Extended Abstract

A Completion–Based Method for Adding Equality to Free Variable Semantic Tableaux

Bernhard Beckert

Institute for Logic, Complexity and Deduction Systems
University of Karlsruhe
Am Fasanengarten 5, 7500 Karlsruhe, Germany
beckert@ira.uka.de

1 Introduction

For both classical and nonclassical first–order logic equality is a crucial feature to increase expressivity of the object language. It is, therefore, of great importance to have a method at hand that allows tableau–based theorem provers to handle equality in an efficient way.

In the first approaches to adding equality to semantic tableaux [8, 10, 11] additional tableau rules were defined that allow the application of equalities occurring on a branch to other formulas. In [4] a completion–based method has been described; it uses, however, as the approaches in [8, 10, 11], the ground version of tableaux. Recent methods, described in [5] and [2, 3], are based on the much more efficient free variable tableau system [5].[1]

In [3] it has been shown that one of the reasons for the inefficiency of previous approaches is that they all mix up the application of equalities and of classical tableau rules. In contrast, the method presented in [3] for closing a tableau branch, consists of two separate tableau expansion stages. In the first stage the tableau is expanded using the classical tableau rules; in the second stage, from a branch B the set of equalities[2]

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

and the set Dis_B are extracted; where the latter consists of disjunctions of inequalities. It is built from the two remaining types of important formulas on B: For every pair $\langle \mathsf{T}\,P(s_1,\ldots,s_n), \mathsf{F}\,P(t_1,\ldots,t_n)\rangle$ of atoms, that potentially close B, in Dis_B there is the n–place disjunction $s_1 \not\approx t_1 \vee \ldots \vee s_n \not\approx t_n$ and for every inequality $\mathsf{F}\,(s \approx t)$ on B, in Dis_B there is the (one–place) disjunction $s \not\approx t$.

Using these two sets, all substitutions that allow to close a branch B can be generated by computing for each inequality $(s \not\approx t)$ in Dis_B a complete set of rigid E_B–unifiers of s and t:

[1] The γ–rule in free variable tableaux does not substitute the bound variable by a "guessed" ground term, but by a new free variable. Therefore, a free variable tableau can contain equalities that are not ground. To close a free variable tableau T, one has to find a (ground) substitution σ that allows to close all branches of T simultaneously, i.e., such that $T\sigma$ is closed.

[2] We use the signed version of tableaux, i.e., the node labels in a tableau are first–order formulas prefixed with either T (true) or F (false). The equality predicate symbol is denoted by $\approx$, such that no confusion with the meta–level equality predicate $=$ can arise.

Definition 1 (Rigid E–Unifier, Complete Set of Rigid E–Unifiers) *Let E be a finite set of equalities and let s and t be terms. A substitution σ is called a rigid E–unifier of s and t iff*

$$E\sigma \models s\sigma \approx t\sigma$$

where the variables in $E\sigma$, $t\sigma$ and $s\sigma$ are treated as constants (held "rigid"), i.e., $E\sigma \models s\sigma \approx t\sigma$ is treated as a ground problem.

A set $\mathcal{C}(E, s \approx t)$ of rigid E–unifiers of s and t is called complete iff for each rigid E–unifier τ of s and t there is a unifier $\sigma \in \mathcal{C}(E, s \approx t)$ that is more general[3] than τ.

In general, there is no *finite* complete set of rigid E-unifiers; it is, however, always possible to enumerate one.

Example 2 *The following table shows some examples for rigid E–unification problems:*

E	s	t	Rigid E–unifiers of s and t
$\{f(x) \approx x\}$	$f(a)$	a	$\{x/a\}$
$\{f(a) \approx a\}$	$f(x)$	a	$\{x/a\}$, $\{x/f(a)\}$, $\{x/f(f(a))\}$, ...
$\{f(x) \approx x\}$	$g(f(a), f(b))$	$g(a, b)$	none[4]

Each rigid E–unifier is as well a non–rigid E–unifier, but, as the last of the above examples shows, the contrary does not hold true.

With complete sets of rigid E–unifiers for each inequality $(s \not\approx t)$ in Dis_B at hand one can easily construct all closing substitutions of the branch B by searching for substitutions τ that are a specialization of some $\sigma_i \in \mathcal{C}(E_B, s_i \approx t_i)$ $(i = 1, \ldots, n)$, where $(s_1 \not\approx t_1 \vee \ldots \vee s_n \not\approx t_n)$ is one of the disjunctions in Dis_B. These substitutions τ allow to *simultaneously* prove all the inequalities in one of the disjunctions to be false.[5]

The method described in [3] for computing complete sets of E–unifiers is based on iteratively constructing sets $\langle t \rangle_B$, whose elements t'_σ are terms labeled with a most general substitution that allows to derive them from t using equalities in E_B, i.e., if $t'_\sigma \in \langle t \rangle_B$ and τ is a specialization of σ, then $E\tau \models (t \approx t')\tau$. With that

$$\mathcal{C}(E_B, s \approx t) = \{\sigma \ : \ \text{there are } s'_{\sigma_1} \in \langle s \rangle_B,\ t'_{\sigma_2} \in \langle t \rangle_B \text{ such that } s',\ t' \text{ are unifiable with MGU } \sigma_3, \text{ and } \sigma \text{ is a most general specialization of } \sigma_1,\ \sigma_2,\ \sigma_3\ \}$$

is a complete set of rigid E–unifiers of s and t.

Even if this method for handling equality within a first–order analytic tableaux framework outperforms all previous approaches considerably, it cannot compete with completion–based equality provers or resolution systems using paramodulation. The main problem is that for computing the sets $\langle t \rangle_B$ equalities are applied unrestricted in both directions. Therefore, $\langle t \rangle_B$ may contain many redundant terms.

In addition, this and all other procedures that allow equalities to be applied unrestricted in both directions have another major drawback: They cannot be used to *decide* the rigid E–unification problem, i.e., to decide whether a rigid E–unifier of two terms s and t exists, although this problem is, in fact, decidable [6].[6] Therefore, the search might continue infinitely even if no rigid E–unifier exists.

[3] σ is more general than τ iff $\tau = \sigma\mu$ for some substitution μ.

[4] There is, however, a *non*–rigid E–unifier of s and t (the empty substitution).

[5] If τ is a specialization of a rigid E–unifier σ, then τ is a rigid E–unifier as well.

[6] The problem has been shown to be NP–complete.

2 A Completion–Based Method

All disadvantages mentioned above can be avoided by using a completion based procedure for computing complete sets of rigid E–unifiers. The algorithm we use is an extension of the algorithm that Gallier et. al. use in [6] to prove the decidability of the rigid E–unification problem. It is based on the unfailing completion procedure [1, 9]. The basic idea — and the main difference to the classical unfailing completion procedure — is that during the completion process variables are never renamed, even if equalities that have variables in common are applied to each other. In addition, *constraints* $\langle \sigma, O \rangle$ consisting of a substitution σ and a set O of *order conditions*[7] are attached to the reduction rules: $l \to r \ll \langle \sigma, O \rangle$ means that $l\tau \to r\tau$ is valid in $E\tau$ whenever τ is a specialization of σ and $O\tau$ is true. If, for example, the commutativity axiom $f(x,y) \approx f(y,x)$ is valid in $E\{z/a\}$, this is represented by $f(x,y) \to f(y,x) \ll \langle \{z/a\}, \{x \succ y\} \rangle$. Using these constraints, every equality is orientable.

The completion algorithm has to be provided with a reduction ordering $\succ$ that is complete on the ground terms.[8] The algorithm computes a set $\mathcal{MC}(E, s \approx t)$ of *minimal* rigid E–unifiers. These unifiers are minimal with respect to a (partial) ordering on the substitutions that is induced by $\succ$ (therefore $\mathcal{MC}(E, s \approx t)$ depends on the chosen ordering $\succ$).

Gallier et. al. proved that $\mathcal{MC}(E, s \approx t)$ is always finite (thus its computation always terminates); and that $\mathcal{MC}(E, s \approx t)$ is complete in the following way:

Theorem 3 *If σ is any rigid E–unifier of s and t, then there is a specialization $\bar{\mu}$ of a minimal rigid E–unifier $\mu \in \mathcal{MC}(E, s \approx t)$, such that $\sigma \rightsquigarrow \bar{\mu}$, where the relation $\rightsquigarrow$ on unifiers is defined by*

$$\sigma \rightsquigarrow \bar{\mu} \quad \textit{iff} \quad \textit{for all variables } x \textit{ in } E,s,t\textit{: } E\sigma \models \sigma(x) \approx \bar{\mu}(x)$$

In other words, $\sigma \rightsquigarrow \bar{\mu}$ means that $\bar{\mu}$ can be generated from σ using the equations in $E\sigma$. Unfortunately, $\rightsquigarrow$ is not symmetric, as the following example illustrates:

Example 4 *Supposed $E = \{x \approx a\}$, then $\{x/b\} \rightsquigarrow \{x/a\}$, since $\{x \approx a\}\{x/b\} \models b \approx a$, but $\{x/a\} \not\rightsquigarrow \{x/b\}$.*

It is, therefore, not possible to generate a complete set $\mathcal{C}(E, s \approx t)$ of rigid E–unifiers just by applying to minimal unifiers $\mu \in \mathcal{MC}(E, s \approx t)$ equalities from $E\mu$.

Using the set $\mathcal{MC}(E, s \approx t)$, it is, however, possible to efficiently *test* whether a given substitution σ is a rigid E–unifier: σ is a rigid E–unifier of s and t iff there is a $\bar{\mu}$ in the (finite) set $\{\bar{\mu} \ : \ \sigma \rightsquigarrow \bar{\mu}, \bar{\mu} \text{ minimal}\}$ of *minimal* unifiers that can be generated from σ such that there is a $\mu \in \mathcal{MC}(E, s \approx t)$ more general than $\bar{\mu}$.

Based on this test a complete set $\mathcal{C}(E, s \approx t)$ of rigid E–unifiers can be computed in the following way: First, a complete set $\mathcal{P} \supset \mathcal{C}(E, s \approx t)$ of possible rigid E–unifiers is generated by applying the equalities in E in a non–rigid way to the unifiers in $\mathcal{MC}(E, s \approx t)$. All the substitutions in $\mathcal{P}$ are, then, tested; those that are rigid E–unifiers are used to build $\mathcal{C}(E, s \approx t)$.

In practice, and in particular in the semantic tableau framework, it is usually not necessary to generate $\mathcal{P}$, but it suffices to try substitutions that are not taken from $\mathcal{P}$ but that are obtained otherwise, e.g., substitutions that are known to close one or more of the other branches of the tableau.

[7] An order condition is of the form $s \succ t$, where s and t are terms.

[8] $\succ$ is a reduction ordering iff it is well–founded, $s \succ t$ implies $u[s] \succ u[t]$, and $s \succ t$ implies $s\sigma \succ t\sigma$. A reduction ordering that is complete on the ground terms exists for all signatures.

3 Conclusion

To our best knowledge, the method presented here is the first that brings together the advantages of free variable semantic tableaux and those of a completion–based handling of equality. It can easily be combined with recent tableau proving techniques, such as lemma generation or the liberalized δ–rule [7].

A problem that remains to be solved is the combination of rigid and non–rigid E–unification: Given two sets $E_\forall$ and $E_\exists$ of equalities and terms s and t, compute (in a complete way) substitutions σ such that

$$(E_\forall \cup E_\exists\sigma) \models s\sigma \approx t\sigma.$$

A completion–based algorithm solving this problem will allow to efficiently handle equality in tableaux with universal formulas [3].

References

[1] 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.

[2] Bernhard Beckert. Konzeption und Implementierung von Gleichheit für einen tableau–basierten Theorembeweiser. IKBS Report 208, Science Center, Institute for Knowledge Based Systems, IBM Germany, 1991.

[3] Bernhard Beckert and Reiner Hähnle. An improved method for adding equality to free variable semantic tableau. In *Proceedings, 11th Conference on Automated Deduction CADE, Albany, NY*. Springer, LNCS, 1992.

[4] 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.

[5] Melvin C. Fitting. *First–Order Logic and Automated Theorem Proving*. Springer, New York, 1990.

[6] 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.

[7] Reiner Hähnle and Peter H. Schmitt. The liberalized δ–rule in free variable semantic tableaux. *Journal of Automated Reasoning*, To appear 1993.

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

[9] 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.

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

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