UPMAIL Technical Report No. 93
January 16, 1995

General Connections via Equality Elimination

Anatoli Degtyarev[*] [†]
Andrei Voronkov[‡]

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

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

[*]Supported by a grant from the Swedish Institute
[†]On leave from Kiev University
[‡]Supported by a TFR grant

Abstract

A new way of handling equality in the connection method is described. Our method is called *equality elimination.* It does not require rigid E-unification which is the basis of most known extension procedures. Equality elimination uses efficient ordering strategies via basic superposition and simplification. In addition, we present a new treatment of the connection method via *named matrices* which gives a number of advantages.

Section 1

Introduction

We propose an approach to adding equality to extension proof methods which are known under two different names *connection method* [BS 75, Bib 87] and the method of matings [And 76, And 81, Gal 90, Gal 92]. Both methods express the same idea going back to the fundamental Herbrand theorem [Her 30, Min 66, Min 67a]. According to these approaches, the proof-search can be considered as a the problem of verifying that each path through a matrix of the goal formula is complementary (in the non-equality case it means that each path contains a literal and its complement). The first justification of this idea has been given by Prawitz [Pra 60, Pra 69, Pra 70].

The main problem addressed by Prawitz when he developed matrix methods was the following. Instead of generating instances of a closed formula $\forall\bar{x}M(\bar{x})$ he proposed to search for a set of substitutions for $\bar{x_1},\ldots,\bar{x_n}$ that makes all paths in $M(\bar{x_1})\wedge\ldots\wedge M(\bar{x_n})$ complementary.

There are many ways to solve this problem. Prawitz considered two of them. In both solutions the matrix expands step by step. Then, for any fixed n in the first (systematic) method a set of substitution conditions is formed for the whole matrix. In the second method the substitution is searched for by an incremental exploration of all paths with the backtracking construction of the substitution conditions.

A generalization of the ideas of Prawitz to the predicate calculus with equality has been proposed by Bibel [Bib 87] and Gallier et al. [Gal 90]. In this generalization complementary literals are replaced by eq-connections instantiated by eq-unifiers [Bib 87] or mated sets instantiated by rigid E-unifiers [Gal 90] which is the same despite different terminology. Unfortunately, this approach preserves all problems created by the non-equality matrix methods and, in addition, creates much more severe problems. First, it is still unclear how to expand the matrix. (Prawitz searched for minimal expansions.) Second, even the problem of finding a mated set with a rigid E-unifier is extremely difficult [GNRS 92]: if a path contains q equations and r disequations then there are $r\cdot 2^q$ mated sets. The check of the rigid unifiability for a given mated set is NP-complete. Third, in order to find a substitution for the whole matrix, rigid E-unification is not enough — one needs to solve the so called simultaneous rigid E-unifiability problem. The rigid E-unification algorithms do not generalize to the simultaneous case immediately, moreover it is not yet known whether simultaneous rigid E-unifiability is decidable (see [BP 94]). Simultaneous rigid E-unification is the crucial problem for traditional approaches to matrix methods.

The major difference of our treatment of equality reasoning for the connection method is that instead of checking paths through an extension of the input formula for $\mathcal{E}$-complementarity (i.e. for the existence of a rigid E-unifier) we first generate a set of $\mathcal{E}$-complementary paths using ordinary unification, and then look for an appropriate extension. Thus, we make the search directed. In [Pet 93] it is noted that the problem of combining paramodulation-based equality handling and extension procedure is non-trivial. What we do is to split equality reasoning and extension procedures into two separate procedures and connect them via solution clauses generated by the equality reasoning part. This allows one to use strong sides of both equality reasoning methods known so far and extension procedures. This approach to adding equality to the connection method has been sketched in [DV 94c] following techniques developed in [DV 94a, DV 94b]. Since we eliminate equality in the first part of our method, the method is called *equality elimination.*

Section 2

Preliminaries.

In this section we present a brief overview of main notions used in the paper. We assume the basic knowledge of substitutions and unification.

All formulas belong to a fixed signature Σ containing at least one constant and a binary predicate symbol $=$. By $\mathcal{X}$ we denote a set of variables. The set of all terms in the signature Σ with variables from $\mathcal{X}$ is denoted by $T_\Sigma(\mathcal{X})$. By T_Σ we denote $T_\Sigma(\emptyset)$, i.e. the set of all ground terms in the signature Σ.

We write $A[s]$ to indicate that an expression A contains s as a subexpression and ambiguously denote by $A[t]$ the result of replacing particular occurrences of s in A by t. By $A\sigma$ we denote the result of applying the substitution σ to A.

We shall sometimes denote tuples of equations $s_1 = t_1, \ldots, s_n = t_n$ by $\bar{s} = \bar{t}$. The overbar notation $\bar{t}$ can also be used to denote sequences of terms. The domain $Dom(\sigma)$ of the substitution σ is the set $\{x \mid x\sigma \neq x\}$. Substitutions θ with the domain $x_1, \ldots, x_n$ will be denoted by $[x_1\theta/x_1, \ldots, x_n\theta/x_n]$.

When we use notation $\models$ or $\vdash$, they state for truth and provability *in classical first order logic.*

Definition 2.1 *A* literal *is either an atomic formula $P(t_1, \ldots, t_n)$ or its negation $\neg P(t_1, \ldots, t_n)$. An* equation *is an atomic formula of the form $s = t$. A* clause *is a set $\{L_1, \ldots, L_n\}$ of literals, denoted $L_1, \ldots, L_n$. By a* ground expression *(i.e. term, equation, clause etc.) we mean an expression containing no variables.*

If L is a literal, C a clause, L, C will denote the clause $\{L\} \bigcup C$. Literals of the form $\neg s = t$ will always be denoted by $s \neq t$.

Definition 2.2 *Let $\succ$ be a partial ordering on $T_\Sigma(\mathcal{X})$. It is called a* reduction ordering *iff*

1. *$\succ$ is well-founded;*
2. *if $s \succ t$ then $u[s\sigma] \succ u[t\sigma]$, for all terms s, t, u and substitutions σ.*

We shall always assume that reduction orderings are total on the set T_Σ.

Definition 2.3 Formulas *and their outermost connectives are defined inductively as follows:*

1. *Any atomic formula is a formula. It has no outermost connective.*
2. *Let $\varphi_1, \ldots, \varphi_n$ be formulas and there is no φ_i with the outermost connective $\wedge$. Then $(\varphi_1 \wedge \ldots \wedge \varphi_n)$ is a formula with the outermost connective $\wedge$.*

3. *Let $\varphi_1, \ldots, \varphi_n$ be formulas and there is no φ_i with the outermost connective $\vee$. Then $(\varphi_1 \vee \ldots \vee \varphi_n)$ is a formula with the outermost connective $\vee$.*

4. *Let φ_1, φ_2 be formulas. Then $(\varphi_1 \supset \varphi_2)$ is a formula with the outermost connective $\supset$.*

5. *Let φ be a formula. Then $\neg\varphi$ is a formula with the outermost connective $\neg$.*

6. *Let φ be a formula whose outermost connective (if any) is not $\forall$. Let $x_1, \ldots, x_n$ be a sequence of variables. Then $\forall x_1 \ldots x_n \varphi$ is a formula with the outermost connective $\forall$.*

7. *Let φ be a formula whose outermost connective (if any) is not $\exists$. Let $x_1, \ldots, x_n$ be a sequence of variables. Then $\exists x_1 \ldots x_n \varphi$ is a formula with the outermost connective $\exists$.*

Definition 2.4 *A formula is called a* sentence, *or a* closed *formula iff it has no free variables. A formula φ is* rectified *iff all occurrences of quantifiers in φ bind different variables and no variable is both free and bound in φ.*

Definition 2.5 *A formula φ is in the* negation Skolem normal form *iff it is constructed from literals using connectives $\wedge, \vee$ and quantifiers $\forall$. Such a formula is* miniscoped *iff it has no occurrences of subformulas of the form $\forall\bar{x}(\varphi_1 \wedge \ldots \wedge \varphi_n)$.*

There is a standard non-increasing transformation of arbitrary formulas without equivalences into refutably equivalent miniscoped formulas in the Skolem negation normal form. Unless otherwise is explicitly stated, in the following we assume all input formulas to be in the Skolem negation normal form and miniscoped.

Section 3

The Connection Method

In this section we recall main notions related to the connection method. There is no uniform notation for the connection method (in fact there is even no uniform name for the method itself), our notation is essentially based on that of [And 81].

Definition 3.1 *Let φ be a formula in the negation normal form. A formula φ' is obtained from φ by* quantifier duplication *iff φ' is the result of replacing a subformula $\forall x\psi$ by $\forall x\psi \wedge \forall x'\psi[x'/x]$, where x' does not occur in φ. If φ' is obtained from φ by a sequence of quantifier duplications and omitting all quantifiers, then φ' is an* amplification *of φ.*

The connection method is often described in terms of *matrices.* In this paper, we introduce a special kind of matrices called *formula matrices.*

Definition 3.2 Formula matrices *are defined inductively as follows:*

1. *Any formula is a formula matrix.*
2. *If $M_1, \ldots, M_n$ are formula matrices then so are*

$$[M_1 \ldots M_n]$$

and

$$\begin{bmatrix} M_1 \\ \vdots \\ M_n \end{bmatrix}$$

If M is formula matrix having a row inside a row:

$$[\varphi_1 \ldots [\psi_1 \ldots \psi_m] \ldots \varphi_n]$$

it will be represented as a single row:

$$[\varphi_1 \ldots \psi_1 \ldots \psi_m \ldots \varphi_n]$$

Similarly, columns inside columns will be represented as single columns.

Definition 3.3 *Let M be a formula matrix. The set $vp(M)$ of* vertical paths through M *is the set of sets of formulas defined inductively as follows:*

$$vp(\varphi) = \{\{\varphi\}\}$$

$$vp(\begin{bmatrix} M_1 \cdots M_n \end{bmatrix}) = vp(M_1) \bigcup \ldots \bigcup vp(M_n)$$

$$vp(\begin{bmatrix} M_1 \\ \vdots \\ M_n \end{bmatrix}) = \{\pi_1 \bigcup \ldots \bigcup \pi_n \mid \pi_i \in vp(M_i)\}$$

Note that any vertical path through M consist of formulas occurring of M.

Similarly, we can define the set of vertical paths through an amplification (i.e. a through a quantifier-free formula). In order to do it, we display the amplification φ as a matrix, conjunctions in φ form columns and disjunctions form rows. Then we just use the set of vertical paths through such a matrix. For example, the formula $A \vee (B \wedge (C \vee (D \wedge E) \vee F))$ is represented by the matrix:

$$\begin{bmatrix} A & \begin{bmatrix} B \\ \begin{bmatrix} C & \begin{bmatrix} D \\ E \end{bmatrix} & F \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Its set of vertical paths consists of the following four sets: $\{A\}$, $\{B, C\}$, $\{B, D, E\}$ and $\{B, F\}$.

For our presentation of the connection method, we need a more general matrix construction:

Definition 3.4 *Let γ be a formula. The notion of a γ-*matrix *M is defined inductively as follows:*

1. *γ is a γ-matrix.*

2. *Let M be a γ-matrix containing a formula φ. Let φ have the form $\varphi_1 \wedge \ldots \wedge \varphi_n$, where $n > 1$. Then the figure obtained from M by replacing the occurrence of φ by*

$$\begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_n \end{bmatrix}$$

   is a γ-matrix.

3. *Let M be a γ-matrix containing a formula φ. Let φ have the form $\varphi_1 \vee \ldots \vee \varphi_n$, where $n > 1$. Then the figure obtained from M by replacing the occurrence of φ by*

$$\begin{bmatrix} \varphi_1 \cdots \varphi_n \end{bmatrix}$$

   is a γ-matrix.

4. *Let M be a γ-matrix containing a formula φ of the form $\forall x_1 \ldots x_n \psi$. Let $y_1, \ldots, y_n$ be variables not occurring in M. Then the figure obtained from M by replacing the occurrence of φ by*

$$\begin{bmatrix} \varphi \\ \psi[y_1/x_1, \ldots, y_n/x_n] \end{bmatrix}$$

   is a γ-matrix.

Example 3.1 *Consider, for example, the formula γ:*

$$\forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x)))$$

The following is a sequence of γ-matrices:

$$\forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x)))$$

$$\left[\begin{array}{c} \forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x))) \\ A(x_1) \vee (\forall y(B(x_1,y) \vee C(y)) \wedge D(x_1)) \end{array}\right]$$

$$\left[\begin{array}{c} \forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x))) \\ [A(x_1) \quad \forall y(B(x_1,y) \vee C(y)) \wedge D(x_1)] \end{array}\right]$$

$$\left[\begin{array}{c} \forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x))) \\ \left[A(x_1) \quad \left[\begin{array}{c} \forall y(B(x_1,y) \vee C(y)) \\ D(x_1) \end{array}\right]\right] \end{array}\right]$$

$$\left[\begin{array}{c} \forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x))) \\ \left[A(x_1) \quad \left[\begin{array}{c} \forall y(B(x_1,y) \vee C(y)) \\ B(x_1,y_1) \vee C(y_1) \\ D(x_1) \end{array}\right]\right] \end{array}\right]$$

$$\left[\begin{array}{c} \forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x))) \\ \left[A(x_1) \quad \left[\begin{array}{c} \forall y(B(x_1,y) \vee C(y)) \\ [B(x_1,y_1) \quad C(y_1)] \\ D(x_1) \end{array}\right]\right] \end{array}\right]$$

Our notions of γ-matrix and of a vertical path corresponds to the set of vertical paths through an amplification used in [And 81] with the following differences:

1. Our "horizontal submatrices"

$$\left[\ \varphi_1 \cdots \varphi_n\ \right]$$

are displayed in [And 81] using disjunctions

$$\left[\ \varphi_1 \vee \cdots \vee \varphi_n\ \right]$$

2. Our paths can contain arbitrary subformulas of γ while paths in [And 81] only contain literals.

This observation gives rise to the following

Theorem 3.1 *Let γ' be an amplification of the formula γ with the set of vertical paths Π. Then there is a γ-matrix with the set of vertical paths Π' such that for every vertical path $\pi \in \Pi$ there is a vertical path $\pi' \in \Pi'$ with $\pi \subseteq \pi'$.*

In our definition paths in a matrix contain formulas — an object difficult for algorithmic study. In order to define provability in a more handy formulation we introduce the notion of literals in a path.

Definition 3.5 *Let M be a γ-matrix and π be a vertical path in M. The set $lit(\pi)$ of* literals in π *is defined as*

$$\begin{array}{lll} lit(\pi) = \{L \mid & L \text{ is a literal and either } L \in \pi \text{ or} \\ & \exists \varphi \in \pi \text{ such that } \varphi \text{ has the form } \varphi_1 \wedge \ldots \wedge L \wedge \ldots \wedge \varphi_k\} \end{array}$$

The connection method is based on the following theorem which is a reformulation of the Herbrand theorem:

Theorem 3.2 *The formula γ is inconsistent iff there are a γ-matrix M and a substitution σ such that for every path π in $M\sigma$ the formula*

$$\exists \bigwedge lit(\pi)$$

is inconsistent.

This theorem says that all paths are simultaneously inconsistent in a very strong sense. It is noted in [Gal 92] that this inconsistency can be checked by the congruence closure algorithm [Koz 76, NO 80].

This theorem can be stated in a different form as

Theorem 3.3 *The formula γ is inconsistent iff there are a γ-matrix M, a substitution σ and a set S of sets of literals such that such that*

1. *For every $s \in S$ the formula*

   $$\exists \bigwedge s\sigma$$

   is inconsistent.

2. *For every vertical path π in M there is a set $s \in S$ such that $s \subseteq lit(\pi)$.*

In the literature on the connection method the following terminology is used (we adapt it for γ-matrices):

Definition 3.6 *Let M be a γ-matrix. A* mated set *for M is any subset of $lit(\pi)$, where $\pi \in vp(M)$. A* mating *for M is a pair $\langle S, \sigma \rangle$, where S is a set of mated sets such that for every $s \in S$ the formula*

$$\exists \bigwedge s\sigma$$

is inconsistent. Such a mating is called spanning *iff for every vertical path π in M there is a set $s \in S$ such that $s \subseteq lit(\pi)$.*

In this terminology Theorem 3.3 can be stated as

Theorem 3.4 *The formula γ is inconsistent iff there are a γ-matrix M and a spanning mating for M.*

This formulation of Theorem 3.2 leaves open the questions of finding a suitable matrix (or, in other terms, a suitable *quantifier duplication*) and finding a substitution σ. These two questions are not fully addressed in other papers on the connection method with equality.

Let us note a few problems that arise here:

1. There is no methodology for quantifier duplication, even though the importance of such a methodology has been stressed since the early paper on the connection method. It has been known that the outermost quantifier duplication is complete but generates unnecessarily large matrices.[1]

2. There is no method for finding a suitable substitution σ. Although an algorithm formulated in [GNRS 92] based on rigid E-unification is supposed to solve this problem, the presentation contains errors, which had been noted in [Gob 93, BP 94]. Even if a correct algorithm for simultaneous rigid E-unification could be found, the approach sketched in [GNRS 92, Gal 92] would still be unfeasible due to high non-determinism in the construction of all potential matings.

3. A solution to the substitution problem could be an *incremental* rigid unification algorithm, i.e. an algorithm which would allow to construct a solution for the simultaneous rigid E-unification problem from solutions to separate rigid E-unification problems. As it is noted in [Bec 94b], "A simultaneous E-unification problem can be solved by searching for *common specialization* to its components"[2]. It is far from obvious that such an incremental algorithm for rigid E-unification can be constructed. In theorem proving without equality, unification perfectly serves as a basis for incremental equality solution due to the notion of a most general unifier and lifting lemma. In (rigid) E-unification the role of most general unifiers is played by the so called *complete* sets of rigid E-unifiers. However there is no such incremental algorithm based on the notion of a complete set of (rigid) E-unifiers[3]. In [DV 94a] we proved that in

[1]We quote [GNRS 92]:

> As Andrews says in discussing this problem in the context of looking for nonequational p-acceptable matings, "One would like to develop a set of heuristics for duplicating quantifiers". However, it is beyond the scope of this paper to consider this issue in detail (as it was beyond the scope of Andrews and Bibel's papers) ...

We add to this quotation that this question was also beyond the scope of all papers we are aware of.

[2]This note was based on a technical report by J. Goubault that stated that simultaneous rigid E-unification is in Nexptime. However, his proof contained errors. It is not even known whether simultaneous rigid E-unification is decidable at all (U.Petermann, private communications).

[3]It is easy to construct an example when the sequential solution of complete sets of (rigid) E-unifiers $\langle E_1; \bar{s_1} = \bar{t_1}\rangle$ and $\langle E_2; \bar{s_2} = \bar{t_2}\rangle$ in any order does not find a simultaneous (rigid) E-unifier. Consider the following simultaneous (rigid) E-unification problem consisting of two subproblems:

$$\langle a = b; x = y, f(u,v) = f(c,d)\rangle \tag{3.1}$$
$$\langle c = d; u = v, f(x,y) = f(a,b)\rangle \tag{3.2}$$

There is a simultaneous rigid E-unifier for this system of rigid E-unification problems: the substitution $[a/x, b/y, c/u, d/v]$. Let us now try to solve the first equation before the second. Problem (3.1) has a complete set of rigid E-unifiers consisting of one substitution $[y/x, c/u, d/v]$. When we apply this solution to (3.2), we obtain a new rigid E-unification problem a slightly different situation with $SLDE^{+}$-resolution [Gal 89], incremental equation solution based on complete sets of E-unifiers is incomplete.

In this paper we propose a method which provides a solution for the problem of finding a spanning mating by an incremental procedure. Our method does not need rigid E-unification, it can take advantage of ordering strategies and it gives a hint of *how* to search for a suitable quantifier duplication. Our method is based on the specialization of Theorem 3.2 which states that mated sets can be chosen from a set of what we call *solution clauses* which will be defined in Section 4. Solution clauses are generated on the basis of the information from the goal γ using *basic superposition* [Deg 79, DV 86b, BGLS 92, NR 92a].

$$\langle c = d; c = d, f(y,y) = f(a,b)\rangle$$

which has no solution. By symmetry, the picture will be the same if we start with solving (3.2).

Section 4

The equality elimination method

Following [BGLS 92] we distinguish terms occurring in the original formula from terms introduced by substitution by using *closures:*

Definition 4.1 *A* closure *is a pair $C \cdot \sigma$, where C is a clause, σ a substitution. The application of a substitution ρ to such a closure is defined by $(C \cdot \sigma)\rho = C \cdot \sigma\rho$.*

The closure $C \cdot \sigma$ is a representation of the clause $C\sigma$. The clause C will often be identified with the closure $C \cdot \varepsilon$, where ε is the empty substitution.

Definition 4.2 *The set $Var(C \cdot \sigma)$ is defined as the set of all variables of $C\sigma$. Two closures $C_1 \cdot \sigma_1$ and $C_2 \cdot \sigma_2$ have* disjoint variables *iff $Var(C_1 \bigcup C_1\sigma_1 \bigcup Dom(\sigma_1)) \bigcap Var(C_2 \bigcup C_2\sigma_2 \bigcup Dom(\sigma_2)) = \emptyset$. An* instance of the closure *$C \cdot \rho$ is any closure $C \cdot \rho\sigma$. The instance is ground iff $Var(C\rho\sigma) = \emptyset$. Two closures are* variants *iff both are instances of each other.*

For the rest of this section we assume that γ is a closed rectified miniscoped formula in skolemized negation normal form. The goal is to refute γ, i.e. to prove its inconsistency. We shall identify subformulas of γ with their *occurrences* in γ. For example, in the formula γ of the form

$$A \wedge (\mathbf{A} \vee \mathbf{B})$$

the boldfaced $\mathbf{A}$ is considered as a subformula of $(\mathbf{A} \vee \mathbf{B})$, but the ordinary A is not a subformula of $(\mathbf{A} \vee \mathbf{B})$.

Definition 4.3 *An occurrence of a subformula φ of γ is called* disjunctive *iff it is an occurrence in a subformula of γ which is a disjunction $\varphi_1 \vee \ldots \vee \varphi \vee \ldots \vee \varphi_n$. A* disjunctive superformula *of φ is a superformula 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 not necessarily exist. For example, γ has no disjunctive superformula. Any formula having a disjunctive superformula has the unique least disjunctive superformula.

We can enumerate all disjunctive subformulas, for example in the order of their occurrences in γ. Thus we can inambiguously use "the nth disjunctive (sub)formula".

Definition 4.4 *Let $A_1, A_2, \ldots$ be a set of predicate symbols not occurring in γ. We say that $A_n(x_1, \ldots, x_m)$ is* the γ-name for a subformula φ *iff there is the subformula ψ such that*

1. *ψ is the n-th disjunctive subformula of γ.*

2. *ψ is the least disjunctive superformula of φ.*

3. *$x_1, \ldots, x_m$ are all variables of ψ in the order of their occurrences in ψ.*

γ-names not always exist. We shall use *the set of γ-names* of a subformula. The set of γ-names is either the empty set $\emptyset$ or a singleton $\{A_i(x_1, \ldots, x_k)\}$. Thus, we can consider sets of γ-names as clauses. Also note that different formulas may have the same γ-names.

The following example illustrates least disjunctive superformulas and γ-names:

Example 4.1 *Let γ be the formula*

$$\forall x(A(x) \vee (B(x) \wedge \forall y C(x,y)) \vee \forall z D(z)) \wedge E$$

All subformulas of γ and their γ-names are shown in the following table:

Subformula	*least disjunctive superformula*	*γ-name*
$\forall x(A(x) \vee (B(x) \wedge \forall y C(x,y)) \vee \forall z D(z)) \wedge E$		
$\forall x(A(x) \vee (B(x) \wedge \forall y C(x,y)) \vee \forall z D(z))$		
$A(x) \vee (B(x) \wedge \forall y C(x,y)) \vee \forall z D(z)$		
$A(x)$	$A(x)$	$A_1(x)$
$B(x) \wedge \forall y C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$A_2(x)$
$B(x)$	$B(x) \wedge \forall y C(x,y)$	$A_2(x)$
$\forall y C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$A_2(x)$
$C(x,y)$	$B(x) \wedge \forall y C(x,y)$	$A_2(x)$
$\forall z D(z)$	$\forall z D(z)$	A_3
$D(z)$	$\forall z D(z)$	A_3
E		

There are many advantages of using γ-names. First, by using atoms to name formulas we obtain simple data structures. Second, we can only restrict to a small subset of all subformulas of γ, and hence a small set of names. Third, formulas with no γ-names allow for unrestricted quantifier duplication and in fact give the same effect as mixed E-unification [Bec 94b].

Lemma 4.1 *Let φ be a subformula of γ. Then*

1. *If ψ is the least disjunctive superformula of φ then $\models \forall(\neg\varphi \supset \neg\psi)$.*

2. *If there is no disjunctive superformula of φ then $\models \forall(\neg\varphi \supset \neg\gamma)$.*

Proof. Immediate from the definition of disjunctive superformulas. □

This lemma partially explains the need in introducing least disjunctive superformulas. There are deterministic chains of inferences in sequent systems, where formulas with $\wedge$ or $\forall$ occur. For example,

$$\cfrac{\cfrac{\Gamma, \varphi \to \Delta}{\Gamma, \varphi \wedge \psi \to \Delta}}{\Gamma, \forall x(\varphi \wedge \psi) \to \Delta}$$

By restricting to disjunctive superformulas only, we eliminate these deterministic chains, making them in one step.

Below we shall reformulate Theorem 3.3 serving as a theoretical basis for the connection method in the language of names. Moreover, we shall show that the search for mated sets can be restricted to *solution clauses* defined below. Solution clauses do not contain equality. The solution clauses will be obtained from a set of *initial closures* using *basic superposition* developed in [Deg 79, Hul 80, BGLS 92, NR 92a].

Definition 4.5 Initial closures *are generated according to one of the three rules:*

1. *Whenever a literal $s \neq t$ occurs in γ and C is the set of γ-names for this occurrence of $s \neq t$, the closure $s \neq t, C \cdot \varepsilon$ is an initial closure.*
2. *Whenever $s = t$ occurs in γ and C is the set of γ-names for this occurrence of $s = t$, the closure $s = t, C \cdot \varepsilon$ is an initial closure.*
3. *Let literals $P(s_1, \ldots, s_n)$ and $\neg P(t_1, \ldots, t_n)$, where P is different from $=$, occur in γ and C_1, C_2 are their sets of γ-names. Let the substitution σ rename variables such that variables of $P(s_1, \ldots, s_n)\sigma, C_1\sigma$ and $P(t_1, \ldots, t_n), C_2$ are disjoint. Then the closure $s_1\sigma \neq t_1, \ldots, s_n\sigma \neq t_n, C_1\sigma, C_2 \cdot \varepsilon$ is an initial closure.*

The *basic superposition* rules derive closures from closures. They are defined as in [BGLS 92]. We always assume that premises of rules have disjoint variables. We also assume for the rest of the paper that $\succ$ is a fixed reduction ordering.

Definition 4.6 *The* basic superposition rules *are the following inference rules:*

Basic right superposition

$$\frac{(C, s = t) \cdot \sigma_1 \qquad (u[s'] = v, D) \cdot \sigma_2}{(u[t] = v, C, D) \cdot \sigma_1\sigma_2\rho}$$

where

1. *ρ is a most general unifier of $s\sigma_1$ and $s'\sigma_2$;*
2. *$t\sigma_1\rho \not\succeq s\sigma_1\rho$;*
3. *$v\sigma_2\rho \not\succeq u[s']\sigma_2\rho$;*
4. *s' is not a variable.*

Basic left superposition

$$\frac{(C, s = t) \cdot \sigma_1 \qquad (u[s'] \neq v, D) \cdot \sigma_2}{(u[t] \neq v, C, D) \cdot \sigma_1\sigma_2\rho}$$

with the same conditions as for basic right superposition.

When there are several disequations in the second premise closure of the basic left superposition, we shall use one more restriction: we only allow for superposition in the leftmost disequation.[1]

We shall introduce two more inference rules. The *equality solution rule* derives closures from closures and the *factoring rule* derives clauses from clauses:

Definition 4.7 *The* equality solution rule *is the following inference rule on closures:*

$$\frac{(C, s \neq t) \cdot \sigma}{C \cdot \sigma\rho}$$

where ρ is a most general unifier of $s\sigma, t\sigma$.

Definition 4.8 *The* factoring rule *is the following inference rule on clauses:*

$$\frac{\Gamma, A, B}{(\Gamma, A)\theta}$$

where θ is a most general unifier of A, B.

Finally, we can define *solution clauses:*

Definition 4.9 Solution clauses *are defined inductively as follows:*

1. *Let $C \cdot \sigma$ be a closure such that*

   (a) *C does not contain equality;*

   (b) *C can be obtained from initial clauses by using basic superposition rules.*

   Then $C\sigma$ is a solution clause.

2. *Let C be a solution clause and C' be obtained from C by the factoring rule. Then C' is a solution clause.*

Solution clauses in our method play the role of $\mathcal{E}$-complementary eq-connections which is confirmed by the following lemma:

Lemma 4.2 *Let ψ_j be disjunctive subformulas of γ, where $j \in \{1, \ldots, m\}$, B_j be the γ-names for ψ_j and σ_j be substitutions. If $B_1\sigma_1, \ldots, B_m\sigma_m$ is a solution clause then*

$$\models \forall(\neg\psi_1 \vee \ldots \vee \neg\psi_m \vee \neg\gamma)$$

Proof. According to Lemma 4.1, for every initial clause $s = t, B_i$ we have $\models \forall(s = t \vee \neg\psi_i)$ and for every initial clause $s = t$ we have $\models \forall(s = t \vee \neg\gamma)$.

Similarly, if an initial clause has the form $\bar{s} \neq \bar{t}, B_i, B_k$ then $\models \forall(\bar{s} \neq \bar{t} \vee \neg\psi_i \vee \psi_k)$, if an initial clause has the form $\bar{s} \neq \bar{t}, B_i$ then $\models \forall(s \neq t \vee \neg\psi_i \vee \neg\gamma)$ and if an initial clause has the form $\bar{s} \neq \bar{t}$ then $\models \forall(\bar{s} \neq \bar{t} \vee \neg\gamma)$. Since superposition rules and equality solution rules are logically sound, the claim of the lemma follows.

□

[1]According to our definitions, a clause is a set of literals, so the leftmost literal may sound as nonsense, but this restriction can easily be formalized using a variant of *locking* [Boy 71].

Let us note that in traditional literature on the connection method an eq-connection $(\bigwedge s_i = t_i \wedge s \neq t)\sigma$ is called $\mathcal{E}$-complementary iff $\exists(\bigwedge s_i = t_i \wedge s \neq t)\sigma$ is inconsistent. In our case inconsistent is the set $\exists(\bigwedge \psi_j \sigma_j \wedge \gamma)$. This addition of γ has the hidden effect of multiple use of universal equations in mixed E-unification [Bec 94a, Bec 94b].

Now we change the notion of γ-matrix introduced in Section 3 in order to use names instead of formulas.

Definition 4.10 *A* named γ-matrix *is defined by induction as follows:*

1. *The empty expression is a named γ-matrix.*
2. *Let the formula $\psi = \forall x_1 \ldots x_m(\varphi_1 \vee \ldots \vee \varphi_n)$, where $m \geq 0, n \geq 2$ be a subformula of γ such that the set of γ-names of ψ is empty. Let M be a named γ-matrix, A_i be the γ-names of φ_i and σ be a renaming such that $A_i\sigma$ has no common variables with M. Then the expression*
$$\left[\begin{array}{c} M \\ \left[\ A_1\sigma \cdots A_n\sigma\ \right] \end{array}\right]$$
is a named γ-matrix.
3. *Let the formula $\psi = \forall x_1 \ldots x_m(\varphi_1 \vee \ldots \vee \varphi_n)$, where $m \geq 0, n \geq 2$ be a subformula of γ such that the set of γ-names of ψ is the singleton set $\{A\}$. Let M be a named γ-matrix such that $A\rho$ occurs in M, A_i be the γ-names of φ_i and $\sigma = [y_1/x_1, \ldots, y_m/x_m]$ be a renaming such that the variables $y_1, \ldots, y_m$ do not occur in M. Then the expression obtained from M by the replacement of $A\rho$ by*
$$A\rho\left[\ \left[\ A_1\rho\sigma \cdots A_n\rho\sigma\ \right]\ \right]$$
is a named γ-matrix.

Note that named γ-matrices may only contain atoms having only variables as arguments.

We adapt Example 3.1 to named matrices. The formula

$$\forall x(A(x) \vee (\forall y(B(x,y) \vee C(y)) \wedge D(x)))$$

gives the following γ-names:

$$\begin{array}{rcl} A_1(x) & \equiv & A(x) \\ A_2(x) & \equiv & \forall y(B(x,y) \vee C(y)) \wedge D(x) \\ A_3(x,y) & \equiv & B(x,y) \\ A_4(y) & \equiv & C(y) \end{array}$$

The following is a sequence of named γ-matrices constructed in the same way as corresponding γ-matrices from Example 3.1:

$$[A_1(x_1) \quad A_2(x_1)]$$

$$\left[A_1(x_1) \quad \left[\begin{array}{c} A_2(x_1) \\ [A_3(x_1, y_1) \quad A_4(y_1)] \end{array}\right] \right]$$

Denote the latter matrix by M' and the last matrix of Example 3.1 by M. Note the following:

1. The matrix M' is much more compact than M.

2. Let Π be all paths through M. Take all paths $\pi \in \Pi$ and replace formulas in π by their γ-names. We obtain exactly all paths in M'.

Thus, named γ-matrices represent some information from γ-matrices in a more compact form.

Theorem 4.1 below is the main theorem of this paper. It characterizes inconsistency in terms of named paths through matrices. For its proof we need a new definition of *subset unification* [DV 94b]. A variant of subset unification has been used in [MRL 91] as θ-subsumption.

Definition 4.11 *A substitution σ is called a* subset unifier *of clauses D_1 and D_2 iff $D_1\sigma \subseteq D_2\sigma$. It is a* most general subset unifier *of D_1 and D_2 iff there is no other subset unifier σ' of D_1 and D_2 such that $\sigma' \leq \sigma$ and not $\sigma \leq \sigma'$.*

Some properties of subset unification are proved in [DV 94b]. In particular, the number of incomparable most general subset unifiers is in the worst case exponent of the length of clauses, and the subset unifiability problem is NP-complete.

Theorem 4.1 *The formula γ is inconsistent iff there are a named γ-matrix M and a substitution σ such that every path π in M contains an instance of a solution clause.*

Proof. The proof of this theorem will be based on a tableau characterization of equality elimination proved in [DV 94b]. The proof of that result is quite long and will not be reproduced here. An interested reader can refer to [DV 94b]. In order to formulate this general result we introduce (with some modifications) a calculus from that paper. Here we shall call it the calculus T. The modifications concern the fact that in this paper we define everything in terms of inconsistency while in [DV 94b] in terms of provability.

The provable objects of the calculus from [DV 94b] are *tableaux,* i.e. expressions of the form

$$C_1 \mid \ldots \mid C_n$$

where C_i are sets of atoms and $n \geq 0$. The initial object of the proof is the tableau consisting of one empty set of atoms. The inference rules are the following:

Branch expansion (Compare these rules with the definition of named γ-matrices)

1. Let the formula $\psi = \forall x_1 \ldots x_m(\varphi_1 \vee \ldots \vee \varphi_n)$, where $m \geq 0, n \geq 2$ be a subformula of γ such that the set of γ-names of ψ is empty. Let A_i be the γ-names of φ_i. Then from the tableau

$$C_1 \mid \ldots \mid C_j \mid \ldots \mid C_m$$

one can derive the tableau

$$C_1 \mid \ldots \mid A_1, C_j \mid \ldots \mid A_n, C_j \mid \ldots \mid C_m$$

where σ is a renaming such that $A_i\sigma$ has no common variables with the first tableau.

2. Let the formula $\psi = \forall x_1 \ldots x_m(\varphi_1 \vee \ldots \vee \varphi_n)$, where $m \geq 0, n \geq 2$ be a subformula of γ such that the set of γ-names of ψ is the singleton set $\{A\}$. Let A_i be the γ-names of φ_i. Then from the tableau

   $$C_1 \mid \ldots \mid A\rho, C_j \mid \ldots \mid C_m$$

   one can derive the tableau

   $$C_1 \mid \ldots \mid A_1\rho\sigma, A\rho, C_j \mid \ldots \mid A_n\rho\sigma, A\rho, C_j \mid \ldots \mid C_m$$

   where $\sigma = [y_1/x_1, \ldots, y_m/x_m]$ be a renaming such that the variables $y_1, \ldots, y_m$ do not occur in the first tableau.

Branch closure Let C be a solution clause and σ be a most general subset unifier of C and a clause C_1. Then from the tableau

$$C_1 \mid C_2 \mid \ldots \mid C_m$$

one can derive the tableau

$$C_2\sigma \mid \ldots \mid C_m\sigma$$

It has been proved in [DV 94b] that γ is inconsistent iff there is an inference of the empty table from the initial table. One can note that all rules in T-calculus are permutable, in particular all branch expansion rules can precede all branch closure rules.

Note the following two lemmas:

Lemma 4.3 *Let the tableau*

$$C_1 \mid C_2 \mid \ldots \mid C_m$$

can be obtained from the initial tableau by the branch expansion rules only. Then there is a named γ-matrix whose set of vertical paths is $\{C_1, \ldots, C_m\}$.

Lemma 4.4 *Let the empty tableau can be obtained from*

$$C_1 \mid C_2 \mid \ldots \mid C_m$$

by a sequence of branch closure rules with substitutions $\sigma_1, \ldots, \sigma_m$ and let σ be the composition $\sigma_1 \ldots \sigma_m$. Then every clause of the form $C_i\sigma$ contains an instance of a solution clause.

These two lemmas and permutability of rules of the T-calculus complete one part of the proof of Theorem 4.1: if the formula γ is inconsistent then there are a named γ-matrix M and a substitution σ such that every path π in M contains an instance of a solution clause.

The other part of Theorem 4.1 is proved similarly. □

The statement of Theorem 4.1 gives no hint on *how* to compute the matrix M and the substitution σ. This question will be considered in Section 5. Here we shall consider examples of the application of Theorem 4.1.

The first example is taken from [Bec 94a]:

Example 4.2 *The formula to be refuted is*

$$\forall x(g(x) = f(x) \vee x \neq a) \wedge \forall x(g(f(x)) = x) \wedge b = c \wedge \neg P(g(g(a)), b) \wedge P(a, c)$$

There are two disjunctive subformulas with the following names:

$$\begin{array}{rcl} A_1(x) & \equiv & g(x) = f(x) \\ A_2(x) & \equiv & x \neq a \end{array}$$

The initial closures are

1. $f(x) = g(x), A_1(x) \cdot \varepsilon$
2. $A_2(x) \wedge x \neq a \cdot \varepsilon$
3. $g(f(x)) = x \cdot \varepsilon$
4. $b = c \cdot \varepsilon$
5. $g(g(a)) \neq a, b \neq c \cdot \varepsilon$

The first four are generated by equalities in the goal, the last one by the pair $\neg P(g(g(a)), b), P(a, c)$.
As the order $\succ$ *we consider the recursive path ordering with* $f > g > b > c > a$. *Basic superposition and equality solution generate the following clauses:*

6. $g(g(x)) = x, A_1(y) \cdot [y/x]$ [right superposition from 1,3]
7. $A_2(a)$ [equality solution from 2]
8. $A_1(y), b \neq c \cdot [a/y]$ [left superposition + equality solution from 6,5]
9. $A_1(a)$ [left superposition + equality solution from 8,4]

The named γ*-matrix for this example is very simple:*

$$[A_1(x) \quad A_2(x)]$$

The corresponding substitution is $[a/x]$.

The next example is a modification of an example from [Mas 71]:

Example 4.3 *The formula to be refuted is*

$$\forall x((a = x \wedge G(f(f(x)))) \vee (\neg G(f(x)) \wedge x \neq b)) \wedge (\neg G(b) \vee \forall z \neg G(f(z)))$$

There are four disjunctive subformulas with the following names:

$$\begin{array}{rcl} A_1(x) & \equiv & a = x \wedge G(f(f(x))) \\ A_2(x) & \equiv & \neg G(f(x)) \wedge x \neq b \\ A_3 & \equiv & \neg G(b) \\ A_4 & \equiv & \forall z \neg G(f(z)) \end{array}$$

The initial closures are

1. $a = x, A_1(x) \cdot \varepsilon$
2. $x \neq b, A_2(x) \cdot \varepsilon$
3. $f(f(x)) \neq f(y), A_1(x), A_2(y) \cdot \varepsilon$
4. $b \neq f(f(x)), A_1(x), A_3 \cdot \varepsilon$
5. $f(f(x)) \neq f(z), A_1(x), A_4 \cdot \varepsilon$

The first two are generated by equalities in the goal, the last two by pairs $G, \neg G$.

As the order $\succ$ we consider the recursive path ordering with $f > b > a$. Basic superposition, equality solution and factoring generate the following clauses:

6. $A_2(b)$ [equality solution from 2]
7. $A_1(x), A_2(f(x)))$ [equality solution from 3]
8. $A_1(x), A_4$ [equality solution from 5]
9. $a \neq b, A_1(x), A_1(y), A_3 \cdot [f(f(y))/x]$ [left superposition from 1,4]
10. $A_1(f(f(y))), A_1(b), A_1(y), A_3$ [left superposition + equality solution from 1,9]
11. $A_1(f(f(b))), A_1(b), A_3$ [factoring from 10]

The named γ-matrix for this example is more complicated:

$$\left[\begin{array}{ll} [A_1(x) & A_2(x)] \\ [A_1(y) & A_2(y)] \\ [A_1(z) & A_2(z)] \\ [A_3 & A_4] \end{array}\right]$$

The corresponding substitution is $[b/x, f(b)/y, f(f(b))/z]$.

There are many optimizations possible in the equality elimination part of our method. The most powerful one is *subsumption* that we can apply in order to prevent generation of unnecessary clauses. Subsumption for the basic strategy has been explained in [Deg 82, BGLS 92]. On solution clauses, we can use ordinary subsumption, or even a stronger subsumption for sequent proofs proposed in [Vor 92].

It is also possible to use arbitrary simplification by equations obtained in the process of generating solution clauses. In general, such simplifications are incomplete for basic superposition. However in our case arguments of names only have non-basic positions. It is known that simplification for non-basic positions is complete [BGLS 92].

Thus, proofs by our method consist of two processes that can be run concurrently. One process generates solution clauses. The other process tries to construct a suitable matrix and a substitution. We shall not consider in details the process of the generation of solution clauses — it can be done by any standard techniques of theorem proving described in [Lus 92].

Section 5

Search strategies

There are several ways of organizing proof search by the equality elimination method. There is high flexibility due to the fact that the search for a suitable substitution can be incremental.

For a given matrix M, the search for a substitution σ can be made incrementally using *subset unification* defined in Section 4. Consider again Example 4.3. Assume that the following set of solution clauses is already constructed:

6. $A_2(b)$
7. $A_1(x), A_2(f(x)))$
8. $A_1(x), A_4$
11. $A_1(f(f(b))), A_1(b), A_3$

and we have the following named γ-matrix:

$$\begin{bmatrix} [A_1(x) & A_2(x)] \\ [A_1(y) & A_2(y)] \\ [A_1(z) & A_2(z)] \\ [A_3 & A_4] \end{bmatrix}$$

Subset-unification of the solution clause $A_2(b)$ with a path containing $A_2(x)$ gives the substitution $[b/x]$. Let us apply this substitution to the whole matrix:

$$\begin{bmatrix} [A_1(b) & A_2(b)] \\ [A_1(y) & A_2(y)] \\ [A_1(z) & A_2(z)] \\ [A_3 & A_4] \end{bmatrix}$$

Now all paths going through $A(b)$ contain an instance of a solution clause. It means that we can safely remove $A_2(b)$ from this matrix:

$$\begin{bmatrix} \multicolumn{2}{c}{A_1(b)} \\ [A_1(y) & A_2(y)] \\ [A_1(z) & A_2(z)] \\ [A_3 & A_4] \end{bmatrix}$$

We can subset-unify any path containing $A_1(b), A_2(y)$ with solution clause 7 obtaining the substitution $[f(b)/y]$. Apply this substitution to the whole matrix:

$$\begin{bmatrix} A_1(b) \\ [A_1(f(b)) \quad A_2(f(b))] \\ [A_1(z) \quad A_2(z)] \\ [A_3 \qquad\quad A_4] \end{bmatrix}$$

Now any path going through $A_2(f(b))$ contains an instance of a solution clause. Thus, we can remove $A_2(f(b))$ from the matrix:

$$\begin{bmatrix} A_1(b) \\ A_1(f(b)) \\ [A_1(z) \quad A_2(z)] \\ [A_3 \qquad\quad A_4] \end{bmatrix}$$

The rest of the search is similar. Subset-unification of solution clause 8 against any branch containing A_4 does not change the substitution. Removing A_4 from the matrix we get

$$\begin{bmatrix} A_1(b) \\ A_1(f(b)) \\ [A_1(z) \quad A_2(z)] \\ A_3 \end{bmatrix}$$

Now we can unify solution clause 11 against the path containing $A_1(z)$. The substitution $[f(f(b))/z]$. allows one to remove $A_1(f(b))$ from the matrix:

$$\begin{bmatrix} A_1(b) \\ A_1(f(b)) \\ A_2(f(f(b))) \\ A_3 \end{bmatrix}$$

The only path of this matrix contains an instance of solution clause 7. The search is completed.

In this example we constructed the suitable matrix in advance, and then started to search for a substitution using the information provided by solution clauses. Instead, we could systematically construct the same matrix by using the same information. Consider, how both the substitution and the matrix could have been constructed. Using solution clause 6, we could have started from a very simple matrix

$$[A_1(b) \quad A_2(b)]$$

in which one way is completely closed by this clause. Of course, this matrix can be replaced by

$$[A_1(b)]$$

We can note that solution clause 7 makes a path contain an instance of this clause if we extend this matrix as

$$\begin{bmatrix} A_1(b) \\ [A_1(y) \quad A_2(y)] \end{bmatrix}$$

and apply the substitution $[f(b)/y]$. The rest of the search is similar. Thus, the search for both a matrix and a substitution can be driven by information taken from solution clauses.

Bibliography

[And 76] P.B. Andrews. Refutations by matings. *IEEE Trans. Comput.*, 25:801–807, 1976.

[And 81] P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.

[And 86] P.B. Andrews. *An introduction to type theory: to truth through proof.* Academic Press, 1986.

[BDP 89] L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In H. Aït Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 2, pages 1–30. Academic Press, 1989.

[BGLS 92] 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.

[BP 94] G. Becher and U. Petermann. Rigid unification by completion and rigid paramodulation. In B. Nebel and Dreschler-Fischer L., 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.

[BH 92] 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.

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

[Bec 94b] 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.

[Ben 90] D. Benanav. Simultaneous paramodulation. In M.E. Stickel, editor, *Proc. 10th Int. Conf. on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 442–455, 1990.

[Bet 58] E.W. Beth. On machines which prove theorems. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 79–92. Springer Verlag, 1983. Originally appeared in 1958.

[Bet 59] E. Beth. *The Foundations of Mathematics.* North Holland, 1959.

[BS 75] W. Bibel and J. Schreiber. Proof search in a Gentzen-like system of first order logic. In E. Gelenbe and D. Potier, editors, *Proc. Int. Computing Symp.*, pages 205–212. North Holland, 1975.

[Bib 87] W. Bibel. *Automated theorem proving.* Vieweg Verlag, 1987. 2nd edition.

[Boy 90] T. Boy de la Tour. Minimizing the number of clauses by renaming. In M.E. Stickel, editor, *Proc. 10th CADE*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 558–572, 1990.

[Boy 71] R.S. Boyer. *Locking: a Restriction to Resolution.* PhD thesis, Univ. of Texas, Austin, 1971.

[Deg 79] A.I. Degtyarev. The strategy of monotone paramodulation (in Russian). In *Fifth Soviet All-Union Conference on Mathematical Logic*, page 39, Novosibirsk, 1979.

[Deg 82] A.I. Degtyarev. On the forms of inference in calculi with equality and paramodulation. In Yu.V. Kapitonova, editor, *Automation of Research in Mathematics*, pages 14–26. Institute of Cybernetics, Kiev, 1982.

[DV 86a] A.I. Degtyarev and A. Voronkov. Automated theorem proving I (in Russian). *Kibernetika*, (3), 1986. The journal is cover-to-cover translated to English.

[DV 86b] A.I. Degtyarev and A. Voronkov. Methods of handling equality in machine theorem proving (in Russian). *Kibernetika*, (3), 1986. The journal is cover-to-cover translated to English.

[DV 87] A.I. Degtyarev and A. Voronkov. Automated theorem proving II (in Russian). *Kibernetika*, (4), 1987. The journal is cover-to-cover translated to English.

[DV 94a] A. Degtyarev and A. Voronkov. A new procedural interpretation of Horn clauses with equality. UPMAIL Technical Report 89, Uppsala University, Computing Science Department, November 1994.

[DV 94b] A. Degtyarev and A. Voronkov. Equality elimination for semantic tableaux. UPMAIL Technical Report 90, Uppsala University, Computing Science Department, November 1994.

[DV 94c] A. Degtyarev and A. Voronkov. Equality elimination for the inverse method and extension procedures. UPMAIL Technical Report 92, Uppsala University, Computing Science Department, December 1994.

[DJ 90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 243–309. North Holland, Amsterdam, 1990.

[Ede 91] E. Eder. Consolution and its relation with resolution. In *Proc. IJCAI'91*, pages 132–136, 1991.

[Fit 90] M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer Verlag, New York, 1990.

[Gal 89] J.H. Gallier and S. Raatz. Extending SLD-resolution to equational horn clauses using E-unification. *Journal of Logic Programming*, 6(3):3–44, 1989.

[Gal 90] 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.

[GNRS 92] 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):3–44, 1992.

[Gal 92] J. Gallier. Unification procedures in automated deduction methods based on matings: a survey. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 439–485. Elsevier Science, 1992.

[Gob 93] 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.

[Her 30] J. Herbrand. *Logical Writings.* Harward University Press, 1972.

[Hul 80] J.M. Hullot. Canonical forms and unification. In *5th CADE*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, 1980.

[Kan 63] 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.

[Koz 76] D. Kozen. Complexity of finitely presented algebras. Technical Report TR 76-294, Cornell University, Ithaca, N.Y., 1976.

[Lus 92] E.L. Lusk. Controlling redundancy in large search spaces: Argonne-style theorem proving through the years. In A. Voronkov, editor, *Logic Programming and Automated Reasoning. International Conference LPAR'92.*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 96–106, St.Petersburg, Russia, July 1992.

[Lif 68] V. Lifschitz. Specialized forms of derivation in predicate calculus with equality and functional symbols (in Russian). In *Trudy MIAN*, volume 98, pages 5–25. 1968. English translation in: Proc. Steklov Institute of Math., AMS, Providence, 1971.

[Lif 89] V. Lifschitz. What is the inverse method? *Journal of Automated Reasoning*, 5(1):1–23, 1989.

[Mas 71] S.Yu. Maslov. The generalization of the inverse method to predicate calculus with equality (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 20:80–96, 1971. English translation in: Journal of Soviet Mathematics 1, no. 1.

[MM 83] S.Yu. Maslov and G. Mints. The proof-search theory and the inverse method (in Russian). In Mints G., editor, *Mathematical Logic and Automatic Theorem Proving*, pages 291–314. Nauka, Moscow, 1983.

[Mas 87] S.Yu. Maslov. *Theory of Deductive Systems and its Applications.* MIT Press, 1987.

[MRL 91] J. Minker, A. Rajasekar, and J. Lobo. Theory of disjunctive logic programs. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic. Essays in Honor of Alan Robinson.*, pages 613–639. The MIT Press, Cambridge, Massachusetts, 1991.

[Min 66] G.E. Mints. The Herbrand theorem for the predicate calculus with equality and function symbols. *Soviet Mathematical Doklady*, (2):273–275, 1969.

[Min 67a] G.E. Mints. Herbrand theorem (in Russian). In A.V. Idelson and G.E. Mints, editors, *Mathematical Theory of Logical Inference*, pages 273–275. Nauka, Moscow, 1967.

[NO 80] G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. *Journal of the Association for Computing Machinery*, 27(2):356–364, 1980.

[NR 92a] 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.

[Ore 69] V.P. Orevkov. On non-increasing rule applications for equality (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 16:152–156, 1969. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 16, Consultants Bureau, NY-London, 1971.

[Pra 60] 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.

[Pra 69] D. Prawitz. Advances and problems in mechanical proof procedures. In Meltzer and Michie, editors, *Machine Intelligence*, volume 4, pages 73–89. Edinburgh University Press, Edinburgh, 1969.

[Pra 70] D. Prawitz. A proof procedure with matrix reduction. In M. Laudet, L. Lacombe, L. Nolin, and M Schutzenberger, editors, *Symp. on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 207–213. Springer Verlag, 1970.

[Pet 93] U. Petermann. A framework for integrating equality reasoning into extension procedure. In *Proc. Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 195–207, 1993.

[Rus 91] M. Rusinowitch. Theorem proving with resolution and superposition: an extension of the Knuth and Bendix completion procedure as a complete set of inference rules. *Journal of Symbolic Computations*, 11:21–49, 1991.

[Vor 92] A. Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 648–662, Saratoga Springs, NY, USA, June 1992. Springer Verlag.