CHAPTER 1

EQUALITY AND OTHER THEORIES BY BERNHARD BECKERT

1 INTRODUCTION

Theory reasoning is an important technique for increasing the efficiency of automated deduction systems. The knowledge from a given domain (or theory) is made use of by applying efficient methods for reasoning in that domain. The general purpose *foreground reasoner* calls a special purpose *background reasoner* to handle problems from a certain theory.

Theory reasoning is indispensable for automated deduction in real world domains. Efficient equality reasoning is essential, but most specifications of real world problems use other theories as well: algebraic theories in mathematical problems and specifications of abstract data types in software verification to name a few.

Following the pioneering work of M. Stickel, theory reasoning methods have been described for various calculi; e.g., resolution [Stickel, 1985; Policriti and Schwartz, 1995], path resolution [Murray and Rosenthal, 1987b], the connection method [Petermann, 1992; Baumgartner and Petermann, 1998], model elimination [Baumgartner, 1992], connection tableaux [Baumgartner *et al.*, 1992; Furbach, 1994; Baumgartner, 1998], and the matrix method [Murray and Rosenthal, 1987a].

In this chapter, we describe how to combine background reasoners with the ground, the free variable, and the universal formula versions of semantic tableaux. All results and methods can be adapted to other tableau versions for first-order logic: calculi with signed formulae, with different δ-rules, with methods for restricting the search space such as connectedness or ordering restrictions, with lemma generation, etc. Difficulties can arise with adaptations to tableau calculi for other logics, in particular if the consequence relation is affected (e.g., nonmonotonic logics and linear logic); and care has to be taken if theory links or theory connections have to be considered [Petermann, 1993; Baumgartner, 1998; Baumgartner and Petermann, 1998].

Background reasoners have been designed for various theories, in particular for equality reasoning; an overview can be found in [Baumgartner *et al.*, 1992; Furbach, 1994; Baumgartner, 1998], for set theory in [Cantone *et al.*, 1989]. Reasoning in single models, e.g. natural numbers, is discussed in [Bürckert, 1990].

One main focus of this chapter is efficient equality reasoning in semantic tableaux. Equality, however, is the only theory that is discussed in detail. There is no uniform way for handling theories, which is, after all, the reason for using a background reasoner but which makes it impossible to present good background reasoners for all possible theories. The second main focus of this chapter is therefore on the interaction between foreground and background reasoners, which plays a critical rôle for the efficiency of the combined system.

The chapter is organized as follows: in Section 2, the basic concepts of theory reasoning are introduced, and the main classifications of theory reasoning methods are discussed. The ground, the free variable, and the universal formula version of semantic tableaux, which are the versions that have to be distinguished for theory reasoning, are defined in Section 3, and methods are presented to add theory reasoning to these versions of tableaux. Soundness of these methods is proven in Section 4. In Section 5, completeness criteria for background reasoners are defined. Total and partial background reasoners for the equality theory are presented in Sections 6 and 7. Incremental theory reasoning, which is a method for improving the interaction between foreground and background reasoners, is introduced in Section 8. Finally, in Section 9, methods for handling equality are described that are based on modifying the input formulae.

2 THEORY REASONING

2.1 First-Order Logic: Syntax and Semantics

We use the logical connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\leftrightarrow$ (equivalence), $\neg$ (negation), and the quantifier symbols $\forall$ and $\exists$.

NOTATION 1 *A* first-order signature $\Sigma = \langle P_\Sigma, F_\Sigma, \alpha_\Sigma \rangle$ *consists of a set* P_Σ *of* predicate symbols, *a set* F_Σ *of* function symbols, *and a function* α_Σ *assigning an* arity $n \geq 0$ *to the predicate and function symbols; for each arity, there are infinitely many function and predicate symbols. Function symbols of arity 0 are called* constants. *In addition, there is an infinite set* V *of* object variables.

$Term_\Sigma$ *is the set of all terms and* $Term^0_\Sigma \subset Term_\Sigma$ *is the set of all ground terms built from* Σ *in the usual manner.* $Form_\Sigma$ *is the set of all first-order formulae over* Σ*; a formula* $\phi \in Form_\Sigma$ *must not contain a variable that is both bound and free in* ϕ *(see Sect. 1.1 in Chap. 3 for formal definitions of* $Term_\Sigma$ *and* $Form_\Sigma$*).* $Lit_\Sigma \subset Form_\Sigma$ *is the set of all literals.*

DEFINITION 2 *A variable* $x \in V$ *is* free *in a first-order formula* ϕ*, if there is an occurrence of* x *in* ϕ *that is not inside the scope of a quantification* $(\forall x)$ *or* $(\exists x)$*;* x *is* bound *in* ϕ *if it occurs in* ϕ *inside the scope of a quantification* $(\forall x)$ *or* $(\exists x)$.

A sentence *is a formula* $\phi \in Form_\Sigma$ *not containing any free variables.*

NOTATION 3 *$Subst_\Sigma$ is the set of all substitutions, and $Subst^*_\Sigma \subset Subst_\Sigma$ is the set of all idempotent substitutions with finite domain.*

A substitution $\sigma \in Subst_\Sigma$ with a finite domain $\{x_1, \ldots, x_n\}$ can be denoted by $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, i.e., $\sigma(x_i) = t_i$ $(1 \leq i \leq n)$.

The restriction of σ to a set $W \subset V$ of variables is denoted by $\sigma_{|W}$.

A substitution σ may be applied to a quantified formula ϕ; however, to avoid undesired results, the bound variables in ϕ must neither occur in the domain nor the scope of σ.

DEFINITION 4 *A formula ϕ' is an* instance *of a formula ϕ if there is a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \in Subst^*_\Sigma$ such that*

1. *$\phi' = \phi\sigma$,*
2. *none of the variables $x_1, \ldots, x_n$ is bound in ϕ, and none of the variables that are bound in ϕ occurs in the terms $t_1, \ldots, t_n$.*

If an instance does not contain any variables, it is a ground instance.

DEFINITION 5 *A formula $\phi \in Form_\Sigma$ is* universally quantified *if it is of the form $(\forall x_1) \cdots (\forall x_n)\psi$, $n \geq 0$, where ψ does not contain any quantifications.*

In this case, if a formula ψ' is an instance of ψ (Def. 4), it is as well called an instance of ϕ.

DEFINITION 6 *A* structure *$M = \langle D, I\rangle$ for a signature Σ consists of a non-empty domain D and an interpretation I which gives meaning to the function and predicate symbols of Σ.*

A variable assignment *is a mapping $\nu : V \to D$ from the set of variables to the domain D.*

The combination of an interpretation I and an assignment ν associates (by structural recursion) with each term $t \in Term_\Sigma$ an element $t^{I,\nu}$ of D.

The evaluation function *$val_{I,\nu}$ maps the formulae in $Form_\Sigma$ to the truth values true and false (in the usual way, see Sect. 1.2 in Chap. 3). If $val_{I,\nu}(\phi) =$ true, which is denoted by $(M, \nu) \models \phi$, holds for all assignments ν, then M* satisfies *the formula ϕ (is a* model *of ϕ); M satisfies a set Φ of formulae if it satisfies all elements of Φ.*

A formula ϕ is a tautology *if it is satisfied by all structures.*

DEFINITION 7 *A formula $\psi \in Form_\Sigma$ is a* (weak) consequence *of a set $\Phi \subset Form_\Sigma$ of formulae, denoted by $\Phi \models \psi$, if all structures that are models of Φ are models of ψ as well.*

In addition to the normal (weak) consequence relation $\models$, we use the notion of strong consequence:

DEFINITION 8 *A formula $\psi \in Form_\Sigma$ is a* strong consequence *of a set $\Phi \subset Form_\Sigma$ of formulae, denoted by $\Phi \models^\circ \psi$, if for all structures $M = \langle D, I\rangle$ and all variable assignments ν:*

$$\textit{If } (M,\nu) \models \phi \textit{ for all } \phi \in \Phi, \textit{ then } (M,\nu) \models \psi \ .$$

A difference between the strong consequence relation $\models^\circ$ and the weak consequence relation $\models$ is that the following holds for $\models^\circ$ (but not for $\models$):

LEMMA 9 *Given a set $\Phi \subset Form_\Sigma$ of formulae and a formula $\psi \in Form_\Sigma$, if $\Phi \models^\circ \psi$, then $\Phi\sigma \models^\circ \psi\sigma$ for all substitutions $\sigma \in Subst^*_\Sigma$.*

2.2 Theories

We define any satisfiable set of sentences to be a theory.

DEFINITION 10 *A* theory *$\mathcal{T} \subset Form_\Sigma$ is a satisfiable set of sentences.*

In the literature, often the additional condition (besides satisfiability) is imposed on theories that they are closed under the logical consequence relation. Without that restriction, we do not have to distinguish between a theory and its defining set of axioms.

EXAMPLE 11 The most important theory in practice is the equality theory $\mathcal{E}$.[1] It consists of the following axioms:

(1) $(\forall x)(x \approx x)$ (reflexivity),

(2) for all function symbols $f \in F_\Sigma$:

$$(\forall x_1)\cdots(\forall x_n)(\forall y_1)\cdots(\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \supset f(x_1,\ldots,x_n) \approx f(y_1,\ldots,y_n))$$

where $n = \alpha_\Sigma(f)$ (monotonicity for function symbols),

(3) for all predicate symbols $p \in P_\Sigma$:

$$(\forall x_1)\cdots(\forall x_n)(\forall y_1)\cdots(\forall y_n)((x_1 \approx y_1 \wedge \ldots \wedge x_n \approx y_n) \supset (p(x_1,\ldots,x_n) \supset p(y_1,\ldots,y_n)))$$

where $n = \alpha_\Sigma(p)$ (monotonicity for predicate symbols),

Symmetry and transitivity of $\approx$ are implied by reflexivity (1) and monotonicity for predicate symbols (3) (observe that $\approx \in P_\Sigma$).

[1] The equality predicate is denoted by $\approx \in P_\Sigma$ such that no confusion with the meta-level equality $=$ can arise.

EXAMPLE 12 The theory $\mathcal{OP}$ of partial orderings consists of the axioms

(1) $(\forall x)\neg(x < x)$ (anti-reflexivity),

(2) $(\forall x)(\forall y)(\forall z)((x < y) \wedge (y < z) \supset (x < z))$ (transitivity).

$\mathcal{OP}$ is a finite theory; contrary to the equality theory, it does not contain monotonicity axioms.

An important class of theories, called *equational theories*, are extensions of the equality theory $\mathcal{E}$ by additional axioms that are universally quantified equalities. An overview of important equational theories and their properties can be found in [Siekmann, 1989].

EXAMPLE 13 The AC-theory for the function symbol f contains (besides $\mathcal{E}$) the additional axioms $(\forall x)(\forall y)(\forall z)(f(f(x,y),z) \approx f(x,f(y,z)))$ and $(\forall x)(\forall y)(f(x,y) \approx f(y,x))$, which state associativity resp. commutativity of f; it is an equational theory.

Other typical examples for equational theories are specifications of algebraic structures:

EXAMPLE 14 Group theory can be defined using, in addition to $\mathcal{E}$, the equalities

$$\begin{array}{rrcl} (\forall x)(\forall y)(\forall z)((x \circ y) \circ z & \approx & x \circ (y \circ z)) \\ (\forall x)(\quad x \circ e & \approx & x \quad) \\ (\forall x)(\quad x \circ x^{-1} & \approx & e \quad) \end{array}$$

The definitions of structure, satisfiability, tautology, and logical consequence are adapted to theory reasoning in a straightforward way:

DEFINITION 15 *Let $\mathcal{T}$ be a theory. A* $\mathcal{T}$-structure *is a structure that satisfies all formulae in $\mathcal{T}$. A formula ϕ (a set Φ of formulae) is* $\mathcal{T}$-satisfiable *if there is a $\mathcal{T}$-structure satisfying ϕ (resp. Φ), else it is* $\mathcal{T}$-unsatisfiable. *A sentence ϕ is a* $\mathcal{T}$-tautology *if it is satisfied by all $\mathcal{T}$-structures.*

A formula ϕ is a (weak) $\mathcal{T}$-consequence *of a set Ψ of formulae, denoted by $\Psi \models_{\mathcal{T}} \phi$, if ϕ is satisfied by all $\mathcal{T}$-structures that satisfy Ψ. A formula ϕ is a* strong $\mathcal{T}$-consequence *of a set Ψ of formulae, denoted by $\Psi \models^{\circ}_{\mathcal{T}} \phi$, if for all $\mathcal{T}$-structures M and all variable assignments ν:*

$$\textit{If } (M,\nu) \models \psi \textit{ for all } \psi \in \Psi\textit{, then } (M,\nu) \models \phi \ .$$

LEMMA 16 *Given a theory $\mathcal{T}$, a set Φ of sentences, and a sentence ψ, the following propositions are equivalent:*

1. $\Phi \models_{\mathcal{T}} \psi$.
2. $\Phi \cup \mathcal{T} \models \psi$.
3. $\Phi \cup \mathcal{T} \cup \{\neg\psi\}$ *is unsatisfiable.*
4. $\Phi \cup \{\neg\psi\}$ *is $\mathcal{T}$-unsatisfiable.*

2.3 Properties of Theories

The following definitions clarify which properties theories should have to be useful in practice:

DEFINITION 17 *A theory $\mathcal{T}$ is* (finitely) axiomatizable *if there is a (finite) decidable set $\Psi \subset Form_\Sigma$ of sentences (the axioms) such that: $\phi \in Form_\Sigma$ is a $\mathcal{T}$-tautology if and only if $\Psi \models \phi$.*

A theory $\mathcal{T}$ is complete *if, for all sentences $\phi \in Form_\Sigma$, either ϕ or $\neg\phi$ is a $\mathcal{T}$-tautology.*

All theories that we are concerned with, including equality, are axiomatizable. An example for a theory that is not axiomatizable is the set $\mathcal{T}$ of all satisfiable sentences.

If a theory $\mathcal{T}$ is axiomatizable, then the set of $\mathcal{T}$-tautologies is enumerable; it may, however, be undecidable (a simple example for this is the empty theory). If $\mathcal{T}$ is both axiomatizable and complete, then the set of $\mathcal{T}$-tautologies is decidable.

Another important method for characterizing a theory $\mathcal{T}$—besides axiomatization—is the model theoretic approach, where $\mathcal{T}$ is defined as the set of all formulae that are true in a given structure M. Theories defined this way are always complete, because $M \models \phi$ or $M \models \neg\phi$ for all sentences ϕ.

DEFINITION 18 *A theory $\mathcal{T}$ is* universal *if it is axiomatizable using an axiom set consisting of universally quantified formulae (Def. 4).*

THEOREM 19 *A set Φ of universally quantified formulae is $\mathcal{T}$-unsatisfiable if and only if there is a finite set of ground instances of formulae from Φ that is $\mathcal{T}$-unsatisfiable.*

In the literature on theory reasoning, all considerations are usually restricted to universal theories, because the Herbrand-type Theorem 19 holds exactly for universal theories [Petermann, 1992]. This theorem is essential for theory reasoning if the background reasoner can only provide formulae without variable quantifications (e.g., only literals or clauses); this is, for example, the case if theory reasoning is added to clausal tableaux or resolution.

EXAMPLE 20 The theory $\mathcal{T} = \{(\exists x)p(x)\}$ is not universal. Consequently, there are sets Φ of universally quantified formulae that are $\mathcal{T}$-unsatisfiable whereas all finite sets of ground instances of formulae from Φ are $\mathcal{T}$-satisfiable. An example is $\Phi = \{(\forall x)(\neg p(x))\}$; even the set $\{\neg p(t) \mid t \in Term^0_\Sigma\}$ of *all* ground instances of Φ is $\mathcal{T}$-satisfiable (using a $\mathcal{T}$-structure where not all elements of the domain are represented by ground terms).

The restriction to universal theories is not a problem in practice, because it is easy to get around using Skolemization.

Key	$\mathcal{E}$-Refuter
$\{\neg(a \approx a)\}$	$\langle id \quad , \emptyset \rangle$
$\{\neg(x \approx a)\}$	$\langle \{x \mapsto a\}, \emptyset \rangle$
$\{(\forall x)(\neg(x \approx a))\}$	$\langle id \quad , \emptyset \rangle$
$\{p(a), \neg p(b)\}$	$\langle id \quad , \{\neg(a \approx b)\} \rangle$
$\{p(f(a), f(b)), f(x) \approx x\}$	$\langle \{x \mapsto a\}, \{p(a, f(b))\} \rangle$ $\langle \{x \mapsto b\}, \{p(f(a), b)\} \rangle$
$\{p(f(a), f(b)), (\forall x)(f(x) \approx x)\}$	$\langle id \quad , \{p(a, b)\} \rangle$

Table 1. Examples for $\mathcal{E}$-refuters.

EXAMPLE 21 An extension of $\mathcal{OP}$ that contains the density axiom

$$(\forall x)(\forall y)((x < y) \supset (\exists z)((x < z) \wedge (z < y)))$$

is not a universal theory. It can be made universal by replacing the above axiom with

$$(\forall x)(\forall y)((x < y) \supset ((x < between(x, y)) \wedge (between(x, y) < y))) \ .$$

2.4 Basic Definitions for Theory Reasoning

The following are the basic definitions for theory reasoning:

DEFINITION 22 *Let $\Phi \subset Form_\Sigma$ be a finite set of formulae, called* key. *A finite set $R = \{\rho_1, \ldots, \rho_k\} \subset Form_\Sigma$ of formulae ($k \geq 0$) is a* $\mathcal{T}$-residue *of Φ if there is a substitution $\sigma \in Subst^*_\Sigma$ such that*

1. *$\Phi\sigma \models^\circ_\mathcal{T} \rho_1 \vee \ldots \vee \rho_k$ (in case R is empty: $\Phi\sigma \models^\circ_\mathcal{T}$ false);*
2. *$R = R\sigma$.*

Then the pair $\langle \sigma, R \rangle$ is called a $\mathcal{T}$-refuter *for Φ. If the residue R is empty, the substitution σ is called a $\mathcal{T}$-refuter for Φ (it is identified with $\langle \sigma, \emptyset \rangle$).*

EXAMPLE 23 Table 1 shows some examples for $\mathcal{E}$-refuters.

DEFINITION 24 *A set $\Phi \subset Form_\Sigma$ of formulae is* $\mathcal{T}$-complementary *if, for all $\mathcal{T}$-structures $\langle D, I \rangle$ and all variable assignments ν, $val_{I,\nu}(\Phi) =$ false.*

EXAMPLE 25 The set $\{\neg(x \approx y)\}$ is $\mathcal{E}$-unsatisfiable; it is, however, not $\mathcal{E}$-complementary because a variable assignment may assign different elements of the domain to x and y. The set $\{\neg(x \approx x)\}$ is both $\mathcal{E}$-unsatisfiable and $\mathcal{E}$-complementary.

In general, it is undecidable whether a formula set is $\mathcal{T}$-complementary; and, consequently, it is undecidable whether a pair $\langle \sigma, R \rangle$ is a refuter for a key Φ.

$\mathcal{T}$-complementarity generalizes the usual notion that formulae ϕ and $\neg\phi$ are complementary. The following lemmata are immediate consequences of the definitions:

LEMMA 26 *Given a theory $\mathcal{T}$, a substitution $\phi \in Subst^*_\Sigma$ is a $\mathcal{T}$-refuter for a set Φ of formulae if and only if the set $\Phi\sigma$ is $\mathcal{T}$-complementary.*

LEMMA 27 *Given a theory $\mathcal{T}$, a substitution σ and a set $R = \{\rho_1, \ldots, \rho_k\}$, $k \geq 0$, of formulae form a refuter $\langle \sigma, R \rangle$ for a set Φ of formulae if and only if*

1. *$\Phi\sigma \cup \{\neg\rho_1, \ldots, \neg\rho_k\}$ is $\mathcal{T}$-complementary;*
2. *$R = R\sigma$.*

There is an alternative characterization of $\mathcal{T}$-complementary sets that do not contain bound variables (e.g., sets of literals or clauses):

THEOREM 28 *Given a theory $\mathcal{T}$, a set Φ of formulae that does not contain any quantifiers is $\mathcal{T}$-complementary if and only if the existential closure $\exists\Phi$ of Φ is $\mathcal{T}$-unsatisfiable.*

Provided that the signature Σ contains enough function symbols not occurring in a universal theory $\mathcal{T}$, a quantifier-free formula set is $\mathcal{T}$-complementary if all its instances are $\mathcal{T}$-complementary:

THEOREM 29 *Given a universal theory $\mathcal{T}$ such that there are infinitely many function symbols of each arity $n \geq 0$ in F_Σ that do not occur in $\mathcal{T}$, then a set Φ of formulae that does not contain any bound variables is $\mathcal{T}$-complementary if and only if all ground instances of Φ are $\mathcal{T}$-unsatisfiable.*

EXAMPLE 30 Let $\mathcal{T}$ be the theory $\{p(t) \mid t \in Term^0_\Sigma\}$ that violates the pre-condition of Theorem 29, as all function symbols occur in $\mathcal{T}$. The formula $\neg p(x)$ is *not* $\mathcal{T}$-complementary because there may be elements in the domain of a $\mathcal{T}$-structure that are not represented by any ground term. Nevertheless, all instances of $\neg p(x)$ are $\mathcal{T}$-unsatisfiable, which shows that the pre-condition of Theorem 29 is indispensable.

By definition there is no restriction on what formulae may occur in keys or refuters. In practice, however, to restrict the search space, background reasoners do not compute refuters for all kinds of keys, and they do not compute all possible refuters (typically, keys are restricted to be sets of literals or universally quantified literals). To model this, we define background reasoners to be partial functions on the set of all possible keys:

DEFINITION 31 *Let $\mathcal{T}$ be a theory; a* background reasoner *for $\mathcal{T}$ is a partial function*

$$\mathcal{R} : 2^{Form_\Sigma} \longrightarrow Subst^*_\Sigma \times 2^{Form_\Sigma}$$

such that, for all keys $\Phi \subset Form_\Sigma$ for which $\mathcal{R}$ is defined, $\mathcal{R}(\Phi)$ is a set of $\mathcal{T}$-refuters for Φ.

A background reasoner $\mathcal{R}$ is total *if, for all keys Φ for which $\mathcal{R}$ is defined, the residues of all refuters in $\mathcal{R}(\Phi)$ are empty, i.e., $\mathcal{R}(\Phi) \subset Subst^*_\Sigma$.*

A background reasoner $\mathcal{R}$ is monotonic *if, for all keys Φ and Ψ such that $\Phi \subset \Psi$: if $\mathcal{R}(\Phi)$ is defined, then $\mathcal{R}(\Psi)$ is defined and $\mathcal{R}(\Phi) \subset \mathcal{R}(\Psi)$.*

EXAMPLE 32 A background reasoner for the theory $\mathcal{PO}$ of partial orderings can be defined as follows: For all keys Φ, let $\mathcal{R}(\Phi)$ be the smallest set such that:

1. for all terms $t, t', t'' \in Term_\Sigma$:
   if $t < t', t' < t'' \in \Phi$, then $\langle id, t < t'' \rangle \in \mathcal{R}(\Phi)$;
2. for all terms $t \in Term_\Sigma$: if $t < t \in \Phi$, then $id \in \mathcal{R}(\Phi)$;
3. for all literals $\phi \in Lit_\Sigma$: if $\phi, \neg\phi \in \Phi$, then $id \in \mathcal{R}(\Phi)$.

The combination of $\mathcal{R}$ and the ground version of tableaux leads to a complete calculus for $\mathcal{PO}$ (see Sect. 3.2).

A background reasoner has to compute refuters that are strong consequences of (an instance of) the key. In contrary to that, for tableau rules it is sufficient to preserve satisfiability. A tableau rule may deduce $p(c)$ from $(\exists x)p(x)$ where c is new, but $\langle id, \{p(c)\}\rangle$ is not a refuter for the key $\{(\exists x)p(x)\}$. A background reasoner may, however, do the opposite: $\langle id, \{(\exists x)p(x)\}\rangle$ is a refuter for the key $\{p(c)\}$ (this deduction usually does not help in finding a proof; see, however, Example 20).

2.5 Total and Partial Theory Reasoning

The central idea behind theory reasoning is the same for all calculi based in some way on Herbrand's theorem (tableau-like calculi, resolution, etc.): A key $\Phi \subset \Psi$ is chosen from the set Ψ of formulae already derived by the foreground reasoner and is passed to the background reasoner, which computes refuters $\langle \sigma, R\rangle$ for Φ.

There are two main approaches: if the background reasoner is total, i.e., only computes refuters with an empty residue R, we speak of *total* theory reasoning else of *partial* theory reasoning.

In the case of partial reasoning, where the residue $R = \{\rho_1, \ldots, \rho_k\}$ is not empty ($k \geq 1$), the formula $\rho_1 \vee \ldots \vee \rho_k$ is added to the set Ψ of derived formulae and the substitution σ is applied. If the foreground reasoner is then able to show that for some substitution τ the set $(\Psi\sigma \cup \{\rho_1 \vee \ldots \vee \rho_k\})\tau$ is $\mathcal{T}$-unsatisfiable, this proves that $\Psi\sigma\tau$ is $\mathcal{T}$-unsatisfiable.

Although total theory reasoning can be seen as a special case of partial theory reasoning, the way the foreground reasoner makes use of the refuter is quite different: no further derivations have to be made by the foreground reasoner; $\Phi\sigma$ and thus $\Psi\sigma$ have been proven to be $\mathcal{T}$-complementary by the background reasoner. In the tableau framework, where (usually) the key Φ is taken from a tableau branch B, this means that B is closed if the substitution σ is applied.

On the one hand, for total theory reasoning, 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 for the foreground reasoner, because there are less refuters for each key and the search is more goal-directed.

2.6 Other Classifications of Theory Reasoning

Besides total and partial theory reasoning, there are several other ways to distinguish different types of background reasoners.

One possibility is to classify according to the information given to the background reasoner: (complex) formulae, literals, or terms [Baumgartner *et al.*, 1992]. Stickel distinguishes *narrow* theory reasoning, where all keys consist of literals, and *wide* theory reasoning, where keys consist of clauses [Stickel, 1985]. This type of classification is not used here, since all these types are subsumed by formula level theory reasoning. We will, however, restrict (nearly) all considerations to keys consisting of literals.

Another possibility is to classify background reasoners according to the type of calculus they use for deductions, the main divisions being *bottom up* and *top down* reasoning.

Local and *non-local* theory reasoning can be distinguished according to the effect that calling the background reasoner has on the tableau [Degtyarev and Voronkov, 1996a]. In particular, the effect of calling the background reasoner is local if only local variables are instantiated by applying the theory expansion or closure rule to a tableau branch B, i.e., no variables occurring on other branches than B are instantiated.

3 THEORY REASONING FOR SEMANTIC TABLEAUX

3.1 Unifying Notation

Following Smullyan [Smullyan, 1995], the set of formulae that are not literals is divided into four classes: α for formulae of conjunctive type, β for formulae of disjunctive type, γ for quantified formulae of universal type, and δ for quantified

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
$\phi \leftrightarrow \psi$	$\phi \wedge \psi$	$\neg\phi \wedge \neg\psi$
$\neg(\phi \leftrightarrow \psi)$	$\phi \wedge \neg\psi$	$\neg\phi \wedge \psi$

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

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

Table 2. Correspondence between formulae and rule types.

formulae of existential type (unifying notation). This classification is motivated by the *tableau expansion rules* which are associated with each (non-literal) formula.

DEFINITION 33 *The non-literal formulae in $Form_\Sigma$ are assigned a* type *according to Table 2. A formula of type $\xi \in \{\alpha, \beta, \gamma, \delta\}$ is called a* ξ-formula*.*

NOTATION 34 *The letters α, β, γ, and δ are used to denote formulae of (and only of) the appropriate type. In the case of γ- and δ-formulae the variable x bound by the (top-most) quantifier is made explicit by writing $\gamma(x)$ and $\gamma_1(x)$ (resp. $\delta(x)$ and $\delta_1(x)$); accordingly $\gamma_1(t)$ denotes the result of replacing all occurrences of x in γ_1 by t.*

3.2 The Ground Version of Semantic Tableaux

We first present the classical *ground* version of tableaux for first-order logic. This version of tableaux is called "ground", because universally quantified variables are replaced by *ground* terms when the γ-rule is applied.

The calculus is defined using a slightly non-standard representation of tableaux: a tableau is multi-sets of branches, which are multi-sets of first-order formulae; as usual, the branches of a tableau are implicitly disjunctively connected and the formulae on a branch are implicitly conjunctively connected. In graphical representations, tableaux are shown in their classical tree form.

DEFINITION 35 *A* tableau *is a (finite) multi-set of tableau branches, where a tableau* branch *is a (finite) multi-set of first-order formulae.*

In Table 3 the ground expansion rule schemata for the various formula types are given schematically. Premisses and conclusions are separated by a horizontal bar, while vertical bars in the conclusion denote different *extensions*. The formulae in

$\dfrac{\alpha}{\alpha_1 \\ \alpha_2}$	$\dfrac{\beta}{\beta_1 \mid \beta_2}$	$\dfrac{\gamma}{\gamma_1(t)}$	$\dfrac{\delta}{\delta_1(t)}$
		where t is any ground term.	where t is a ground term new to the tableau.

Table 3. Rule schemata for the ground version of tableaux.

an extension are implicitly conjunctively connected, and different extensions are implicitly disjunctively connected.

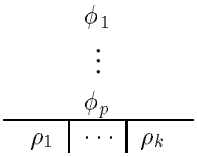
To prove a sentence ϕ to be a tautology, we apply the expansion rules starting from the initial tableau $\{\{\neg\phi\}\}$. A tableau T is expanded by choosing a branch B of T and a formula $\phi \in B$ and replacing B by as many updated branches as the rule corresponding to ϕ has extensions. Closed branches are removed from the tableau instead of just marking them as being closed; thus, a proof is found when the empty tableau has been derived.

There is a theory expansion and a theory closure rule. For both rules, a key $\Phi \subset B$ is chosen from a branch B, and a refuter $\langle \sigma, R \rangle$ for Φ is computed. Since formulae in ground tableaux do not contain free variables, the formulae in the residue, too, have to be sentences. The application of substitutions to formulae without free variables does not have any effect. Thus, if $\langle \sigma, R \rangle$ is a refuter for a key Φ taken from a ground tableau, then $\langle id, R \rangle$ is a refuter for Φ as well; thus, it is possible to use only refuters of this form for ground tableaux.

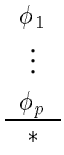
DEFINITION 36 *A background reasoner $\mathcal{R}$ for a theory $\mathcal{T}$ is a* ground background reasoner *for $\mathcal{T}$ if, for all keys $\Phi \subset Form_\Sigma$ for which $\mathcal{R}$ is defined, all formulae in $\mathcal{R}(\Phi)$ are sentences, i.e., do not contain free variables.*

Whether an expansion or a closure rule is to be applied depends on whether the residue $R = \{\rho_1, \ldots, \rho_k\}$ is empty or not. If $k \geq 1$, then the tableau is expanded. The old branch is replaced by k new branches, one for each ρ_i (since the ρ_i are implicitly disjunctively connected). The closure rule is applied if the residue is empty ($k = 0$); it can be seen as a special case of the expansion rule: the old branch is replaced by 0 new branches, i.e., it is removed. The rule schemata are shown in Table 4; in this and all following schematic representations of rules, the symbol $*$ is used to denote that a branch is closed if the rule is applied.

If the residue R is empty, the key is $\mathcal{T}$-complementary and the branch it has been taken from is $\mathcal{T}$-closed. This is a straightforward extension of the closure rule for tableaux without theory reasoning, where a branch is closed if it contains complementary formulae ϕ and $\neg\phi$, i.e., the $\emptyset$-complementary key $\{\phi, \neg\phi\}$ (therefore, the rule that a branch containing complementary formulae ϕ and $\neg\phi$ is closed does not have to be considered separately).



where $\langle id, \{\rho_1, \ldots, \rho_k\}\rangle$ ($k \geq 1$) is a refuter for the key $\{\phi_1, \ldots, \phi_p\}$, and $\rho_1, \ldots, \rho_k$ do not contain free variables.



where id is a refuter for the key $\{\phi_1, \ldots, \phi_p\}$.

Table 4. Theory expansion and closure rules (ground version).

DEFINITION 37 (Ground tableau proof.) *Given a theory $\mathcal{T}$ and a ground background reasoner $\mathcal{R}$ for $\mathcal{T}$ (Def. 36), a* ground tableau proof *for a first-order sentence $\phi \in Form_\Sigma$ consists of a sequence*

$$\{\{\neg\phi\}\} = T_0, T_1, \ldots, T_{n-1}, T_n = \emptyset \qquad (n \geq 0)$$

of tableaux such that, for $1 \leq i \leq n$, the tableau T_i is constructed from T_{i-1}

1. *by applying one of the expansion rules for ground tableaux from Table 3, i.e., there is a branch $B \in T_{i-1}$ and a formula $\phi \in B$ (that is not a literal) such that*

$$T_i \;=\; (T_{i-1} \setminus \{B\}) \cup \begin{cases} \{(B \setminus \{\alpha\}) \cup \{\alpha_1, \alpha_2\}\} & \text{if } \phi = \alpha \\ \{(B \setminus \{\beta\}) \cup \{\beta_1\}, (B \setminus \{\beta\}) \cup \{\beta_2\}\} & \text{if } \phi = \beta \\ \{B \cup \{\gamma_1(s)\}\} & \text{if } \phi = \gamma(x) \\ \{(B \setminus \{\delta(x)\}) \cup \{\delta_1(t)\}\} & \text{if } \phi = \delta(x) \end{cases}$$

   where $s \in Term^0_\Sigma$ is any ground term, and $t \in Term^0_\Sigma$ ground term not occurring in T_{i-1};

2. *by applying the ground theory expansion rule, i.e., there is a branch $B \in T_{i-1}$ and, for a key $\Phi \subset B$, there is a $\mathcal{T}$-refuter $\langle id, \{\rho_1, \ldots, \rho_k\}\rangle$ ($k \geq 1$) in $\mathcal{R}(\Phi)$ such that*

$$T_i = (T_{i-1} \setminus \{B\}) \cup \{B \cup \{\rho_1\}, \ldots, B \cup \{\rho_k\}\} \;;$$

3. *or by closing a branch $B \in T_{i-1}$, i.e., $T_i = T_{i-1} \setminus \{B\}$ where B is $\mathcal{T}$-closed (i.e., $id \in \mathcal{R}(\Phi)$ for a key $\Phi \subset B$).*

It is possible to describe a background reasoner using tableau rule schemata. The reasoner from Example 32 then takes the form that is shown in Table 5.

$$\frac{t < t' \quad t' < t''}{t < t''} \qquad \frac{t < t}{*} \qquad \frac{\phi \quad \neg\phi}{*}$$

Table 5. Expansion and closure rules for the theory $\mathcal{PO}$ of partial orderings.

Even without theory reasoning, the construction of a closed tableau is a highly non-deterministic 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 non-determinism (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).

The theory expansion rule makes things even worse, because it is highly non-deterministic. In which way it has to be applied to be of any use, in particular when and how often the rule is applied, and which types of keys and refuters are used depends on the particular theory and is part of the domain knowledge (see Sect. 5.3).

3.3 Free Variable Semantic Tableaux

Using free variable quantifier rules is crucial for efficient implementation—even more so if a theory 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, which has to be "guessed". Free variables can later be instantiated "on demand" when a tableau branch is closed or the theory expansion rule 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 6).

Again, there is both a theory expansion and a theory closure rule. The difference to the ground version is that now there are free variables both in the tableau and in the refuter (the formulae that are added). When theory reasoning is used for expansion or for closing, the substitution σ of a refuter $\langle \sigma, R \rangle$ has to be applied to the whole tableau; the theory rule schemata are shown in Table 7.

In case there is a refuter σ with an empty residue for a key Φ taken from a branch, it is $\mathcal{T}$-closed under the substitution σ, i.e., it is closed when σ is applied

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

Table 6. Tableau expansion rule schemata for free variable tableau.

$\dfrac{\begin{array}{c}\phi_1\\ \vdots\\ \phi_p\end{array}}{\rho_1 \mid \cdots \mid \rho_k}$	$\dfrac{\begin{array}{c}\phi_1\\ \vdots\\ \phi_p\end{array}}{*}$
where $\langle\sigma,\{\rho_1,\ldots,\rho_k\}\rangle$ ($k \geq 1$) is a refuter for the key $\{\phi_1,\ldots,\phi_p\}$, and σ is applied to the whole tableau.	where σ is a refuter for the key $\{\phi_1,\ldots,\phi_p\}$, and σ is applied to the whole tableau.

Table 7. Theory expansion and closure rules (free variable version).

to the whole tableau.

It is often difficult to find a substitution σ that instantiates the variables in a tableau T such that *all* branches of T are $\mathcal{T}$-closed. The problem is simplified (as is usually done in practice) by 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. This is not a restriction because, if a substitution is known to $\mathcal{T}$-close several branches, it can be applied to close one of them; after that the other branches are closed under the empty substitution.

DEFINITION 38 (Free variable tableau proof.) *Let $\mathcal{T}$ be a theory and let $\mathcal{R}$ be a background reasoner for $\mathcal{T}$, a* free variable tableau proof *for a first-order sentence ϕ consists of a sequence*

$$\{\{\neg\phi\}\} = T_0, T_1, \ldots, T_{n-1}, T_n = \emptyset \qquad (n \geq 0)$$

of tableaux such that, for $1 \leq i \leq n$, the tableau T_i is constructed from T_{i-1}

1. *by applying one of the free variable expansion rules from Table 6, that is, there is a branch $B \in T_{i-1}$ and a formula $\phi \in B$ (that is not a literal) such that*

$$T_i \quad = \quad (T_{i-1} \setminus \{B\})$$

$$\cup \begin{cases} \{(B \setminus \{\alpha\}) \cup \{\alpha_1, \alpha_2\}\} & \text{if } \phi = \alpha \\ \{(B \setminus \{\beta\}) \cup \{\beta_1\}, (B \setminus \{\beta\}) \cup \{\beta_2\}\} & \text{if } \phi = \beta \\ \{B \cup \{\gamma_1(y)\}\} & \text{if } \phi = \gamma(x) \\ \{(B \setminus \{\delta(x)\}) \cup \{\delta_1(f(x_1, \ldots, x_n))\}\} & \text{if } \phi = \delta(x) \end{cases}$$

where $y \in V$ is a new variable not occurring in T_{i-1}, $f \in F_\Sigma$ is a Skolem function symbol not occurring in T_{i-1}, and $x_1, \ldots, x_n$ are the free variables in ϕ;

2. *by applying the free variable theory expansion rule, that is, there is a branch $B \in T_{i-1}$, a key $\Phi \subset B$, and a $\mathcal{T}$-refuter $\langle \sigma, \{\rho_1, \ldots, \rho_k\}\rangle$ ($k \geq 1$) in $\mathcal{R}(\Phi)$ such that*

$$T_i = (T_{i-1} \setminus \{B\})\sigma \cup \{B\sigma \cup \{\rho_1\}, \ldots, B\sigma \cup \{\rho_k\}\} \; ;$$

3. *or by closing a branch $B \in T_{i-1}$, that is, $T_i = (T_{i-1} \setminus \{B\})\sigma$ where the branch B is $\mathcal{T}$-closed under σ, i.e., $\sigma \in \mathcal{R}(\Phi)$ for a key $\Phi \subset B$.*

3.4 Semantic Tableaux with Universal Formulae

Free variable semantic tableaux can be further improved by using the concept of universal formulae [Beckert and Hähnle, 1992]: γ-formulae (in particular axioms that extend the theory) have often to be used multiply in a tableau proof, with different instantiations for the free variables they contain. An example is the axiom $(\forall x)(\forall y)((x < y) \supset (p(x) \supset p(y)))$, that extends the theory of partial orderings by defining the predicate symbol p to be monotonous. The associativity axiom $(\forall x)(\forall y)(\forall z)((x \cdot y) \cdot z \approx x \cdot (y \cdot z))$ is another typical example. Usually, it has to be applied several times with different substitutions for x, y and z to prove even very simple theorems from, for example, group theory. 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. Free variables in tableaux are *not* implicitly universally quantified (as it is, for instance, the case with variables in clauses when using a resolution calculus) but are *rigid*, which is the reason why a substitution must be applied to all occurrences of a free variable in the whole tableau.

Supposed a tableau branch B contains a formula $p(x)$, and the expansion of the tableau proceeds with creating new branches. Some of these branches contain occurrences of x; for closing the generated branches, the same substitution for x has to be used on all of them. Figure 1 gives an example for the situation: this tableau for cannot be closed immediately as no single substitution closes both branches. To find a proof, the γ-rule has to be applied again to create another instance of $(\forall x)p(x)$.

In particular situations, a logical consequence of the formulae already on the tableau (in a sense made precise in Def. 39) may be that $(\forall y)p(y)$ can be added

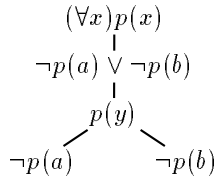


Figure 1. The advantage of using universal formulae.

to B. This is trivially true in Figure 1. In such cases, different substitutions for y can be used without destroying soundness of the calculus. The tableau in Figure 1 then closes immediately. Recognizing such situations and exploiting them allows to use more general closing substitutions, yields shorter tableau proofs, and in most cases reduces the search space.

DEFINITION 39 *Let $\mathcal{T}$ be a theory, and let ϕ be a formula on a branch B of a tableau T. Let T' be the tableau that results from adding $(\forall x)\phi$ to B for some $x \in V$. The formula ϕ is* $\mathcal{T}$-universal *on B with respect to x if $T \models_{\mathcal{T}} T'$, where T and T' are identified with the formulae that are the disjunctions of their branches, respectively, and a branch is the conjunction of the formulae it contains. Let $UVar(\phi) \subset V$ denote the universal variables of ϕ.*[2]

The above definition is an adaptation of the definition given in [Beckert and Hähnle, 1998] to theory reasoning.

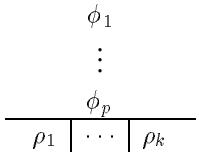
The problem of recognizing universal formulae is of course undecidable in general. However, a wide and important class can be recognized quite easily (using this class can already shorten tableau proofs exponentially): If tableaux are seen as trees, a formula ϕ on a branch B of a tableau T is ($\mathcal{T}$-)universal w.r.t. x if all branches B' of T are closed which contain an occurrence of x that is not on B as well; this holds in particular if the branch B contains all occurrences of x in T.

Assume there is a sequence of tableau rule applications that introduces a variable x by a γ-rule application and does not contain a rule application distributing x over different subbranches; then the above criterion is obviously satisfied and all formulae that are generated by this sequence are universal w.r.t. x.

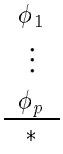
THEOREM 40 *Given a theory $\mathcal{T}$, a formula ϕ on a branch B of a tableau T is $\mathcal{T}$-universal w.r.t. x on B if in the construction of T the formula ϕ was added to B by either*

1. *applying a γ-rule and x is the free variable that was introduced by that application;*

[2] When the context is clear, a formula ϕ which is universal on a branch B w.r.t. a variable x is just referred to by "the universal formula ϕ," and the variable x by "the universal variable x."



where $\langle \sigma, \{\rho_1, \ldots, \rho_k\} \rangle$ ($k \geq 1$) is a refuter for the key $\{(\forall x^i_1) \cdots (\forall x^i_{m_i})\phi_i \mid 1 \leq i \leq p\}$, the formula ϕ_i is $\mathcal{T}$-universal w.r.t. the variables $x^i_1, \ldots, x^i_{m_i}$; and σ is applied to the whole tableau.



where σ is a refuter for the key $\{(\forall x^i_1) \cdots (\forall x^i_{m_i})\phi_i \mid 1 \leq i \leq p\}$, the formula ϕ_i is $\mathcal{T}$-universal w.r.t. the variables $x^i_1, \ldots, x^i_{m_i}$; and σ is applied to the whole tableau.

Table 8. Theory expansion and closure rules (universal formula version).

2. *applying an α-, δ- or γ-rule to a formula ψ that is $\mathcal{T}$-universal w.r.t. x on B;*
3. *applying a β-rule to a formula ψ that is $\mathcal{T}$-universal w.r.t. x on B, and x does not occur in any formula except ϕ that has been added to the tableau by that β-rule application;*
4. *applying the theory expansion rule to a refuter $\{\rho_1, \ldots, \rho_k\}$ for a key $\Phi \subset B$ (i.e., $\phi = \rho_i$ for some $i \in \{1, \ldots, k\}$), and all formulae in Φ are $\mathcal{T}$-universal w.r.t. x on B, and x does not occur in any of the ρ_j for $j \neq i$.*

The knowledge that formulae are $\mathcal{T}$-universal w.r.t. variables they contain can be taken advantage of by universally quantifying the formulae in a key w.r.t. (some of) their universal variables. Similar to the ground and free variable cases, the closure rule can be seen as a special case of the expansion rule. The new theory rule schemata are shown in Table 8.

DEFINITION 41 (Universal formula version.) *Let $\mathcal{T}$ be a theory, and let $\mathcal{R}$ be a background reasoner for $\mathcal{T}$. A* universal formula tableau proof *for a first-order sentence ϕ consists of a sequence*

$$\{\{\neg\phi\}\} = T_0, T_1, \ldots, T_{n-1}, T_n = \emptyset \qquad (n \geq 0)$$

of tableaux such that, for $1 \leq i \leq n$, the tableau T_i is constructed from T_{i-1}

1. *by applying one of the free variable expansion rules from Table 6 (see Theorem 38 for a formal definition);*
2. *by applying the universal formula theory expansion rule to a branch $B \in T_{i-1}$, that is,*

$$T_i = (T_{i-1} \setminus \{B\})\sigma \cup \{B\sigma \cup \{\rho_1\}, \ldots, B\sigma \cup \{\rho_k\}\}$$

where $\langle \sigma, \{\rho_1, \ldots, \rho_k\}\rangle$ *is a* $\mathcal{T}$*-refuter in* $\mathcal{R}(\Phi)$ *for a key*

$$\Phi = \{(\forall x_1^i) \cdots (\forall x_{m_i}^i)\phi_i \mid 1 \leq i \leq p\}$$

such that

(a) $\phi_1, \ldots, \phi_p \in B$,

(b) $\{x_1^i, \ldots, x_{m_i}^i\} \subset UVar(\phi_i)$ *for* $1 \leq i \leq p$,

3. *or by closing a branch* $B \in T_{i-1}$*, that is,*

$$T_i = (T_{i-1} \setminus \{B\})\sigma$$

where the branch B *is* $\mathcal{T}$*-closed under* σ*, i.e.,* $\sigma \in \mathcal{R}(\Phi)$ *for a key*

$$\Phi = \{(\forall x_1^i) \cdots (\forall x_{m_i}^i)\phi_i \mid 1 \leq i \leq p\}$$

such that

(a) $\phi_1, \ldots, \phi_p \in B$,

(b) $\{x_1^i, \ldots, x_{m_i}^i\} \subset UVar(\phi_i)$ *for* $1 \leq i \leq p$.

Although it is easier to add theory reasoning to the ground version of tableau, to prove even simple theorems, free variable tableaux have to be used. These are sufficient for simple theories. If, however, finding a refuter requires complex deductions where the formulae both in the key and in the theory have to be used multiply with different instantiations, then universal formula tableau have to be used (free variable tableaux are a special case of universal formula tableau),

The following example illustrates the advantage of using the universal formula expansion rule as compared to the free variable rule:

EXAMPLE 42 Consider again the tableau T shown in Figure 1. The substitution $\sigma = \{y \mapsto a\}$ is a refute for the key $\{p(y), \neg p(a)\}$, which is taken from the left branch of T. If σ is used to close the left branch, then the variable y is instantiated with a in the whole tableau T. However, if the formula $p(y)$ is recognized to be universal w.r.t. y, then the key $\{(\forall y)p(y), \neg p(a)\}$ can be used instead, for which the empty substitution id is a refuter; thus using the universal formula closure rule, the left branch can be closed without instantiating y. Then the right branch can be closed, too, without generating a second free variable instance $p(y')$ of $(\forall x)p(x)$.

Using the universal formula technique is even more important in situations like the following:

EXAMPLE 43 Supposed the equality $f(x) \approx x$ and the literals $p(f(a), f(b))$ and $\neg p(a, b)$ are $\mathcal{E}$-universal w.r.t. x on a tableau branch. In that case, the key $\{(\forall x)(f(x) \approx x),\ p(f(a), f(b))$ can be used, for which id is a refuter.

In free variable tableaux, the key $\{(f(x) \approx x),\ p(f(a), f(b)),\ p(a, b)\}$ has to be used, which allows to derive the refuters $\langle\{x \mapsto a\}, \{f(b) \approx b\}\rangle$ and $\langle\{x \mapsto b\}, \{f(a) \approx a\}\rangle$ only; a refuter with an empty residue, which closes the branch immediately, cannot be deduced anymore.

4 SOUNDNESS

In this section, soundness of semantic tableaux with theory reasoning is proven for the universal formula version. Soundness of the free variable version follows as a corollary because free variable tableaux are a special case of universal formula tableaux. For the ground version, soundness can be proven completely analogously.

First, satisfiability of tableaux is defined (Def. 44), then it is proven that satisfiability is preserved in a sequence of tableaux forming a tableau proof (Lemma 46).

DEFINITION 44 *Given a theory $\mathcal{T}$, a tableau T is* $\mathcal{T}$-satisfiable *if there is a $\mathcal{T}$-structure M such that, for every variable assignment ν, there is a branch $B \in T$ with*

$$(M, \nu) \models B\ .$$

LEMMA 45 *If a tableau T is $\mathcal{T}$-satisfiable, then $T\sigma$ is $\mathcal{T}$-satisfiable for all substitutions $\sigma \in Subst^*_\Sigma$.*

Proof. By hypothesis there is a $\mathcal{T}$-structure $M = \langle D, I\rangle$ such that, for all variable assignments ν, there is a branch $B_\nu \in T$ with $(M, \nu) \models B_\nu$. We claim that, for the same structure M, we have also for all variable assignments ξ that there is a branch B with $(M, \xi) \models B\sigma$.

To prove the above claim, we consider a given variable assignment ξ. Let the variable assignment ν be defined by

$$\nu(x) = (x\sigma)^{I,\xi} \text{ for all } x \in V\,.$$

That implies for all terms $t \in Term_\Sigma$, and in particular for all terms t occurring in B_ν,

$$(t\sigma)^{I,\xi} = t^{I,\nu}$$

and therefore

$$val_{I,\xi}(B_\nu\sigma) = val_{I,\nu}(B_\nu) = \textit{true}\ .$$

■

LEMMA 46 *Given a universal formula tableau proof $(T_j)_{0 \le j \le n}$, if the tableau T_i ($0 \le i < n$) is $\mathcal{T}$-satisfiable, then the tableau T_{i+1} is $\mathcal{T}$-satisfiable as well.*

Proof. We use the notation from Definition 41. Let B be the branch in T_i to which one of the classical expansion rules or the theory expansion rule has applied or that has been removed by applying the theory closure rule to derive the tableau T_{i+1}. Let $M = \langle D, I\rangle$ be a $\mathcal{T}$-structure satisfying T_i.

β-rule: Let ν be an arbitrary variable assignment. There has to be a branch B' in T_i such that $(M, \nu) \models B'$. If B' is different from B then $B' \in T_{i+1}$ and we are through.

If, on the other hand, $B' = B$, then $(M, \nu) \models B$. Let β be the formula in B to which the β-rule has been applied. By the property of β-formulae, $(M, \nu) \models \beta$ entails $(M, \nu) \models \beta_1$ or $(M, \nu) \models \beta_2$; and, therefore, $(M, \nu) \models (B \setminus \{\beta\}) \cup \{\beta_1\}$ or $(M, \nu) \models (B \setminus \{\beta\}) \cup \{\beta_2\}$. This concludes the proof for the case of a β-rule application, because $(B \setminus \{\beta\}) \cup \{\beta_1\}$ and $(B \setminus \{\beta\}) \cup \{\beta_2\}$ are branches in T_{i+1}.

α- and γ-rule: Similar to the β-rule.

δ-rule: Let δ be the δ-formula to which the δ-rule is applied to derive T_{i+1} from T_i; $\delta_1(f(x_1, \ldots, x_m))$ is the formula added to the branch (f is a new Skolem function symbol and $x_1, \ldots, x_m$ are the free variables in δ). We define a structure $M' = \langle D, I'\rangle$ that is identical to M, except that the new function symbol f is interpreted by I' in the following way: For every set $d_1, \ldots, d_m$ of elements from the domain D, if there is an element d such that $(M, \xi) \models \delta_1(x)$ where $\xi(x_j) = d_j$ $(1 \leq j \leq m)$ and $\xi(x) = d$, then $f^{I'}(d_1, \ldots, d_m) = d$. If there are several such elements d, one of them may be chosen; and if there is no such element, an arbitrary element from the domain is chosen. It follows from this construction that for all variable assignments ν: if $(M, \nu) \models \delta$, then $(M', \nu) \models \delta_1(f(x_1, \ldots, x_m))$. Since f does not occur in $\mathcal{T}$, M' is a $\mathcal{T}$-structure.

We proceed to show that M' satisfies T_{i+1}. Let ν be an arbitrary variable assignment. There has to be a branch B' in T_i with $(M, \nu) \models B'$. If B' is different from B, then we are done because $(M', \nu) \models B'$ (as f does not occur in B') and $B' \in T_{i+1}$.

In the interesting case where $\delta \in B' = B$, we have $(M, \nu) \models \delta$ which entails $(M', \nu) \models \delta_1(f(x_1, \ldots, x_m))$. Thus, $(B \setminus \{\delta\}) \cup \{\delta_1(f(x_1, \ldots, x_m))\}$, which is a branch in T_{i+1}, is satisfied by M'.

Theory Expansion Rule: Let $\langle \sigma, \{\rho_1, \ldots, \rho_k\}\rangle\}$ be the refuter used to expand the tableau. Since T_i is $\mathcal{T}$-satisfiable, the tableau $T_i\sigma$ is $\mathcal{T}$-satisfiable as well (Lemma 45). Let M be a $\mathcal{T}$-structure satisfying $T_i\sigma$, and let ν be an arbitrary variable assignment. There has to be a branch $B' \in T_i\sigma$ with $(M, \nu) \models B'$. Again, the only interesting case is where $B' = B\sigma$, and $B\sigma$ is the only branch in $T_i\sigma$ satisfied by (M, ν).

By definition of universal formulae, that implies $B \models^{\circ}_{\mathcal{T}} (\forall x^j_1) \cdots (\forall x^j_{m_j})\, \phi_j$ $(1 \leq j \leq p)$ and, thus, $B\sigma \models^{\circ}_{\mathcal{T}} (\forall x^j_1) \cdots (\forall x^j_{m_j})\, \phi_j\sigma$ (Lemma 9), which implies $(M, \nu) \models \Phi\sigma$ where $\Phi = \{(\forall x^j_1) \cdots (\forall x^j_{m_j})\, \phi_j \mid 1 \leq j \leq p\}$, as $(M, \nu) \models B\sigma$. Because $\langle \sigma, \{\rho_1, \ldots, \rho_k\}\rangle$ is a refuter for Φ and, thus, $\Phi\sigma \models^{\circ}_{\mathcal{T}} \rho_1 \vee \ldots \vee \rho_k$, we

have $(M, \nu) \models \rho_j$ for some $j \in \{1, \ldots, k\}$. This, finally, implies that M satisfies the branch $B\sigma \cup \{\rho_j\}$ in T_{i+1}.

Theory Closure Rule: In the same way as in the case of the theory expansion rule, we conclude that $(M, \nu) \models \Phi\sigma$. But now this leads to a contradiction: because the residue is empty, $val_{I,\nu}(\Phi\sigma) = \mathit{false}$ by definition. Thus the assumption that $B' = B\sigma$ has to be wrong, and the branch $B\sigma$ can be removed from the tableau. ■

Based on this lemma, soundness of semantic tableaux with theory reasoning can easily be proven:

THEOREM 47 *If there is a universal formula tableau proof*

$$\{\{\neg\phi\}\} = T_0, T_1, \ldots, T_{n-1}, T_n = \emptyset$$

for a sentence $\phi \in Form_\Sigma$ (Def. 41), then ϕ is a $\mathcal{T}$-tautology.

Proof. None of the tableaux in the sequence which the tableau proof consists of can be $\mathcal{T}$-satisfiable, otherwise the empty tableau $T_n = \emptyset$ had to be $\mathcal{T}$-satisfiable as well (according to Lemma 46); this, however, is impossible because the empty tableau has no branches.

Thus, the first tableau $\{\{\neg\phi\}\}$ in the sequence is $\mathcal{T}$-unsatisfiable, i.e., $\neg\phi$ is $\mathcal{T}$-unsatisfiable, which is equivalent to ϕ being a $\mathcal{T}$-tautology. ■

5 COMPLETENESS

5.1 Complete Background Reasoners

The most important feature of a background reasoner is completeness—besides soundness which is part of the definition of background reasoners. We define a background reasoner to be complete if its combination with the foreground reasoner leads to a complete calculus.

DEFINITION 48 *A (ground) background reasoner for a theory $\mathcal{T}$ is*

- *a* complete ground *background reasoner for $\mathcal{T}$ if, for every $\mathcal{T}$-tautology ϕ, a ground tableau proof (Def. 37) can be built using $\mathcal{R}$.*
- *a* complete free variable *background reasoner for $\mathcal{T}$ if, for every $\mathcal{T}$-tautology ϕ, a free variable tableau proof (Def. 38) can be built using $\mathcal{R}$.*
- *a* complete universal formula *background reasoner for $\mathcal{T}$ if, for every $\mathcal{T}$-tautology ϕ, a universal formula tableau proof (Def. 41) can be built using $\mathcal{R}$.*

Because free variable tableaux are a special case of universal formula tableaux, a complete universal variable background reasoner has to be a complete free variable background reasoner as well.

The existence of complete background reasoners is trivial, because an oracle-like background reasoner that detects all kinds of inconsistencies (and thus does all the work) is complete for all versions of tableau:

THEOREM 49 *Let $\mathcal{T}$ be a theory. If a background reasoner $\mathcal{R}$ satisfies the condition:*

$$id \in \mathcal{R}(\{\phi\})$$

for all $\mathcal{T}$-unsatisfiable sentences $\phi \in Form_\Sigma$, then $\mathcal{R}$ is a complete ground, free variable, and universal variable background reasoner.

Proof. If ϕ is a $\mathcal{T}$-tautology, then its negation $\neg\phi$ is $\mathcal{T}$-unsatisfiable and thus $id \in \mathcal{R}(\{\neg\phi\})$. By applying the theory closure rule using this refuter, the empty tableau $T_1 = \emptyset$ can be derived from the initial tableau $T_0 = \{\{\neg\phi\}\}$. ■

This completeness result is only of theoretical value. In practice, theory reasoners are needed that, on the one hand, lead to short tableau proofs and, on the other hand, can be computed easily (i.e., fast and at low cost). Of course, there is a trade off between these two goals.

There is a complete background reasoner for a theory $\mathcal{T}$ such that $\mathcal{R}(\Phi)$ is enumerable for all keys Φ if and only if the theory $\mathcal{T}$ is axioamatizable. Thus, it is not possible to implement a complete background reasoner for a non-axiomatizable theory.

5.2 Completeness Criteria

General completeness criteria that work for all theories such as "a background reasoner that computes all existing refuters is complete" are not useful in practice. For many theories, and in particular for equality, highly specialized background reasoners have to be used to build an efficient prover. These exploit domain knowledge to restrict the number of refuters (and thus the search space); domain knowledge has to be used to prove such background reasoners to be complete.

Therefore, the completeness criteria presented in the following refer to the semantics of the particular theory, and there is no uniform way to prove that a background reasoner satisfies such a criterion. Nevertheless, the criteria give some insight in what has to be proven to show completeness of a background reasoner.

Fairness of the Foreground Reasoner

First, a characterization of fair tableau construction rules (i.e., fairness of the foreground reasoner) is given for the ground version. This notion is used in the proof

that a background reasoner that satisfies a completeness criterion can be combined with a fair foreground reasoner to form a complete calculus.

For the multi-set representation of tableaux, the notion of fair tableau construction is somewhat more difficult to formalize than for the tree representation, but this has no effect on which construction rules are fair.

The condition for α-, β-, and δ-formulae is that the respective tableau rule is applied sooner or later. The γ-rule has to be applied to each γ-formula infinitely often, and—this is specific for the ground version—each ground term has to be used for one of these applications. The background reasoner has to be called with all keys for which it is defined and all refuters have to be used sooner or later.

DEFINITION 50 *Given a ground background reasoner $\mathcal{R}$, a* ground tableau construction rule *for $\mathcal{R}$ is a rule that, when supplied with a formula ϕ, deterministically specifies in which way a sequence $(T_i)_{i\geq 0}$ of ground tableaux starting from $T_0 = \{\{\neg\phi\}\}$ is to be constructed. The rule is* fair *if for all Φ:*

1. *If there is a branch B that occurs in all tableaux T_i, $i > n$ for some $n \geq 0$, then B is exhausted, i.e., no rule expansion or closure rule can be applied to B.*
2. *For all infinite sequences $(B_i)_{i\geq 0}$ of branches such that $B_i \in T_i$ and either $B_{i+1} = B_i$ or the tableau T_{i+1} has been constructed from T_i by applying an expansion rule to B_i and B_{i+1} is one of the resulting new branches in T_{i+1} ($i \geq 0$):*
   (a) *for all α-, β-, and δ-formulae $\phi \in B_i$ ($i \geq 0$), there is a $j \geq i$ such that the tableau T_{j+1} has been constructed from T_j by applying the α-, β-, and δ-rule to $\phi \in B_j$.*
   (b) *for all γ-formulae $\phi \in B_i$ ($i \geq 0$) and all terms $t \in Term^0_\Sigma$, there is a $j \geq 0$ such that the tableau T_{j+1} has been constructed from T_j by applying the γ-rule to $\phi \in B_j$ and t is the ground term that has been substituted for the universally quantified variable in ϕ.*
   (c) *for all keys $\Phi \subset B_i$ ($i \geq 0$) such that $\mathcal{R}(\Phi)$ is defined and all refuters $\langle id, R\rangle \in \mathcal{R}(\Phi)$, there is a $j \geq 0$ such that the tableau T_{j+1} has been constructed from T_j by applying the theory expansion or the theory closure rule to B_j using the key $\Phi \subset B_j$ and the refuter $\langle id, R\rangle$ (if the background reasoner is monotonic, a key $\Phi' \supset \Phi$ may be used as well).*

A Completeness Criterion for Ground Background Reasoners

The criterion we are going to prove is that a background reasoner is complete if, for all $\mathcal{T}$-unsatisfiable downward saturated sets Ξ, it can either derive a residue consisting of new formulae that are not yet in Ξ, or it can detect the $\mathcal{T}$-unsatisfiability of Ξ.

DEFINITION 51 *A set $\Phi \subset Form_\Sigma$ is downward saturated if the following conditions hold for all formulae $\phi \in \Phi$ that are not a literal:*

1. *if $\phi = \alpha$, then $\alpha_1, \alpha_2 \in \Phi$;*
2. *if $\phi = \beta$, then $\beta_1 \in \Phi$ or $\beta_2 \in \Phi$;*
3. *if $\phi = \delta(x)$, then $\delta_1(t) \in \Phi$ for some term $t \in Term^0_\Sigma$;*
4. *if $\phi = \gamma(x)$, then $\gamma_1(t) \in \Phi$ for all terms $t \in Term^0_\Sigma$.*

THEOREM 52 *A ground background reasoner $\mathcal{R}$ for a theory $\mathcal{T}$ is complete if, for all (finite or infinite) $\mathcal{T}$-unsatisfiable downward saturated sets $\Xi \subset Form_\Sigma$ that do not contain free variables:*

1. *there is a key $\Phi \subset \Xi$ such that $id \in \mathcal{R}(\Phi)$; or*
2. *there is a key $\Phi \subset \Xi$ such that there is a refuter $\langle id, \{\rho_1, \ldots, \rho_k\}\rangle$ in $\mathcal{R}(\Phi)$ with $\{\rho_1, \ldots, \rho_k\} \cap \Xi = \emptyset$ ($k \geq 1$).*

Proof. Let $\mathcal{R}$ be a background reasoner satisfying the criterion of the theorem, and let ϕ be a $\mathcal{T}$-tautology. We prove that, using $\mathcal{R}$ and an arbitrary fair ground tableau construction rule (Def. 50), a tableau proof for ϕ is constructed. Let $(T_i)_{i\geq 0}$ be the sequence of ground tableaux that is constructed according to the fair rule starting from $T_0 = \{\{\neg\phi\}\}$.

Supposed $(T_i)_{i\geq 0}$ is not a tableau proof. If the sequence is finite, then there has to be at least one finite exhausted branch B^* in the final tableau T_n. If the sequence is infinite, then there is a sequence $(B_i)_{i\geq 0}$, $B_i \in T_i$, of branches as described in Condition 2 in the definition of fairness (Def. 50). In that case, $B^* = \bigcup_{i\geq 0} B_i$ is the union of these branches. We proceed to prove that the set B^* is $\mathcal{T}$-satisfiable. Because of the fairness conditions, B^* is downward saturated. If it were $\mathcal{T}$-unsatisfiable, then there had to be a key $\Phi \subset B^*$ such that $id \in \mathcal{R}(\Phi)$ or such that there is a refuter $\langle id, \{\rho_1, \ldots, \rho_k\}\rangle \in \mathcal{R}(\Phi)$ ($k \geq 1$) where $\{\rho_1, \ldots, \rho_k\} \cap B^* = \emptyset$. Because keys are finite, there then had to be an $i \geq 0$ such that $B_i \supset \Phi$; thus, for some $j \geq 0$, the tableau T_{j+1} had to be constructed from T_j applying the theory closure rule to B_j—which is according to the construction of the sequence $(B_i)_{i\geq 0}$ not the case—, or T_{j+1} had to be constructed from T_j applying the theory rule to B_j such that $\rho_m \in B_{j+1}$ for some $m \in \{1, \ldots, k\}$—which is impossible because $\{\rho_1, \ldots, \rho_k\} \cap B^* = \emptyset$.

We have shown B^* to be $\mathcal{T}$-satisfiable. Because $\neg\phi \in B^*$, $\neg\phi$ is $\mathcal{T}$-satisfiable as well. This, however, is a contradiction to ϕ being a $\mathcal{T}$-tautology, and the assumption that $(T_i)_{i\geq 0}$ is not a tableau proof has to be wrong. ■

The criterion can be simplified if the residues a background reasoner computes for keys consisting of literals consist of literals as well. This is a reasonable assumption, which is usually satisfied in practice (and is true for all total background reasoners).

COROLLARY 53 *A ground background reasoner $\mathcal{R}$ for a theory $\mathcal{T}$ is complete if, for all keys $\Phi \subset Lit_\Sigma$ for which $\mathcal{R}$ is defined, $R \subset Lit_\Sigma$ for all $\langle \sigma, R\rangle \in \mathcal{R}(\Phi)$ and for all (finite or infinite) $\mathcal{T}$-unsatisfiable sets $\Xi \subset Form_\Sigma$ of* literals *that do not contain free variables:*

1. *there is a key $\Phi \subset \Xi$ such that $id \in \mathcal{R}(\Phi)$; or*

2. *there is a key $\Phi \subset \Xi$ such that there is a refuter $\langle id, \{\rho, \ldots, \rho_k\}\rangle$ in $\mathcal{R}(\Phi)$ with $\{\rho_1, \ldots, \rho_k\} \cap \Xi = \emptyset$ ($k \geq 1$).*

Proof. Let Ξ be a downward saturated $\mathcal{T}$-unsatisfiable set of formulae and define $\Xi' = \Xi \cap Lit_\Sigma$ to be the set of literals in Ξ. Then Ξ' is $\mathcal{T}$-unsatisfiable, because a $\mathcal{T}$-structure satisfying Ξ' would satisfy Ξ as well.

Thus, there is a key $\Phi \subset \Xi' \subset \Xi$ such that $id \in \mathcal{R}(\Phi)$—in which case we are done—, or there is a key $\Phi \subset \Xi'$ and a refuter $\langle id, \{\rho_1, \ldots, \rho_k\}\rangle \in \mathcal{R}(\Phi)$ with $\{\rho_1, \ldots, \rho_k\} \cap \Xi' = \emptyset$. In the latter case, the refuter satisfies the condition $\{\rho_1, \ldots, \rho_k\} \cap \Xi = \emptyset$, because by assumption $\{\rho_1, \ldots, \rho_k\} \subset Lit_\Sigma$. ■

There is a strong relation between the criterion from Theorem 52 and the definition of Hintikka sets for theory reasoning: similar to classical first-order Hintikka sets (see Sect. 2.3 in Chap. 1), any set is satisfiable that does not contain "obvious" inconsistencies (inconsistencies that can be detected by the background reasoner) and is downward saturated (the background reasoner cannot add new formulae).

EXAMPLE 54 The complete background reasoner for the theory $\mathcal{OP}$ of partial orderings from Example 32 can be turned into a definition of Hintikka sets for $\mathcal{OP}$: A set Ξ that is downward saturated and, in addition, satisfies the following conditions is $\mathcal{OP}$-satisfiable:

1. For all terms $t, t', t'' \in Term_\Sigma$:
   if $(t < t'), (t' < t'') \in \Xi$, then $(t < t'') \in \Xi$.

2. There is no literal of the form $(t < t)$ in Ξ.

3. There are no literals $\phi, \neg\phi$ in Ξ.

A Completeness Criterion for Free Variable Background Reasoners

The criterion for free variable background reasoners is based on lifting completeness of a ground background reasoner. If the ground background reasoner computes a refuter $\langle id, R\rangle$ for a ground instance $\Phi\tau$ of a key Φ, then, to be complete, the free variable background reasoner has to compute a refuter for Φ that is more general than $\langle \tau, R\rangle$.

THEOREM 55 *Let $\mathcal{R}$ be a free variable background reasoner for a theory $\mathcal{T}$; $\mathcal{R}$ is complete if there is a complete ground background reasoner $\mathcal{R}^g$ for $\mathcal{T}$ such that, for all keys $\Phi \subset Form_\Sigma$, all ground substitutions τ, and all refuters $\langle id, R^g \rangle \in \mathcal{R}^g(\Phi\tau)$, there is a refuter $\langle \sigma, R \rangle \in \mathcal{R}(\Phi)$ and a substitution τ' with*

1. $\tau = \tau' \circ \sigma$,
2. $R\tau' = R^g$.

Proof. Let the sentence ϕ be a $\mathcal{T}$-tautology. Since $\mathcal{R}^g$ is a complete ground background reasoner, using $\mathcal{R}^g$ a ground tableau proof

$$\{\{\neg\phi\}\} = T_0^g, \ldots, T_n^g = \emptyset \ ,$$

can be constructed, where the new terms introduced by δ-rule applications have been chosen in an arbitrary way (see below).

By induction we prove that there is a free variable tableau proof

$$\{\{\neg\phi\}\} = T_0, \ldots, T_n = \emptyset$$

such that $T_i\tau_i = T_i^g$ for substitutions $\tau_i \in Subst_\Sigma^*$ $(0 \leq i \leq n)$.

$i = 0$: Since ϕ is a sentence, $T_0\tau_0 = T_0^g$ for $\tau_0 = id$.

$i \to i+1$: Depending on how T_{i+1}^g has been derived from T_i^g, there are the following subcases:

If T_{i+1}^g has been derived from T_i^g by applying an expansion rule to a formula ϕ^g on a branch $B_i^g \in T_i^g$, then there has to be a formula ϕ_i on a branch $B_i \in T_i$ such that $\phi_i\tau_i = \phi_i^g$ and $B_i\tau_i = B_i^g$. If an α- or β-rule has been applied, then apply the same rule to ϕ_i to derive T_{i+1} from T_i and set $\tau_{i+1} = \tau_i$. If a γ-rule has been applied and the term t has been substituted for the quantified variable in the ground tableau, then derive T_{i+1} from T_i by applying a γ-rule to ϕ_i and substituting a new free variable x for the quantified variable; set $\tau_{i+1} = \tau_i \cup \{x \mapsto t\}$. If a δ-rule has been applied, then apply the free variable δ-rule to ϕ_i to derive T_{i+1} from T_i and set $\tau_{i+1} = \tau_i$. In addition, the new ground term introduced in the ground tableau—that we are free to choose as long as it does not occur in T_i—shall be $f(x_1, \ldots, x_n)\tau_i$ where $f(x_1, \ldots, x_n)$ is the Skolem term that has been substituted for the existentially quantified variable in the free variable tableau.

If T_{i+1}^g has been derived from T_i^g by applying the theory closure or the theory expansion rule using a key Φ^g taken from a branch $B_i^g \in T_i^g$ and a refuter id or $\langle id, R^g \rangle$ in $\mathcal{R}^g(\Phi^g)$, then there has to be a key Φ_i on a branch $B_i \in T_i$ such that $\Phi_i\tau_i = \Phi_i^g$ and $B_i\tau_i = B_i^g$. Thus. there is a refuter $\langle \sigma, R \rangle \in \mathcal{R}(\Phi)$ and a substitution τ' such that $\tau_i = \tau' \circ \sigma$ and $R\tau' = R^g$. In that case, derive T_{i+1} from T_i by applying the theory expansion or closure rule using the key Φ_i and the refuter $\langle \sigma, R \rangle$, and set $\tau_{i+1} = \tau'$. ■

EXAMPLE 56 The criterion from Theorem 55 can be used to prove completeness of the free variable background reasoner $\mathcal{R}$ for the theory $\mathcal{OP}$ of partial orderings that satisfies the following conditions. The proof is based on the completeness of the ground background reasoner for $\mathcal{OP}$ from Example 32. The conditions for $\mathcal{R}$ are:

1. For all terms $s_1, t, t', s_2 \in Term_\Sigma$ where t and t' are unifiable:
   if $(s_1 < t), (t' < s_2) \in \Phi$, then $\langle \mu, \{(s_1 < s_2)\mu\} \rangle \in \mathcal{R}(\Phi)$ where μ is a most general unifier (MGU) of t and t'.

2. For all terms $t, t' \in Term_\Sigma$ that are unifiable:
   if $(t < t') \in \Phi$, then $\mu \in \mathcal{R}(\Phi)$ where μ is an MGU of t and t'.

3. For all atoms $\phi, \phi' \in Form_\Sigma$ that are unifiable:
   If $\phi, \neg\phi' \in \Phi$, then $\mu \in \mathcal{R}(\Phi)$ where μ is an MGU of ϕ and ϕ'.

In the ground case, a tableau proof can be constructed deterministically using a fair tableau construction rule. In the free variable case, however, there are additional choice points because there may be refuters with incompatible substitutions. Thus lifting the notion of fairness to the free variable case such that no backtracking at all is needed to construct a tableau proof is very difficult (though not impossible).

A Completeness Criterion for Universal Formula Background Reasoners

A criterion for the completeness of universal formula background reasoners can be defined based on completeness of free variable background reasoners (the proof of the theorem is similar to that of Thereom 55).

DEFINITION 57 *Let $\Phi \subset Form_\Sigma$ be a key, and let $(\forall x_1) \cdots (\forall x_k)\phi$ be a universally quantified literal in Φ. If ϕ' is constructed from ϕ by replacing the variables $x_1, \ldots, x_k$ by free variables $y_1, \ldots, y_k$ that do not occur in Φ, then ϕ' is a* free variable instance *of $(\forall x_1) \cdots (\forall x_k)\phi$ (w.r.t. Φ).*

THEOREM 58 *Let $\mathcal{R}$ be a universal formula background reasoner for a theory $\mathcal{T}$; $\mathcal{R}$ is complete if there is a complete free variable background reasoner $\mathcal{R}^{fv}$ for $\mathcal{T}$ such that, for all keys $\Phi \subset Form_\Sigma$, the following holds:*

Let the key Φ^{fv} be constructed from Φ by replacing all formulae $\phi \in \Phi$ of the form $(\forall x_1) \cdots (\forall x_n)\psi$ by a free variable instance of ϕ, and let F be the set of all the free variables occurring in Φ^{fv} but not in Φ. Then, for all refuters $\langle \sigma^{fv}, R^{fv} \rangle \in \mathcal{R}^{fv}(\Phi^{fv})$, there is a refuter $\langle \sigma, R \rangle \in \mathcal{R}(\Phi)$ where

1. ${\sigma^{fv}}_{|(V \setminus F)} = \sigma$,
2. $R^{fv} = R({\sigma^{fv}}_{|F})$.

5.3 Completeness Preserving Refinements

Restrictions on Keys

In this section, additional refinements are discussed that are indispensable for an efficient implementation of theory reasoning.

An important simplification usually used in implementations is to impose the restriction on keys that they must consist of literals (universally quantified literals in the case of universal formula tableau). The proof for Theorem 52 shows that completeness is preserved if this restriction is combined with any complete background reasoner.

COROLLARY 59 *Let $\mathcal{R}$ be a complete ground, free variable, or universal formula background reasoner. Then the restriction $\mathcal{R}'$ of $\mathcal{R}$ to keys Φ that consist of (universally quantified) literals is complete ($\mathcal{R}'$ is undefined for other keys).*

The set of keys that have to be considered can be further restricted. The background reasoner has only to be defined for keys that contain a pair of complementary literals or at least one formula in which a symbol occurs that is defined by the theory:

DEFINITION 60 *A set of function or predicate symbols is* defined *by a theory $\mathcal{T}$ if for all sets Φ of formulae that do not contain these symbols: Φ is satisfiable if and only if Φ is $\mathcal{T}$-satisfiable.*

For example, the equality theory $\mathcal{E}$ defines the equality predicate $\approx$; the theory of partial orderings defines the predicate symbol $<$.

Similarly, only keys have to be considered that contain a pair of complementary literals or consist of formulae that *all* have a predicate symbol in common with the theory (which may or may not be defined by the theory). Thus, for the theory $\mathcal{OP}$, all formulae in keys have to contain the predicate symbol $<$ as it is the only predicate symbol in $\mathcal{OP}$. For the equality theory $\mathcal{E}$, however, this restriction is useless because $\mathcal{E}$ contains all predicate symbols.

COROLLARY 61 *Let $\mathcal{R}$ be a complete ground, free variable, or universal formula background reasoner for a theory $\mathcal{T}$. Then the restriction $\mathcal{R}'$ of $\mathcal{R}$ to keys Φ that*

1. *(a) contain at least one occurrence of a function or predicate symbol defined by $\mathcal{T}$, and*
   (b) consist of formulae that all have at least one predicate symbol in common with $\mathcal{T}$,
2. *or contain a pair ϕ and $\neg\psi$ (resp. $(\forall\overline{x})\phi$ and $(\forall\overline{y})\neg\psi$), where ϕ and ψ are unifiable,*

is complete ($\mathcal{R}'$ is undefined for other keys).

Most General Refuters

There is another important refinement that can be combined with all complete background reasoners: completeness is preserved if only most general refuters are computed (this is a corollary to Theorem 55). The subsumption relation on refuters may or may not take the theory $\mathcal{T}$ into account:

DEFINITION 62 *Let $\mathcal{T}$ be a theory; and let $W \subset V$ be a set of variables. The subsumption relations $\leq^W$ and $\leq^W_\mathcal{T}$ on refuters are defined by:*

- $\langle \sigma, R\rangle \leq^W \langle \sigma', R'\rangle$ *if there is a substitution $\tau \in Subst^*_\Sigma$ such that*
  1. $\sigma'(x) = \sigma(x)\tau$ *for all $x \in W$, and*
  2. $R'\tau \supset R$.
- $\langle \sigma, R\rangle \leq^W_\mathcal{T} \langle \sigma', R'\rangle$ *if there is a refuter $\langle \sigma'', R''\rangle$ such that*
  1. $\langle \sigma, R\rangle \leq^W \langle \sigma'', R''\rangle$, *and*
  2. $\Phi\sigma'' \cup \{\bigvee R''\} \models^\circ_\mathcal{T} \Phi\sigma' \cup \{\bigvee R'\}$ *for all formula sets $\Phi \subset Form_\Sigma$ (including, in particular, the empty set).*

In addition, we use the abbreviations $\leq = \leq^V$ and $\leq_\mathcal{T} = \leq^V_\mathcal{T}$ where V is the set of all variables.

The set W contains the "relevant" variables, including *at least* those occurring in the two refuters that are compared. If, for example, the theory expansion rule is used to extend a tableau branch, then W contains all free variables occurring in the tableau. It is of advantage to keep the set W as small as possible; but, if the context is not known, the set $W = V$ of all variables has to be used.

The intuitive meaning of $\langle \sigma, R\rangle \leq^W_\mathcal{T} \langle \sigma', R'\rangle$ is that the effects of using the refuter $\langle \sigma', R'\rangle$ can be simulated by first applying a substitution τ and then using the resulting refuter $\langle \sigma'', R''\rangle$ of which the refuter $\langle \sigma', R'\rangle$ is a logical consequence.

EXAMPLE 63 The refuter $\langle id, \{p(x)\}\rangle$ subsumes $\langle \{x \mapsto a\}, \{p(a)\}\rangle$ w.r.t. the subsumption relation $\leq^W$ (and thus w.r.t. $\leq^W_\mathcal{T}$) for all variable sets W; however, it subsumes the refuter $\langle id, \{p(a)\}\rangle$ only if $x \notin W$.

The refuter $\langle \sigma, \{\phi\}\rangle$ is more general than $\langle \sigma, \{\phi, \psi\}\rangle$ w.r.t. all subsumption relations, i.e., only refuters with a minimal residue are most general.

Let $\mathcal{T}$ be the equational theory $\mathcal{E} \cup \{a \approx b\}$. Then $\langle id, p(a)\rangle$ and $\langle id, p(b)\rangle$ resp. $\{x \mapsto a\}$ and $\{x \mapsto b\}$ subsume each other w.r.t. $\leq_\mathcal{T}$.

COROLLARY 64 *Let $\mathcal{R}$ be a complete free variable or universal formula background reasoner for a theory $\mathcal{T}$. Then a background reasoner $\mathcal{R}'$ is complete as well if, for all keys Φ and refuters $\langle \sigma, R\rangle \in \mathcal{R}(\Phi)$, there is a refuter $\langle \sigma', R'\rangle \in \mathcal{R}'(\Phi)$ that subsumes $\langle \sigma, R\rangle$ w.r.t. $\leq$ or $\leq_\mathcal{T}$ (Def. 62).*

If the subsumption relations $\leq^W$ and $\leq^W_{\mathcal{T}}$ are used, the context in which a background reasoner is used has to be taken into consideration:

THEOREM 65 *Let $\mathcal{R}$ be a complete free variable or universal formula background reasoner for a theory $\mathcal{T}$. Then, for every $\mathcal{T}$-tautologies ϕ, a free variable tableau proof resp. a universal formula tableau proof can be built using $\mathcal{R}$ observing the restriction that each $\mathcal{T}$-refuter that is used in a theory expansion or closure rule application is minimal in $\mathcal{R}(\Phi)$ w.r.t. $\leq^W$ or $\leq^W_{\mathcal{T}}$, where Φ is the key that has been chosen for that rule application and W is the set of free variables in the tableau to which the rule is applied.*

The number of refuters that have to be considered is closely related to the number of choice points when the theory expansion or closure rule is applied to a tableau. Therefore, it is desirable to compute a *minimal* set of refuters. Nevertheless, it is often not useful to ensure minimality since there is a trade-off between the gain of computing a minimal set and the extra cost for checking minimality and removing subsumed refuters. While it is relatively easy to decide whether $\langle \sigma, R \rangle \leq^W \langle \sigma', R' \rangle$, it can (depending on the theoy $\mathcal{T}$) be difficult to decide and is in general undecidable whether $\langle \sigma, R \rangle \leq^W_{\mathcal{T}} \langle \sigma', R' \rangle$.

Other Search Space Restrictions

There are other useful restrictions that, however, cannot be imposed on an arbitrary background reasoner without destroying completeness. Nevertheless, for every theory, there are background reasoners that have at least some of the following features:

- To avoid branching when the theory expansion rule is applied, only refuters $\langle \sigma, R \rangle$ are computed where the residue is either empty or a singleton.
- Only total refuters are computed, i.e., the residues are empty.
- The sets of refuters computed for a key are restricted to be
  - finite (in which case their computation terminates);
  - empty or a singleton (then theory expansion or closure rules are—at least for a single key—deterministic);

There is, of course, a trade-off between these desirable features, in particular between total and partial theory reasoning (see Sect. 2.5).

$$\frac{t \approx s \quad \phi[t]}{\phi[s]} \qquad \frac{s \approx t \quad \phi[t]}{\phi[s]} \qquad \frac{\neg(t \approx t)}{*} \qquad \frac{\phi \quad \neg\phi}{*}$$

Table 9. Jeffrey's equality theory expansion and closure rules.

6 PARTIAL EQUALITY REASONING

6.1 Partial Equality Reasoning for Ground Tableaux

Virtually all approaches to handling equality can be regarded as a special case of the general methods for theory reasoning in semantic tableaux. Exception are, for example, the method of *equality elimination* [Degtyarev and Voronkov, 1996a] and applying transformations from first-order logic with equality into first-order logic without equality to the input formulae [Brand, 1975; Bachmair *et al.*, 1997] (see Sect. 9).

The first methods for adding equality to the ground version of semantic tableaux have been developed in the 1960s [Jeffrey, 1967; Popplestone, 1967], following work by S. Kanger on how to add equality to sequent calculi [Kanger, 1963]. R. Jeffrey introduced the additional tableau expansion and closure rules shown in Table 9 (i.e., a partial reasoning method); a similar set of rules has been described by Z. Lis in [Lis, 1960]. 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 $\phi[s]$ (which is constructed by substituting one occurrence of t in $\phi[t]$ by s), then $\phi[s]$ may be added to B.

There are two closure rules. The first one is the usual closure rule for ground tableaux with and without theory reasoning: a branch B is closed if there are formulae ϕ and $\neg\phi$ in B. The second one is an additional equality theory closure rule: a branch is closed if it contains a formula of the form $\neg(t \approx t)$.

THEOREM 66 (Jeffrey, 1967) *A ground background reasoner $\mathcal{R}$ for the theory $\mathcal{E}$ of equality is complete if it satisfies the following conditions:*

1. *For all terms $t, s \in Term^0_\Sigma$ and sentences $\phi \in Form_\Sigma$:*
   if $\phi[t], (t \approx s) \in \Phi$ or $\phi[t], (s \approx t) \in \Phi$, then $\langle id, \{\phi[s]\}\rangle \in \mathcal{R}(\Phi)$.
2. *For all terms $t \in Term^0_\Sigma$: if $\neg(t \approx t) \in \Phi$, then $id \in \mathcal{R}(\Phi)$.*
3. *For all sentences $\phi \in Form_\Sigma$: if $\phi, \neg\phi \in \Phi$, then $id \in \mathcal{R}(\Phi)$.*

EXAMPLE 67 Figure 2 shows an example for the application of Jeffrey's equality 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.

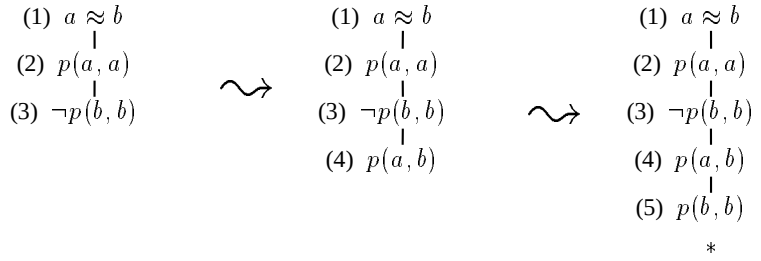


Figure 2. The application of Jeffrey's additional rules to expand and close a tableau branch (Example 67).

The background reasoner is still complete if the formula ϕ to which an equality is applied is restricted to be (a) an inequality $\neg(s \approx t)$, or (b) a literal $p(t_1, \ldots, t_n)$ or $\neg p(t_1, \ldots, t_n)$ where $p \neq \approx$; i.e., equalities do not have to be applied to complex formulae or to equalities.

Jeffrey's rules resemble paramodulation [Robinson and Wos, 1969] (see [Snyder, 1991] for an overview on various techniques for improving paramodulation).

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 non-determinism and a huge search space; an enormous number of irrelevant formulae (residues) can be derived. 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 [Reeves, 1987] (see Table 10) generate a smaller search space. They are the tableau counterpart of RUE-resolution [Digricoli and Harrison, 1986] and are more goal-directed than Jeffrey's expansion rules: only literals that are potentially complementary are used for expansion. Like RUE-resolution, the rules are based upon the following fact: If an $\mathcal{E}$-structure M satisfies the inequality $\neg(f(a_1, \ldots, a_k) \approx f(b_1, \ldots, b_k))$ or it satisfies the formulae $p(a_1, \ldots, a_k)$ and $\neg p(b_1, \ldots, b_k)$, then at least one of the inequalities

$$\neg(a_1 \approx b_1), \ldots, \neg(a_k \approx b_k)$$

is satisfied by M. In addition, a rule is needed that implements the symmetry of equality, i.e., that allows to deduce $s \approx t$ from $t \approx s$. With these equality theory expansion rules, it is sufficient to use the same closure rules as in Theorem 66:

THEOREM 68 (Reeves, 1987) *If a ground background reasoner $\mathcal{R}$ for the theory $\mathcal{E}$ of equality satisfies the following conditions, it is complete:*

1. *For all terms $t = f(t_1, \ldots, t_k)$ and $s = f(s_1, \ldots, s_k)$ ($k \geq 1$):*
   if $\neg(t \approx s) \in \Phi$, then $\langle id, \{\neg(s_1 \approx t_1), \ldots, \neg(s_k \approx t_k)\}\rangle \in \mathcal{R}(\Phi)$.

$$\frac{p(t_1,\ldots,t_k) \quad \neg p(s_1,\ldots,s_k)}{\neg(t_1 \approx s_1) \mid \cdots \mid \neg(t_k \approx s_k)} \qquad \frac{\neg(f(t_1,\ldots,t_k) \approx f(s_1,\ldots,s_k))}{\neg(t_1 \approx s_1) \mid \cdots \mid \neg(t_k \approx s_k)}$$

$$\frac{t \approx s}{s \approx t} \qquad \frac{\neg(t \approx t)}{*} \qquad \frac{\phi \quad \neg\phi}{*}$$

Table 10. Reeves's equality expansion and closure rules.

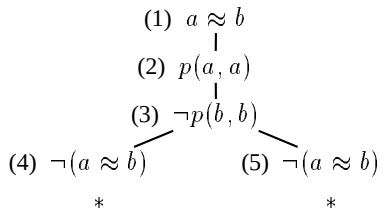


Figure 3. Applying Reeves's equality expansion rule (Example 69).

2. *For all literals* $\psi = p(t_1,\ldots,t_k)$ *and* $\psi' = \neg p(s_1,\ldots,s_k)$ $(k \geq 1)$*:* *if* $\psi, \psi' \in \Phi$*, then* $\langle id, \{\neg(s_1 \approx t_1),\ldots,\neg(s_k \approx t_k)\}\rangle \in \mathcal{R}(\Phi)$.
3. *For all terms* $s,t \in Term^0_\Sigma$*: if* $(s \approx t) \in \Phi$*, then* $\langle id, \{t \approx s\}\rangle \in \mathcal{R}(\Phi)$.
4. *For all terms* $t \in Term^0_\Sigma$*: if* $\neg(t \approx t) \in \Phi$*, then* $id \in \mathcal{R}(\Phi)$.
5. *For all literals* $\phi \in Lit_\Sigma$*: if* $\phi, \neg\phi \in \Phi$*, then* $id \in \mathcal{R}(\Phi)$.

EXAMPLE 69 Figure 3 shows the application of Reeves's rule to expand and close the same tableau branch as in Figure 2: 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.

6.2 Partial Equality Reasoning for Free Variable Tableaux

M. Fitting extended Jeffrey's approach and adapted it to free variable tableaux [Fitting, 1996]. The main difference is that equality rule applications may require

$$\frac{t \approx s \qquad \phi[t']}{(\phi[s])\,\mu} \qquad \frac{s \approx t \qquad \phi[t']}{(\phi[s])\,\mu} \qquad \frac{\neg(t \approx t')}{*} \qquad \frac{\phi \qquad \phi'}{*}$$

where μ is an MGU of t and t' resp. ϕ and ϕ'
and μ is applied to the whole tableau.

Table 11. Fitting's equality reasoning rules for free variable tableaux.

instantiating free variables, i.e., the substitution that is part of a refuter may not be the identity. These substitutions can be obtained using unification: If an equality $t \approx s$ is to be applied to a formula $\phi[t']$, the application of a most general unifier μ of t and t' is sufficient to derive $(\phi[s])\,\mu$ (see Table 11).

Unification can become necessary as well 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 $\{x \mapsto a\}$ is applied (to the whole tableau):

THEOREM 70 (Fitting, 1990) *$\mathcal{R}$ is a complete free variable background reasoner for the equality theory $\mathcal{E}$ if it satisfies the following conditions:*

1. *For all terms $t, t' \in Term_\Sigma$ that are unifiable and all $\phi \in Form_\Sigma$:*
   if $(t \approx s), \phi[t'] \in \Phi$, then $\langle \mu, \{(\phi[s])\,\mu\}\rangle \in \mathcal{R}(\Phi)$ where μ is an MGU of t and t'.
2. *For all terms $t, t' \in Term_\Sigma$ that are unifiable:*
   if $\neg(t \approx t') \in \Phi$, then $\mu \in \mathcal{R}(\Phi)$ where μ is an MGU of t and t'.
3. *For all literals $\phi, \phi' \in Lit_\Sigma$ that are unifiable:*
   If $\phi, \neg\phi' \in \Phi$, then $\mu \in \mathcal{R}(\Phi)$ where μ is an MGU of ϕ and ϕ'.

EXAMPLE 71 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 formula (2), (7) from (1), and (8) and (9) from (7). The framed formulae are added to the left branch by applying Fitting's equality expansion rules: Formula (10) is derived by applying equality (8) to (4) (the substitution $\{x_2 \mapsto a\}$ has to be applied), formula (11) is derived by applying (6) to (10) (the substitution $\{x_1 \mapsto a\}$ 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 $\{x_2 \mapsto a\}$ has already been applied).

The example demonstrates a difficulty involved in using free variable equality 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.

Completeness is preserved if the restriction is made that the formulae ϕ und ϕ' in Theorem 70 which the equality expansion rule is applied to have to be literals (similar to the ground case). However, the restriction that equalities must not be applied to equalities (that can be employed in the ground case) would destroy completeness, as the following example demonstrates.

EXAMPLE 72 Let the tableau branch B contain the formulae

$$a \approx b,\ f(h(a), h(b)) \approx g(h(a), h(b)),\ \neg(f(x, x) \approx g(x, x))\ .$$

A refuter with the residue $\{f(h(a), h(a)) \approx g(h(a), h(a))\}$ can be derived, provided it is allowed to apply equalities to equalities. After this formula has been added to the branch, the closing refuter $\{x \mapsto h(a)\}$ can be found.

If the application of equalities to equalities is prohibited, completeness is lost: then the only possibility is to apply $a \approx b$ to the inequality in B. All refuters that can be derived that way instantiate the variable x either with a or with b, which in the sequel makes it impossible to close the branch. Note that the criterion from Theorem 55, which would guarantee completeness, is not satisfied.

6.3 Partial Equality Reasoning for Tableaux with Universal Formulae

Fitting's method can easily be extended to free variable tableaux *with universal formulae* [Beckert, 1997]. 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 equality expansion rules from Section 6.2 is that, instead of the MGU μ of t and t', only its restriction μ' to variables is applied w.r.t. which *not* all formulae in the precondition of the rule are $\mathcal{T}$-universal (apart from that, the rule schemata are the same as the free variable schemata in Table 11). 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.

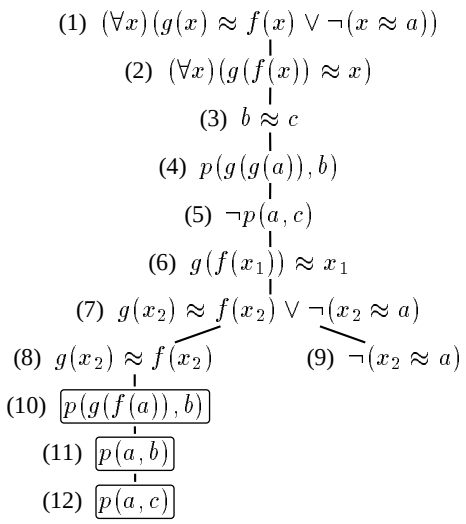


Figure 4. Using Fitting's expansion rules (Example 71).

EXAMPLE 73 If the method from Theorem 40 for recognizing universal formulae is used, the tableau in Figure 4 (without the framed formulae) can be closed using the substitution $\{x_2 \mapsto a\}$. The variable x_1 does not have to be instantiated, because equality (6) is recognized to be universal w.r.t. to x_1.

The background reasoner does not have to cope with the problem of recognizing universal formulae, because in keys the universal formulae are explicitly universally quantified (Def. 41).

THEOREM 74 *A background reasoner $\mathcal{R}$ that satisfies the following conditions is a complete universal formula background reasoner for the equality theory $\mathcal{E}$:*

1. *For all $s, t, t' \in Term_\Sigma$ such that t, t' are unifiable, and all $\phi \in Form_\Sigma$: if $(\forall\overline{x})(t \approx s), (\forall\overline{y})\phi[t'] \in \Phi$, then $\langle \mu_{|F}, \{(\phi[s])\mu\}\rangle \in \mathcal{R}(\Phi)$ where μ is an MGU of t and t' and F is the set of variables that are free in $(\forall\overline{x})(t \approx s)$ or $(\forall\overline{y})\phi[t']$.*[3]

2. *For all $t, t' \in Term_\Sigma$ that are unifiable: if $(\forall\overline{x})\neg(t \approx t') \in \Phi$, then $\mu_{|F} \in \mathcal{R}(\Phi)$ where μ is an MGU of t and t' and F is the set of variables that are free in $(\forall\overline{x})\neg(t \approx t')$.*

3. *For all atoms $\phi, \phi' \in Form_\Sigma$ that are unifiable: if $(\forall\overline{x})\phi, (\forall\overline{y})\neg\phi' \in \Phi$, then $\mu_{|F} \in \mathcal{R}(\Phi)$ where μ is an MGU of ϕ and ϕ' and F is the set of variables that are free in $(\forall\overline{x})\phi$ or $(\forall\overline{y})\phi'$.*

[3] $(\forall\overline{x})$ is an abbreviation for $(\forall x_1)\cdots(\forall x_m)$ $(m \geq 0)$. Without making a real restriction, we assume the sets of free and bound variables occurring in Φ to be disjoint.

7 TOTAL EQUALITY REASONING

7.1 Total Equality Reasoning and E-unification

The common problem of all the partial reasoning methods described in Section 6.1, which 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 (i.e., partial background reasoners). A set of rules that implement a completion procedure for the ground version of tableaux has been described in [Browne, 1988]; however, these equality 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 refuters that close a tableau branch is equivalent to solving E-unification problems.

Depending on the version of semantic tableaux to which equality handling is added, different types of E-unification problems have to be solved. These are introduced in the following section.

7.2 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, "rigid" E-unification, and "mixed" E-unification, which is a combination of both. The different versions allow equalities to be used differently in an equational deduction: 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 compute refuters, depends on the version of semantic tableaux that equality reasoning 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. For tableaux with universal formulae, both versions have to be combined [Beckert, 1994]; then equalities contain two types of variables, namely universal (bound) and rigid (free) ones.

DEFINITION 75 *A* mixed E-unification problem $\langle E, s, t\rangle$ *consists of a finite set* E *of universally quantified equalities* $(\forall x_1)\cdots(\forall x_m)(l \approx r)$ *and terms* s *and* t. *A*

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

Table 12. Examples for the different versions of E-unification.

substitution $\sigma \in Subst^*_\Sigma$ *is a* solution *to the problem* $\langle E, s, t\rangle$ *if*

$$E\sigma \models^\circ_\varepsilon (s\sigma \approx t\sigma) \ .^4$$

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 $\models^\circ_\varepsilon$ is used instead of $\models_\varepsilon$.
- The substitution σ is applied not only to the terms s und t but also to the set E.

A mixed E-unification problem $\langle E, s, t\rangle$ is *universal* if there are no free variables in E, and it is *rigid* if there are no bound variables in E (if E is ground, the problem is both rigid and universal).

EXAMPLE 76 Table 12 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 be 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.

Syntactical unification is a special case of E-unification, namely the case where the set E of equalities is empty.

For handling equality in free variable tableaux, the problem of finding a simultaneous solution to several mixed E-unification problems plays an important rôle, as it corresponds to the problem of finding a substitution that allows to simultaneously close several tableau branches.

DEFINITION 77 *A finite set* $\{\langle E_1, s_1, t_1\rangle, \ldots, \langle E_n, s_n, t_n\rangle\}$ *(*$n \geq 1$*) of* E*-unification problems is called* simultaneous E*-unification problem. A substitution* σ *is a solution to the simultaneous problem if it is a solution to every component* $\langle E_k, s_k, t_k\rangle$ $(1 \leq k \leq n)$.

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

7.3 Extracting E-unification Problems from Keys

The important formulae in a key from which E-unification problems are extracted are: equalities, inequalities, and pairs of potentially complementary literals:

DEFINITION 78 *Literals*

$$(\forall x_1)\cdots(\forall x_k)p(s_1,\ldots,s_n) \text{ and } (\forall y_1)\cdots(\forall y_l)\neg p(t_1,\ldots,t_n)\ ,$$

where $p \not\equiv \approx$, are called a pair of potentially complementary literals *($n \geq 0$ and $k, l \geq 0$, i.e., the literals may or may not be universally quantified).*

We now proceed to define the set of equalities and the set of E-unification problems of a key. All considerations here are restricted to keys consisting of universally quantified literals.

DEFINITION 79 *Let $\Phi \subset Form_\Sigma$ be a key. The* set $E(\Phi)$ of equalities *consists of the universally quantified equalities in Φ, i.e., all formulae in Φ of the form $(\forall x_1)\cdots(\forall x_k)(s \approx t)$ ($k \geq 0$).*

EXAMPLE 80 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 40 is used and keys Φ_1 and Φ_2 are built from the literals on the branches B_1 and B_2, respectively (according to Theorem 41), then both $E(\Phi_1)$ and $E(\Phi_2)$ contain the equalities $b \approx c$ and $(\forall x)(g(f(x)) \approx x)$; $E(\Phi_1)$ contains, in addition, the equality $g(x_2) \approx f(x_2)$.

DEFINITION 81 *Let $\Phi \subset Form_\Sigma$ be a key. The* set $P(\Phi)$ of E-unification problems *consists exactly of:*

1. *for each pair $\phi, \psi \in \Phi$ of potentially complementary literals, the problem*

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

   where $p(t_1,\ldots,t_n)$ and $\neg p(s_1,\ldots,s_n)$ are free variable instances of ϕ resp. ψ (Def. 57);

2. *for each inequality $\phi = (\forall x_1)\cdots(\forall x_k)(\neg(t' \approx s'))$ in Φ ($k \geq 0$), the problem*

   $$\langle E(\Phi), t, s\rangle$$

   where $\neg(t \approx s)$ is a free variable instance of ϕ (Def. 57).

The problems in $\mathcal{P}(B)$ of the form $\langle E(B), \langle s_1,\ldots,s_k\rangle, \langle t_1,\ldots,t_k\rangle\rangle$ are actually simultaneous E-unification problems (sharing the same set of equalities), since the non-simultaneous problems $\langle E(B), s_i, t_i\rangle$ ($1 \leq i \leq k$) have to be solved simultaneously.

LEMMA 82 *A substitution is a solution to a simultaneous mixed E-unification problem of the form $\{\langle E, s_1, t_1\rangle, \ldots, \langle E, s_n, t_n\rangle\}$ ($n \geq 1$) iff*

- *it is a solution to the non-simultaneous mixed E-unification problem $\langle E, f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)\rangle$ (the function symbol f must not occur in the original problem), and*
- *it does not instantiate variables with terms containing f.*

All substitutions that are $\mathcal{E}$-refuters, i.e., that close a tableau branch B, can be computed by extracting the set $P(\Phi)$ of mixed E-unification problems from a key $\Phi \subset B$ according to the above definition and solving the problems in $P(\Phi)$. If one of the problems in $P(\Phi)$ has a solution σ, all instances of $\Phi\sigma$ are $\mathcal{E}$-unsatisfiable; therefore, σ is a refuter for Φ. The pair of potentially complementary literals corresponding to the solved unification problem has been proven to actually be $\mathcal{E}$-complementary; or the corresponding inequality has been proven to be $\mathcal{E}$-complementary (provided the refuter is applied).

EXAMPLE 83 We continue Example 80. Again, B_1 denotes the left and B_2 the right branch of the tableau in Figure 4 (without the framed formulae), and Φ_1 and Φ_2 are keys extracted from these branches. Then both $P(\Phi_1)$ and $P(\Phi_2)$ contain the problem $\langle E(\Phi_i), \langle g(g(a)), b\rangle, \langle a, c\rangle\rangle$. $P(\Phi_2)$ contains, in addition, the problem $\langle E(\Phi_2), x_2, a\rangle$.

Apart from 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 84 *A total (universal formula, free variable, or ground) background reasoner $\mathcal{R}$ is complete for the equality theory $\mathcal{E}$ if it satisfies the following condition for all keys $\Phi \subset Lit_\Sigma$: If a substitution σ is a most general solution w.r.t. the subsumption relation $\leq^W$ (or $\leq^W_\varepsilon$) of one of the problems in $P(\Phi)$, then $\mathcal{R}(\Phi)$ contains the restriction of σ to the variables occurring in Φ.*

EXAMPLE 85 We continue from Examples 80 and 83: $\sigma = \{x_2 \mapsto a\}$ is a solution to the two mixed E-unification problems

$$\begin{aligned} \langle E(\Phi_1), \langle g(g(a)), b\rangle, \langle a, c\rangle\rangle &\in P(\Phi_1) \ , \\ \langle E(\Phi_2), x_2, a\rangle &\in P(\Phi_2) \ . \end{aligned}$$

When the theory closure rule is used to close one of the branches (and thus σ is applied to the tableau), the other branch can then be closed using the empty substitution.

7.4 Solving Ground E-unification Problems

In [Shostak, 1978], it is proven that *ground* E-unification is decidable; consequently, by considering all variables to be constants, it is decidable whether the empty substitution id is a solution to a given *rigid* E-unification problem $\langle E, s, t\rangle$, i.e., whether $E \models^{\circ} s \approx t$. This can be decided by computing a congruence closure, namely the equivalence classes of the terms (and subterms) occurring in $\langle E, s, t\rangle$ w.r.t. the equalities in E.

DEFINITION 86 *Let $\langle E, s, t\rangle$ be a ground (or rigid) E-unification problem; and let $T_{\langle E,s,t\rangle} \subset Term_{\Sigma}$ be the set of all (sub-)terms occurring in $\langle E, s, t\rangle$. The equivalence class $[t]_{\langle E,s,t\rangle}$ of a term $t \in T_{\langle E,s,t\rangle}$ is defined by:*

$$[t]_{\langle E,s,t\rangle} = \{s \in T_{\langle E,s,t\rangle} \mid E \models^{\circ}_{\varepsilon} s \approx t\} \ .$$

Since a ground E-unification problem $\langle E, s, t\rangle$ is solvable (and id a solution) if and only if $[s]_{\langle E,s,t\rangle} = [t]_{\langle E,s,t\rangle}$, one can decide whether $\langle E, s, t\rangle$ is solvable by computing these equivalence classes. Shostak proved that for computing the equivalence classes of all terms in $T_{\langle E,s,t\rangle}$, no terms that are not in $T_{\langle E,s,t\rangle}$ have to be considered: If s can be derived from t using the equalities in E, then this can be done without using an intermediate term that does not occur in the original problem, i.e., there is a sequence of terms $s = r_0, r_1, \ldots, r_k = t$, $k \geq 0$, all occurring in $\langle E, s, t\rangle$ such that r_i is derivable in one step from r_{i-1} using the equalities in E.

Since the number of subterms in a given problem is polynomial in its size, and the congruence closure can be computed in time polynomial in the number of subterms and the number of equalities, the solvability of a ground E-unification problem can be decided in polynomial time.

There are very efficient and sophisticated methods for computing the congruence closure, for example the algorithm described in [Nelson and Oppen, 1980], which is based on techniques from graph theory.

7.5 Solving Universal E-unification Problems

To solve a universal E-unification problem, the question has to be answered whether the equality of two given terms (or of instances of these terms) follows from E or, equivalently, whether the terms are equal in the free algebra of E. Overviews of methods for universal E-unification can be found in [Siekmann, 1989; Gallier and Snyder, 1990; Jouannaud and Kirchner, 1991; Snyder, 1991].

7.6 Solving Rigid E-unification Problems

Rigid E-unification and its significance for automated theorem proving was first described in [Gallier *et al.*, 1987]. It can be used for equality handling in semantic

tableaux and other *rigid variable* calculi for first-order logic, including the mating method [Andrews, 1981], the connection method [Bibel, 1987], and model elimination [Loveland, 1969]; an overview of rigid E-unification can be found in [Beckert, 1998].

The solution to a rigid E-unification problem $\langle E, s, t\rangle$ is a substitution representing the instantiations of free variables that have been necessary to show that the two given terms are equal; it is an $\mathcal{E}$-refuter for the key $E \cup \{\neg(s \approx t)\}$. A single variable can only be instantiated once by a substitution and, accordingly, to solve a rigid E-unification problem, the equalities of the problem can only be used with (at most) one instantiation for each variable they contain; a variable is either instantiated or not, that is, uninstantiated variables have to be treated as constants.

Rigid E-unification does not provide an answer to the question of how many different instantiations of an equality are needed to solve a problem. If a single instance is not sufficient, then the answer is "not unifiable". If several different instances of an equality are needed, a sufficient number of copies of that equality (with different rigid variables) has to be provided for the rigid E-unification problem to be solvable.

The following theorem clarifies the basic properties of rigid E-unification by listing different characterizations of the set of solutions of a given problem:

THEOREM 87 *Given a rigid E-unification problem $\langle E, s, t\rangle$ and a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \in Subst^*_\Sigma$, the following are equivalent conditions for σ being a solution to $\langle E, s, t\rangle$:*

1. *$E\sigma \models^0_\varepsilon s\sigma \approx t\sigma$, i.e., σ is by definition a solution to $\langle E, s, t\rangle$;*
2. *$E\sigma \models_\varepsilon s\sigma \approx t\sigma$ over a set V^0 of variables and a signature Σ^0 such that the variables occurring in $\langle E, s, t\rangle$ are constants, i.e., $V^0 = V \setminus W$ and $\Sigma^0 = \langle P_\Sigma, F_\Sigma \cup W, \alpha_\Sigma \cup \{x \mapsto 0 \mid x \in W\}\rangle$ where W is the set of variables occurring in $\langle E, s, t\rangle$.*
3. *$(E\sigma)\tau \models_\varepsilon (s\sigma)\tau \approx (t\sigma)\tau$ for all substitutions $\tau \in Subst^*_\Sigma$;*
4. *$E \cup \{x_1 \approx t_1, \ldots, x_n \approx t_n\} \models^0_\varepsilon s \approx t$; provided that none of the variables x_i occurs in any of the terms t_j ($1 \leq i, j \leq n$);*
5. *σ is the restriction to the variables occurring in $\langle E, s, t\rangle$ of a substitution which is a solution to the rigid E-unification problem $\langle E', yes, no\rangle$ where $E' = E \cup \{eq(x, x) \approx yes,\ eq(s, t) \approx no\}$, and (a) the constants yes, no, the predicate eq, and the variable x do not occur in $\langle E, s, t\rangle$, and (b) the constants yes, no do not occur in the terms $t_1, \ldots, t_n$.*

The last characterization of solutions in the above theorem shows that it is always possible to solve a rigid E-unification problem by transforming it into a problem in which the terms to be unified are constants.

If a rigid E-unification problem is solvable, then it has infinitely many solutions. But there are, for each problem, *finite* sets of solutions w.r.t. the subsumption relation $\leq^W_{\mathcal{E},E}$ that is defined as follows:

DEFINITION 88 *Let $E \subset Form_\Sigma$ be a set of rigid (i.e., quantifier-free) equalities; and let $W \subset V$ be a set of variables. Then the subsumption relations $\sqsubseteq^W_{\mathcal{E},E}$ and $\leq^W_{\mathcal{E},E}$ are on $Subst^*_\Sigma$ defined by:*

- *$\sigma \sqsubseteq^W_{\mathcal{E},E} \tau$ iff $E\sigma \models^0_{\mathcal{E}} \sigma(x) \approx \tau(x)$ for all $x \in W$;*
- *$\sigma \leq^W_{\mathcal{E},E} \tau$ iff there is a substitution $\sigma' \in Subst^*_\Sigma$ such that*

$$\sigma \leq^W \sigma' \quad \text{and} \quad \sigma' \sqsubseteq^W_{\mathcal{E},E} \tau \ .$$

The intuitive meaning of $\sigma \leq^W_{\mathcal{E},E} \tau$ is that the effects of applying τ to the set E of equalities can be simulated by first applying σ, then some other substitution ρ, and then equalities form $(E\sigma)\rho$.

LEMMA 89 *Let $E \subset Form_\Sigma$ be a set of rigid equalities, and let σ, τ be substitutions such that $\sigma \leq^W_{\mathcal{E},E} \tau$ where the set W contains all variables occurring in E. Then there is a substitution ρ such that $(E\sigma)\rho \models^0_{\mathcal{E}} E\tau$.*

It is possible to effectively compute a *finite* set $\mathcal{U}$ of solutions for a rigid E-unification problem $\langle E, s, t\rangle$ that is complete w.r.t. the subsumption relation $\leq^W_{\mathcal{E},E}$, i.e., for every solution σ of $\langle E, s, t\rangle$ there is a solution τ in $\mathcal{U}$ such that $\tau \leq^W_{\mathcal{E},E} \sigma$. This immediately implies the decidability of the question whether a given rigid E-unification problem $\langle E, s, t\rangle$ is solvable or not. On first sight this might be somewhat surprising since universal E-unification is undecidable; however, the additional restriction of rigid E-unification, that variables in E may only be instantiated once, is strong enough to turn an undecidable problem into a decidable one.

The problem of deciding whether a rigid E-unification problem has a solution is, in fact, NP-complete. This was first proven in [Gallier *et al.*, 1988] and then, more detailed, in [Gallier *et al.*, 1990; Gallier *et al.*, 1992]. The NP-hardness of the problem was already shown in [Kozen, 1981]. An alternative proof for the decidability of rigid E-unification was presented in [de Kogel, 1995], it is easy to understand but uses an inefficient decision procedure. More efficient methods using term rewriting techniques are described in [Gallier *et al.*, 1992; Becher and Petermann, 1994; Plaisted, 1995]. The procedure described in [Becher and Petermann, 1994] has been implemented and integrated into a prover for first-order logic with equality [Grieser, 1996].

7.7 Rigid Basic Superposition

In [Degtyarev and Voronkov, 1998], a method called *rigid basic superposition* has been presented for computing a *finite* (incomplete) set of solutions for rigid E-unification problems that is "sufficient" for handling equality in rigid variable calculi, i.e., can be used to build a complete free variable background reasoner for the equality theory $\mathcal{E}$. The procedure is an adaptation of basic superposition (in the formulation presented in [Nieuwenhuis and Rubio, 1995]) to rigid variables. It uses the concept of ordering constraints:

DEFINITION 90 *An* (ordering) constraint *is a (finite) set of expressions of the form $s \simeq t$ or $s \succ t$ where s and t are terms. A substitution σ is a solution to a constraint C iff (a) $s\sigma = t\sigma$ for all $s \simeq t \in C$, i.e., σ is a unifier of s and t, (b) $s\sigma > t\sigma$ for all $s \succ t \in C$, where $>$ is an arbitrary but fixed term reduction ordering, and (c) σ instantiates all variables occurring in C with ground terms.*

There are efficient methods for deciding the satisfiability of an ordering constraint C and for computing most general substitutions satisfying C in case the reduction ordering $>$ is a lexicographic path ordering (LPO) [Nieuwenhuis and Rubio, 1995].

The rigid basic superposition calculus consists of the two transformation rules shown below. They are applied to a rigid E-unification problem $\langle E, s, t\rangle \cdot C$ that has an ordering constraint C attached to it. The computation starts initially with the unification problem that is to be solved and the empty constraint. A transformation rule may be applied to $\langle E, s, t\rangle \cdot C$ only if the constraint is satisfiable before and after the application.

Left rigid basic superposition. If there are an equality $l \approx r$ or $r \approx l$ and an equality $u \approx v$ or $v \approx u$ in E and l' is a subterm of u, then replace the latter equality by $u[r] \approx v$ (where $u[r]$ is the result of replacing one occurrence of l' in u by r) and add $l \succ r$, $u \succ v$, and $l \simeq l'$ to C.

Right rigid basic superposition. If there is an equality $l \approx r$ or $r \approx l$ in E and l' is a subterm of s or of t, then replace s (resp. t) with $s[r]$ (resp. $t[r]$) and add $l \succ r$, $s \succ t$ (resp. $t \succ s$) and $l \simeq l'$ to C.

As the constraint expressions that are added by a rule application have to be satisfiable, they can be seen as a pre-condition for that application; for example, since $l \simeq l'$ is added to C, the terms l and l' have to be unifiable.

The two transformation rules are repeatedly applied, forming a non-deterministic procedure for transforming rigid E-unification problems. The process terminates when (a) the terms s and t become identical or (b) no further rule application is possible without making C inconsistent. Provided that no transformation is allowed that merely replaces an equality by itself, all transformation sequences are finite.

It is possible to only allow transformations where the term l' is *not* a variable, thus improving the efficiency of the procedure and reducing the number of solutions that are computed.

Let $\langle E, s, t\rangle \cdot C$ be any of the unification problems that are reachable by applying rigid basic superposition transformations to the original problem. Then, any solution to $C \cup \{s \simeq t\}$ is a solution to the original problem. Let $\mathcal{U}$ be the set of all such solutions that are most general w.r.t. $\leq^W$. The set $\mathcal{U}$ is finite because the application of rigid basic superposition rules always terminates.

EXAMPLE 91 Consider the rigid E-unification problem[5]

$$\langle E, s, t\rangle = \langle \{fa \approx a,\, g^2x \approx fa\},\, g^3x,\, x\rangle \ ,$$

and let $>$ be the LPO induced by the ordering $g > f > a$ on the function symbols.

The computation starts with

$$\langle E, s, t\rangle \cdot C = \langle \{fa \approx a,\, g^2x \approx fa\},\, g^3x,\, x\rangle \cdot \emptyset \ .$$

The only possible transformation is to use the right rigid basic superposition rule, applying the equality $(l \approx r) = (g^2x \approx fa)$ to reduce the term g^3x (all other transformations would lead to an inconsistent constraint). The result is the unification problem $\langle E, gfa, x\rangle \cdot \{g^2x \succ fa,\, g^3x \succ x,\, g^2x \simeq g^2x\}$; its constraint can be reduced to $C_1 = \{g^2x \succ fa\}$. A most general substitution satisfying $C_1 \cup \{gfa \simeq x\}$ is $\sigma_1 = \{x \mapsto gfa\}$.

A second application of the right rigid basic superposition rule leads to the unification problem $\langle E, ga, x\rangle \cdot \{g^2x \succ fa,\, fa \succ a,\, gfa \succ x,\, fa \simeq fa\}$; its constraint can be reduced to $C_2 = \{g^2x \succ fa,\, gfa \succ x\}$. A most general substitution satisfying $C_2 \cup \{ga \simeq x\}$ is $\sigma_2 = \{x \mapsto ga\}$.

At that point the process terminates because no further rule application is possible. Thus, σ_1 and σ_2 are the only solutions that are computed by rigid basic superposition for this example.

7.8 Solving Mixed E-unification Problems

Since universal E-unification is already undecidable, *mixed* E-unification is—in general—undecidable as well. It is, however, possible to enumerate a complete set of MGUs.

EXAMPLE 92 The following example requires only very little non-equality reasoning. A powerful equality handling technique is needed to find a closed tableau,

[5] In this example, we use g^2x as an abbreviation for $g(g(x))$, etc.

and the universal formula version of tableaux has to be used to restrict the search space: If Γ consists of the axioms[6]

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

then

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

To prove this, the tableau shown in Figure 5 has to be closed. Formula (2) is derived from the negated theorem (1) by three δ- and one γ-rule application; (3) and (4) are derived from (2).

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 ${}_3T^AP$ [Beckert *et al.*, 1996], that uses a completion-based method for finding solutions of mixed E-unification problems, proceeds as follows. The reduction rule $(\forall x)(i(x, x) \rightarrow tr)$ is one of the first rules that are deduced from Γ. Using this rule, the solution $\sigma = \{w_1 \mapsto c_1\}$ 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 $\sigma' = \{w_1 \mapsto tr\}$ 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 $(\forall x)(\forall y)(i(y, i(x, y)) \rightarrow tr)$ which can be applied to show that the empty substitution is a solution to $P_r\sigma'$ and that therefore the right branch is closed.

7.9 Simultaneous E-unification

Instead of closing one branch after the other, one can search for a simultaneous refuter for all branches of a tableau. However, this is much more difficult than closing a single branch. Although (non-simultaneous) *rigid* E-unification is decidable, it is undecidable whether a simultaneous solution to several E-unification problems exists [Degtyarev and Voronkov, 1996b]. It is as well undecidable whether

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

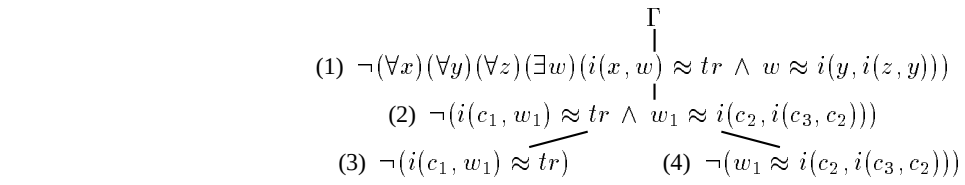


Figure 5. The tableau that has to be closed to prove the theorem from Example 92.

there is a substitution closing all branches of a given free variable tableau simultaneously after it has been expanded by a *fixed* number of copies of the universally quantified formulae it contains [Voda and Komara, 1995; Gurevich and Veanes, 1997].

In the same way as it may be surprising on first sight that simple rigid E-unification is decidable, it may be surprising that moving from simple to simultaneous problems destroys decidability—even more so considering that the simultaneous versions of other decidable types of unification (including syntactical unification and ground E-unification) are decidable. However, simultaneous rigid E-unification turns out to have a much higher expressiveness than simple rigid E-unification; it is even possible to encode Turing Machines into simultaneous rigid E-unification problems [Veanes, 1997]. For an overview of simultaneous rigid E-unification see [Degtyarev and Voronkov, 1998; Beckert, 1998].

Since simultaneous rigid E-unification is undecidable, sets of unifiers can only be enumerated; in general they are not finite. Solutions to a simultaneous problem can be computed combining solutions to its constituents $\langle E_i, s_i, t_i \rangle$; however, it is not possible to compute a finite complete set of unifiers of the simultaneous problem by combining solutions from finite sets of unifiers of the constituents that are complete w.r.t. the subsumption relation $\leq^W_{\mathcal{E},E}$, because they are complete w.r.t. different relations $\leq^W_{\mathcal{E},E_i}$. Thus, the subsumption relation $\leq^W_{\mathcal{E}}$ has to be used, which is the same for all i (but does not allow to construct *finite* complete sets of unifiers).

The undecidability of simultaneous rigid E-unification implies that, if a background reasoner produces only a *finite* number of solutions to any (non-simultaneous) rigid E-unification problem, then closing a tableau T may require to extend T by additional instances of equalities and terms even if there is a substitution that closes all branches of T simultaneously and there is, thus, a solution to a simultaneous rigid E-unification problem extracted from T. That notwithstanding, the background reasoner *may* be complete; and in that case the advantages of finite sets of solutions prevail. A complete background reasoner of this type can be built using rigid basic superposition (Sect. 7.7). It is not known whether the same can be achieved using (finite) sets of unifiers that are complete w.r.t. the subsumption relation $\leq^W_{\mathcal{E},E}$.

8 INCREMENTAL THEORY REASONING

Besides the efficiency of the foreground and the background reasoner, the interaction between them plays a critical rôle for the efficiency of the combined system: It is a difficult problem to decide whether it is useful to call the background reasoner at a certain point or not, and how much time and other resources to spend for its computations. In general, giving a perfect answer to these questions is as difficult as the theory reasoning problem itself. Even with good heuristics at hand, one cannot avoid calling the background reasoner at the wrong point: either too early or too late.

This problem can (at least partially) be avoided by using incremental methods for background reasoning [Beckert and Pape, 1996], i.e., algorithms that—after a futile try to solve a theory reasoning problem—allow to save the results of the background reasoner's computations and to reuse this data for a later call.[7] Then, in case of doubt, the background reasoner can be called early without running the risk of doing useless computations. In addition, an incremental background reasoner can reuse data multiply if different extensions of a problem have to be handled. An important example are completion-based methods for equality reasoning, which are inherently incremental.

As already mentioned in Section 2.5, one of the main problems in using theorem reasoning techniques in practice is the efficient combination of foreground and background reasoner and their interaction—in particular if (a) the computation steps of the background reasoner are comparatively complex, and (b) in case calling the background reasoner may be useless because no refuter exists or can be found.

On the one hand, a late call to the background reasoner can lead to bigger tableaux and redundancy. Although several branches may share the same subbranch and thus contain the same key for which a refuter exists, the background reasoner is called separately for these branches and the refuter has to be computed repeatedly. On the other hand, an early call to the background reasoner may not be successful and time consuming; this is of particular disadvantage if the existence of a refuter is undecidable and, as a result, the background reasoner does not terminate although no refuter exists.

Both these phenomena may considerably decrease the performance of a prover, and it is very difficult to decide (resp. to develop good heuristics which decide)

1. when to call the background reasoner;

2. when to stop the background reasoner if it does not find a refuter.

[7] This should not be confused with deriving a refuter and handing it back to the foreground reasoner. The information derived by an incremental background reasoner cannot be used by the foreground reasoner, but only by the background reasoner during later calls.

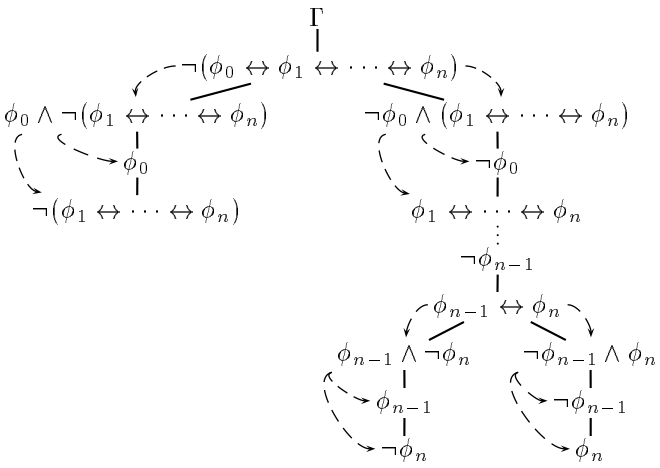


Figure 6. Short tableau proof for $\Gamma \models_{\mathcal{T}} \phi_0 \leftrightarrow \cdots \leftrightarrow \phi_n$ (Example 93).

EXAMPLE 93 The following example shows that earlier calls to the background reasoner can reduce the size of a tableau proof exponentially. Let $\Gamma \subset Form_\Sigma$ be a set of formulae and let $\phi_n \in Form_\Sigma$, $n \geq 0$, be formulae such that, for some theory $\mathcal{T}$, $\Gamma \models_{\mathcal{T}} \neg\phi_n$ $(n \geq 0)$. Figure 6 shows a proof for

$$\Gamma \models_{\mathcal{T}} \phi_0 \leftrightarrow \phi_1 \leftrightarrow \cdots \leftrightarrow \phi_n \ ,$$

where the background reasoner is called when a literal of the form ϕ_n appears on a branch (with the key $\Phi = \Gamma \cup \{\phi_n\}$). As a result, all the left-hand branches are closed immediately and the tableau is of linear size in n.

If the background reasoner were only called when a branch is exhausted, i.e., when no further expansion is possible, then the tableau would have 2^n branches and the background reasoner would have to be called 2^n times (instead of n times).

An *incremental* background reasoner can be of additional advantage if the computations that are necessary to show that $\Gamma \models_{\mathcal{T}} \neg\phi_n$ are similar for all n. In that case, a single call to the background reasoner in the beginning may provide information that later can be reused to close all the branches with less effort.

Even the best heuristics cannot avoid calls to the background reasoner at the wrong time. However, under certain conditions, it is possible to avoid the adverse consequences of early calls: If the algorithm that the background reasoner uses is *incremental*, i.e., if the data produced by the background reasoner during a futile try to compute refuters can be reused for a later call.

If early calls have no negative effects, the disadvantages of late calls can easily be avoided by using heuristics that, in case of doubt, call the background reasoner

at an early time. The problem of not knowing when to stop the background reasoner is solved by calling it more often with less resources (time, etc.) for each call.

An additional advantage of using incremental background reasoners in the tableau framework is that computations can be reused repeatedly for different extensions of a branch—even if the computation of refuters proceeds differently for these extensions.

8.1 Incremental Keys and Algorithms

Obviously, there has to be some strong relationship between the keys transferred to the background reasoner, to make it possible to reuse the information computed. Since, between calls to the background reasoner, (1) the tableau may be extended by new formulae and (2) substitutions (refuters) may be applied (to the tableau), these are the two operations we allow for changing the key:

DEFINITION 94 *A sequence $(\Phi_i)_{i\geq 0}$ of keys is* incremental *if, for $i \geq 0$, there is a set $\Psi_i \subset Form_\Sigma$ of formulae and a substitution σ_i such that $\Phi_{i+1} = \Phi_i\sigma_i \cup \Psi_i$, where $\Psi_i = \Psi_i\sigma_i$.*

In general, not all refuters of Φ_i are refuters of Φ_{i+1} (because a substitution is applied); nor are all refuters of Φ_{i+1} refuters of Φ_i (because new formulae are added).

To be able to formally denote the state the computation of a background reasoner has reached and the data generated, we use the following notion of incremental background reasoner:

DEFINITION 95 *An* incremental background reasoner $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ *is a background reasoner (Def. 31) that can be described using*

1. *an* algorithm *(a function) $\mathcal{A} : \mathcal{D} \to \mathcal{D}$ operating on a data structure $\mathcal{D}$,*
2. *an initialization function $\mathcal{I} : 2^{Form_\Sigma} \to \mathcal{D}$ that transforms a given key into the data structure format, and*
3. *an output function $\mathcal{S} : \mathcal{D} \to 2^{Subst^*_\Sigma}$ that extracts computed refuters from the data structure,*

such that for every key $\Phi \subset Form_\Sigma$ for which $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ is defined

$$\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}(\Phi) = \bigcup_{i\geq 0} \mathcal{S}(\mathcal{A}^i(\mathcal{I}(\Phi))) \ .$$

The above definition does not restrict the type of algorithms that may be used; every background reasoner whose computations proceed in steps can be described this way. If a background reasoner applies different transformations to the data at each step of its computation, this can be modeled by adding the state of the reasoner to the data structure such that the right operation or sub-algorithm can be applied each time the background reasoner is invoked.

Of course, the input and output functions have to be reasonably easy to compute; in particular, the cost of their computation has to be much smaller than that of applying the algorithm $\mathcal{A}$, which is supposed to do the actual work.

The goal is to be able to stop the background reasoner when it has reached a certain state in its computations for a key Φ, and to proceed from that state with a new key $\Phi' = \Phi\sigma \cup \Psi$. For that purpose, an update function is needed that adapts the data structure representing the state of the computation to the new formulae Ψ and the substitution σ.

DEFINITION 96 *Let $\mathcal{T}$ be a theory and $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ a complete incremental background reasoner for $\mathcal{T}$. An update function*

$$\mathcal{U} : \mathcal{D} \times 2^{Form_\Sigma} \times Subst^*_\Sigma \longrightarrow \mathcal{D}$$

is correct *(for $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$) if a complete background reasoner $\mathcal{R}'_{\mathcal{A},\mathcal{I},\mathcal{S}}$ is defined by: for every key Φ*

1. *choose $\Phi' \subset Form_\Sigma$ and $\sigma \in Subst^*_\Sigma$ such that $\Phi = \Phi'\sigma \cup \Psi$ arbitrarily;*
2. *compute $D_n = \mathcal{U}(\mathcal{A}^n(\mathcal{I}(\Phi')),\ \Psi,\ \sigma)$ for an arbitrary $n \geq 0$;*
3. *set $\mathcal{R}'_{\mathcal{A},\mathcal{I},\mathcal{S}}(\Phi) = \bigcup_{i\geq 0} \mathcal{S}(\mathcal{A}^i(D_n))$.*

According to the above definition, a correct update function behaves as expected when used for a single incremental step. Theorem 97 shows that this behavior extends to sequences of incremental steps. In addition, the algorithm can be applied arbitrarily often between incremental steps:

THEOREM 97 *Let $\mathcal{T}$ be a theory, $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ a complete incremental background reasoner for $\mathcal{T}$, and $\mathcal{U}$ a correct update function for $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$.*

Then $\mathcal{R}^*_{\mathcal{A},\mathcal{I},\mathcal{S}}$ is a complete background reasoner for $\mathcal{T}$ that is defined by: for every key Φ

1. *choose an arbitrary incremental sequence $(\Phi_i)_{i\geq 0}$ of keys where*

$$\Phi_{i+1} = \Phi_i\sigma_i \cup \Psi_i \qquad (i \geq 0)\ ,$$

   and $\Phi = \Phi_k$ for some $k \geq 0$;

2. *let $(D_i)_{i\geq 0} \subset \mathcal{D}$ be defined by*

(a) $D_0 = \mathcal{I}(\Phi_0)$,

(b) $D_{i+1} = \mathcal{U}(\mathcal{A}^{n_i}(D_i),\ \sigma_{i+1},\ \Psi_{i+1})$ *for some* $n_i \geq 0$*;*

3. *set* $\mathcal{R}^*_{\mathcal{A},\mathcal{I},\mathcal{S}}(\Phi) = \bigcup_{j\geq 0} \mathcal{S}(\mathcal{A}^j(D_k))$.

EXAMPLE 98 Let $(\Phi_i)_{i\geq 0}$ be an incremental sequence of keys such that $\Phi_{i+1} = \Phi_i\sigma_i \cup \Psi_i$ ($i \geq 0$). Then, for every sound and complete incremental background reasoner $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$, the trivial update function defined by

$$\mathcal{U}(D, \Psi_i, \sigma_i) = \mathcal{I}(\Phi_i\sigma_i \cup \Psi_i)$$

is correct.

The above example shows that it is not sufficient to use any correct update function to achieve a better performance of the calculus, because using the trivial update function means that no information is reused. A useful update function has to preserve the information contained in the computed data.

Whether there actually is a useful and reasonably easy to compute update function depends on the theory $\mathcal{T}$, the background reasoner, and its data structure.

Such a useful update function exists for a background reasoner for completion-based equality handling [Beckert and Pape, 1996]. Another important example are background reasoners based on resolution: if a resolvent can be derived from a key Φ, then it is valid for all extensions $\Phi \cup \Psi$ of Φ; resolvents may be invalid for an instance $\Phi\sigma$ of the key, but to check this is much easier than to re-compute all resolvents. In [Baumgartner, 1996], a uniform translation from Horn theories to partial background reasoners based on unit-resulting positive hyper-resolution with input restriction is described. This procedure can be used to generate incremental background reasoners for a large class of theories.

8.2 Semantic Tableaux and Incremental Theory Reasoning

The incremental theory reasoning method presented in the previous section is easy to use for tableau-like calculi, because the definition of incremental sequences of keys matches the construction of tableau branches. The keys of a sequence are taken from an expanding branch, and the substitutions are those applied to the whole tableau.

The keys used in calls to the background reasoner as well as the information computed so far by the background reasoner have to be attached to the tableau branches:

DEFINITION 99 *A* tableau for incremental theory reasoning *is a (finite) multi-set of tableau branches where a tableau* branch *is a triple* $\langle\Theta, D, \Phi\rangle$*;* Θ *is a (finite) multi-set of first-order formulae,* $D \in \mathcal{D}$ *(where* $\mathcal{D}$ *is the data structure used by the background reasoner), and* $\Phi \subset Form_\Sigma$ *is a key.*

Now, the tableau calculus with theory reasoning introduced in Section 3.4 can be adapted to *incremental* theory reasoning: calling the background reasoner is added as a further possibility of changing the tableau (besides expanding and closing branches).

DEFINITION 100 (Incremental reasoning version.) *Given a theory $\mathcal{T}$, an incremental background reasoner $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ for $\mathcal{T}$ (Def. 95), and a correct update function $\mathcal{U}$ for $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ (Def. 96), an* incremental theory reasoning tableau proof *for a first-order sentence ϕ consists of a sequence*

$$\{\{\neg\phi\}\} = T_0, T_1, \ldots, T_{n-1}, T_n = \emptyset \qquad (n \geq 0)$$

of tableaux such that, for $1 \leq i \leq n$, the tableau T_i is constructed from T_{i-1}

1. *by applying one of the expansion rules from Table 6, i.e., there is a branch $B = \langle\Theta, D, \Phi\rangle \in T_{i-1}$ and a formula $\phi \in \Theta$ (that is not a literal) such that*

$$T_i = (T_{i-1} \setminus \{B\}) \cup \begin{cases} \{\langle(\Theta \setminus \{\alpha\}) \cup \{\alpha_1, \alpha_2\}, D, \Phi\rangle\} & \text{if } \phi = \alpha \\ \{\langle(\Theta \setminus \{\phi\}) \cup \{\beta_1\}, D, \Phi\rangle, \\ \ \langle(\Theta \setminus \{\beta\}) \cup \{\beta_2\}, D, \Phi\rangle\} & \text{if } \phi = \beta \\ \{\langle\Theta \cup \{\gamma_1(y)\}, D, \Phi\rangle\} & \text{if } \phi = \gamma(x) \\ \{\langle(\Theta \setminus \{\phi\}) \cup \{\delta_1(f(x_1, \ldots, x_n))\}, D, \Phi\rangle\} & \text{if } \phi = \delta(x) \end{cases}$$

   where $y \in V$ is a new variable not occurring in T_{i-1}, $f \in F_\Sigma$ is a Skolem function symbol not occurring in T_{i-1}, and $x_1, \ldots, x_n$ are the free variables in ϕ;

2. *by applying the incremental theory expansion rule, i.e., there is a branch $B = \langle\Theta, D, \Phi\rangle$ in T_{i-1} and a $\mathcal{T}$-refuter $\langle\sigma, \{\rho_1, \ldots, \rho_k\}\rangle$ ($k \geq 1$) in the set $\mathcal{S}(D)$, and*

$$T_i = \{\langle\Theta'\sigma, D', \Phi'\rangle \mid \langle\Theta', D', \Phi'\rangle \in (T_{i-1} \setminus \{B\})\} \cup \{\langle\Theta \cup \{\rho_j\}, D, \Phi\rangle \mid 1 \leq j \leq k\}$$

3. *by applying the incremental theory closure rule, i.e., there is a branch $B = \langle\Theta, D, \Phi\rangle$ in T_{i-1} that is $\mathcal{T}$-closed under σ, i.e., $\sigma \in \mathcal{S}(D)$, and*

$$T_i = \{\langle\Theta'\sigma, D', \Phi'\rangle \mid \langle\Theta', D', \Phi'\rangle \in (T_{i-1} \setminus \{B\})\}$$

4. *or by calling the background reasoner, i.e., there is a branch $B = \langle\Theta, D, \Phi\rangle$ in T_{i-1}, a number $c > 0$ of applications, and a key*

$$\Phi' = \Phi\sigma \cup \Psi \subset \Theta = \{(\forall x_1^i) \cdots (\forall x_{m_i}^i)\phi_i \mid 1 \leq i \leq p\}$$

   where

(a) $\phi_1, \ldots, \phi_p \in \Theta$,

(b) $\{x^i_1, \ldots, x^i_{m_i}\} \subset UVar(\phi_i)$ *for* $1 \leq i \leq p$.

and

$$T_i = (T_{i-1} \setminus \{B\}) \cup \{\langle \Theta, \mathcal{A}^c(\mathcal{U}(D, \Psi, \sigma)), \Phi' \rangle\} \ .$$

Soundness and completeness of the resulting calculus are a corollary of Theorems 47, 58, and 97:

THEOREM 101 *Let $\phi \in Form_\Sigma$ be a sentence. If there is an incremental tableau proof for ϕ (Def. 100), then ϕ is a $\mathcal{T}$-tautology.*

If $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$ is a complete incremental background reasoner for a theory $\mathcal{T}$ and the formula ϕ is a $\mathcal{T}$-tautology, then an incremental tableau proof for ϕ can be constructed using $\mathcal{R}_{\mathcal{A},\mathcal{I},\mathcal{S}}$.

The maximal cost reduction that can be achieved by using an incremental reasoner is reached if the costs are those of the non-incremental background reasoner called neither too early nor too late, i.e., if always the right key in the incremental sequence is chosen and the background reasoner is only called for that key (which is not possible in practice).

In practice, the costs of an incremental method are between the ideal value and the costs of calling a non-incremental reasoner for each of the keys in an incremental sequence (without reusing).

But even if the costs for one sequence, i.e., for closing one tableau branch, are higher than those of using a non-incremental method, the overall costs for closing the whole tableau can be small, because information is reused for more than one branch.

9 EQUALITY REASONING BY TRANSFORMING THE INPUT

Methods based on transforming the input are inherently not specific for semantic tableaux (although they might be more suitable for tableaux than for other calculi). They do not require the tableau calculus to be adopted to theory reasoning.

The simplest—however useless—method is to just add the theory axioms to the input formulae. A better way to "incorporate" the equality axioms into the formulae to be proven is D. Brand's STE-modification [Brand, 1975], which is described below. An improved transformation using term orderings has been presented in [Bachmair *et al.*, 1997].

Usually, STE-modification is only defined for formulae in clausal form; but since a transformation to clausal form may be of disadvantage for non-normal form calculi like semantic tableaux, we present an adaptation of Brand's method for formulae in Skolemized negation normal form.

DEFINITION 102 *Let ϕ be a formula in Skolemized negation normal form. The* E-modification *of ϕ is the result of applying the following transformations iteratively as often as possible:*

1. *If a literal of the form $p(\ldots, s, \ldots)$ or $\neg p(\ldots, s, \ldots)$ occurs in the formula where $s \notin V$, then replace it by*

$$(\forall x)(\neg(s \approx x) \vee p(\ldots, x, \ldots)) \quad \textit{resp.}$$
$$(\forall x)(\neg(s \approx x) \vee \neg p(\ldots, x, \ldots))$$

   where x is a new variable.

2. *If an equality of the form $f(\ldots, s, \ldots) \approx t$ or $t \approx f(\ldots, s, \ldots)$ occurs in the formula where $s \notin V$, then replace it by*

$$(\forall x)(\neg(s \approx x) \vee f(\ldots, x, \ldots) \approx t)$$

   where x is a new variable.

The STE-modification *of ϕ is the result of (non-iteratively) replacing in the E-modification ϕ' of ϕ all equalities $s \approx t$ by*

$$(\forall x)(\neg(t \approx x) \vee s \approx x) \wedge (\forall x)(\neg(s \approx x) \vee t \approx x)$$

where x is a new variable.

EXAMPLE 103 The STE-modification of $(\forall x)(p(f(a, g(x))))$ is

$$(\forall x)(\forall u)(\forall v)(\forall w)(\neg(a \approx u) \vee \neg(g(x) \approx v) \vee \neg(f(u, v) \approx w) \vee p(w)) \ .$$

The STE-modification of $(\forall x)(\forall y)(f(x, y) \approx g(a))$ is

$$(\forall x)(\forall y)(\forall z)(\neg(a \approx z) \vee ((\forall u)(\neg(f(x, y) \approx u) \vee g(z) \approx u) \wedge (\forall u)(\neg(g(z) \approx u) \vee f(x, y) \approx u))) \ .$$

To prove the $\mathcal{E}$-unsatisfiability of the STE-modification of a formula, it is sufficient to use the reflexivity axiom; symmetry, transitivity and monotonicity axioms are not needed any more.

THEOREM 104 (Brand, 1975) *Let ϕ be a sentence in Skolemized negation normal form, and let ϕ' be the STE-modification of ϕ. Then ϕ is $\mathcal{E}$-unsatisfiable if and only if*

$$\phi' \wedge (\forall x)(x \approx x)$$

is unsatisfiable.

10 CONCLUSION

We have given an overview of how to design the interface between semantic tableaux (the foreground reasoner) and a theory background reasoner. The problem of handling a certain theory has been reduced to finding an efficient background reasoner for that theory. The search for efficient methods has not come to an end, however, because there is no universal recipe for designing background reasoners. Nevertheless, some criteria have been presented that a background reasoner should satisfy and useful features it should have.

Specialized methods have been presented for handling equality; the most efficient of these are based on E-unification techniques. Similar to the design of background reasoners in general, the problem of developing E-unification procedures is difficult to solve in a uniform way. The research in the field of designing such procedures for certain equality theories has produced a huge amount of results, that is still rapidly growing, in particular for rigid and mixed E-unification.

ACKNOWLEDGEMENTS

I would like to thank Peter Baumgartner, Marcello D'Agostino, Paliath Narendran, and Christian Pape for fruitful comments on earlier versions of this chapter.

REFERENCES

[Andrews, 1981] Andrews, P. B. Theorem proving through general matings. *Journal of the ACM*, **28**, 193–214.

[Bachmair *et al.*, 1997] Bachmair, L., Ganzinger, H., and Voronkov, A. *Elimination of Equality via Transformation with Ordering Constraints*. Technical Report MPI-I-97-2-012. MPI für Informatik, Saarbrücken.

[Baumgartner, 1992] Baumgartner, P. A model elimination calculus with built-in theories. *Pages 30–42 of:* Ohlbach, H.-J. (ed.), *Proceedings, German Workshop on Artificial Intelligence (GWAI)*. LNCS 671. Springer.

[Baumgartner, 1996] Baumgartner, P. Linear and unit-resulting refutations for Horn theories. *Journal of Automated Reasoning*, **16**(3), 241–319.

[Baumgartner, 1998] Baumgartner, P. *Theory Reasoning in Connection Calculi*. LNCS. Springer. To appear.

[Baumgartner and Petermann, 1998] Baumgartner, P., and Petermann, U. Theory reasoning. *In:* Bibel, W., and Schmitt, P. H. (eds.), *Automated Deduction – A Basis for Applications*, vol. I. Kluwer.

[Baumgartner *et al.*, 1992] Baumgartner, P., Furbach, U., and Petermann, U. *A Unified Approach to Theory Reasoning*. Forschungsbericht 15/92. University of Koblenz.

[Becher and Petermann, 1994] Becher, G., and Petermann, U. Rigid unification by completion and rigid paramodulation. *Pages 319–330 of:* Nebel, B., and Dreschler-Fischer, L. (eds.), *Proceedings, 18th German Annual Conference on Artificial Intelligence (KI-94), Saarbrücken, Germany*. LNCS 861. Springer.

[Beckert, 1994] Beckert, B. A completion-based method for mixed universal and rigid E-unification. *Pages 678–692 of:* Bundy, A. (ed.), *Proceedings, 12th International Conference on Automated Deduction (CADE), Nancy, France*. LNCS 814. Springer.

[Beckert, 1997] Beckert, B. Semantic tableaux with equality. *Journal of Logic and Computation*, **7**(1), 39–58.

[Beckert, 1998] Beckert, B. Rigid E-unification. *In:* Bibel, W., and Schmitt, P. H. (eds.), *Automated Deduction – A Basis for Applications*, vol. I. Kluwer.

[Beckert and Hähnle, 1992] Beckert, B., and Hähnle, R. An improved method for adding equality to free variable semantic tableaux. *Pages 507–521 of:* Kapur, D. (ed.), *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY, USA*. LNCS 607. Springer.

[Beckert and Hähnle, 1998] Beckert, B., and Hähnle, R. Analytic tableaux. *In:* Bibel, W., and Schmitt, P. H. (eds.), *Automated Deduction – A Basis for Applications*, vol. I. Kluwer.

[Beckert and Pape, 1996] Beckert, B., and Pape, C. Incremental theory reasoning methods for semantic tableaux. *Pages 93–109 of:* Miglioli, P., Moscato, U., Mundici, D., and Ornaghi, M. (eds.), *Proceedings, 5th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Palermo, Italy*. LNCS 1071. Springer.

[Beckert *et al.*, 1996] Beckert, B., Hähnle, R., Oel, P., and Sulzmann, M. The tableau-based theorem prover $_3TAP$, version 4.0. *Pages 303–307 of: Proceedings, 13th International Conference on Automated Deduction (CADE), New Brunswick, NJ, USA*. LNCS 1104. Springer.

[Bibel, 1987] Bibel, W. *Automated Theorem Proving*. Second edn. Vieweg, Braunschweig. First edition published in 1982.

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

[Browne, 1988] Browne, R. J. *Ground Term Rewriting in Semantic Tableaux Systems for First-Order Logic with Equality*. Technical Report UMIACS-TR-88-44. College Park, MD.

[Bürckert, 1990] Bürckert, H. A resolution principle for clauses with constraints. *Pages 178–192 of: Proceedings, 10th International Conference on Automated Deduction (CADE)*. LNCS 449. Springer.

[Cantone *et al.*, 1989] Cantone, D., Ferro, A., and Omodeo, E. *Computable Set Theory*. International Series of Monographs on Computer Science, vol. 6. Oxford University Press.

[de Kogel, 1995] de Kogel, E. Rigid E-unification simplified. *Pages 17–30 of: Proceedings, 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, St. Goar*. LNCS 918. Springer.

[Degtyarev and Voronkov, 1996a] Degtyarev, A., and Voronkov, A.a. Equality elimination for the tableau method. *Pages 46–60 of:* Calmet, J., and Limongelli, C. (eds.), *Proceedings, International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO), Karlsruhe, Germany*. LNCS 1128.

[Degtyarev and Voronkov, 1996b] Degtyarev, A., and Voronkov, A.b. Simultaneous rigid E-unification is undecidable. *Pages 178–190 of:* Kleine Büning, H. (ed.), *Proceedings, Annual Conference of the European Association for Computer Science Logic (CSL'95)*. LNCS 1092. Springer.

[Degtyarev and Voronkov, 1998] Degtyarev, A., and Voronkov, A. What you always wanted to know about rigid E-unification. *Journal of Automated Reasoning*, **20**(1), 47–80.

[Digricoli and Harrison, 1986] Digricoli, V. J., and Harrison, M. C. Equality-based binary resolution. *Journal of the ACM*, **33**(2), 253–289.

[Fitting, 1996] Fitting, M. C. *First-Order Logic and Automated Theorem Proving*. Second edn. Springer.

[Furbach, 1994] Furbach, U. Theory reasoning in first order calculi. *Pages 139–156 of:* v. Luck, K., and Marburger, H. (eds.), *Proceedings, Third Workshop on Information Systems and Artificial Intelligence, Hamburg, Germany*. LNCS 777. Springer.

[Gallier and Snyder, 1990] Gallier, J. H., and Snyder, W. Designing unification procedures using transformations: A survey. *Bulletin of the EATCS*, **40**, 273–326.

[Gallier *et al.*, 1987] Gallier, J. H., Raatz, S., and Snyder, W. Theorem proving using rigid E-unification, equational matings. *In: Proceedings, Symposium on Logic in Computer Science (LICS), Ithaka, NY, USA*. IEEE Press.

[Gallier *et al.*, 1988] Gallier, J. H., Narendran, P., Plaisted, D., and Snyder, W. Rigid E-unification is NP-complete. *In: Procceedings, Symposium on Logic in Computer Science (LICS)*. IEEE Press.

[Gallier *et al.*, 1990] Gallier, J. H., Narendran, P., Plaisted, D., and Snyder, W. Rigid E-unification: NP-completeness and application to equational matings. *Information and Computation*, 129–195.

[Gallier *et al.*, 1992] Gallier, J. H., Narendran, P., Raatz, S., and Snyder, W. Theorem proving using equational matings and rigid E-unification. *Journal of the ACM*, **39**(2), 377–429.

[Grieser, 1996] Grieser, G. *An Implementation of Rigid E-Unification Using Completion and Rigid Paramodulation*. Forschungsbericht FITL-96-4. FIT Leipzig e.V.

[Gurevich and Veanes, 1997] Gurevich, Y., and Veanes, M. *Some Undecidable Problems Related to the Herbrand Theorem*. UPMAIL Technical Report 138. Uppsala University.

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

[Jouannaud and Kirchner, 1991] Jouannaud, J.-P., and Kirchner, C. Solving equations in abstract algebras: A rule-based survey of unification. *Pages 257–321 of:* Lassez, J., and Plotkin, G. (eds.), *Computational Logic – Essays in Honor of Alan Robinson*. MIT Press.

[Kanger, 1963] Kanger, S. A simplified proof method for elementary logic. *Pages 87–94 of:* Braffort, P., and Hirschberg, D. (eds.), *Computer Programming and Formal Systems*. North Holland. *Reprint as pages 364–371 of:* Siekmann, J., and Wrightson, G. (eds.), *Automation of Reasoning. Classical Papers on Computational Logic*, vol. 1. Springer, 1983.

[Kozen, 1981] Kozen, D. Positive first-order logic is NP-complete. *IBM Journal of Research and Development*, **25**(4), 327–332.

[Lis, 1960] Lis, Z. Wynikanie semantyczne a wynikanie formalne. *Studia Logica*, **10**, 39–60. In Polish with English summary.

[Loveland, 1969] Loveland, D. W. A simplified format for the model elimination procedure. *Journal of the ACM*, **16**(3), 233–248.

[Murray and Rosenthal, 1987a] Murray, N. V., and Rosenthal, E.a. Inference with path resolution and semantic graphs. *Journal of the ACM*, **34**(2), 225–254.

[Murray and Rosenthal, 1987b] Murray, N. V., and Rosenthal, E.b. Theory links: Applications to automated theorem proving. *Journal of Symbolic Computation*, **4**, 173–190.

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

[Nieuwenhuis and Rubio, 1995] Nieuwenhuis, R., and Rubio, A. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computation*, **19**, 321–351.

[Petermann, 1992] Petermann, U. How to build-in an open theory into connection calculi. *Journal on Computer and Artificial Intelligence*, **11**(2), 105–142.

[Petermann, 1993] Petermann, U. Completeness of the pool calculus with an open built-in theory. *Pages 264–277 of:* Gottlob, G., Leitsch, A., and Mundici, D. (eds.), *Proceedings, 3rd Kurt Gödel Colloquium (KGC), Brno, Czech Republic*. LNCS 713. Springer.

[Plaisted, 1995] Plaisted, D. A. *Special Cases and Substitutes for Rigid E-Unification*. Technical Report MPI-I-95-2-010. Max-Planck-Institut für Informatik, Saarbrücken.

[Policriti and Schwartz, 1995] Policriti, A., and Schwartz, J. T. T-theorem proving I. *Journal of Symbolic Computation*, **20**, 315–342.

[Popplestone, 1967] Popplestone, R. J. Beth-tree methods in automatic theorem proving. *Pages 31–46 of:* Collins, N., and Michie, D. (eds.), *Machine Intelligence*, vol. 1. Oliver and Boyd.

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

[Robinson and Wos, 1969] Robinson, J. A., and Wos, L. Paramodulation and theorem proving in first order theories with equality. *In:* Meltzer, B., and Michie, D. (eds.), *Machine Intelligence*. Edinburgh University Press.

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

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

[Smullyan, 1995] Smullyan, R. M. *First-Order Logic*. Second corrected edn. Dover Publications, New York. First published in 1968 by Springer.

[Snyder, 1991] Snyder, W. *A Proof Theory for General Unification*. Boston: Birkhäuser.

[Stickel, 1985] Stickel, M. E. Automated deduction by theory resolution. *Journal of Automated Reasoning*, **1**, 333–355.

[Veanes, 1997] Veanes, M. On Simultaneous Rigid E-Unification. PhD Thesis, Uppsala University, Sweden.

[Voda and Komara, 1995] Voda, P., and Komara, J. *On Herbrand Skeletons*. Technical Report mff-ii-02-1995. Institute of Informatics, Comenius University, Bratislava, Slovakia.

Index