeffrey's method. It uses the same tableau expansion rules. The only improvements made essentially regard implementation. The main idea is to attach a graph to each formula in a tableau. The nodes of this graph are labelled with terms that occur above the formula on the branch or that can be derived from these terms using equalities present on the branch. Two nodes in the graph are adjacent if one can be transformed into the other by a single equality application. Thus, two nodes in the same graph are known to be equal if and only if they are connected. Therefore, a branch is closed if it contains a formula $\mathsf{F}\,(t \approx s)$ which has a graph attached to it proving t to be equal to s. A closure caused by complementary atoms can be found in a similar way.

This method has several advantages. Any heuristic can be used to search in the graphs and to expand them. Also, the information about the equality of certain terms can be reused in new formulægenerated by application of a tableau expansion rule.
But there is still a problem: information about the equality of all the terms in all the formulæin a tableau is generated. As will be shown later, this is not necessary. It suffices to take a closer look on some of the *atomic formulæ* and the terms they contain.

Fitting's Approach

Like Popplestone's method Fitting's is also an improvement of Jeffrey's approach. In [4] a complete implementation of a tableau–based theorem prover with equality in PROLOG is contained.
Fitting combines free variable tableaux with treatment of equality. The advantages of using free variables have already been discussed — only free variable substitutions that are necessary are performed. But with equalities present in a tableau, substitutions are also necessary at other points than closure of branches. There might be equality rule applications that require substitutions of free variables. Fortunately, these substitutions are easy to obtain, namely in a similar way as those substitutions that are needed to close a branch. If an equality $\mathsf{T}\,(t \approx s)$ is to be applied to a formula $\mathsf{T}\,Z(t')$, the application of an MGU σ of t and t' to the tableau is sufficient and necessary. Fitting's tableau expansion rules are thus the rules shown on the right. Fitting also addresses the problem of indeterminism embodied in the tableau expansion rules. As indeterminism is difficult to implement and inevitably leads to expensive backtracking, its elimination is an important improvement. To achieve this the application of equality rules is separated from the application of the standard tableau expansion rules. The application of the latter has to be exhausted before equalities are being applied to a tableau. In this process γ–rules may be applied to a certain formula only a finite number of times, say q. As a consequence, completeness has to be relativized: If ϕ is an unsatisfiable formula, then a closed tableau for ϕ will only be found when the limit on the number of γ–rule applications is sufficiently high. In practice one could implement an incremental prover which tries to find a proof by increasing the γ–limit after each try.

$$\frac{\mathsf{T}\,(t \approx s) \quad \phi(t')}{(\phi(s))\sigma} \qquad \frac{\mathsf{T}\,(s \approx t) \quad \phi(t')}{(\phi(s))\sigma}$$

where σ is an MGU of t and t'.

Fitting proves his method to be complete in the above sense if only the order of rule applications is "fair", i.e. if there is a standard tableau expansion rule that can be applied to a certain formula, this application will eventually happen.
Besides the elimination of indeterminism, there is another important point involved here. After the first stage of tableau expansion is finished and the tableau is exhausted (observing the limit q for γ–rule applications), it is sufficient to expand the tableau in the second stage solely by equality applications to *atomic formulæ*. That greatly decreases the number of possible equality applications.
In addition, the method has advantages for implementation, as it is not necessary to switch between the application of equalities and the application of other expansion rules. Thus an appropriate data structure for dealing with equalities can be chosen. Indeterminism is only partly resolved yet, since there may be several ways to close a branch or to apply equalities that require different free variable substitutions. Fitting's system subsequently tries each of these possibilities by means of PROLOG's backtracking. The resulting inefficiency presumably is the main reason for his concluding remark "...But remember the resulting system, while complete, may take years to prove anything interesting. Good heuristics are vital now."
Fitting proposes something similar to a heuristic: an orientation is assigned to the equalities. They are applied only from left to right. This makes it possible to avoid those applications of equalities like $\mathsf{T}\,(f(a) \approx a)$ which yield a large number of new formulæ. But there is a drawback, of course. Completeness would be lost if

there were not the following new expansion rule: The formula $\mathsf{T}\,(\forall x)(x \approx x)$ may be added to every branch. Unfortunately, this enables one to reverse the orientation of equalities. If a branch contains an equality $\mathsf{T}\,(f(a) \approx a)$, one can add $\mathsf{T}\,(\forall x)(x \approx x)$ and deduce $\mathsf{T}\,(x \approx x)$ from the latter. Applying the first equality to the *left* side of $\mathsf{T}\,(x \approx x)$ yields $\mathsf{T}\,(a \approx f(a))$. Therefore, this technique can only be described as a heuristic that specifies a preferred ordering of equalities in which they are tried first.

Completion–Based Methods

An essential prerequisite for using methods that are based on the completion of an equality theory is that the equalities are universally closed. This condition does not hold for the equalities in tableaux with free variables; there has to be a *single* substitution which, when applied, allows to close all branches simultaneously.
A method of equality theory completion would be needed, where the resulting complete theory not only provides the information whether two terms are equal, but also the set of all free variable substitutions that allow to prove equality of these terms using the initial equalities. This problem has, to our best knowledge, not been addressed so far.
On the other hand, for adding equality to ground semantic tableaux rewrite systems may well be used. In [2] an example of such a system is given. The problem of course is that the success of the rewrite part depends on the right guesses of terms in γ–rule applications. This will inevitably lead to frequent backtracking over the whole tableau and is hopelessly inefficient in practice.

3 An Improved Equality Module for Tableau–Based Provers

Two Separate Tableau Expansion Stages

From Fitting's approach we adopted the idea of separating the tableau expansion into two stages. In the first stage the standard rules are applied until the tableau is exhausted (observing the limit for γ–rule applications). Thus, in the second stage, it is possible to restrict equality applications and to avoid the generation of useless new formulæ. Equality rule applications can be limited to inequalities $\mathsf{F}\,(t \approx s)$ and pairs of atomic formulæ$\langle \mathsf{T}\,P(t_1, \ldots, t_n), \mathsf{F}\,P(s_1, \ldots, s_n)\rangle$ that potentially close a branch, where P is not the equality predicate $\approx$. In particular, for preserving completeness it is not necessary to apply equalities to other equalities.
When a tableau is exhausted, most of its formulæare not needed any more. From then on only equalities, inequalities, and pairs of potentially closing atoms are of interest. Therefore the tableau can be left and equality reasoning is done on a more suitable (and efficient) data structure. For this optimization it is crucial that backtracking to the first building stage of the tableau does not occur too frequently.

Disjunctions of Inequalities

The new data structure mentioned in the previous section consists of two sets of certain formulæfor each branch B of an exhausted tableau. The first one is the set $\mathrm{Gl}(B)$ defined as

$$\mathrm{Gl}(B) := \{s \approx t \;:\; \mathsf{T}\,(s \approx t) \in B\},$$

i.e. the set of all equalities present on B. The second is the set $\mathrm{Dis}(B)$, which consists of disjunctions of inequalities. It embodies the two remaining types of important formulæ. For every pair $\langle \mathsf{T}\,P(t_1, \ldots, t_n), \mathsf{F}\,P(s_1, \ldots, s_n)\rangle$ of atoms that potentially close B, in $\mathrm{Dis}(B)$ there is the n–place disjunction $t_1 \not\approx s_1 \vee \ldots \vee t_n \not\approx s_n$; and for every inequality $\mathsf{F}\,(t \approx s)$ on B, in $\mathrm{Dis}(B)$ there is the (one–place) disjunction $t \not\approx s$.

Example 3.1 *Consider the branch shown in the left part of Figure 2. The corresponding set of equalities is* $\{a \approx b, b \approx c\}$*; the set of disjunctions of inequalities is* $\{a \not\approx c\}$.

A tableau T with branches $B_1, \ldots, B_k$ is closed if there is a substitution σ of free variables such that for each branch B_i there is a disjunction $D_i \in \mathrm{Dis}(B_i)$ which can be proved to be unsatisfiable using the equalities in $\mathrm{Gl}(B_i)$. Soundness of this new closure rule is justified by the following consideration that holds for each branch B_i: If a disjunction $D_i = (t \not\approx s)$ has emerged from an inequality $\mathsf{F}\,(t \approx s) \in B_i$ and D_i can be proved to be unsatisfiable, we have in fact proved that, when σ is applied, $\mathsf{F}\,(t \approx t)$ is valid in every canonical model of B_i, which is clearly a contradiction. If, on the other hand, $D_i = (t_1 \not\approx s_1 \vee \ldots \vee t_n \not\approx s_n)$ has emerged from a pair of potentially complementary atoms $\langle \mathsf{T}\,P(t_1, \ldots, t_n), \mathsf{F}\,P(s_1, \ldots, s_n)\rangle$, unsatisfiability of D_i implies that $\mathsf{F}\,P(t_1, \ldots, t_n)$ holds and the branch can be closed as usual.
A straightforward method to prove an inequality $t \not\approx s$ to be unsatisfiable is to calculate step by step the equivalence classes of the terms t and s using equalities in $\mathrm{Gl}(B_i)$ and look in these classes for common elements.
One has to take into account that different free variable substitutions may lead to different equivalence classes. The problem of finding a suitable substitution that allows to refute all inequalities in a disjunction simultaneously is discussed in the next section.
The transformation of pairs of potentially closing atoms into disjunctions of inequalities corresponds to the application of Reeves's expansion rule; but the disadvantage of Reeves's approach is avoided, as disjunctions of inequalities are no longer allowed to be built up from pairs of equalities and inequalities.
The gain of efficiency in closing a branch by calculating equivalence classes instead of adding new formulæis illustrated by the following example.

Example 3.2 *If the branch shown in the left part of Figure 2 is expanded and closed according to Jeffrey's method, the information that c can be derived from a has to be generated twice to derive the formula* $\mathsf{T}\,P(c, c)$*, that is used to close the branch. Irrelevant formulæ, such as* $\mathsf{T}\,P(a, c)$*, are repeatedly added to the branch. With no heuristic at hand to avoid useless equality applications, up to eight new formulæare added to close the branch.*
In Figure 2 the branch is closed based on the transformation into disjunctions of inequalities. There is only one disjunction, containing two identical inequalities, that are merged into one. It is possible to prove the inequality to be unsatisfiable after no more than two new terms have been derived.

Example 3.2 demonstrates that for Jeffrey's method the number of formulægenerated in order to close a branch grows exponentially with the arity n of the involved predicate symbols, even if there is only one pair of potentially closing atoms, if for each occurring term there is only one equality that can be applied to it, and if no free variable substitutions have to be applied. It is very difficult to avoid the worst case, because a heuristic would be needed to recognize irrelevant formulæ. Only in the best case the number of generated formulægrows linearly with n, whereas the number of equality applications, when the new method is used, *always* grows linearly, even in the worst case.
In addition it is possible to use any heuristic to direct the search for common elements in the equivalence classes.

Most General Substitutions

The problem of finding a free variable substitution that allows to close all inequalities in one of the disjunctions of each branch simultaneously will be addressed next. A possible but inefficient solution would be to subsequently try each grounding substitution.
Fortunately, there is a much better method, namely, not to restrict the search to grounding substitutions. Instead, a closer look is taken at those substitutions σ that are an MGU of one side of an equality and an already derived term t or one of its subterms, i.e. that allow to apply this equality to t. This proves to be sufficient for preserving completeness as the following holds: If the term s can be derived from t

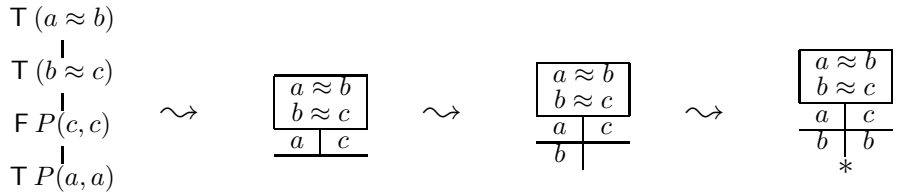

Figure 2: The branch of a tableau being closed using the method based on the transformation into disjunctions of inequalities and the calculation of equivalence classes. The new data structures are represented by a box containing the equalities having below it the two sides of the inequality with the terms in the equivalence classes of the left– and the right–hand term computed so far.

with the help of σ, then s subsumes every term s' that can be derived from t with the help of a grounding substitution τ more special than σ, i.e. there is a substitution σ' such that $s' = s\sigma'$. Therefore, an inequality that can be closed using the term s' can be closed using s as well.
Since equivalence classes may contain non–closed terms when they are built up using most general substitutions, an inequality is not only closed if the equivalence classes corresponding to its left– and right–hand side terms have a common element, but it is already closed if two *unifiable* terms occur in these classes.

Example 3.3 *$\sigma = \{y \leftarrow x\}$ is a most general substitution that allows to apply the equality $\mathsf{T}\,(f(x) \approx x)$ to the term $g(f(y))$. The term $g(x)$ would be the result of this application. If the grounding substitution $\sigma' = \{y \leftarrow a, x \leftarrow a\}$, which is more special than σ, was used, the term $g(a)$ could be derived from $g(f(y))$.*
Now, if an inequality becomes closed, because $g(a)$ is a common element of the equivalence classes corresponding to its left– and right–hand sides, that inequality may as well be closed using $g(x)$, since $g(x)$ subsumes $g(a)$.

Breadth–First–Search for Substitutions

Using most general substitutions to build up equivalence classes leads to a search tree for each side of an inequality whose edges are coloured with most general substitutions and whose nodes are the sets of terms that can be derived from the present equalities using terms already on the branch and the substitution on its incoming edge.
The edge leading to the root node of a search tree is labelled with the most general of all, the empty substitution and the root node itself consists of the terms that can be derived in one step without applying any substitutions, in particular including the starting term itself. The tree branches when the application of equalities to a term on one of the leaf nodes requires additional substitutions that are more special than those associated with its incoming edge.
Each of the terms in a search tree is associated with the substitution σ on its incoming edge and as well with the substitutions that are more special than σ. Substitutions become more special towards the leaves. Figure 3 shows an example.
One may prove that while using these search trees all substitutions that allow to close an inequality can be found. A depth–first–search could easily be implemented based on PROLOG's backtracking.
In general, however, search trees have infinitely long branches. It is therefore very difficult to realize when another branch should be tried, the problem is in fact undecidable. In addition, it is often necessary to look for more than one closing substitution for each inequality, as a disjunction is only closed provided that *all* substitutions used to close its inequalities are compatible. Consequently, heuristics would have to be used in order to decide at which point the search in a branch should be abandoned and backtracking should be initiated. Obviously, either these

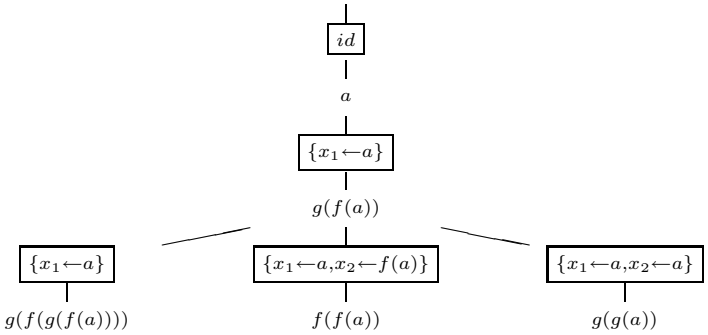

Figure 3: The search tree that is built from the right side of the inequality $g(g(a)) \not\approx a$ if $\mathrm{Gl}(B)$ contains the equalities $g(f(x_1)) \approx x_1$ and $g(x_2) \approx f(x_2)$.

heuristics react too quickly such that many possibilities for closing inequalities are being missed; or they react too slowly and one has to wait too long before the right branches are tried.
All these problems are avoided if breadth–first–search is used, i.e. all branches are searched simultaneously. Breadth–first–search is much more powerful, because any heuristic can be used to push ahead the search in some of the branches more quickly than in others. Therefore, virtually any other search method can easily be simulated. Since breadth–first–search cannot be based on PROLOG's backtracking, it is, however, slightly more difficult to implement in PROLOG.

Data Structures for Breadth–First–Search

Breadth–first–search as proposed in the previous section can be implemented using sets $\langle t \rangle_B$ that contain all the equivalence classes of a term t on a branch B associated with different substitutions. The elements s_σ of $\langle t \rangle_B$ are terms labelled with the substitution which is needed to derive them from t using equalities in $\mathrm{Gl}(B)$. If, for example, the application of $\{x_1 \leftarrow a\}$ leads to an equivalence class of the term a that contains $g(f(a))$, the element $g(f(a))_{\{x_1 \leftarrow a\}}$ is in $\langle a \rangle_B$.
One can view the sets $\langle t \rangle_B$ as a representation of the search trees introduced in the previous section; $s_\sigma \in \langle t \rangle_B$ if and only if s occurs on some node of the search tree whose incoming edge is coloured with σ.
Thus all required equivalence classes for checking an inequality are handled simultaneously. $\langle t \rangle_B$ is in general an infinite set, but there is a sequence of approximations $(\langle t \rangle_B^n)_{n \geq 0}$ to $\langle t \rangle_B$ that can be computed with a deterministic algorithm.
The only element in the first set $\langle t \rangle_B^0$ is t_{id}. The additional elements in $\langle t \rangle_B^{n+1}$ are those that can be derived in one step from a certain element $t' \in \langle t \rangle_B^n$ using the equalities in $\mathrm{Gl}(B)$, where $t' = \mathcal{H}(\langle t \rangle_B^n)$ is chosen by a certain heuristic $\mathcal{H}$. Elements that are subsumed by others are not included, since an inequality that can be proved to be unsatisfiable using a term $s'_{\sigma'}$ can as well be proved to be unsatisfiable using any term s_σ that subsumes $s'_{\sigma'}$.

Definition 3.4 (Subsumption) *Suppose s, s' are terms and σ, σ' are substitutions. s_σ* **subsumes** *$s'_{\sigma'}$ if s and s' are unifiable with an MGU τ such that $s\tau = s'$ and both σ and τ are more general than σ'.*

Example 3.5 *$f(x)_{\{y \leftarrow a\}}$ subsumes $f(a)_{\{x \leftarrow a, y \leftarrow a\}}$; $a_{\{x \leftarrow y\}}$ subsumes $a_{\{x \leftarrow b, y \leftarrow b\}}$. On the other hand, $a_{\{x \leftarrow f(y)\}}$ does not subsume $a_{\{x \leftarrow f(b)\}}$; both of these terms do,*

however, subsume $a_{\{x \leftarrow f(b), y \leftarrow b\}}$.

Definition 3.6 (Sequence of Sets $\langle t \rangle_B^n$) *The sets* $\langle t \rangle_B^n$ *are inductively defined as follows:*

- $\langle t \rangle_B^0 = \{t_{id}\}$
- *Let the set* Θ_n *contain all the elements from* $\langle t \rangle_B^n$ *and in addition the terms* r_τ *that can be derived in one step from* $s_\sigma = \mathcal{H}(\langle t \rangle_B^n)$*, where* r_τ *can be derived in one step from* s_σ *if*
  1. τ *is an MGU of a subterm of* s *and one side of an equality* $G \in \mathrm{Gl}(B)$*;*
  2. r *can be derived from* $s\tau$ *by application of* $G\tau$*;*
  3. σ *is more general than* τ*.*

  Then $\langle t \rangle_B^{n+1}$ *contains all elements from* Θ_n *that are not subsumed by another element in* Θ_n*. If there are several elements subsuming each other, an arbitrary one is chosen.*

Example 3.7 *Table 2 shows the computation of* $\langle a \rangle_B^n$ *(for* $n = 0, 1, 2$*) using the set of equalities* $\mathrm{Gl}(B) = \{g(f(x_1)) \approx x_1,\ g(x_2) \approx f(x_2)\}$.

n	$\langle a \rangle_B^n$	$\mathcal{H}(\langle a \rangle_B^n)$	Θ_n
0	a_{id}	a_{id}	a_{id} $g(f(a))_{\{x_1 \leftarrow a\}}$
1	a_{id} $g(f(a))_{\{x_1 \leftarrow a\}}$	$g(f(a))_{\{x_1 \leftarrow a\}}$	a_{id} $g(f(a))_{\{x_1 \leftarrow a\}}$ $a_{\{x_1 \leftarrow a\}}$ $g(f(g(f(a))))_{\{x_1 \leftarrow a\}}$ $g(g(a))_{\{x_1 \leftarrow a, x_2 \leftarrow a\}}$ $f(f(a))_{\{x_1 \leftarrow a, x_2 \leftarrow f(a)\}}$
2	a_{id} $g(f(a))_{\{x_1 \leftarrow a\}}$ $g(f(g(f(a))))_{\{x_1 \leftarrow a\}}$ $g(g(a))_{\{x_1 \leftarrow a, x_2 \leftarrow f(a)\}}$ $f(f(a))_{\{x_1 \leftarrow a, x_2 \leftarrow a\}}$		

Table 2: Computation of $\langle a \rangle_B^n$ $(n = 0, 1, 2)$ (Example 3.7).

Definition 3.8 (Closed Tableau) *A tableau* T *with branches* $B_1, \ldots, B_k$ *is* **closed** *if for each branch* B_i $(1 \leq i \leq k)$ *there is a disjunction*

$$D_i = (t_{i1} \not\approx s_{i1} \vee \ldots \vee t_{in_i} \not\approx s_{in_i}) \in \mathrm{Dis}(B_i)$$

such that for $1 \leq j \leq n_i$ *elements*

$$r^{ij}_{\rho_{ij}} \in \langle t_{ij} \rangle_{B_i}^{l_{ij}} \quad \text{and} \quad \overline{r}^{ij}_{\overline{\rho}_{ij}} \in \langle s_{ij} \rangle_{B_i}^{\overline{l}_{ij}}$$

can be found for some $l_{ij}, \overline{l}_{ij} \geq 0$*, where* r^{ij} *and* $\overline{r}^{ij}$ *are unifiable with an MGU* $\overline{\overline{\rho}}_{ij}$*, and there is a grounding substitution* σ *such that all of the substitutions* $\rho_{ij}, \overline{\rho}_{ij}, \overline{\overline{\rho}}_{ij}$ $(1 \leq i \leq k, 1 \leq j \leq n_i)$ *are more general than* σ*.*

If one shows a tableau to be closed, the substitution σ mentioned in the above definition allows to prove one disjunction of inequalities for each branch of the tableau to be unsatisfiable.
One can check whether a tableau T is closed according to Definition 3.8 by gradually computing the sets $\langle t\rangle^0_{B_i}$, $\langle t\rangle^1_{B_i}$, ... for every term t occurring in $\mathrm{Dis}(B_1)$, ..., $\mathrm{Dis}(B_k)$, where $B_1, \ldots, B_k$ are the branches of T. Definition 3.6 provides an effective way to do this.

Heuristics for Term Search

Before soundness and completeness of the method based on the sets $\langle t\rangle^n_B$ can be stated, we have to say something about the heuristics. The heuristic $\mathcal{H}$ for choosing an element from $\langle t\rangle^n_B$ to which equalities are applied has to be "fair" in the following sense:

Definition 3.9 (Fair Heuristic) *A heuristic $\mathcal{H}$ is* **fair** *if for each term t, each $n \geq 0$, and each element $s_\sigma \in \langle t\rangle^n_B$ there is an $m \geq 0$ such that $\mathcal{H}(\langle t\rangle^m_B)$ subsumes s_σ.*

For example, the heuristic that always chooses the syntactically shortest term which has not been chosen before is fair. We propose that the heuristic in Definition 3.6 is fair in the above sense. The heuristic we have been using for our implementation is described below.

Theorem 3.10 *Suppose T is a tableau with root formula ϕ.*
If T is closed according to Definition 3.8, then ϕ is not satisfiable in a normal model.
If ϕ is not satisfiable in a normal model and if the limit q for γ–rule applications is sufficiently high, T is closed according to Definition 3.8.

The proof is based on the following lemma which clarifies the connection between the sets $\langle t\rangle^n_B$ and the equivalence classes of the term t that are associated with different substitutions.

Lemma 3.11 *If $r_\sigma \in \langle t\rangle^n_B$ and σ is more general than the grounding substitution τ, then the equivalence class of t that is associated with τ contains the term $r\tau$.*
If the equivalence class of t that is associated with the grounding substitution τ contains a term s, then for some $n \geq 0$ there is an element $r_\sigma \in \langle t\rangle^n_B$ that subsumes s_τ.

For our implementation we used the following heuristic for choosing the next element from $\langle t\rangle^n_B$ to which equalities are applied:

Definition 3.12 (Implemented Heuristic) *The criteria for selection, ordered by their importance, are as follows:*

1. *Elements that have been chosen before are not considered again.*
2. *The term weight $G(s)$ and the distance $D(s) = G(s') - G(s)$ to the weight of the term s' from which s has been derived.*[3] *Terms s are preferred that*
   (a) *have a positive weight distance $D(s)$,*
   (b) *have a lower weight $G(s)$,*
   (c) *have a higher weight distance $D(s)$.*
3. *The number of steps necessary to derive a term. Terms that can be derived in fewer steps are preferred.*

We remark that this heuristic is fair in the sense of Definition 3.9. The term weight is by default given by the lexicographic length of the terms, but can easily be changed in order to prove theorems over specific domains.

[3] $D(s)$ is not defined for the single element t_{id} in $\langle t\rangle^0_B$, since it is not derived from a term. However, t_{id} is always chosen, as it is the only element in $\langle t\rangle^0_B$.

Universal Formulæ

An equality has often to be applied more than once in order to close a branch, each time with different substitutions for the variables occurring in it. A typical example is the associativity axiom $As = (\forall x)(\forall y)(\forall z)[(x \cdot y) \cdot z \approx x \cdot (y \cdot z)]$ from group theory. In most cases it has to be applied several times with different instantiations of x, y and z to prove even simple theorems of group theory.
In semantic tableaux the mechanism to do so usually is to apply the γ–rule more than once to As and thus generate several instances of As, each with different free variables substituted for x, y and z.
Consequently, to prove a theorem the γ–limit q has to be at least as high as the maximal number of necessary applications of the same equality with different substitutions for the free variables it contains. Before equalities can be applied, however, the tableau has to be exhausted, but the higher the limit q is, the more branches have to be closed and the bigger the tableau becomes. Moreover, it is quite difficult to choose the limit q appropriately, because one does not know how many equality applications will be needed.
The problem could be avoided if equations in the initial tableau were not allowed to occur nested in other formulæ, but appeared only on the top–level. We did, however, not want to employ this restriction in order to allow for a natural formulation of problems.[4] Nevertheless, the problem can at least partly be solved if one is able to recognize formulæand in particular equalities that are "universal", i.e. that can be used repeatedly with different substitutions for the variables they contain:

Definition 3.13 (Universal Formula) *Suppose ϕ is a formula on some tableau branch B. If $\phi = \mathsf{T}\, F$ for some F let $\phi_u = F$ else if $\phi = \mathsf{F}\, F$ let $\phi_u = \neg F$.*
ϕ is **universal** *with respect to x if the following holds for every normal model $\mathbf{M}$ and every grounding substitution σ:*

$$\textit{If } \mathbf{M} \models B\sigma, \textit{ then } \mathbf{M} \models ((\forall x)\phi_u)\sigma.$$

The problem of recognizing universal formulea is in general undecidable. However, a wide and important class of universal formulæcan be recognized easily:

Theorem 3.14 *A formula ϕ on a branch B is universal with respect to x if either*

1. *ϕ has been generated by applying a γ–rule to a γ–formula and x is the free variable that has been substituted for the variable bound by the leading quantifier in the γ–formula or*
2. *ϕ has been generated by applying an α– or a γ–rule to a formula which is universal with respect to x.*

Once formulæare recognized as being universal this knowledge can be taken advantage of in the following way:

- If an equality $\mathsf{T}\,(t \approx s)$ on a branch B is universal with respect to a variable x, the equality is implicitly universally quantified by marking all occurrences of x in the equality as being "universal".[5] Thus, later on, the variable x does not have to be instantiated to apply the equality.
- To make use of the universal validity of other formulæthe variables with respect to which they are universal are substituted by new ones that hitherto do not occur on the tableau. The occurrences of the same variable in different formulæare substituted by different new variables. This is done before the sets $\mathrm{Dis}(B)$ are being built. Therefore, it is less likely that a branch is prevented from being closed by a substitution which interferes with the closure of other branches.

[4] Cf. the example at the end of the next section.
[5] To mark an occurrence of a variable x as being "universal" it is given in the following a different typeface.

Theorem 3.10 still holds if one takes advantage of the universal validity of formulæ, i.e. soundness is preserved. Since elements subsuming each other are removed, the sets $\langle t \rangle_B^n$ are still finite, although $\mathrm{Gl}(B)$ may contain implicitly universally quantified formulæ.

4 Examples

In this section we are going through an example in detail which shows how the method works and which also demonstrates most (though not all) of its advantages. The formula we are going to establish as a tautology is

$$(\forall x)[g(f(x)) \approx c \wedge (f(x) \approx g(x) \vee \phi)] \supset g(g(a)) \approx g(f(b)),$$

where ϕ is an arbitrary formula that can be proved to be unsatisfiable without the use of equalities.
It will turn out that a γ–limit of 1 is sufficient, thus γ–formulæwill be removed after the first rule application to them.
The first stage of tableau expansion yields the tableau shown in Figure 4. According to Theorem 3.14 some of the formulæare universal with respect to x. Since the formulæmarked with an asterisk are no longer present on the tableau, the expansion using logical rules is finished, but the tableau cannot be closed.

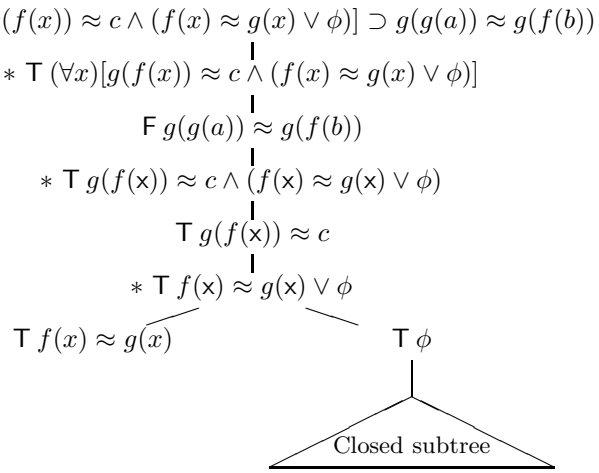

Figure 4: Expanded tableau after first stage.

At this stage the equality reasoning part is invoked. Let us refer with the letter B to the open branch. Then B is converted into the following data structure:

$$\begin{aligned} \mathrm{Gl}(B) &= \{g(f(x)) \approx c,\ f(x) \approx g(x)\} \\ \mathrm{Dis}(B) &= \{g(g(a)) \not\approx g(f(b))\} \end{aligned}$$

This can be conveniently represented as in the top left part of Figure 5. The algorithm now tries to find unifiable elements in the sets $\langle g(g(a)) \rangle_B^n$ and $\langle g(f(b)) \rangle_B^n$. In the first step by equation (2) the terms $g(f(a))$ and $f(g(a))$ are derived from $g(g(a))$. If depth–first– instead of a breadth–first–search was used, and the useless term $f(g(a))$ was derived first, this would prevent one from deriving $g(f(a))$, since the required substitutions $\{x \leftarrow a\}$ and $\{x \leftarrow g(a)\}$ are incompatible. Backtracking would become necessary. Instead, both $f(g(a))_{\{x \leftarrow g(a)\}}$ and $g(f(a))_{\{x \leftarrow a\}}$ are included in $\langle g(g(a)) \rangle_B^1$.

In the second step, first, the useless elements $f(f(b))_{\{x \leftarrow f(b)\}}$, $g(g(b))_{\{x \leftarrow b\}}$ are added to $\langle g(f(b))\rangle_B^1$ using equality (2). Then the term c is derived from $g(f(b))$ using equation (1). Since equation (1) is marked as being universal with respect to x, the required substitution $\{x \leftarrow b\}$ has not been actually performed. Thus the resulting term c is indexed with the empty substitution. In the next step we can see this is crucial, because the substitution $\{x \leftarrow a\}$ is needed to derive c from $g(f(a))$. If equality (1) had not been universal, the derivation would be stuck at this point with two incompatible substitutions. Backtracking to the first tableau expansion stage would be required to produce another copy of equation (1). Moreover, a γ–limit of 1 would not be sufficient, causing the fully expanded tableau and in particular the subtree for $\mathsf{T}\,\phi$ to be much bigger. In addition two copies of the subtree would have to be closed. If ϕ is a complex formula, this is very expensive.
Now, the disjunction $g(g(a)) \not\approx g(f(b))$ and thus branch B can be closed using the compatible elements $c_{\{x \leftarrow a\}} \in \langle g(g(a))\rangle_B^2$, $c_{id} \in \langle g(f(b))\rangle_B^1$.

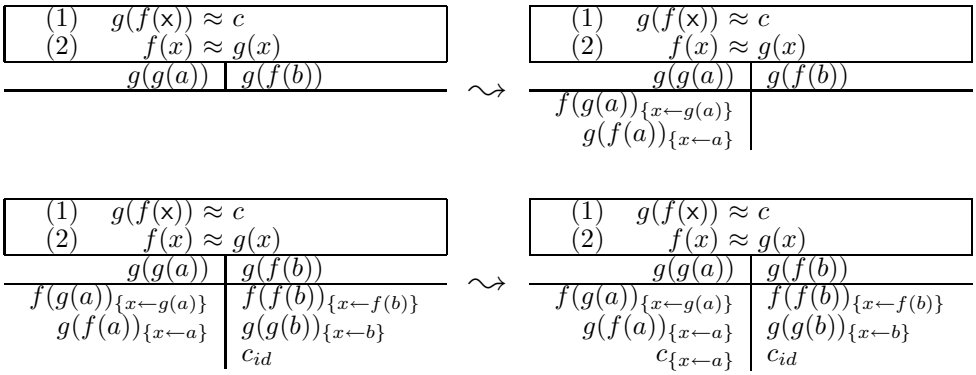

Figure 5: Second stage of tableau expansion.

If the same formula was to be proven to be a tautology using Fitting's method, first of all, the γ–limit q would have to be raised and more branches would have to be closed in the subtree for $\mathsf{T}\,\phi$. In addition, more equality applications would be necessary to prove the inequality $g(g(a)) \not\approx g(f(b))$ to be unsatisfiable. If the same heuristic was used to select the terms and equalities that we used for our implementation, then about 15 equality applications would be needed. Several times free variable substitutions would be applied that prevent the branch from being closed resulting in backtracking.
If Jeffrey's approach were used, the number of equality rule applications would heavily depend on the order of the equalities (there would be four equalities on the branch). In the best case only two branches have to be closed; a slightly different order, however, can lead to several hundred branches.
Our implementation solves most of the problems stated by Pelletier [10] in a few seconds. Below Pelletier's 55th problem[6] is stated both in natural language and in first–orde logic:

[6] Although a γ–limit of $q = 1$ is sufficient, the resulting tableau is too large to be depicted here. Our implementation closes 33 branches, and succeeds in 4.18s in proving that the problem as it is stated above is a tautology (running on a SUN–4 SPARC SLC workstation).

$$\frac{\begin{array}{c}(\exists x)(L(x)\wedge K(x,a))\\ L(a)\wedge L(b)\wedge L(c)\\ (\forall x)(L(x)\supset(x\approx a\vee x\approx b\vee x\approx c))\\ (\forall x)(\forall y)(K(x,y)\supset H(x,y))\\ (\forall x)(\forall y)(K(x,y)\supset\neg R(x,y))\\ (\forall x)(H(a,x)\supset\neg H(c,x))\\ (\forall x)(\neg(x\approx b)\supset H(a,x))\\ (\forall x)(\neg R(x,a)\supset H(b,x))\\ (\forall x)(H(a,x)\supset H(b,x))\\ (\forall x)(\exists y)(\neg H(x,y))\\ \neg(a\approx b)\end{array}}{K(a,a)}$$

Someone who lives in Dreadsbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadsbury Mansion, and are the only people who live therein. A killer always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Agatha hates. No one hates everyone. Agatha is not the butler.
Therefore: Agatha killed herself.

5 Conclusion

We have presented a method for handling equality within a first–order analytic tableaux framework that outperformances all previous approaches considerably. Recent tableau proving techniques, such as free variables, universal formulæ, and a liberalized δ–rule are made full use of. Although performance cannot compete with completion–based equality provers or resolution systems using paramodulation, we think that the effort is justified, since tableau–based systems offer superior possibilities for many non–classical logics. We stress that, in contrary to most other approaches, the equality theory needs not to be specified explicitly, but equalities may occur arbitrarily nested in formulæ.
Our considerations suggest that it might be better to abandon the depth–first branch–by–branch closure of tableaux in favour of a breadth-first approach which expands a tableau fully, before attempting to close all branches simultaneously.
Other topics for further investigations are the use of successful equality reasoning methods, such as completion–based algorithms (though one has to take into account the problem of non–universal variables), and implementation of many–valued "equality" predicates.

References

[1] B. Beckert. Konzeption und Implementierung von Gleichheit für einen tableau–basierten Theorembeweiser. Studienarbeit, Univ. Karlsruhe, July 1991.

[2] 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.

[3] M. Fitting. First-order modal tableaux. *JAR*, 4:191 – 213, 1988.

[4] M. Fitting. *First–Order Logic and Automated Theorem Proving*. Springer, 1990.

[5] R. Hähnle. Spezifikation eines Theorembeweisers für dreiwertige First–Order Logik. IWBS Report 136, Wiss. Zentrum, IWBS, IBM Deutschland, 1990.

[6] R. Hähnle. Towards an efficient tableau proof procedure for multiple–valued logics. In *Proc. WS Computer Science Logic, Heidelberg*, pp 248 – 260. Springer LNCS 533, 1990.

[7] R. Hähnle and P. H. Schmitt. The liberalized δ–rule in free variable semantic tableaux. *to appear*, 1991.

[8] R. C. Jeffrey. *Formal Logic. Its Scope and Limits*. McGraw–Hill, New York, 1967.

[9] F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *JAR*, 4:69 – 100, 1988.

[10] F. J. Pelletier. Seventy-five problems for testing automatic theorem provers. *JAR*, 2:191 – 216, 1986.

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

[12] S. V. Reeves. Adding equality to semantic tableaux. *JAR*, 3:225 – 246, 1987.

[13] R. Smullyan. *First–Order Logic*. Springer, New York, second edition, 1968.