UPMAIL Technical Report No. 143
April 4, 1997

What You Always Wanted to Know About Rigid E-Unification

Anatoli Degtyarev[*]
Andrei Voronkov[†]

Computing Science Department
Uppsala University
Box 311, S-751 05 Uppsala,
Sweden

email {anatoli,voronkov}@csd.uu.se

[*]Supported by grants from INTAS, TFR and the Swedish Royal Academy of Sciences
[†]Supported by a TFR grant

Abstract

This paper solves an open problem posed by a number of researchers: the construction of a complete calculus for matrix-based methods with rigid E-unification. The use of rigid E-unification and simultaneous rigid E-unification for such methods has been proposed by Gallier, Raatz and Snyder [36]. After our proof of the undecidability of simultaneous rigid E-unification [27] it has become clear that one should look for more refined techniques to deal with equality in matrix-based methods. In this article, we define a complete proof procedure for first-order logic with equality based on an incomplete but terminating procedure for rigid E-unification. Our approach is applicable to the connection method and the tableau method and illustrated on the tableau method.

Section 1

Introduction

Algorithms for theorem-proving based on matings or tableaux in first-order logic without equality comprise two kinds of rules. Rules of the first kind construct matrices or tableaux from a given formula using a suitable amplification. Rules of the second kind try to close paths or branches using substitutions making the paths or branches inconsistent. These substitutions are unifiers of some atoms laying on a path or branch. Until recently, most approaches to introducing equality in such *matrix-based* methods tried to generalize such algorithms by a suitable modification of the notion of a unifier.

Such a modification using *simultaneous rigid E-unification* was introduced by Gallier, Raatz and Snyder [36] for the method of matings due to Andrews [1] or the connection method due to Bibel [12]. It can easily be represented in the tableau formalism. The method of matings interleaves two steps: amplification by quantifier duplication and search for mating for a given amplification. For formulas in disjunctive normal form this method was formulated earlier by Prawitz [59]. In this case amplification is represented by a matrix and mating is represented by a set of simultaneously satisfiable substitution conditions (mated pairs). Prawitz proposed a procedure for constructing substitution conditions one by one, closing the corresponding paths in the matrix through search with backtracking. Procedures of this kind were used in later formalizations and implementations of the tableau method, the method of matings or the connection method for formulas without equality. For example, Fitting [32] formulates this via an inference rule called MGU atomic closure rule.

Gallier, Raatz and Snyder [36] tried to describe a similar procedure for logic with equality. For example, Gallier et.al. [34] describe such a procedure in which a substitution condition is formalized via rigid E-unification, and the set of substitution conditions via simultaneous rigid E-unification.

Simultaneous rigid E-unification can be formulated as follows. Given equations $s_i \simeq t_i$ and finite sets of equations E_i, $i \in \{1, \dots, n\}$, find a substitution σ such that $\vdash (\bigwedge_{e \in E_i} e\sigma) \supset s_i\sigma \simeq t_i\sigma$, for all i (here $\vdash$ means provability in first-order logic with equality). The corresponding instance of the simultaneous rigid E-unification problem is denoted by the system of *rigid equations* $E_i \vdash_\forall s_i \simeq t_i$.

Example 1.1 In this and further examples we shall often omit parentheses in terms with unary function symbols, for example we write ffb instead of $f(f(b))$. Assume that we want to prove the formula $\varphi = \exists xyzu((a \simeq b \supset g(x,u,v) \simeq g(y,fc,fd)) \wedge (c \simeq d \supset g(u,x,y) \simeq g(v,fa,fb)))$. After several applications of tableau expansion rules to the negation normal form of $\neg\varphi$ (α-, β- and γ-rules in the terminology of Smullyan [63] or Fitting [32]) we obtain the following tableau (we only consider the part of the tableau containing literals, omitting non-literal formulas):

$$\begin{tikzcd} & {} \arrow[dl, no head] \arrow[dr, no head] & \\ a \simeq b \arrow[d, no head] & & c \simeq d \arrow[d, no head] \\ g(x,u,v) \not\simeq g(y,fc,fd) & & g(u,x,y) \not\simeq g(v,fa,fb) \end{tikzcd}$$

Collecting formulas lying on the two branches in this tableau we obtain the following two *rigid equations* expressing inconsistency of this tableau:

$$\begin{array}{c} a \simeq b \vdash_{\forall} g(x,u,v) \simeq g(y,fc,fd) \\ c \simeq d \vdash_{\forall} g(u,x,y) \simeq g(v,fa,fb) \end{array}$$

This system of rigid equations has one solution $\{fa/x, fb/y, fc/u, fd/v\}$. This substitution can be found by applying the functional reflexivity rule and MGU replacement rule of [32] (in fact, the reformulation of the paramodulation rule for tableaux). Obviously, this tableau cannot be closed without the use of the functional reflexivity.[1]

Since the invention of simultaneous rigid E-unification by Gallier, Raatz and Snyder [36], there were a number of publications on simultaneous rigid E-unification itself and its use in theorem proving, for example Gallier et.al. [35, 37, 33, 34], Baumgartner [5], Beckert and Hähnle [11], Becher and Petermann [8], Beckert [9], Goubault [39] and Petermann [56]. Some of these articles were based on the conjecture that simultaneous rigid E-unification is decidable. There were several faulty proofs of the decidability of this problem (e.g. [35, 33, 39]).

The refutation procedure for first-order logic with equality using simultaneous rigid E-unification (e.g. Gallier et.al. [34]) was based on a faulty assumption that solutions to simultaneous rigid E-unification can be found by consecutive combination of finite *complete* sets of solutions for (non-simultaneous) rigid E-unification [33, 35]. Later we [23, 27] proved that simultaneous rigid E-unification is undecidable, which implied that Gallier et.al.'s procedure cannot, in general, find solutions to simultaneous rigid E-unification. However, it is not clear whether this implies incompleteness of this procedure for first-order logic with equality: there are examples when their procedure cannot find a solution for a given amplification in spite that such a solution exists, but can find a solution for a bigger amplification. Completeness of Gallier et.al.'s procedure or existence of a procedure complete for first-order logic with equality based on some set of solutions to rigid E-unification was an open problem (see e.g. Petermann [56] and Beckert [10]). Our paper gives a positive solution to this problem.

An advantage of Gallier et.al.'s procedure is that it allows one to extend the proof-search technology developed for tableaux without equality to the case with equality, using solutions to rigid E-unification instead of most general unifiers. In particular, for a given amplification Gallier

[1]The system of clauses corresponding to this example improves a result proved in Plaisted [58] by using a more complicated example. In this system of clauses

$$\begin{array}{l} a \simeq b \vee c \simeq d \\ a \simeq b \vee g(u,x,y) \not\simeq g(v,fa,fb) \\ c \simeq d \vee g(x,u,v) \not\simeq g(y,fc,fd) \\ g(x,u,v) \not\simeq g(y,fc,fd) \vee g(u,x,y) \not\simeq g(v,fa,fb) \end{array}$$

there is no refutation even by *unrestricted* rigid paramodulation (i.e. using non-ordered rigid paramodulation and paramodulation into variables), while Plaisted [58] gives an example showing incompleteness of *ordered* rigid paramodulation only.

et.al.'s procedure always terminates. A procedure of this kind is used in the theorem prover ${}_3T^AP$ [40] (R.Hähnle, private communication).

In this paper we define a procedure extending the tableau method to logic with equality based on an incomplete procedure for rigid E-unification. Nevertheless, our procedure is complete for first-order logic with equality. Hence, we rehabilitate Gallier et.al.'s program for adding equality to semantic tableaux. Moreover, our procedure solves rigid equations laying on different tableau branches independently. This strongly improves Gallier et.al.'s procedure which uses solutions of some rigid equations to solve rigid equations on other branches.

A similar approach has already been defined by Kanger [41] based on a more straightforward way of variable instantiation. As a method for finding a closing substitution, Kanger proposed an algorithm which can now be characterized as an incomplete (but terminating) algorithm for simultaneous rigid E-unifiability. Variables in a matrix (or a tableau) could be consecutively substituted by ground terms already occurring in the matrix. This procedure does not solve simultaneous rigid E-unifiability, but it gives a procedure complete for first-order logic with equality. In the terminology of Fitting [32] it means that a closing substitution can be found after a sufficiently high (but not necessarily minimal) number of applications of the γ-rule. The approach to substitutions based on this idea has been characterized as minus-normalization in Matulis [49] and Maslov [48].

However, for a language with function symbols minus-normalization is interesting mostly theoretically. Even in simplest cases, minus-normalization requires a huge number of instantiations. For example, in the tableau of Example 1.1, we have to consider 8^4 possible instantiations of variables x, y, u, v by terms in the set $\{a, b, c, d, fa, fb, fc, fd\}$. Moreover, it was proved that the use of minus-normalization can lead to considerable growth of derivations. Some results on minus-normalization are proved by Norgela [55].

In this paper we describe a logical calculus $\mathcal{BSE}$ for rigid E-unification based on the *rigid basic superposition* rule that is an adaptation of basic superposition of Bachmair et.al. [3] and Nieuwenhuis and Rubio [52], for "rigid" variables. For a given rigid E-unification problem (called *rigid equation* in this paper), there is only a finite number of $\mathcal{BSE}$-derivations for this problem. Thus, $\mathcal{BSE}$ gives us an algorithm returning a finite set of solutions to this rigid equation. We use these solutions to close a tableau branch in the same way as most general unifiers are used to close a branch in the MGU atomic closure rule of Fitting [32].

Section 2

Preliminaries

We present here a brief overview of notions and preliminary definitions necessary for understanding the paper. We assume basic knowledge of substitutions and unification.

Let Σ be a signature, and X be a set of variables. $T(\Sigma, X)$ denotes the set of all terms in the signature Σ with variables from X. The set of all *ground* terms in the signature Σ is denoted by $T(\Sigma)$.

A *literal* is either an atomic formula or a negation of an atomic formula. An *equation* is a literal $s \simeq t$, where $s, t \in T(\Sigma, X)$. *We do not distinguish equations $s \simeq t$ and $t \simeq s$.* Literals of the form $\neg(s \simeq t)$ are denoted by $s \not\simeq t$ and called *disequations.* For simplicity, we assume that $\simeq$ is the only predicate symbol of our first-order language. As usual, arbitrary first-order languages can be represented in such language by introducing a sort `bool` and replacing any non-equational atom A by $A \simeq$ `true` (for details see e.g. [4]).

By a *ground expression* (i.e. term or literal) we mean an expression containing no variables. For any expression E, $var(E)$ denotes the set of all variables occurring in E. For a sequence of variables $\bar{x}$, we shall sometimes denote $\bar{x}$ also the corresponding *set* of variables. We write $A[s]$ to indicate that an expression A contains s as a subexpression and denote by $A[t]$ the result of replacing this occurrence of s in A by t. By $A\sigma$ we denote the result of applying the substitution σ to A. If A is a formula, we can as usual rename bound variables in A before applying σ. We shall denote $\varphi(x)$ a formula φ with zero or more free occurrences of a variable x and write $\varphi(t)$ to denote the formula $\varphi\{t/x\}$.

A substitution θ whose domain is a subset of $\{x_1, \ldots, x_n\}$ is denoted by $\{x_1\theta/x_1, \ldots, x_n\theta/x_n\}$. A substitution σ is called *grounding for a set of variables V* iff for every variable $v \in V$ the term $v\sigma$ is ground.

Let θ_1 and θ_2 be two substitutions with disjoint domains, The *union of θ_1 and θ_2,* denoted $\theta_1 \cup \theta_2$ is the substitution σ defined as follows. For every variable v we have

$$\sigma(v) \rightleftharpoons \begin{cases} \theta_1(v) & \text{if } v \in dom(\theta_1) \\ \theta_2(v) & \text{if } v \in dom(\theta_2) \\ v & \text{if } v \notin dom(\theta_1) \cup dom(\theta_2) \end{cases}$$

Note that we use the union notation $\theta_1 \cup \theta_2$ *only* for substitutions with disjoint domains.

The inference systems used in this paper are defined with respect to a reduction ordering, denoted by $\succ$ which is total on ground terms. Our results are valid for any such ordering.

A formula is in the *Skolem negation normal form* iff it is constructed from literals using the connectives $\wedge, \vee$ and the quantifier $\forall$. There is a satisfiability-preserving structure-preserving translation of formulas without equivalences into formulas in Skolem negation normal form consisting of the standard skolemization and a translation into negation normal form used e.g. in Andrews [1]. In order to prove an arbitrary formula φ, we translate $\neg\varphi$ in Skolem negation normal form obtaining a formula ψ and try to establish unsatisfiability of ψ. For this reason, theorems in this paper are formulated in terms of unsatisfiability.

For an equation $s \simeq t$ and a multiset of equations E we write $E \vdash s \simeq t$ to denote that the formula $(\bigwedge_{e \in E} e) \supset s \simeq t$ is provable in first-order logic with equality. For such formulas provability can be tested by the congruence closure algorithm [62]. For the inclusion of multisets we shall use notation $S_1 \sqsubseteq S_2$.

Section 3

Rigid basic superposition

The term "rigid paramodulation" has already been used by Becher and Petermann [8] and Plaisted [58] for systems of inference rules in which all variables are treated as "rigid". For example, rigid clause paramodulation of Plaisted [58] is essentially paramodulation and resolution over a set of clauses, where all substitutions are applied to the whole set of clauses. A similar system for resolution has been proposed earlier by Chang [15] as V-resolution and by Chang and Lee [43] for resolution with paramodulation as V-resolution and V-paramodulation. We shall use the term "rigid basic superposition" to denote a "rigid" version of basic superposition. We formalize rigid basic superposition using constraints that is close to the presentation of Nieuwenhuis and Rubio [54].

Definition 3.1 (Constraints)
By an (ordering) *constraint* we mean a set of expressions which can be of two kinds: an *equality constraint* $s \simeq t$, or an *inequality constraint* $s \succ t$, where s, t are terms. A substitution θ is a *solution* to a constraint $s \simeq t$ (respectively, a constraint $s \succ t$) iff θ is grounding for $var(s) \cup var(t)$ and $s\theta$ coincides with $t\theta$ (respectively, $s\theta \succ t\theta$).

A substitution θ is a solution to a constraint $\mathcal{C}$ iff θ is a solution to every equality or inequality constraint in $\mathcal{C}$. A constraint $\mathcal{C}$ is *satisfiable* iff it has a solution. Constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ are called *equivalent* iff they have the same sets of solutions.

We assume that there is an effective procedure for checking constraint satisfiability. For example, there are efficient methods for solving ordering constraints for lexicographic path orderings given by Nieuwenhuis [53] and Nieuwenhuis and Rubio [54].

Definition 3.2 A *rigid equation* is an expression of the form $E \vdash_{\forall} s \simeq t$, where E is a finite multiset of equations and s, t are terms. Its *solution* is any substitution θ such that $E\theta \vdash s\theta \simeq t\theta$[1].

Below we shall introduce a system $\mathcal{BSE}$ for solving rigid equations. The derivable objects of $\mathcal{BSE}$ are *constraint rigid equations:*

Definition 3.3 (Constraint rigid equation)
A *constraint rigid equation* is a pair consisting of a rigid equation R and a constraint $\mathcal{C}$. Such a constraint rigid equation will be denoted $R \cdot \mathcal{C}$.

[1]The term "rigid equation" could be more adequately expressed as "instance of a (non-simultaneous) rigid E-unification problem", but this would be too lengthy.

Definition 3.4 (Calculus $\mathcal{BSE}$)
The *calculus* $\mathcal{BSE}$ of constraint rigid equations consists of the following inference rules:

Left rigid basic superposition:

$$\frac{E \cup \{l \simeq r, s[p] \simeq t\} \vdash_\forall e \cdot \mathcal{C}}{E \cup \{l \simeq r, s[r] \simeq t\} \vdash_\forall e \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l \simeq p\}}\ (lrbs)$$

Right rigid basic superposition:

$$\frac{E \cup \{l \simeq r\} \vdash_\forall s[p] \simeq t \cdot \mathcal{C}}{E \cup \{l \simeq r\} \vdash_\forall s[r] \simeq t \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l \simeq p\}}\ (rrbs)$$

Equality resolution:

$$\frac{E \vdash_\forall s \simeq t \cdot \mathcal{C}}{\vdash_\forall s \simeq s \cdot \mathcal{C} \cup \{s \simeq t\}}\ (er)$$

Application of all the rules is restricted to the following conditions:

1. The constraint at the conclusion of the rule is satisfiable;
2. The right-hand side of the rigid equation at the premise of the rule does not have the form $q \simeq q$.
3. In the basic superposition rules, the term p is not a variable.
4. In the left basic superposition rule, $s[r] \neq t$.

The basic restriction in $\mathcal{BSE}$ is formalized by representing most general unifiers through equality constraints. Condition 1 has two purposes. The satisfiability of equations in constraints is needed to preserve correctness of the method. The satisfiability of inequality constraints is needed to ensure termination (Theorem 3.9 below). Conditions 3–4 are not essential, they is added as standard optimizations used in paramodulation-based methods. Condition 2 prohibits to apply any rules to rigid equations of the form $E \vdash_\forall q \simeq q$.

We denote by $R \cdot \mathcal{C} \leadsto R' \cdot \mathcal{C}'$ the fact that $R' \cdot \mathcal{C}'$ is obtained from $R \cdot \mathcal{C}$ by an application of one of the inference rules of $\mathcal{BSE}$. The symbol $\leadsto^*$ denotes the reflexive and transitive closure of $\leadsto$.

Example 3.5 Consider the rigid equation $ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq gfy$. The ordering $\succ$ is the Knuth-Bendix ordering (see Martin [46]) in which all weights of symbols are equal to 1 and which uses the precedence relation $f > h > b > a$. Under this ordering we have $ht \succ a$ and $ft \succ hb$ for every ground term t. The following is a $\mathcal{BSE}$-derivation for this rigid equation:

$$\dfrac{\dfrac{\dfrac{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq gfy \cdot \emptyset}{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq ghb \cdot \{fy \succ hb, gfy \succ y, fy \simeq fy\}}\ (rrbs)}{\begin{array}{c} ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq ga \\ \cdot \{fy \succ hb, gfy \succ y, fy \simeq fy, hx \succ a, ghb \succ y, hx \simeq hb\}\end{array}}\ (rrbs)}{\vdash_\forall y \simeq y \cdot \{fy \succ hb, gfy \succ y, fy \simeq fy, hx \succ a, ghb \succ y, hx \simeq hb, y \simeq ga\}}\ (er)$$

By using *constraint simplification,* i.e. replacement of constraints by equivalent "more simple" constraints we can rewrite this derivation as

$$\dfrac{\dfrac{\dfrac{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq gfy \cdot \emptyset}{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq ghb \cdot \emptyset}\,(rrbs)}{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq ga \cdot \{ghb \succ y, x \simeq b\}}\,(rrbs)}{\vdash_\forall y \simeq y \cdot \{x \simeq b, y \simeq ga\}}\,(er)$$

Theorem 3.6 (Soundness of $\mathcal{BSE}$)
Let $R \cdot \emptyset \rightsquigarrow^* E \vdash_\forall t \simeq t \cdot \mathcal{C}$. Then any substitution satisfying $\mathcal{C}$ is a solution to R. In particular, R is solvable.

Proof. For any constraint $\mathcal{C}$, denote by $\mathcal{C}^{\simeq}$ the constraint obtained from $\mathcal{C}$ be removing all inequality constraints. First we note that for every application of an inference rule of $\mathcal{BSE}$ of the form $E_1 \vdash_\forall e_1 \cdot \mathcal{C}_1 \rightsquigarrow E_2 \vdash_\forall e_2 \cdot \mathcal{C}_2$ we have $E_1, \neg e_1, \mathcal{C}_2^{\simeq} \vdash E_2, \neg e_2$. By induction on the number of inference steps and using the fact $C_i \subseteq C_{i+1}$, we prove the same statement for multi-step derivations $E_1 \vdash_\forall e_1 \cdot \mathcal{C}_1 \rightsquigarrow^* E_2 \vdash_\forall e_2 \cdot \mathcal{C}_2$.

Let R have the form $E_0 \vdash_\forall r \simeq s$. Applying the obtained statement to multi-step derivations, we get $E_0, r \not\simeq s, \mathcal{C}^{\simeq} \vdash E, t \not\simeq t$. Hence, $E_0, \mathcal{C}^{\simeq} \vdash r \simeq s$. Let θ be any solution to $\mathcal{C}$. We have $E_0\theta, \mathcal{C}^{\simeq}\theta \vdash r\theta \simeq s\theta$. Any constraint in $\mathcal{C}^{\simeq}\theta$ has the form $u \simeq u$. Hence, $E_0\theta \vdash r\theta \simeq s\theta$, i.e. θ is a solution to $E_0 \vdash_\forall r \simeq s$. □

This theorem leads to the following definition:

Definition 3.7 (Answer constraint)
A constraint $\mathcal{C}$ is called an *answer constraint for a rigid equation* R iff for some rigid equation $E \vdash_\forall t \simeq t$ we have $R \cdot \emptyset \rightsquigarrow^* E \vdash_\forall t \simeq t \cdot \mathcal{C}$.

We note that $\mathcal{BSE}$ is an incomplete calculus for solving rigid equations. It means that there are solvable rigid equations R that have no answer constraint. For instance, consider the rigid equation[2] $x \simeq a \vdash_\forall gx \simeq x$. It has one solution $\{ga/x\}$. However, no rule of $\mathcal{BSE}$ is applicable to $x \simeq a \vdash_\forall gx \simeq x \cdot \emptyset$.

This means that $\mathcal{BSE}$ can yield less solutions to a rigid equation than any other known procedure, for example that of Gallier et.al. [34] because all these procedures are existentially complete. At the same time, $\mathcal{BSE}$ can yield more solutions than the procedure of [34] as the following example shows. For the rigid equation $a \simeq fa \vdash_\forall x \simeq fa$ the procedure of [34] will find one solution $\{a/x\}$, but there are two answer constraints whose solutions are the substitutions $\{a/x\}$ and $\{fa/x\}$ respectively.

In order to show that there is only finite number of derivations in $\mathcal{BSE}$ from a given constraint rigid equation, we prove an auxiliary statement.

Lemma 3.8 Let $t_0, t_1, \ldots$ be an infinite sequence of terms in a finite signature all whose variables belong to a finite set. Then there are numbers i, j such that $i < j$ and the constraint $t_i \succ t_j$ is unsatisfiable.

Proof. Following Kruskal [42] we introduce a partial ordering $\geq$ on terms as the smallest reflexive and transitive relation satisfying

[2]Suggested by G.Becher (private communication).

1. $f(s_1, \dots, s_n) \geq s_i$ for all $i \in \{1, \dots, n\}$;
2. if $s \geq t$ then $r[s] \geq r[t]$.

By Kruskal's Tree Theorem [42] there exist i, j such that $i < j$ and $t_j \geq t_i$. It is easy to see that the constraint $t_i \succ t_j$ is unsatisfiable. □

Similar statements have been proven by Dershowitz [16] and Plaisted [57].

Theorem 3.9 (Termination of $\mathcal{BSE}$)
For any constraint rigid equation $R \cdot \mathcal{C}$, there exists a finite number of derivations from $R \cdot \mathcal{C}$.

Proof. Suppose that there exists an infinite number of derivations from $R \cdot \mathcal{C}$. Then, by König's lemma there exists an infinite derivation $R \cdot \mathcal{C} = R_0 \cdot \mathcal{C}_0 \leadsto R_1 \cdot \mathcal{C}_1 \leadsto \dots$ consisting of superpositions. Consider any application of superposition $R_i \cdot \mathcal{C}_i \leadsto R_{i+1} \cdot \mathcal{C}_{i+1}$. Let it have the form

$$\frac{E \cup \{l \simeq r, s[p] \simeq t\} \vdash_\forall e \cdot \mathcal{C}_i}{E \cup \{l \simeq r, s[r] \simeq t\} \vdash_\forall e \cdot \mathcal{C}_{i+1}} \ (lrbs)$$

(the case of right rigid basic superposition is similar). We prove that for every $n \geq i + 1$ the constraint $\mathcal{C}_n$ is equivalent to $\mathcal{C}_n \cup \{s[p] \succ s[r]\}$. Indeed, the constraint $\mathcal{C}_{i+1}$ (and hence the constraint $\mathcal{C}_n$) contains $\{l \succ r, l \simeq p\}$. By the definition of reduction orderings, if $p \succ r$, then $s[p] \succ s[r]$. This implies that $\mathcal{C}_n$ is equivalent to $\mathcal{C}_n \cup \{s[p] \succ s[r]\}$.

Since every application of rigid basic superposition replaces a literal $s[p] \simeq t$ (or $s[p] \not\simeq t$) in R_i by a literal $s[r] \simeq t$ (respectively, $s[r] \not\simeq t$), there is an infinite sequence of terms $t_0, t_1, \dots$ and an increasing sequence of natural numbers $n_1, n_2, \dots$ with the following property. For every positive natural number k the constraint $\mathcal{C}_{n_k} \cup \{t_{k-1} \succ t_k\}$ is equivalent to $\mathcal{C}_{n_k}$. Since all terms t_k are in the same finite signature and have variables in the same finite set, by Lemma 3.8, there are i, j such that $i < j$ and the constraint $t_i \succ t_j$ is unsatisfiable. Since $\mathcal{C}_{n_k} \subseteq \mathcal{C}_{n_j}$ for all $k \leq j$, the constraint $\mathcal{C}_{n_j} \cup \{t_{k_1} \succ t_k\}$ is equivalent to $\mathcal{C}_{n_j}$, for all $k \leq j$. Hence, the constraint $\mathcal{C} = \mathcal{C}_{n_j} \cup \{t_i \succ t_{i+1}, \dots, t_{j-1} \succ t_j\}$ is equivalent to $\mathcal{C}_{n_j}$. Thus, $\mathcal{C}$ is satisfiable. But satisfiability of $\mathcal{C}$ implies satisfiability of $t_i \succ t_j$. Contradiction. □

Note. In Degtyarev and Voronkov [25] the left rigid basic superposition has been formulated incorrectly in the following way:

$$\frac{E \cup \{l \simeq r, s[p] \simeq t\} \vdash_\forall e \cdot \mathcal{C}}{E \cup \{l \simeq r, s[p] \simeq t, s[r] \simeq t\} \vdash_\forall e \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l \simeq p\}} \ (lrbs)$$

With this formulation, termination is not guaranteed as the following example shows. Consider the rigid equation $fgx \simeq gx, gx \simeq a \vdash_\forall a \simeq b$ and arbitrary reduction ordering $\succ$ total on ground terms. We have the following is an infinite sequence of applications of $(lrbs)$:

$$\cfrac{\cfrac{fgx \simeq gx, gx \simeq a \vdash_\forall a \simeq b \cdot \emptyset}{fgx \simeq gx, gx \simeq a, gx \simeq fa \vdash_\forall a \simeq b \cdot \{gx \simeq gx, fgx \succ gx, gx \succ a\}} \ (lrbs)}{fgx \simeq gx, gx \simeq a, gx \simeq fa, gx \simeq ffa \vdash_\forall a \simeq b \cdot \{gx \simeq gx, fgx \succ gx, gx \succ a, gx \succ fa\}} \ (lrbs)$$

$$\vdots$$

$$\begin{array}{l} fgx \simeq gx, gx \simeq a, gx \simeq fa, gx \simeq ffa, \dots, gx \simeq f^n a \\ \quad \vdash_\forall a \simeq b \cdot \{gx \simeq gx, fgx \succ gx, gx \succ a, gx \succ fa, \dots, gx \succ f^{n-1} a\} \end{array}$$

$$\vdots$$

It is easy to see that the constraint $\{gx \simeq gx, fgx \succ gx, gx \succ a, gx \succ fa, \ldots, gx \succ f^{n-1}a\}$ is satisfied by the substitution $\{f^{n-1}a/x\}$.

Inequality constraints are not needed for soundness or completeness of our method. The pragmatics behind inequality constraints is to ensure that the search for solutions of a rigid equation is finite. In addition, the use of ordering constraints prunes the search space.

To illustrate this theorem, we consider Example 3.5. The rigid equation of this example has an infinite number of solutions including $\{b/x, gh^n a/y\}$, for every natural number n. However, all possible $\mathcal{BSE}$-derivations starting with $ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq gfy \cdot \emptyset$ give only two answer constraints, one is

$$\{fy \succ hb, gfy \succ y, fy \simeq fy, hx \succ a, ghb \succ y, hx \simeq hb, y \simeq ga\}$$

shown in Example 3.5, another is $\{fy \succ hb, gfy \succ y, fy \simeq fy, y \simeq ghb\}$ obtained from the following derivation:

$$\cfrac{\cfrac{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq gfy \cdot \emptyset}{ha \simeq a, hx \simeq a, hb \simeq fy \vdash_\forall y \simeq ghb \cdot \{fy \succ hb, gfy \succ y, fy \simeq fy\}}\;(rrbs)}{\vdash_\forall y \simeq y \cdot \{fy \succ hb, gfy \succ y, fy \simeq fy, y \simeq ghb\}}\;(er)$$

This answer constraint can be simplified to $\{y \simeq ghb\}$.

Theorem 3.9 yields

Theorem 3.10 Any rigid equation has a finite number of answer constraints. There is an algorithm giving by any rigid equation R the set of all answer constraints for R.

Section 4

Answer constraints and the tableau method

In this section we consider how to use the system $\mathcal{BSE}$ for theorem proving by the tableau method. Since we only consider skolemized formulas, we have no δ-rules in tableau calculi.

Definition 4.1 (Branch and tableau)
A *branch* is a finite multiset of formulas. A *tableau* is a finite multiset of branches. A tableau with branches $\Gamma_1, \ldots, \Gamma_n$ will be denoted by $\Gamma_1 \mid \ldots \mid \Gamma_n$. The tableau with $n = 0$ is called *the empty tableau* and denoted by $\#$.

Often, tableaux are presented in the tree form. Representation of tableaux as multisets of branches is more convenient for us for several reasons. For this representation we introduce a logical system allowing to *expand* tableaux:

Definition 4.2 (Tableau expansion rules)
The rules (α), (β) and (γ) of Figure 4.1 are called *tableau expansion rules*.

Definition 4.3 Let Γ be a branch of a tableau. The set of *rigid equations on* Γ is defined in the following way. A rigid equation $E \vdash_\forall s \simeq t$ is on Γ iff E is the multiset of all equations in Γ and $(s \not\simeq t) \in \Gamma$.

We extend the notion of answer constraints to tableau branches:

$$\frac{\Gamma_1, \varphi \wedge \psi \mid \ldots \mid \Gamma_n}{\Gamma_1, \varphi \wedge \psi, \varphi, \psi \mid \ldots \mid \Gamma_n}\ (\alpha) \qquad \frac{\Gamma_1, \varphi \vee \psi \mid \ldots \mid \Gamma_n}{\Gamma_1, \varphi \mid \Gamma_1, \psi \mid \ldots \mid \Gamma_n}\ (\beta)$$

$$\frac{\Gamma_1, \forall x \varphi(x) \mid \ldots \mid \Gamma_n}{\Gamma_1, \forall x \varphi(x), \varphi(y) \mid \ldots \mid \Gamma_n}\ (\gamma)$$

In the rules (γ) the variable y does not occur in the premise.

Figure 4.1: Tableau Expansion Rules

Definition 4.4 A constraint $\mathcal{C}$ is an *answer constraint for a tableau branch* Γ iff $\mathcal{C}$ is an answer constraint for some rigid equation on Γ.

By Theorem 3.10, we obtain

Theorem 4.5 Any tableau branch has a finite number of answer constraints. There is an algorithm giving by any tableau branch Γ the set of all answer constraints for Γ.

The following theorem states soundness and completeness of the tableau method with answer constraints:

Theorem 4.6 (Soundness and completeness)
Let ξ be a sentence in Skolem negation normal form. Then ξ is unsatisfiable iff there is a tableau T obtained from ξ by tableau expansion rules with the following property. Let $\Gamma_1, \ldots, \Gamma_n$ be all branches of T. Then there exist answer constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ for $\Gamma_1, \ldots, \Gamma_n$, respectively, such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$ is satisfiable.

Proof. Soundness follows from soundness of $\mathcal{BSE}$.

The proof of completeness is quite lengthy and is given in Appendix A. It is based on the completeness of the equality elimination method [19, 21, 26].

□

To illustrate this theorem, consider the formula of Example 1.1. Assume that we want to prove the formula $\xi = \exists xyzu((a \simeq b \supset g(x,u,v) \simeq g(y,fc,fd)) \wedge (c \simeq d \supset g(u,x,y) \simeq g(v,fa,fb)))$. The negation normal form of $\neg\xi$ is $\forall xyzu((a \simeq b \wedge g(x,u,v) \not\simeq g(y,fc,fd)) \vee (c \simeq d \wedge g(u,x,y) \not\simeq g(v,fa,fb)))$

The ordering $\succ$ is the lexicographic path ordering (see e.g. [54]) based on the precedence $g > f > a > b > c > d$. For purely illustrative purpose, we shall display tableaux in the tree form. After one quantifier duplication (application of a γ-rule) and some other tableau expansion rules we obtain the following tableau:

$$\begin{tikzcd}
 & {} \arrow[dl, no head] \arrow[dr, no head] & \\
a \simeq b \arrow[d, no head] & & c \simeq d \arrow[d, no head] \\
g(x,u,v) \not\simeq g(y,fc,fd) & & g(u,x,y) \not\simeq g(v,fa,fb)
\end{tikzcd}$$

There is one rigid equation on each branch of the tableau:

$$a \simeq b \vdash_{\forall} g(x,u,v) \simeq g(y,fc,fd) \tag{4.1}$$
$$c \simeq d \vdash_{\forall} g(u,x,y) \simeq g(v,fa,fb) \tag{4.2}$$

Rigid basic superposition is applicable to none of this rigid equations. Rigid equation (4.1) has one answer constraint $\{g(x,u,v) \simeq g(y,fc,fd)\}$ obtained by an application of the equality resolution rule:

$$\frac{a \simeq b \vdash_\forall g(x,u,v) \simeq g(y,fc,fd) \cdot \emptyset}{\vdash_\forall g(x,u,v) \simeq g(x,u,v) \cdot \{g(x,u,v) \simeq g(y,fc,fd)\}}\ (er)$$

Similarly, rigid equation (4.2) has one answer constraint $\{g(u,x,y) \simeq g(v,fa,fb)\}$. The union of these constraints $\{g(x,u,v) \simeq g(y,fc,fd), g(u,x,y) \simeq g(v,fa,fb)\}$ is unsatisfiable. Thus, our method does not find solution after one quantifier duplication. After three quantifier duplications and some other tableau expansion steps we obtain the following tableau:

$$\begin{tikzcd}
 & & & & {} \arrow[dll, no head] \arrow[drr, no head] & & & \\
 & & a \simeq b \arrow[d, no head] & & & & c \simeq d \arrow[d, no head] & \\
 & & g(x_1,u_1,v_1) \not\simeq g(y_1,fc,fd) \arrow[dl, no head] \arrow[dr, no head] & & & & g(u_1,x_1,y_1) \not\simeq g(v_1,fa,fb) \arrow[dl, no head] \arrow[dr, no head] & \\
 & a \simeq b \arrow[d, no head] & & c \simeq d \arrow[dd, no head] & & a \simeq b \arrow[dd, no head] & & c \simeq d \arrow[d, no head] \\
 & g(x_2,u_2,v_2) \not\simeq g(y_2,fc,fd) & & & & & & g(u_3,x_3,y_3) \not\simeq g(v_3,fa,fb) \\
 & & & g(u_2,x_2,y_2) \not\simeq g(v_2,fa,fb) & & g(x_3,u_3,v_3) \not\simeq g(y_3,fc,fd) & &
\end{tikzcd}$$

It has four branches:

$$\begin{array}{ll}
\Gamma_1: & \{a \simeq b, a \simeq b, g(x_1,u_1,v_1) \not\simeq g(y_1,fc,fd), g(x_2,u_2,v_2) \not\simeq g(y_2,fc,fd)\} \\
\Gamma_2: & \{a \simeq b, c \simeq d, g(x_1,u_1,v_1) \not\simeq g(y_1,fc,fd), g(u_2,x_2,y_2) \not\simeq g(v_2,fa,fb)\} \\
\Gamma_3: & \{a \simeq b, c \simeq d, g(u_1,x_1,y_1) \not\simeq g(v_1,fa,fb), g(x_3,u_3,v_3) \not\simeq g(y_3,fc,fd)\} \\
\Gamma_4: & \{c \simeq d, c \simeq d, g(u_1,x_1,y_1) \not\simeq g(v_1,fa,fb), g(u_3,x_3,y_3) \not\simeq g(v_3,fa,fb)\}
\end{array}$$

Consider the following rigid equations R_1–R_4 on the branches Γ_1–Γ_4, respectively:

$$\begin{array}{ll}
R_1: & a \simeq b, a \simeq b \vdash_\forall g(x_2,u_2,v_2) \simeq g(y_2,fc,fd) \\
R_2: & a \simeq b, c \simeq d \vdash_\forall g(x_1,u_1,v_1) \simeq g(y_1,fc,fd) \\
R_3: & a \simeq b, c \simeq d \vdash_\forall g(u_1,x_1,y_1) \simeq g(v_1,fa,fb) \\
R_4: & c \simeq d, c \simeq d \vdash_\forall g(u_3,x_3,y_3) \simeq g(v_3,fa,fb)
\end{array}$$

We can apply the following $\mathcal{BSE}$-derivations to R_1–R_4:

$$\frac{a \simeq b, a \simeq b \vdash_\forall g(x_2,u_2,v_2) \simeq g(y_2,fc,fd) \cdot \emptyset}{\vdash_\forall g(x_2,u_2,v_2) \simeq g(x_2,u_2,v_2) \cdot \{g(x_2,u_2,v_2) \simeq g(y_2,fc,fd)\}}\ (er)$$

$$\cfrac{\cfrac{a \simeq b, c \simeq d \vdash_\forall g(x_1,u_1,v_1) \simeq g(y_1,fc,fd) \cdot \emptyset}{\begin{array}{l} a \simeq b, c \simeq d \vdash_\forall g(x_1,u_1,v_1) \simeq g(y_1,fd,fd) \\ \quad \cdot\{c \succ d, g(y_1,fc,fd) \succ g(x_1,u_1,v_1), c \simeq c\} \end{array}}\ (rrbs)}{\begin{array}{l} \vdash_\forall g(x_1,u_1,v_1) \simeq g(x_1,u_1,v_1) \\ \quad \cdot\{c \succ d, g(y_1,fc,fd) \succ g(x_1,u_1,v_1), c \simeq c, g(x_1,u_1,v_1) \simeq g(y_1,fd,fd)\} \end{array}}\ (er)$$

$$\dfrac{\dfrac{a \simeq b, c \simeq d \vdash_\forall g(u_1, x_1, y_1) \simeq g(v_1, fa, fb) \cdot \emptyset}{\begin{array}{l} a \simeq b, c \simeq d \vdash_\forall g(u_1, x_1, y_1) \simeq g(v_1, fb, fb) \\ \quad \cdot \{a \succ b, g(v_1, fa, fb) \succ g(u_1, x_1, y_1), a \simeq a\} \end{array}}\ (rrbs)}{\begin{array}{l} \vdash_\forall g(u_1, x_1, y_1) \simeq g(u_1, x_1, y_1) \\ \quad \cdot \{a \succ b, g(v_1, fa, fb) \succ g(u_1, x_1, y_1), a \simeq a, g(u_1, x_1, y_1) \simeq g(v_1, fb, fb)\} \end{array}}\ (er)$$

$$\dfrac{c \simeq d, c \simeq d \vdash_\forall g(u_3, x_3, y_3) \simeq g(v_3, fa, fb) \cdot \emptyset}{\vdash_\forall g(u_3, x_3, y_3) \simeq g(u_3, x_3, y_3) \cdot \{g(u_3, x_3, y_3) \simeq g(v_3, fa, fb)\}}\ (er)$$

The union of the answer constraints of these derivations is

$$\begin{array}{l} \{g(x_2, u_2, v_2) \simeq g(y_2, fc, fd), \\ \ c \succ d, g(y_1, fc, fd) \succ g(x_1, u_1, v_1), c \simeq c, g(x_1, u_1, v_1) \simeq g(y_1, fd, fd), \\ \ a \succ b, a \simeq a, g(v_1, fa, fb) \succ g(u_1, x_1, y_1), g(u_1, x_1, y_1) \simeq g(v_1, fb, fb), \\ \ g(u_3, x_3, y_3) \simeq g(v_3, fa, fb)\ \} \end{array}$$

This constraint is satisfiable. To check this, we can consider the following substitution:

$$\{fb/x_1, fb/y_1, fd/u_1, fd/v_1, b/x_2, b/y_2, fc/u_2, fd/v_2, d/u_3, d/v_3, fa/x_3, fb/y_3\}.$$

Section 5

Tableau basic superposition

As a simple consequence of our results, we prove a completeness result for a paramodulation rule working on tableaux. A paramodulation rule working directly on tableaux was proposed by Loveland [45] in the context of model elimination and later by Fitting [32]. However, their formulations have all disadvantages of the early paramodulation rule of Robinson and Wos [60]:

1. Functional reflexivity rule is used;
2. Paramodulation into variables is allowed;
3. Increasing applications of paramodulation are allowed (for example, x can be rewritten to $f(x)$.

As a consequence, for a given tableau expansion there may be an infinite sequence of paramodulations, in particular due to the use of functional reflexivity or increasing applications of paramodulation. Since the publication of Loveland's book [45], no improvements of the paramodulation-based tableau calculi have been described except for Plaisted [58] who has shown how to transform derivations with resolution and paramodulation to tableaux by introducing a tableau factoring rule.

Here we show that paramodulation is complete under the following restrictions:

1. No functional reflexivity is needed;
2. Paramodulation into variables is not allowed;
3. Orderings are used so that there are no increasing applications of paramodulation;
4. Basic restriction on paramodulation that allows us to prohibit paramodulation into non-variables terms introduced by unification.

All these refinements are a consequence of our main result (Theorem 4.6).

In order to formalize the basic strategy, we keep the substitution condition as a set of constraints, as before. Thus, we work with *constraint tableaux:*

Definition 5.1 (Constraint tableau)
A *constraint tableau* is a pair consisting of a tableau T and a constraint $\mathcal{C}$, denoted $T \cdot \mathcal{C}$.

Now we adapt the tableau rules of [32] to the case of constraint tableaux. For simplicity, we only consider signatures whose only predicate symbol is $\simeq$. When we prove a formula φ, we construct the Skolem negation normal form ψ of $\neg\varphi$ and, starting with the constraint tableau $\psi \cdot \emptyset$ try to derive the empty tableau # with some satisfiable constraint.

$$\frac{\Gamma_1, \varphi \wedge \psi \mid \ldots \mid \Gamma_n \cdot C}{\Gamma_1, \varphi \wedge \psi, \varphi, \psi \mid \ldots \mid \Gamma_n \cdot C}\ (\alpha) \qquad \frac{\Gamma_1, \varphi \vee \psi \mid \ldots \mid \Gamma_n \cdot C}{\Gamma_1, \varphi \mid \Gamma_1, \psi \mid \ldots \mid \Gamma_n \cdot C}\ (\beta)$$

$$\frac{\Gamma_1, \forall x \varphi(x) \mid \ldots \mid \Gamma_n \cdot C}{\Gamma_1, \forall x \varphi, \varphi(y) \mid \ldots \mid \Gamma_n \cdot C}\ (\gamma) \qquad \frac{\Gamma_1, s \not\simeq t \mid \Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C}}{\Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C} \cup \{s \simeq t\}}\ (er)$$

$$\frac{\Gamma_1, l \simeq r, s[p] \simeq t \mid \Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C}}{\Gamma_1, l \simeq r, s[r] \simeq t \mid \Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l \simeq p\}}\ (lrbs)$$

$$\frac{\Gamma_1, l \simeq r, s[p] \not\simeq t \mid \Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C}}{\Gamma_1, l \simeq r, s[r] \not\simeq t \mid \Gamma_2 \mid \ldots \mid \Gamma_n \cdot \mathcal{C} \cup \{l \succ r, s[p] \succ t, l \simeq p\}}\ (rrbs)$$

In the rules (γ) the variable y does not occur in the premise. The conditions on the rules $(lrbs)$ and $(rrbs)$ are the same as for the corresponding rules of $\mathcal{BSE}$.

Figure 5.1: Calculus $\mathcal{TBSE}$

Definition 5.2 (Calculus $\mathcal{TBSE}$)
The *free-variable tableau calculus* $\mathcal{TBSE}$ is shown in Figure 5.1.

Definition 5.3 (Constraint tableau expansion rules)
The rules (α), (β) and (γ) of $\mathcal{TBSE}$ are called *constraint tableau expansion rules.*

The calculus $\mathcal{TBSE}$ has the required completeness property:

Theorem 5.4 (Soundness and completeness)
Let φ be a formula in the Skolem negation normal form. It is unsatisfiable iff there is a derivation from the constraint tableau $\varphi \cdot \emptyset$ of a constraint tableau $\# \cdot \mathcal{C}$.

Proof. Straightforward from Theorem 4.6 by noting that the rules of $\mathcal{BSE}$ can be simulated by the corresponding tableau rules. □

This logical system has one more pleasant property:

Theorem 5.5 (Termination)
For any constraint tableau $T \cdot \mathcal{C}$ there is only a finite number of derivations from $T \cdot \mathcal{C}$ not using constraint tableau expansion rules.

Proof. Similar to that of Theorem 3.9. □

This means, that for a given amplification, we cannot have infinite search. Infinite search without any expansion steps is possible in the Fitting's system.

To illustrate the connection between the tableau rigid basic superposition rule and rules of $\mathcal{BSE}$, we reconsider the example of Section 4. On the branch containing the literal $g(x_1, u_1, v_1) \not\simeq g(y_1, fc, fd)$ and the equation $c \simeq d$, we can apply rigid basic superposition that adds $g(x_1, u_1, v_1) \not\simeq g(y_1, fd, fd)$ to the branch. Similarly, we can apply rigid basic superposition to the branch containing $g(u_1, x_1, y_1) \not\simeq g(v_1, fa, fb)$ and $a \simeq b$, obtaining $g(u_1, x_1, y_1) \not\simeq g(v_1, fb, fb)$. This results in the following tableau (the picture below does not include the constraint, it is discussed below).

$$\begin{tikzcd}[column sep=tiny]
 & & & & {} \arrow[dll, dash] \arrow[drr, dash] & & & & \\
 & & a \simeq b \arrow[dl, dash] & & & & c \simeq d \arrow[dr, dash] & & \\
 & g(x_1,u_1,v_1) \not\simeq g(y_1,fc,fd) \arrow[dl, dash] \arrow[dr, dash] & & & & & & g(u_1,x_1,y_1) \not\simeq g(v_1,fa,fb) \arrow[dl, dash] \arrow[dr, dash] & \\
a \simeq b \arrow[d, dash] & & c \simeq d \arrow[dd, dash] & & & & a \simeq b \arrow[dd, dash] & & c \simeq d \arrow[d, dash] \\
g(x_2,u_2,v_2) \not\simeq g(y_2,fc,fd) & & & & & & & & g(u_3,x_3,y_3) \not\simeq g(v_3,fa,fb) \\
 & & \begin{array}{c} g(u_2,x_2,y_2) \not\simeq g(v_2,fa,fb) \\ g(x_1,u_1,v_1) \not\simeq g(y_1,fd,fd) \end{array} & & & & \begin{array}{c} g(x_3,u_3,v_3) \not\simeq g(y_3,fc,fd) \\ g(u_1,x_1,y_1) \not\simeq g(v_1,fb,fb) \end{array} & &
\end{tikzcd}$$

After four application of the (er) rules all branches of this tableau become closed. The resulting constraint of this derivation is the same as the union of the answer constraints shown at the end of Section 4.

Section 6

Related work

The problem of extending tableaux with equality rules is crucial for enriching the deductive capabilities of the tableau method. Despite the fact that this problem is attacked by a growing number of researchers during the last years, known solutions are not yet convincing. At the same time tableau methods of automated deduction play an important role in various areas of artificial intelligence and computer science — see e.g. special issues of the Journal of Automated Reasoning, v. 13, no. 2,3, 1994. These issues contain a survey by Schumann [61] of implementations of tableau-based provers. Among 24 systems mentioned in the survey only two are able to handle equality.

The system PROTEIN [7] (and also KoMeT [13]) implement the modification method of Brand [14]. This method transforms a set of clauses with equality into a set of clauses without equality. This transformation usually yields a considerably larger set of clauses. In particular, the symmetry and the transitivity axioms must be explicitly applied to all positive occurrences of the equality predicate. Recently, we proposed a new translation method based on the so-called *basic folding* demonstrated for Horn clauses in [22].

According to Schumann [61], the system ${}_3T^AP$ uses the method of Beckert and Hähnle [11]. Paper [11] claims the completeness of the method, but this claim is not true. The method expands the tableau using the standard tableau rules, including γ-rules. For finding a closing substitution, an analog of linear paramodulation without function reflexivity has been proposed. As it is well known, linear paramodulation is incomplete without function reflexivity. The same is true for the method of Beckert and Hähnle [11], as the following example shows. Suppose that we prove the formula $\exists x(a \simeq b \wedge g(fa, fb) \simeq h(fa, fb) \supset g(x,x) \simeq h(x,x))$. In order to prove it using paramodulation, we need to paramodulate $a \simeq b$ into $g(fa, fb) \simeq h(fa, fb)$. The method of Beckert and Hähnle [11] only allows for paramodulation into copies of $g(x,x) \simeq h(x,x)$ obtained by the application of γ-rules. Thus, this (provable) formula cannot be proved using the method of Beckert and Hähnle [11].

Consider now approaches based on the simultaneous rigid E-unifiability by Gallier et.al. [36, 34] and related methods. We do not consider numerous works dedicated to the non-simultaneous rigid E-unifiability. This problem is NP-complete and there exist a number of complete algorithms for its solution (Gallier et.al. [35, 33], Goubault [38], Becher and Petermann [8], De Kogel [29] and Plaisted [58]). Since simultaneous rigid E-unification is undecidable (Degtyarev and Voronkov [23, 27]), their completeness is useless from the viewpoint of general purpose theorem proving as proposed by Gallier et.al. [36, 34]. Our system $\mathcal{BSE}$ can easily be extended to a calculus complete for rigid E-unifiability, but such completeness was not our aim. We tried to restrict the number of possible $\mathcal{BSE}$-derivations preserving completeness of the general-purpose method of Section 4.

It is not known whether the procedure described in Gallier et.al. [34] is complete for theorem proving[1]. Even if it is complete, our procedure based on $\mathcal{BSE}$ has some advantages over Gallier et.al.'s procedure. For example, for every tableau branch with p equations and q disequations, we consider q rigid equations, while Gallier et.al.'s procedure checks $q \cdot 2^p$ rigid equations.

Gallier et.al. [35, 33] introduced the notion of a complete set of solutions for rigid E-unification, proved finiteness of such sets and gave an algorithm computing finite complete set of solutions. According to this result, Goubault [39] proposed to solve simultaneous rigid E-unifiability by using finite complete sets of solutions to the components of the simultaneous problem. Paper [39] contained faulty results. The undecidability of simultaneous rigid E-unification shows that finite complete sets of solutions do not give a solution to the simultaneous problem. The reason for this is that substitutions belonging to complete sets of solutions for different rigid equations are minimal modulo *different* congruences.

Petermann [56] introduces a "complete connection calculus with rigid E-unification". Here the completeness is achieved by changing the notion of a complete set of unifiers so that solutions to all subproblems are compared modulo the same congruence (generated by the empty equality theory). In this case, a non-simultaneous problem can have an infinite number of solutions and no finite complete set of solutions. For example, for the rigid E-unification problem $f(a) \simeq a \vdash_{\forall} x \simeq a$ the complete set of solutions in the sense of Gallier et.al. [34] consists of one substitution $\{a/x\}$ (and there is only one answer constraint $\{x \simeq a\}$ obtained by our method), but the complete set of solutions in the sense of Petermann [56] is infinite and consists of substitutions $\{f^n(a)/x\}$, for all $n \in \{0, 1, \ldots\}$. This implies that the proof-search by the method of Petermann [56] can be nonterminating even for a limited number of applications of γ-rule (i.e. for a particular tableau), unlike algorithms based on the finite complete sets of unifiers in the sense of Gallier et.al. [34] or based on minus-normalization (Kanger [41]). The implementation of the method of [56] uses a completion-based procedure by Beckert [9] of generation of complete sets of rigid E-unifiers. This procedure is developed with the aim of solving a more general problem — so-called *mixed E-unification* and has been implemented as part of the tableau-based theorem prover ${}_3T^AP$. Complete sets of unifiers both in the sense of Gallier et.al. [34] and in the sense of Petermann [56] can be computed by this procedure in the case when all variables are treated as rigid. However, the termination is not guaranteed even for complete sets of rigid E-unifiers in the sense of Gallier et.al. [34].

Plaisted [58] gives "techniques for incorporating equality into theorem proving; these techniques have a rigid flavor". His method called *path paramodulation* guarantees termination for a given amplification and, in the case of success "solves the simultaneous rigid E-unification problem", in a sense. However, this does not solve the problem attacked by a number of researchers: extend the method of matings to languages with equality by rigid E-unification. First, unlike [34] the search for solutions for a given amplification is not incremental (the method does not allow "branch-wise" computation of solutions to rigid E-unification for separate branches). Second, within a given amplification Plaisted uses factoring rules which involves two branches (paths). As a consequence, even when the original formula contains no equality, his method results in the standard tableau calculus *plus* the factoring rule.

In fact, path paramodulation of Plaisted [58] simulates resolution-paramodulation inference in a connection-like calculus. Although it is not noted in [58], but this technique has been demonstrated for resolution in many papers, for example by Bibel [12], Eder [30, 31], Mints [50], Baumgartner and Furbach [6] and Avron [2]. The generalization of this simulation to paramodulation is straightforward.

However, this simulation technique is insufficient for proving results of our paper since, in

[1]For example, the completeness of Gallier et.al.'s procedure does not follow from our method because, as noted above, our calculus $\mathcal{BSE}$ can give more solutions to some rigid equations.

particular, it gives no insight on how to avoid factoring in tableaux with equality. The use of factoring prevents not only from the independent search for solutions for tableau branches, but even from the incremental solving of rigid equations on tableau branches as proposed by Gallier et.al.

Our *equality elimination method* [26, 20, 21] is based on extending a tableau prover by a bottom-up equation solver using basic superposition. Solutions to equations are generated by this solver and used to close branches of a tableau. Thus, the method combines (non-local) tableau proof search with the (local) equation solving. Only completely solved equations are used in the tableau part of the proof, thus reducing non-determinism created by applications of MGU replacement rule of Fitting [32]. The equation solution is even more restricted by the use of orderings, basic simplification and subsumption.

A similar idea: combination of proof-search in tableaux with a bottom-up equality saturation of the original formula, is used in [51] for constructing a goal-directed version of model elimination and paramodulation.

One of advantages of the tableau method is its applicability to non-classical logics. However, handling equality in non-classical logics seems to be much more difficult problem than that in classical logic. For example, it is shown by Voronkov [64] that procedures for intuitionistic logic with equality must handle simultaneous rigid E-unification. This implies that our method based on $\mathcal{BSE}$ *does not* give a complete procedure for intuitionistic logic with equality. Other results on relations between simultaneous rigid E-unification and intuitionistic logic are considered by Degtyarev and Voronkov [24], Degtyarev, Matiyasevich and Voronkov [18].

Bibliography

[1] P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.

[2] A. Avron. Gentzen-type systems, resolution and tableaux. *Journal of Automated Reasoning*, 10:256–281, 1993.

[3] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 462–476, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

[4] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basis paramodulation. *Information and Computation*, 121:172–192, 1995.

[5] Peter Baumgartner. An ordered theory resolution calculus. In A. Voronkov, editor, *Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 119–130, 1992.

[6] P. Baumgartner and U. Furbach. Consolution as a framework for comparing calculi. *Journal of Symbolic Computations*, 16:445–477, 1993.

[7] P. Baumgartner and U. Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *IN*terface. In A. Bundy, editor, *Automated Deduction — CADE-12. 12th International Conference on Automated Deduction.*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 769–773, Nancy, France, June/July 1994.

[8] G. Becher and U. Petermann. Rigid unification by completion and rigid paramodulation. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence. 18th German Annual Conference on Artificial Intelligence*, volume 861 of *Lecture Notes in Artificial Intelligence*, pages 319–330, Saarbrücken, Germany, September 1994. Springer Verlag.

[9] B. Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, *Automated Deduction — CADE-12. 12th International Conference on Automated Deduction.*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Nancy, France, June/July 1994.

[10] B. Beckert. Are minimal solutions to simultaneous rigid E-unification sufficient for adding equality to semantic tableauxΓ Privately circulated manuscript, University of Karlsruhe, 1995.

[11] B. Beckert and R. Hähnle. An improved method for adding equality to free variable semantic tableaux. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

[12] W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, 1981.

[13] W. Bibel. Issues in theorem proving based on the connection method. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 1–16, Schloß Rheinfels, St. Goar, Germany, May 1995.

[14] D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4:412–430, 1975.

[15] C.L. Chang. Theorem proving with variable-constrained resolution. *Information Sciences*, 4:217–231, 1972.

[16] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[17] A. Degtyarev, Yu. Gurevich, and A. Voronkov. Herbrand's theorem and equational reasoning: Problems and solutions. UPMAIL Technical Report 128, Uppsala University, Computing Science Department, September 1996.

[18] A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification and related algorithmic problems. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 494–502, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[19] A. Degtyarev and A. Voronkov. Equality elimination for semantic tableaux. UPMAIL Technical Report 90, Uppsala University, Computing Science Department, December 1994.

[20] A. Degtyarev and A. Voronkov. General connections via equality elimination. In M. De Glas and Z. Pawlak, editors, *Second World Conference on the Fundamentals of Artificial Intelligence (WOCFAI-95)*, pages 109–120, Paris, July 1995. Angkor.

[21] A. Degtyarev and A. Voronkov. Equality elimination for the inverse method and extension procedures. In C.S. Mellish, editor, *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 342–347, Montréal, August 1995.

[22] A. Degtyarev and A. Voronkov. Handling equality in logic programs via basic folding. In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Extensions of Logic Programming (5th International Workshop, ELP'96)*, volume 1050 of *Lecture Notes in Computer Science*, pages 119–136, Leipzig, Germany, March 1996.

[23] A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. In H. Kleine Büning, editor, *Computer Science Logic. 9th International Workshop, CSL'95*, volume 1092 of *Lecture Notes in Computer Science*, pages 178–190, Paderborn, Germany, September 1995, 1996.

[24] A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 503–512, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

[25] A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. In J.J. Alferes, L.M. Pereira, and E. Orlowska, editors, *Logics in Artificial Intelligence. European Workshop, JELIA'96*, volume 1126 of *Lecture Notes in Artificial Intelligence*, pages 50–69, Évora, Portugal, September/October 1996.

[26] A. Degtyarev and A. Voronkov. Equality elimination for the tableau method. In J. Calmet and C. Limongelli, editors, *Design and Implementation of Symbolic Computation Systems. International Symposium, DISCO'96*, volume 1128 of *Lecture Notes in Computer Science*, pages 46–60, Karlsruhe, Germany, September 1996.

[27] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

[28] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi: a tutorial. Upmail technical report, Uppsala University, Computing Science Department, August 1996. To appear.

[29] E. De Kogel. Rigid E-unification simplified. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 17–30, Schloß Rheinfels, St. Goar, Germany, May 1995.

[30] E. Eder. A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In E. Börger, G. Jäger, H. Kleine Büning, and M.M. Richter, editors, *CSL'88 (Proc. 2nd Workshop on Computer Science Logic)*, volume 385 of *Lecture Notes in Computer Science*, pages 80–98. Springer Verlag, 1988.

[31] E. Eder. Consolution and its relation with resolution. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 132–136, 1991.

[32] M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer Verlag, New York, 1990.

[33] J. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification: NP-completeness and applications to equational matings. *Information and Computation*, 87(1/2):129–195, 1990.

[34] J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the Association for Computing Machinery*, 39(2):377–429, 1992.

[35] J.H. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, July 1988.

[36] J.H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid E-unification: Equational matings. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, 1987.

[37] J.H. Gallier, S. Raatz, and W. Snyder. Rigid E-unification and its applications to equational matings. In H. Aït Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 1, pages 151–216. Academic Press, 1989.

[38] J. Goubault. A rule-based algorithm for rigid E-unification. In Georg Gottlob, Alexander Leitsch, and Daniele Mundici, editors, *Computational Logic and Proof Theory. Proceedings of the Third Kurt Gödel Colloquium, KGC'93*, volume 713 of *Lecture Notes in Computer Science*, pages 202–210, Brno, August 1993.

[39] J. Goubault. Rigid $\bar{E}$-unifiability is DEXPTIME-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1994.

[40] R. Hähnle, B. Beckert, and S. Gerberding. The many-valued tableau-based theorem prover ${}_3T^AP$. Technical Report 30/94, Universität Karlsruhe, Fakultät für Informatik, November 1994.

[41] S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 364–371. Springer Verlag, 1983. Originally appeared in 1963.

[42] J. Kruskal. Well quasi ordering, the tree problem and Vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95:210–225, 1960.

[43] R.C.T. Lee and C.L. Chang. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[44] D.W. Loveland. Mechanical theorem proving by model elimination. *Journal of the Association for Computing Machinery*, 15:236–251, 1968.

[45] D.W. Loveland. *Automated Theorem Proving: a Logical Basis.* North Holland, 1978.

[46] U. Martin. How to choose weights in the Knuth-Bendix ordering. In *Rewriting Technics and Applications*, volume 256 of *Lecture Notes in Computer Science*, pages 42–53, 1987.

[47] S.Yu. Maslov. The inverse method of establishing deducibility in the classical predicate calculus. *Soviet Mathematical Doklady*, 5:1420–1424, 1964.

[48] S.Yu. Maslov. An invertible sequential variant of constructive predicate calculus (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 4, 1967. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 4, Consultants Bureau, NY-London, 1969, p.36–42.

[49] V.A. Matulis. On variants of classical predicate calculus with the unique deduction tree (in Russian). *Soviet Mathematical Doklady*, 148:768–770, 1963.

[50] G. Mints. Gentzen-type systems and resolution rules. part I. propositional logic. In P. Martin-Löf and G. Mints, editors, *COLOG-88*, volume 417 of *Lecture Notes in Computer Science*, pages 198–231. Springer Verlag, 1990.

[51] M. Moser, C. Lynch, and J. Steinbach. Model elimination with basic ordered paramodulation. Technical Report AR-95-11, Fakultät für Informatik, Technische Universität München, München, 1995.

[52] R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In *ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer Verlag, 1992.

[53] R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.

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

[55] S.A. Norgela. On the size of derivations under minus-normalization (in Russian). In V.A. Smirnov, editor, *The Theory of Logical Inference*. Institute of Philosophy, Moscow, 1974.

[56] U. Petermann. A complete connection calculus with rigid E-unification. In *JELIA'94*, volume 838 of *Lecture Notes in Computer Science*, pages 152–166, 1994.

[57] D.A. Plaisted. A simple non-termination test for the Knuth-Bendix method. In *Proc. 8th CADE*, volume 230 of *Lecture Notes in Computer Science*, pages 79–88, 1986.

[58] D.A. Plaisted. Special cases and substitutes for rigid E-unification. Technical Report MPI-I-95-2-010, Max-Planck-Institut für Informatik, November 1995.

[59] D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 162–201. Springer Verlag, 1983. Originally appeared in 1960.

[60] G. Robinson and L.T. Wos. Paramodulation and theorem-proving in first order theories with equality. In Meltzer and Michie, editors, *Machine Intelligence*, volume 4, pages 135–150. Edinburgh University Press, Edinburgh, 1969.

[61] J. Schumann. Tableau-based theorem provers: Systems and implementations. *Journal of Automated Reasoning*, 13(3):409–421, 1994.

[62] R. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21:583–585, July 1978.

[63] R.M. Smullyan. *First-Order Logic*. Springer Verlag, 1968.

[64] A. Voronkov. Proof search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. In M.A. McRobbie and J.K. Slaney, editors, *Automated Deduction — CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 32–46, New Brunswick, NJ, USA, 1996.

Appendix A

Proof of the completeness theorem

This appendix proves the completeness part of Theorem 4.6. It is based on the completeness theorem for the equality elimination method (Degtyarev and Voronkov [19, 20, 21, 26]). We shall introduce several notions.

A.1 Clauses

Definition A.1 (Clause)
A *clause* is a finite multiset of literals $\{L_1, \ldots, L_n\}$, denoted $L_1, \ldots, L_n$. If L is a literal and C a clause, then L, C denotes the clause $\{L\} \cup C$. The *empty clause* is denoted by $\Box$.

Definition A.2 (Constraint clause)
A *constraint clause* is a pair consisting of a clause C and a constraint $\mathcal{C}$. Such a constraint clause will be denoted $C \cdot \mathcal{C}$.

A.2 The equality elimination method

For the rest of this section we assume that ξ denotes a closed formula in the Skolem negation normal form to be checked for unsatisfiability. We assume that all different occurrences of quantifiers in ξ bind different variables. For example, ξ cannot have the form $\forall x A \wedge \forall x B$. All formulas in this section are assumed to be subformulas of the formula ξ. We shall identify subformulas of ξ and their superformulas with their *occurrences* in ξ. For example, in the formula ξ of the form $A \wedge (A \vee B)$ the second occurrence of A is considered a subformula of $(A \vee B)$, but the first occurrence of A is not.

Definition A.3 (Subformula and fresh-variable subformulas)
The set of *subformulas* and *fresh-variable subformulas* of a formula in negation normal form is defined inductively as follows.

1. Any formula is a subformula and a fresh-variable subformula of itself.
2. Both formulas φ_1 and φ_2 are subformulas and fresh-variable subformulas of $\varphi_1 \wedge \varphi_2$, and similar for $\vee$ instead of $\wedge$.
3. The formula $\varphi(x)$ is a subformula of $\forall x \varphi(x)$; any formula $\varphi(y)$, where $y \notin var(\forall x \varphi(x))$, is a fresh-variable subformula of $\forall x \varphi(x)$, and similar for $\exists$ instead of $\forall$.

4. If φ_1 is a subformula of φ_2 and φ_2 is a subformula of φ_3, then φ_1 is a subformula of φ_3, and similar for fresh-variable subformulas.

Note that we do not consider an atom A to be a subformula of $\neg A$. The reason is that we want to restrict ourselves with only positive subformulas. Also note that a fresh-variable subformula is not necessarily a subformula.

We shall only deal with some subformulas of ξ called disjunctive subformulas.

Definition A.4 (Disjunctive subformula)
The occurrence of a subformula φ of ξ is called *disjunctive* iff it is an occurrence in a subformula $\varphi \vee \psi$ or in $\psi \vee \varphi$. A *disjunctive superformula* of φ is a superformula[1] ψ of φ that is disjunctive. The *least disjunctive superformula* of φ is the disjunctive superformula ψ of φ such that any other disjunctive superformula of φ is a superformula of ψ.

Let us note that disjunctive superformulas do not necessarily exist. For example, ξ has no disjunctive superformula. Any formula having a disjunctive superformula has the unique least disjunctive superformula.

We can enumerate all disjunctive subformulas $\xi_1, \dots, \xi_n$ of ξ, for example in the order of their occurrences in ξ. Thus we can unambiguously use "the kth disjunctive (sub)formula" ξ_k of ξ.

Let $\mathcal{A}_1, \dots, \mathcal{A}_n$ be predicate symbols not occurring in ξ.

Definition A.5 (ξ-name of a subformula)
An atomic formula $\mathcal{A}_k(x_1, \dots, x_m)$ is *the ξ-name of a subformula* φ of ξ iff

1. The least disjunctive superformula of φ is ξ_k;

2. $x_1, \dots, x_m$ are all free variables of ξ_k in the order of their occurrences in ξ_k.

If a ξ-name of a formula φ exists, then it is unique. Note that different formulas may have the same ξ-names. Also note that some subformulas of ξ do not have ξ-names. We can use *the set of ξ-names* of a subformula. The set of ξ-names of a formula φ is either $\emptyset$ or a singleton $\{\mathcal{A}_k(x_1, \dots, x_m)\}$.

Figure A.1 illustrates least disjunctive superformulas and ξ-names.

Lemma A.6 Let φ, ψ be subformulas of ξ whose sets of ξ-names coincide, φ is a proper subformula of ψ, and $\psi' = \psi\eta'$, where η' is a substitution with $dom(\eta') = var(\psi)$. Let φ' be a fresh-variable subformula of ψ' which is also a variant of $\varphi\eta'$.

Let $T = \Gamma_1, \psi' \mid \Gamma_2 \mid \dots \mid \Gamma_n$ such that $var(T) \cap (var(\varphi') \setminus var(\psi')) = \emptyset$. Then there is a derivation

$$\begin{array}{c} \Gamma_1, \psi' \mid \Gamma_2 \mid \dots \mid \Gamma_n \\ \vdots \\ \Gamma_1', \psi', \varphi' \mid \Gamma_2 \mid \dots \mid \Gamma_n \end{array}$$

consisting only of applications of α- and γ-rules of Figure 4.1 such that $\Gamma_1 \sqsubseteq \Gamma_1'$.

Proof. Induction by ψ'. Note that ψ' cannot be a disjunction because then the set of ξ-names of φ and ψ would be different. Hence, there are two possible cases: either ψ' is a conjunction or begins with a universal quantifier. Let $\varphi' = \varphi\eta'\{z_1/x_1, \dots, z_m/x_m\}$, where $\{x_1, \dots, x_m\} = var(\varphi\eta') \setminus var(\psi\eta')$. Consider the two cases mentioned above.

[1] φ is a superformula of ψ iff ψ is a subformula of φ (not necessarily proper).

Subformula	least disjunctive superformula	
$(\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y))) \vee \forall z D(z)) \wedge E$	no	$\emptyset$
$\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y))) \vee \forall z D(z)$	no	$\emptyset$
$\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y)))$	$\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y)))$	$\{\mathcal{A}_1\}$
$F(x) \vee (B(x) \wedge \forall y C(x,y))$	$\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y)))$	$\{\mathcal{A}_1\}$
$F(x)$	$F(x)$	$\{\mathcal{A}_2(x)\}$
$B(x) \wedge \forall y C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$\{\mathcal{A}_3(x)\}$
$B(x)$	$B(x) \wedge \forall y C(x,y)$	$\{\mathcal{A}_3(x)\}$
$\forall y C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$\{\mathcal{A}_3(x)\}$
$C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$\{\mathcal{A}_3(x)\}$
$\forall z D(z)$	$\forall z D(z)$	$\{\mathcal{A}_4\}$
$D(z)$	$\forall z D(z)$	$\{\mathcal{A}_4\}$
E	no	$\emptyset$

Figure A.1: Least disjunctive superformulas and sets of ξ-names of subformulas of the formula $(\forall x(F(x) \vee (B(x) \wedge \forall y C(x,y))) \vee \forall z D(z)) \wedge E$

1. ψ' has the form $\forall x \chi(x)$. If for some $i \in \{1, \ldots, m\}$ we have $x = x_i$, then we apply the rule

$$\frac{\ldots \forall x_i \chi(x_i) \ldots}{\ldots \forall x_i \chi(x_i), \chi(z_i) \ldots} \; (\gamma)$$

Otherwise, choose any fresh variable z and apply the rule

$$\frac{\ldots \forall x_i \chi(x_i) \ldots}{\ldots \forall x_i \chi(x_i), \chi(z) \ldots} \; (\gamma)$$

If $\chi(z_i)$ (respectively, $\chi(z)$) coincides with φ', then the required derivation is found. Otherwise, apply the induction hypothesis to $\chi(z_i)$ (respectively, $\chi(z)$).

2. ψ' has the form $\chi_1 \wedge \chi_2$. Then we apply the rule

$$\frac{\ldots \chi_1 \wedge \chi_2 \ldots}{\ldots \chi_1 \wedge \chi_2, \chi_1, \chi_2 \ldots} \; (\gamma)$$

If φ' coincides with χ_1 or χ_2 then the required derivation is found. Otherwise, φ' is a fresh-variable proper subformula of some χ_i, where $i \in \{1, 2\}$. Then apply the induction hypothesis to χ_i.

□

Lemma A.7 Let $T = \Gamma_1 \mid \ldots \mid \Gamma_n$ be a tableau obtained from ξ by tableau expansions rules and φ a proper subformula of ξ whose set of γ-names is empty. Let φ' be a fresh-variable subformula of ξ that is also a variant of φ such that $var(\varphi') \cap var(T) = \emptyset$. Then there is derivation

$$\begin{array}{c}\Gamma_1 \mid \Gamma_2 \mid \ldots \mid \Gamma_n \\ \vdots \\ \Gamma_1', \varphi' \mid \Gamma_2 \mid \ldots \mid \Gamma_n\end{array}$$

consisting only of applications of α- and γ-rules of Figure 4.1 such that $\Gamma_1 \sqsubseteq \Gamma_1'$.

Proof. If $\xi \in \Gamma_1$, then the claim is immediate by Lemma A.6, where $\psi = \xi$ and η' is the empty substitution. Suppose (by contradiction) $\xi \notin \Gamma_1$. Evidently, ξ is a disjunction. But then ξ has no proper subformulas whose sets of ξ-names are empty. Contradiction. □

Theorem 4.6 that we prove in this section connects two calculi: (i) calculus $\mathcal{BSE}$ of rigid equations; and (ii) tableau expansion rules for tableaux. The main theorem on the completeness of the equality elimination method [19, 26] (Theorem A.15 below) also connects two calculi, both introduced below: (i) the calculus $\mathcal{CBSE}$ of constraint clauses; and (ii) ξ-expansion rules for named ξ-tableaux. Both calculi depend on the goal formula ξ.

Definition A.8 (Calculus $\mathcal{CBSE}$)
The calculus $\mathcal{CBSE}$ of constraint clauses consists of the following inference rules. As usual, we assume that premises of rules have disjoint variables which can be achieved by renaming variables.

Left rigid basic superposition:

$$\frac{l \simeq r, C_1 \cdot \mathcal{C}_1 \quad s[p] \simeq t, C_2 \cdot \mathcal{C}_2}{s[r] \simeq t, C_1, C_2 \cdot \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{l \succ r, s[p] \succ t, l \simeq p\}} \; (lrbs)$$

Right rigid basic superposition:

$$\frac{l \simeq r, C_1 \cdot \mathcal{C}_1 \quad s[p] \not\simeq t, C_2 \cdot \mathcal{C}_2}{s[r] \not\simeq t, C_1, C_2 \cdot \mathcal{C}_1 \cup \mathcal{C}_2 \cup \{l \succ r, s[p] \succ t, l \simeq p\}} \; (rrbs)$$

Equality resolution:

$$\frac{s \not\simeq t, C \cdot \mathcal{C}}{C \cdot \mathcal{C} \cup \{s \simeq t\}} \; (er)$$

Application of all the rules is restricted to the following conditions:

1. The constraint at the conclusion of the rule is satisfiable;
2. In the basic superposition rules, the term p is not a variable.
3. In the left basic superposition rule, $s[r] \neq t$.

Definition A.9 (Named ξ-tableau)
A *named ξ-tableau* is a tableau all whose formulas are atomic formulas of the form $\mathcal{A}_i(t_1, \ldots, t_m)$.

Definition A.10 (Initial named ξ-tableau)
The *initial named ξ-tableau of T_ξ* is the tableau □ which has one branch consisting of the empty set of formulas.

Definition A.11 (ξ-expansion rules)
Let $\{\mathcal{A}_i(\bar{x}_i)\}, \{\mathcal{A}_j(\bar{x}_j)\}$ and D be the sets of ξ-names of φ, ψ and $\varphi \vee \psi$, where $\varphi \vee \psi$ is a subformula of ξ. Then the following are ξ-expansion rules.

1. if $D = \emptyset$ then

$$\frac{C_1 \mid C_2 \mid \ldots \mid C_m}{C_1, \mathcal{A}_i(\bar{x}_i) \mid C_1, \mathcal{A}_j(\bar{x}_j) \mid C_2 \mid \ldots \mid C_m}$$

2. if $D = \{\mathcal{A}_k(\bar{x}_k)\}$ then

$$\frac{C_1, \mathcal{A}_k(\bar{y}) \mid C_2 \mid \ldots \mid C_m}{(C_1, \mathcal{A}_i(\bar{x}_i), \mathcal{A}_k(\bar{y}) \mid C_1, \mathcal{A}_j(\bar{x}_j), \mathcal{A}_k(\bar{y}) \mid C_2 \mid \ldots \mid C_m)\{\bar{x}_k/\bar{y}\}}$$

We assume that the variables of the premises are disjoint from the variables of $\bar{x}_i, \bar{x}_j, \bar{x}_k$).

Definition A.12 (Initial constraint ξ-clause)
Initial constraint ξ-clauses are defined as follows:

1. Whenever a literal $s \not\simeq t$ occurs in ξ and C is the set of ξ-names of this occurrence of $s \not\simeq t$, the constraint clause $s \not\simeq t, C \cdot \{\}$ is an initial constraint ξ-clause.
2. Whenever a literal $s \simeq t$ occurs in ξ and C is the set of ξ-names of this occurrence of $s \simeq t$, the constraint clause $s \simeq t, C \cdot \{\}$ is an initial constraint ξ-clause.

Definition A.13 (Solution constraint ξ-clause)
A *solution constraint ξ-clause* is any constraint clause $C \cdot \mathcal{C}$ which is derivable from initial constraint ξ-clauses using rules $(lrbs), (rrbs)$ and (er) and such that C does not contain $\simeq$.

Definition A.14 (Answer constraint)
Let $C' \cdot \mathcal{C}'$ be a solution constraint ξ-clause and C be a branch of a named ξ-tableau, so that $var(C) \cap (var(C') \cup var(\mathcal{C}')) = \emptyset$. If η is a substitution such that (i) $dom(\eta) = var(C')$; (ii) $C'\eta \sqsubseteq C$ and (iii) $\mathcal{C}'\eta$ is satisfiable, then $\mathcal{C}'\eta$ is an *answer constraint for C.*

The following theorem is a reformulation of Soundness and Completeness theorems of [19] for named ξ-tableaux and constraint clauses.

Theorem A.15 The formula ξ is unsatisfiable if and only if there exists a named ξ-tableau $C_1 \mid \ldots \mid C_n$ obtained from the initial named ξ-tableau by ξ-expansion rules with the following property. There exist answer constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ for $C_1, \ldots, C_n$, respectively, such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$ is satisfiable.

Definition A.16 (Image)
Let the atomic formula $\mathcal{A}_k(x_1, \ldots, x_m)$ be the ξ-name of a disjunctive subformula $\varphi(x_1, \ldots, x_m)$ of ξ. For any terms $t_1, \ldots, t_m$, the *image of the formula* $\mathcal{A}_k(t_1, \ldots, t_m)$ is the formula $\varphi(t_1, \ldots, t_m)$. The *image* of a named ξ-tableau $C_1 \mid \ldots \mid C_n$ is the tableau $\Gamma_1 \mid \ldots \mid \Gamma_n$ obtained from $C_1 \mid \ldots \mid C_n$ by replacing every atomic formula $\mathcal{A}_k(t_1, \ldots, t_m)$ by its image.

Note that the image of a named ξ-tableau is uniquely defined.

Lemma A.17 Let $C_1 \mid \ldots \mid C_n$ be any named ξ-tableau obtained from $\square$ by ξ-expansion rules and $\Gamma_1 \mid \ldots \mid \Gamma_n$ be its image. Then there exists a tableau $\Gamma'_1 \mid \ldots \mid \Gamma'_n$ obtained from ξ by tableau expansion rules such that $\Gamma_i \sqsubseteq \Gamma'_i$, for all $i \in \{1, \ldots, n\}$.

Proof. By induction on the number of applications of expansion rules. The basic case is obvious: we can take ξ as the required tableau. Consider the induction step. The following two cases are possible.

1. The expansion rule has the form

$$\frac{C_1 \mid C_2 \mid \ldots \mid C_n}{C_1, \mathcal{A}_i(\bar{x}_i) \mid C_1, \mathcal{A}_j(\bar{x}_j) \mid C_2 \mid \ldots \mid C_n}$$

   Denote the image of $C_1 \mid \ldots \mid C_n$ by $\Gamma_1 \mid \ldots \mid \Gamma_n$. By the induction hypothesis, there is a tableau $\Gamma'_1 \mid \ldots \mid \Gamma'_n$ obtained from ξ by tableau expansion rules such that $\Gamma_i \sqsubseteq \Gamma'_i$, for all $i \in \{1, \ldots, n\}$. Let $\xi_i(\bar{x}_i)$ be the image of $\mathcal{A}_i(\bar{x}_i)$ and $\xi_j(\bar{x}_j)$ be the image of $\mathcal{A}_j(\bar{x}_j)$. Then $\xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ has the empty set of ξ-names and it is a fresh-variable subformula of ξ. By the definition ξ-expansion rules, variables $\bar{x}_i, \bar{x}_j$ do not occur in $C_1 \mid \ldots \mid C_n$. Evidently, we can assume that $\bar{x}_i, \bar{x}_j$ do not occur in $\Gamma'_1 \mid \ldots \mid \Gamma'_n$ (we can rename variables in Γ'_k distinct from variables of $C_1, \ldots, C_n$). Applying Lemma A.7 to the tableau $T = \Gamma'_1 \mid \ldots \mid \Gamma'_n$ and formula $\varphi = \xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ we see that there is a tableau $\Gamma''_1, \xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n$ obtained from ξ be tableau expansion rules such that $\Gamma'_1 \sqsubseteq \Gamma''_1$. Applying β-rule

$$\frac{\Gamma''_1, \xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n}{\Gamma''_1, \xi_i(\bar{x}_i) \mid \Gamma''_1, \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n}$$

   we obtain the required tableau.

2. The expansion rule has the form

$$\frac{C_1, \mathcal{A}_k(\bar{y}) \mid C_2 \mid \ldots \mid C_n}{(C_1, \mathcal{A}_i(\bar{x}_i), \mathcal{A}_k(\bar{y}) \mid C_1, \mathcal{A}_j(\bar{x}_j), \mathcal{A}_k(\bar{y}) \mid C_2 \mid \ldots \mid C_n)\{\bar{x}_k/\bar{y}\}}$$

   Denote the image of $C_1 \mid \ldots \mid C_n$ by $\Gamma_1 \mid \ldots \mid \Gamma_n$ and the image of $\mathcal{A}_k(\bar{y})$ by $\xi_k(\bar{y})$. By the induction hypothesis, there is a tableau $\Gamma'_1, \xi_k(\bar{y}) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n$ obtained from ξ be tableau expansion rules such that $\Gamma_i \sqsubseteq \Gamma'_i$, for all $i \in \{1, \ldots, n\}$. Since ξ contains no free variables, any variant of this tableau can also be obtained from ξ be tableau expansion rules, in particular the tableau $(\Gamma'_1, \xi_k(\bar{y}) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}$. The formula $\xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ is a subformula of $\xi_k(\bar{x}_k)$, and their sets of ξ-names coincide. Then either $\xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ coincides with, or is a proper subformula of $\xi_k(\bar{x}_k)$. Consider the two corresponding cases

   (a) $\xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ coincides with $\xi_k(\bar{x}_k)$. Applying β-rule

$$\frac{(\Gamma'_1, \xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}}{(\Gamma'_1, \xi_i(\bar{x}_i) \mid \Gamma'_1, \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}}$$

   we obtain the required tableau.

(b) $\xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j)$ is a proper subformula of $\xi_k(\bar{x}_k)$. It is also a fresh-variable subformula. To apply Lemma A.7, we have to guarantee the condition

$$var((\Gamma'_1, \xi_k(\bar{y}) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}) \cap ((\bar{x}_i \cup \bar{x}_j) \setminus \bar{x}_k) = \emptyset. \qquad \text{(A.1)}$$

By the conditions of this lemma, $\bar{x}_i, \bar{x}_j$ do not occur in $\Gamma_1, \xi_k(\bar{y}) \mid \Gamma_2 \mid \ldots \mid \Gamma_n$. Since in the derivation of $\Gamma'_1, \xi_k(\bar{y}) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n$ we could introduce fresh variables different from $\bar{x}_i, \bar{x}_j$, we have

$$(\bar{x}_i \cup \bar{x}_j) \cap var((\Gamma'_1, \xi_k(\bar{y}) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}) \subseteq \bar{x}_k$$

Thus, condition (A.1) can be guaranteed. By Lemma A.7, there is a required derivation of the form

$$\begin{array}{c}
\xi \\
\vdots \\
(\Gamma'_1, \xi_k(\bar{x}_k) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\} \\
\vdots \\
\dfrac{(\Gamma''_1, \xi_k(\bar{x}_k), \xi_i(\bar{x}_i) \vee \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}}{(\Gamma''_1, \xi_k(\bar{x}_k), \xi_i(\bar{x}_i) \mid \Gamma''_1, \xi_k(\bar{x}_k), \xi_j(\bar{x}_j) \mid \Gamma'_2 \mid \ldots \mid \Gamma'_n)\{\bar{x}_k/\bar{y}\}}
\end{array}$$

□

Lemma A.18 Let $\mathcal{C}$ be an answer constraint for a rigid equation R and η be a substitution such that $\mathcal{C}\eta$ is satisfiable. Then there exists $\mathcal{C}' \sqsubseteq \mathcal{C}$ such that $\mathcal{C}'\eta$ is an answer constraint for $R\eta$.

Proof. Consider the derivation

$$R \cdot \emptyset = E_0 \vdash_\forall s_0 \simeq t_0 \cdot \mathcal{C}_0 \rightsquigarrow E_1 \vdash_\forall s_1 \simeq t_1 \cdot \mathcal{C}_1 \rightsquigarrow \ldots \rightsquigarrow E_n \vdash_\forall s_n \simeq t_n \cdot \mathcal{C}_n \qquad \text{(A.2)}$$

such that $\mathcal{C}_n = \mathcal{C}$. Consider the figure

$$\begin{array}{c}
\dfrac{(E_0 \vdash_\forall s_0 \simeq t_0 \cdot \mathcal{C}_0)\eta}{(E_1 \vdash_\forall s_1 \simeq t_1 \cdot \mathcal{C}_1)\eta} \\
\vdots \\
(E_n \vdash_\forall s_n \simeq t_n \cdot \mathcal{C}_n)\eta
\end{array} \qquad \text{(A.3)}$$

The figure

$$\frac{(E_i \vdash_\forall s_i \simeq t_i \cdot \mathcal{C}_i)\eta}{(E_{i+1} \vdash_\forall s_{i+1} \simeq t_{i+1} \cdot \mathcal{C}_{i+1})\eta}$$

is not a correct inference rule of $\mathcal{BSE}$ in one of the following cases.

1. $E_{i+1} \setminus E_i$ contains an equation $s[r] \simeq t$ such that $s[r]\eta = t\eta$. In this case condition (4) on the rules of $\mathcal{BSE}$ is violated. Note that $s[r] \simeq t$ cannot be used in the inference rules of (A.2). Indeed, otherwise $\mathcal{C}$ would contain either $s[r] \succ t$ or $t \succ s[r]$ which contradicts to the condition $\mathcal{C}\eta$ is satisfiable. Thus, we can assume without loss of generality that (A.2) does not contain applications of left basic superposition giving such equations $s[r] \simeq t$. Then (A.3) is a correct $\mathcal{BSE}$-derivation satisfying the conditions.

2. $s_{i+1}\eta = t_{i+1}\eta$, where $i+1 \neq n$. In this case condition (2) on the rules of $\mathcal{BSE}$ is violated. This case is considered similarly.

□

Let τ be a tree derivation of a solution constraint ξ-clause $C \cdot \mathcal{C}$. Without loss of generality we can assume that the leaves of τ have disjoint variables. Denote by $L(\tau)$ the multiset of the leaves of τ.

We shall associate with τ a rigid equation R_τ such that $\mathcal{C}$ is an answer constraint for R_τ.

Definition A.19 A rigid equation $E \vdash_\forall s \simeq t$ is *associated with* τ if E is the multiset of all equations occurring in $L(\tau)$, and $s \not\simeq t$ is the only disequation occurring in $L(\tau)$.

We recall that any element of $L(\tau)$ has either the form $p \simeq q, D \cdot \{\}$, or the form $p \not\simeq q, D \cdot \{\}$, where D does not contain equations or disequations.

Lemma A.20 Let $E \vdash_\forall s \simeq t$ is a rigid equation associated with a derivation τ of a solution constraint clause $C \cdot \mathcal{C}$. Then $\mathcal{C}$ is an answer constraint for $E \vdash_\forall s \simeq t$.

Proof. Straightforward. □

Combining Lemmas A.18 and A.20, we obtain

Lemma A.21 Let $E \vdash_\forall s \simeq t$ is a rigid equation associated with a derivation τ of a solution constraint clause $C' \cdot \mathcal{C}'$ and η be a substitution such that $\mathcal{C}' \cdot \eta$ is satisfiable. Then there exists a constraint $\mathcal{C}'' \sqsubseteq \mathcal{C}'$ such that $\mathcal{C}''\eta$ is an answer constraint for $(E \vdash_\forall s \simeq t)\eta$.

Definition A.22 ($\alpha\gamma$-expansion)
Let a tableau T' be obtained from a tableau T by a sequence of α- and γ-rules. Then T' is called an *$\alpha\gamma$-expansion of T*.

Lemma A.23 Let $\Gamma_1' \mid \ldots \mid \Gamma_n'$ be an $\alpha\gamma$-expansion of $\Gamma_1 \mid \ldots \mid \Gamma_n$. Then $\Gamma_i \sqsubseteq \Gamma_i'$, for all $i \in \{1, \ldots, n\}$.

Proof. Obvious. □

Let us introduce a technical definition.

Definition A.24 Let $E \vdash_\forall s \simeq t$ be a rigid equation. Its *clause form* is the clause $E, s \not\simeq t$.

Finally, we come to the proof of the main theorem.

Theorem 4.6 Let ξ be a sentence in Skolem negation normal form. Then ξ is unsatisfiable iff there is a tableau T obtained from ξ by tableau expansion rules with the following property. Let $\Gamma_1, \ldots, \Gamma_n$ be all branches of T. Then there exist answer constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ for $\Gamma_1, \ldots, \Gamma_n$, respectively, such that $\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_n$ is satisfiable.

Proof. The soundness part is proven in Section 4. We only prove the completeness part. Let ξ be unsatisfiable. By Theorem A.15, there exist a named ξ-tableau $C_1 \mid \ldots \mid C_n$ obtained from the initial named ξ-tableau by ξ-expansion rules and answer constraints $\mathcal{C}_1^0, \ldots, \mathcal{C}_n^0$ for $C_1, \ldots, C_n$, respectively, such that $\mathcal{C}_1^0 \cup \ldots \cup \mathcal{C}_n^0$ is satisfiable. By Definition A.14 of answer constraints, there exist solution constraint ξ-clauses $C_1' \cdot \mathcal{C}_1', \ldots, C_n' \cdot \mathcal{C}_n'$ and substitutions $\eta_1, \ldots, \eta_n$ such that

1. $var(C_i) \cap var(C_i' \cdot \mathcal{C}_i') = \emptyset$;
2. $dom(\eta_i) = var(C_i')$;
3. $C_i'\eta_i = C_i^0 \sqsubseteq C_i$.

Without loss of generality we can assume that $var(C_i' \cdot \mathcal{C}_i') \cap var(C_j' \cdot \mathcal{C}_j') = \emptyset$, whenever $i \neq j$. Hence, we can introduce the substitution $\eta = \eta_1 \cup \ldots \cup \eta_n$.

Let $\Gamma_1' \mid \ldots \mid \Gamma_n'$ be the image of $C_1 \mid \ldots \mid C_n$. By Lemma A.17, there is a tableau $\Gamma_1'' \mid \ldots \mid \Gamma_n''$ obtained from ξ by tableau expansion rules such that $\Gamma_i' \sqsubseteq \Gamma_i''$, for all $i \in \{1, \ldots, n\}$.

Let $\tau_1, \ldots, \tau_n$ be derivations of $C_1' \cdot \mathcal{C}_1', \ldots, C_n' \cdot \mathcal{C}_n'$, respectively in the tree form. Let τ be the multiset $\tau_1, \ldots, \tau_n$ of trees. Let $L(\tau)$ be the multiset of leaves in τ. We assume any two members of $L(\tau)$ have disjoint sets of variables. Since $\Gamma_1'' \mid \ldots \mid \Gamma_n''$ is derived from the closed formula ξ, we can also assume that

$$var(\tau) \cap var(\Gamma_1'' \mid \ldots \mid \Gamma_n'') = \emptyset. \tag{A.4}$$

Let for all $i \in \{1, \ldots, n\}$ the rigid equation $E_i \vdash_\forall s_i \simeq t_i$ be associated with τ_i and $E_i, s_i \not\simeq t_i$ be its clause form.

Now we shall construct an $\alpha\gamma$-expansion $\Gamma_1 \mid \ldots \mid \Gamma_n$ of $\Gamma_1'' \mid \ldots \mid \Gamma_n''$ such that $(E_i, s_i \not\simeq t_i)\eta \sqsubseteq \Gamma_i$, for all $i \in \{1, \ldots, n\}$.

Let $L_1, \ldots, L_l$ be the multiset of all equations and disequations in the leaves of τ. We shall construct the required $\Gamma_1 \mid \ldots \mid \Gamma_n$ using a sequence of $\alpha\gamma$-expansions

$$\begin{array}{rl} \Gamma_1'' \mid \ldots \mid \Gamma_n'' = & \Gamma_1^{(0)} \mid \ldots \mid \Gamma_n^{(0)} \\ & \Gamma_1^{(1)} \mid \ldots \mid \Gamma_n^{(1)} \\ & \ldots \\ & \Gamma_1^{(l)} \mid \ldots \mid \Gamma_n^{(l)} = \Gamma_1 \mid \ldots \mid \Gamma_n \end{array}$$

During the construction we shall satisfy the following conditions:

1. For every i, j with $1 \leq i \leq l$ and $1 \leq j \leq n$, if L_i is in the leaf of τ_j, then $L_i\eta$ is on the branch $\Gamma_j^{(i)}$;
2. For every i with $1 \leq i < l$ we have

   $$var(\Gamma_1^{(i)} \mid \ldots \mid \Gamma_n^{(i)}) \cap var(L_{i+1}, \ldots, L_l) = \emptyset$$

   (Initially, this condition holds by (A.4).)

It is straightforward to check that the first condition implies that $\Gamma_1, \ldots, \Gamma_n$ is the required tableau.

Suppose that we have already constructed the tableau $\Gamma_1^{(k)} \mid \ldots \mid \Gamma_n^{(k)}$ for $k < l$. We show how to construct $\Gamma_1^{(k+1)} \mid \ldots \mid \Gamma_n^{(k+1)}$. Let L_{k+1} is in the leaf of the tree τ_i. Then this leaf has the form $L_{k+1}, D \cdot \{\}$ such that $D \sqsubseteq C_i'$ and $C_i'\eta = C_i^0 \sqsubseteq C_i$.

There are two possible cases: either $D \neq \emptyset$ or $D = \emptyset$.

1. Let $\bar{y} = var(L_{k+1})$ and $\bar{x} = var(D)$ and $L_{k+1}(\bar{y}), D(\bar{x}) \cdot \{\}$ be a variant of an initial constraint ξ-clause $L_{k+1}(\bar{z}), \mathcal{A}_m(\bar{x}_m) \cdot \{\}$. Then $D\eta = \mathcal{A}_m(\bar{x}\eta) \sqsubseteq C_i$. Let $\xi_m(\bar{x}\eta)$ be the image of $\mathcal{A}_m(\bar{x}\eta)$. By the construction of Γ_i'', we have $\xi_m(\bar{x}\eta) \in \Gamma_i''$ since $\mathcal{A}_m(\bar{x}\eta) \in C_i$.

   There are two possible cases.

(a) $L_{k+1}(\bar{z}) = \xi_m(\bar{x}_m)$. In this case we define $\Gamma_1^{(k+1)} \mid \ldots \mid \Gamma_n^{(k+1)}$ as $\Gamma_1^{(k)} \mid \ldots \mid \Gamma_n^{(k)}$.

(b) $L_{k+1}(\bar{z})$ is a proper subformula of $\xi_m(\bar{x}_m)$ (whose ξ-name is $\mathcal{A}_m(\bar{x}_m)$. Then apply Lemma A.6. To this end we let

$$\begin{aligned}
&\psi = \xi_m(\bar{x}_m)\\
&\varphi = L_{k+1}(\bar{z})\\
&\psi' = \xi_m(\bar{x}\eta) = \xi_m(\bar{x}_m)\{\bar{x}\eta/\bar{x}_m\}\\
&\eta' = \{\bar{x}\eta/\bar{x}_m\}\\
&\varphi' = L_{k+1}(\bar{y}\eta) = L_{k+1}(\bar{z})\eta'\\
&T = \Gamma_1^{(k)} \mid \ldots \mid \Gamma_n^{(k)}
\end{aligned}$$

Let us check that the condition of Lemma A.6

$$\mathit{var}(T) \cap (\mathit{var}(\varphi') \setminus \mathit{var}(\psi')) = \emptyset$$

is satisfied. Let v be any variable such that $v \in \mathit{var}(\bar{y}\eta)$ and $v \notin \mathit{var}(\bar{x}\eta)$. If v belongs to $\bar{y}$ then $v \notin \mathit{var}(T)$ since by the construction of T we have $\mathit{var}(T) \cap \mathit{var}(L_{k+1}) = \emptyset$. Hence, it is enough to prove that v belongs to $\bar{y}$.

Suppose, by contradiction, that v does not belong to $\bar{y}$. Using $v \in \mathit{var}(\bar{y}\eta)$, we obtain $v \in \mathit{var}(\bar{x}\eta)$, since $\mathit{dom}(\eta) \cap (\bar{x} \cup \bar{y}) = \bar{x}$. This contradicts $v \notin \mathit{var}(\bar{x}\eta)$.

Thus, we can apply Lemma A.6 and deduce the tableau $\Gamma_1^{(k+1)} \mid \ldots \mid \Gamma_n^{(k+1)}$.

To guarantee the condition

$$\mathit{var}(\Gamma_1^{(k+1)} \cap \ldots \cap \Gamma_n^{(k+1)}) \cap \mathit{var}(L_{k+2}, \ldots, L_l) = \emptyset$$

we can require that variables introduced by γ-rules when we come from $\Gamma_1^{(k)} \mid \ldots \mid \Gamma_n^{(k)}$ to $\Gamma_1^{(k+1)} \mid \ldots \mid \Gamma_n^{(k+1)}$ do not belong to the set $\mathit{var}(L_{k+2}, \ldots, L_l)$.

2. *Case* $D = \emptyset$. Analogous, but using Lemma A.7 instead of Lemma A.6.

Thus, we have constructed an $\alpha\gamma$-expansion $\Gamma_1 \mid \ldots \mid \Gamma_n$ of $\Gamma_1'' \mid \ldots \mid \Gamma_n''$ such that $(E_i, s_i \not\simeq t_i)\eta \sqsubseteq \Gamma_i$, for all $i \in \{1, \ldots, n\}$. Then the set of rigid equations on Γ_i contains a rigid equation $E_i', E_i\eta \vdash_\forall (s_i \simeq t_i)\eta$, where E_i' is a multiset of equations. Evidently, the set of answer constraints for $E_i', E_i\eta \vdash_\forall (s_i \simeq t_i)\eta$ contains all answer constraints for $E_i\eta \vdash_\forall (s_i \simeq t_i)\eta$. By Lemma A.20, $\mathcal{C}_i'$ is an answer constraint for $E_i \vdash_\forall s_i \simeq t_i$. Since $\mathcal{C}_i'\eta$ is satisfiable, by Lemma A.21 there exists a constraint $\mathcal{C}_i'' \sqsubseteq \mathcal{C}_i'$ such that $\mathcal{C}_i''\eta$ is an answer constraint for $(E_i \vdash_\forall s_i \simeq t_i)\eta$. Hence, $\mathcal{C}_i''\eta$ is an answer constraint for Γ_i. Since $\bigcup_{i\in\{1,\ldots,n\}} \mathcal{C}_i'\eta$ is satisfiable, then $\bigcup_{i\in\{1,\ldots,n\}} \mathcal{C}_i''\eta$ is also satisfiable. Evidently, the constraints $\mathcal{C}_1 = \mathcal{C}_1''\eta, \ldots, \mathcal{C}_n = \mathcal{C}_n''\eta$ satisfy the claim of the theorem. $\square$