General Connections via Equality Elimination

Anatoli Degtyarev* Andrei Voronkov†‡

Abstract

The *equality elimination method* for 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 with selection function, basic subsumption and simplification. In addition, we present a new treatment of the connection method via *named matrices* which gives a number of advantages.

1 Introduction

We propose an approach to adding equality to extension proof methods which are known under two different names *connection method* [Bi87] and the *method of matings* [An81, GNRS92]. Both methods express the same idea going back to the fundamental Herbrand theorem. According to this idea, the proof-search can be considered as 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 [Pr60].

The main problem addressed by Prawitz when he developed matrix methods was the following. Instead of generating instances of a quantifier-free formula $M(\bar{x})$ he proposed to search for a substitution σ for $\bar{x}_1, \ldots, \bar{x}_n$ such that every path in $(M(\bar{x}_1) \wedge \ldots \wedge M(\bar{x}_n))\sigma$ becomes complementary.

A generalization of the ideas of Prawitz to the predicate calculus with equality has been proposed by Bibel [Bi87] and Gallier et al. [GNRS92]. In this generalization complementary literals are replaced by eq-connections instantiated by eq-unifiers [Bi87] or mated sets instantiated by rigid E-unifiers [GNRS92] which is the same despite different terminology. For some time, there was a hope that rigid E-unification can be used for these methods. However, recently it was proved that the simultaneous rigid E-unification is undecidable [DeVo95b] and that even its quite elementary special cases are very complicated [DeMaVo95].

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. We split equality reasoning and matrix extension 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. Our approach

*Supported by a grant from the Swedish Institute

†Supported by a TFR grant

‡Anatoli Degtyarev, Andrei Voronkov, Computing Science Department, Uppsala University, Box 311, S-751 05 Uppsala, Sweden, email {anatoli,voronkov}@csd.uu.se

to adding equality to the connection method has been sketched in [DeVo94c] following techniques developed in [DeVo94a, DeVo94b]. Since we eliminate equality in the first part of our method, the method is called *equality elimination.*

Proofs are omitted from this article. They can be found in an extended version of this paper available as a technical report [DeVo95a]. The report can be obtained by WWW at `http://www.csd.uu.se/~thomasl/reports.html` and by anonymous ftp at `ftp.csd.uu.se` in the directory `pub/papers/reports`.

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 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. 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 $[x_1\theta/x_1, \ldots, x_n\theta/x_n]$. For any expression E, $Var(E)$ denotes the set of all variables of E. For a substitution σ of the form $[t_1/x_1, \ldots, t_n/x_n]$, $Var(\sigma)$ is the set $Var(x_1, \ldots, x_n, t_1, \ldots, t_n)$.

Notation $\vdash$ stands for the provability *in classical first order logic with equality.*

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 finite 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 $s \neq t$.

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.

Formulas are defined in the usual way, we allow for conjunction and disjunction to have any finite number of arguments. 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. 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.

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 [An81].

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* defined inductively as follows:

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

$$[M_1 \ldots M_n] \quad \text{and} \quad \begin{bmatrix} M_1 \\ \vdots \\ M_n \end{bmatrix}$$

If M is formula matrix having a row inside a row, it will be represented as a single row, for example

$$[\varphi_1 \ldots [\psi_1 \ldots \psi_m] \ldots \varphi_n] \quad \text{as} \quad [\varphi_1 \ldots \psi_1 \ldots \psi_m \ldots \varphi_n]$$

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

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:

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

$$vp([\ M_1 \cdots M_n\ ]) = vp(M_1) \bigcup \ldots \bigcup vp(M_n)$$

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:

$$\left[A \quad \begin{bmatrix} B \\ \left[C \quad \begin{bmatrix} D \\ E \end{bmatrix} \quad F \right] \end{bmatrix} \right]$$

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 special kind of matrices, called *γ-matrices:*

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

$$\begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_n \end{bmatrix} \quad \text{(respectively} \quad [\ \varphi_1 \cdots \varphi_n\ ]\ )$$

is a γ-matrix.

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

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

is a γ-matrix.

Example 3.1 *Consider the following 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:

$$\forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \qquad \begin{bmatrix} \forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \\ A(x_1) \vee (\forall y (B(x_1,y) \vee C(y)) \wedge D(x_1)) \end{bmatrix}$$

$$\begin{bmatrix} \forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \\ [A(x_1) \quad \forall y (B(x_1,y) \vee C(y)) \wedge D(x_1)] \end{bmatrix} \qquad \begin{bmatrix} \forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \\ \begin{bmatrix} A(x_1) & \begin{bmatrix} \forall y (B(x_1,y) \vee C(y)) \\ D(x_1) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \\ \begin{bmatrix} A(x_1) & \begin{bmatrix} \forall y (B(x_1,y) \vee C(y)) \\ B(x_1,y_1) \vee C(y_1) \\ D(x_1) \end{bmatrix} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \forall x (A(x) \vee (\forall y (B(x,y) \vee C(y)) \wedge D(x))) \\ \begin{bmatrix} A(x_1) & \begin{bmatrix} \forall y (B(x_1,y) \vee C(y)) \\ [B(x_1,y_1) \quad C(y_1)] \\ D(x_1) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Our notions of a γ-matrix and of a vertical path correspond to the set of vertical paths through an amplification used in [An81] with the following differences:

1. Our "horizontal submatrices" are displayed in [An81] using disjunctions, for example

$$\begin{bmatrix} \varphi_1 \cdots \varphi_n \end{bmatrix} \quad \text{as} \quad \begin{bmatrix} \varphi_1 \vee \cdots \vee \varphi_n \end{bmatrix}$$

2. Our paths can contain arbitrary subformulas of γ while paths in [An81] 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.

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

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

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 $\exists \bigwedge_{L \in lit(\pi)} L$ is inconsistent.*

This theorem says that all paths are simultaneously inconsistent in a very strong sense. It is noted in [GNRS92] that this inconsistency can be checked by the congruence closure algorithm.

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*

1. *For every $s \in S$ the formula $\exists \bigwedge_{L \in s} L\sigma$ is inconsistent.*

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

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 $\exists \bigwedge_{L \in s} L\sigma$ 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 theorem 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 properly addressed in other papers on the connection method with equality.

In this paper we propose a method which provides a solution for the problem of finding a spanning mating by an incremental procedure, i.e. a procedure that closes paths one by one. 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 3.4 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* [DeVo86, BGLS92, NiRu92] with selection function.

4 The equality elimination method

Following [BGLS92] we distinguish terms occurring in the original formula from terms introduced by substitution by using closures. 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 set $Var(C \cdot \sigma)$ is defined as $Var(C) \bigcup Var(\sigma)$. Two closures $\mathcal{C}_1, \mathcal{C}_2$ have *disjoint variables* iff $Var(\mathcal{C}_1) \bigcap Var(\mathcal{C}_2) = \emptyset$. Two closures $C_1 \cdot \sigma_1$ and $C_2 \cdot \sigma_2$ are *variants* iff C_1 is a variant of C_2 and $C_1\sigma_1$ is a variant of $C_2\sigma_2$.

In the following we identify closures which are variants of each other. One can note that every closure has a variant $C \cdot \sigma$ such that $Dom(\sigma) \subseteq Var(C)$. 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.

For the rest of this section we assume that γ is a closed rectified miniscoped formula in the Skolem 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 $A \wedge (A \vee B)$ the second occurrence of A is considered a subformula of $(A \vee B)$, but the first occurrence of A is not.

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.

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".

Let $A_1, A_2, \ldots$ be a set of predicate symbols not occurring in γ. Then $A_n(x_1, \ldots, x_m)$ 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. For more examples see [DeVo95a]. For example, in Example 3.1 the formulas $\forall y(B(x,y) \vee C(y)) \wedge D(x)$, $\forall y(B(x,y) \vee C(y))$, $B(x,y) \vee C(y)$ and $D(x)$ have the same γ-name.

There are many advantages of using γ-names. First, by using atoms to name formulas we obtain simple data structures. Second, we can 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 [BeHä92]. The essence of least disjunctive superformulas can partially be explained by the following facts:

1. If ψ is the least disjunctive superformula of φ then $\vdash \forall(\neg\varphi \supset \neg\psi)$.
2. If there is no disjunctive superformula of φ then $\vdash \forall(\neg\varphi \supset \neg\gamma)$.

Another explanation of least disjunctive superformulas is that their use removes deterministic chains of inferences in sequent systems (see [Vo92, DeVo95a]).

Below we shall reformulate Theorem 3.3 serving as a theoretical basis for the connection method in the language of γ-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 [BGLS92, NiRu92].

Definition 4.1 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 set of all initial clauses is denoted by $Init_\gamma$.

The *basic superposition* rules derive closures from closures. They are defined as in [BGLS92]. 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 4.2 *The* basic (right and left) superposition rules *are the following inference rules on closures:*

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

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.*

For basic left superposition, we also require that $u[s'] \neq v$ *be the leftmost disequation in the second premise (although clauses are defined as sets of literals, the notion of the leftmost literal can be formalized using a* selection function*).*

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

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

where ρ *is a most general unifier of* $s\sigma, t\sigma$ *and* $s \neq t$ *is a leftmost disequation in the premise.*

The equality elimination calculus $\mathcal{EE}_\gamma$ *is the calculus of closures whose axioms are* $Init_\gamma$ *and whose inference rules are basic superposition and equality solution.*

Finally, we can define *solution clauses:*

Definition 4.3 *The set* Sol_γ *of* solution clauses *is the set of all clauses* $C\theta$ *such that* $C \cdot \theta$ *is provable in* $\mathcal{EE}_\gamma$ *and* C *does not contain equality.*

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

Lemma 4.1 *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*

$$\vdash \forall(\neg\psi_1\sigma_1 \vee \ldots \vee \neg\psi_m\sigma_m \vee \neg\gamma)$$

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 a multiple use of universal equations in mixed E-unification [BeHä92].

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

Definition 4.4 *A* named γ-matrix *is defined by induction as follows:*

1. *The empty expression is a named γ-matrix.*
2. *Let the formula $\psi = \forall x_1 \dots x_m(\varphi_1 \vee \dots \vee \varphi_n)$, where $m \geq 1, n \geq 2$ 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 $\psi = \forall x_1 \dots x_m(\varphi_1 \vee \dots \vee \varphi_n)$, where $m \geq 1, n \geq 2$ 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 $\sigma = [y_1/x_1, \dots, y_m/x_m]$ be a renaming such that the variables $y_1, \dots, y_m$ do not occur in M. Then the expression obtained from M by the replacement of $A\rho$ by*

$$\left[\begin{array}{c} A\rho \\ \left[\; A_1\rho\sigma \cdots A_n\rho\sigma \;\right] \end{array}\right]$$

is a named γ-matrix.

Note that named γ-matrices may only contain atoms having 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:

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

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

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

The first two matrices of Example 3.1 correspond to the empty named γ-matrix, the next three matrices of Example 3.1 correspond to the first named γ-matrix here, and the last matrix of Example 3.1 correspond to the last named γ-matrix here. The corresponding named γ-matrices are obtained by the replacement of formulas by their γ-names, renaming variables when necessary. The path $\{(\forall y(B(x,y) \vee C(y)) \wedge D(x)), \forall y(B(x_1,y) \vee C(y)), C(y_1), D(x_1)\}$ in the last matrix of Example 3.1 corresponds to the path $\{A_2(x_1), A_4(y_1)\}$ in the named γ-matrix.

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

The theoretical basis of the equality elimination method is provided by

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 $M\sigma$ contains an instance of a solution clause.*

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 an example of the application of Theorem 4.1.

Example 4.1 *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:

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

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 lexicographic path ordering with $f > b > a$. Basic superposition, and equality solution generate the following closures:

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

This closures give the following set of solution clauses:

s1. $A_2(b)$
s2. $A_1(x), A_2(f(x)))$
s3. $A_1(x), A_4$
s4. $A_1(b), A_1(f(f(y))), A_1(y), A_3$

Consider the following named γ-matrix M

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

and the substitution $\sigma = [b/x, f(b)/y, f(f(b))/z]$. They satisfy the claim of Theorem 4.1.

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 [BGLS92]. On solution clauses, we can use ordinary subsumption, or even a stronger subsumption for sequent proofs proposed in [Vo92].

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 [BGLS92].

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 detail the process of the generation of solution clauses — it can be done by any standard techniques of theorem proving.

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* [DeVo94b]. A variant of subset unification has been used in [MiRaLo91] as θ-subsumption. A substitution σ is called a *subset unifier* of a clause D_1 against a clause D_2 iff $D_1\sigma \subseteq D_2\sigma$. It is a *minimal subset unifier* of D_1 against D_2 iff there is no other subset unifier σ' of D_1 against D_2 such that $\sigma' \leq \sigma$ and not $\sigma \leq \sigma'$.

Some properties of subset unification are proved in [DeVo94b]. In particular, the number of incomparable minimal subset unifiers is in the worst case exponent of the length of clauses, and the subset unifiability problem is NP-complete.

Consider again Example 4.1. Assume that the solution clauses *s1*–*s4* from Example 4.1 and the following named γ-matrix are already obtained:

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

Subset-unification of the solution clause $A_2(b)$ against any 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 solution clause *s2* against any path containing $A_1(b), A_2(y)$ 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 *s3* 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 *s4* 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 *s2*. 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 *s1*, we could have started from a very simple matrix $[A_1(b) \quad A_2(b)]$ in which one path is completely closed by this clause. Of course, this matrix can be replaced by $[A_1(b)]$ We can note that solution clause *s2* 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.

References

[An81] P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.

[BGLS92] 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.

[BeHä92] 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.

[Bi87] W. Bibel. *Automated theorem proving.* Vieweg Verlag, 1987. 2nd edition.

[DeVo86] A. 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.

[DeVo94a] 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.

[DeVo94b] A. Degtyarev and A. Voronkov. Equality elimination for semantic tableaux. UPMAIL Technical Report 90, Uppsala University, Computing Science Department, December 1994.

[DeVo94c] 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.

[DeVo95a] A. Degtyarev and A. Voronkov. General connections via equality elimination. UPMAIL Technical Report 93, Uppsala University, Computing Science Department, January 1995.

[DeVo95b] A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, Computing Science Department, May 1995.

[DeMaVo95] A. Degtyarev, Yu. Matijasevich, and A. Voronkov. Simultaneous rigid E-unification is not so simple. UPMAIL Technical Report 104, Uppsala University, Computing Science Department, April 1995.

[GNRS92] 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.

[MiRaLo91] 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.

[NiRu92] 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.

[Pr60] 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.

[Vo92] A.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.