AN EXTENSION OF UNIFICATION TO SUBSTITUTIONS WITH AN APPLICATION TO AUTOMATIC TIIEOREM PROVING

Jophien van Vaalen
Mathematisch Centrum, Amsterdam, The Netherlands

Abstract

The notion of unifiability is extended to substitutions. Theorems concerning this notion are derived together with an algorithm for computing the most general unifier of a set of substitutions. Especially fruitfull is the application in the case of the and-or tree approach to theorem proving where the subgoals are not independent but contain the same variables. Here the ultimate solution is shown to be the most general instance of the solutions to the individual subproblems. Another application concerns connection graphs where the arcs are substitutions and new arcs can be generated from old arcs.

1. Introduction

In [5] ROBINSON introduced resolution theorem proving. He defines the notion of unifiability of expressions as follows: two expressions are called unifiable if we can find a substitution such that by applying that substitution to the expressions the expressions become equal. In §2 the definitions, the *unification algorithm* and the unification theorem are reviewed. In §3 we extend the notion of unifiability to substitutions. Two substitutions are called unifiable if we can find a substitution such that if we can compute the composition of the original substitutions with this substitution the resulting substitutions become equal in the set theoretical sense. Some theorems connecting unifiability of expressions and unifiability of substitutions are presented. Furthermore, an algorithm for computing the unifier of substitutions is described. Applications of the notion of unifiability of substitutions to resolution theorem proving problems such as and-or tree representation, connection graphs [3,4] and the structure sharing way of representing clauses [1] are dealt with in §4.

2. The Unification of Expressions

Definitions

The following definitions are taken from [5]:

A *term* is either a variable or a string of symbols consisting of a function symbol of degree $n \geq 0$ followed by n terms.

An *expression* is either a term or a string of symbols consisting of a predicate symbol of degree $n > 0$ followed by n terms.

A *substitution component* is any construct of the form $v \rightarrow t$ where v is a variable and t is a term different from v; v is called the *variable* of the substitution component $v \rightarrow t$ and t is called the *term* (Hence $v \rightarrow v$ is not a substitution component for any variable v).

A *substitution* is a finite (possibly empty) set of substitution components with distinct left-hand sides. Therefore in a substitution $\{v_1 \rightarrow t_1, v_2 \rightarrow t_2, \ldots, v_k \rightarrow t_k\}$ the order of the constituent components is immaterial. Substitutions are denoted by lower case greek letters, ϵ is used to denote the empty substitution.

In the following variables will be denoted by strings commencing with v,w,x,y,z; constants or nullary functions by a,b,c,d,e and strings commencing with f,g,h denote n-ary functions with $n > 1$.

Let E be a finite string of symbols and $\theta = \{v_i \rightarrow t_i\}_{i=1}^{k}$ be a substitution. The *instantiation* of E by θ is the operation of simultaneously replacing each occurrence of v_i in E by an occurrence of the term t_i for all i, $1 \leq i \leq k$. The resulting string $E\theta$ is called the *instance* of E by θ. If C is a set of strings and θ a substitution then the instance of C by θ is defined by $C\theta = \{E\theta \mid E \in C\}$.

let $\theta = \{v_i \rightarrow t_i\}_{i=1}^{k}$ and λ be substitutions. The *composition* of θ and λ is the substitution $\theta' \cup \lambda'$ where θ' consists of the components $v_i \rightarrow t_i\lambda$ such that $t_i\lambda \neq v_i$, $1 \leq i \leq k$ and λ' consists of the substitution components of λ whose left-hand sides are not left-hand sides of θ.

It is easily verified that $\epsilon\theta = \theta\epsilon = \theta$ for all substitutions θ. Similarly, composition of substitutions is associative, i.e. $(\theta\lambda)\mu = \theta(\lambda\mu)$ and therefore we may omit the parentheses.
(Hint: for any expression E and a composition of substitutions $\sigma = \theta\lambda$; $E\sigma$ is the string $E\theta\lambda$, that is, the instance of $E\theta$ by λ).

We now summarize some properties of the composition of substitutions which we need in the sequel.
(E is an expression and σ,μ,λ are substitutions.)

1. $(E\sigma)\lambda = E(\sigma\lambda)$ for all strings E and all substitutions σ,λ.
2. $\sigma = \lambda$ iff $E\sigma = E\lambda$ for all strings E.
3. $(\sigma\lambda)\mu = \sigma(\lambda\mu)$ for all substitutions σ,λ,μ.
4. $(A \cup B)\lambda = A\lambda \cup B\lambda$ for all sets of strings A,B and all substitutions λ. Note that the composition of substitutions is not commutative $\sigma\lambda \neq \lambda\sigma$.

The *disagreement set* of a non-empty set of expressions A consists of the set of well-formed subexpressions extracted from the expressions in A by deleting the initial parts which are common to all expressions in A.

UNIFIER. Let A be a set of expressions and θ be a substitution; θ is said to *unify* A (or to be a *unifier* of A) if $A\theta$ is a singleton. A set of expressions which has a unifier is said to be unifiable.

MOST GENERAL UNIFIER. A unifier θ of a set of expressions A is called a *most general unifier* of A if for all unifiers σ of A there exists a substitution λ such that $\sigma = \theta\lambda$. If A has a unifier then there exists always a most general unifier θ; $A\theta$ then is called the most general instance of A. (The most general unifier of A is unique up to isomorphisms.)

The Algorithm and the Unification Theorem

The unification algorithm and theorem are quoted from the ROBINSON [5]. The *Unification algorithm* computes the most general unifier of A, where A is a finite non-empty set of expressions:
Step 1 : $\sigma_0 := \epsilon$ (the empty substitution); k := 0;

*) This work was supported by the Mathematisch Centrum under number IW 24/74.

Step 11 : If Ao_k is a singleton stop: o_k. is the most general unifier of A.
Step III: Compute the disagreement set of Ao_k; order this set such that the variables are in front, the rest of the ordering being immaterial.
If v_k and u_k are the two earliest elements of the disagreement set and v_k is a variable that does not occur in u_k then o_{k+1} := o_k. $\{v_k \to u_k\}$; k := k+1 ; go to step 2 otherwise stop then A is not unifiable.

UNIFICATION THEOREM, *let* A *be a non-empty finite eet of expressions. If* A *is unifiable then there exists a most general unifier* o *of* A *which is computed by the unification algorithm.*

During the computation of a deduction in automatic theorem proving a great amount of unification computations has to be performed. The implementation of the unification algorithm can be done much more efficient than a straightaway implementation; See 16, 2 and 7 J.

3. An Extension of Unification to Substitutions

Definitions

Substitution unifier. Let o = I^fJ;*i-i be a set of substitutions and o a substitution; o is said to unify 0 or to be a unifier of o if e-o tt ri+i° ror all I< i < k. A set of substitutions which has a unifier is said to be *unifiable* or to be *compatible.* (By equality of substitutions is meant the set theoretical equality of the ordered pairs).

Instantiation, If Uj and o_2 are substitutions then the composition O_fB_2 *s called an instance of 0..

Most general unifier. A unifier o of a set of substitutions 0 = IO^t^.j is called a most general unifier if for all unifiers i of 0 there exists a substitution A such that T= nA; the instance fljo is called the most general instance of 0.

Normal form, A substitution 6 ≃ ^vi^ci^i«I ^s *-n normal form if for all i,j: 1 ≤ i, j *: k, v_i does not occur in t;.

Note

1. By definition it holds that all vj's in a substitution are different.
2. The most general unifier of expressions computed by the unification algorithm as quoted in 2.2 *is* in normal form.
3. The most general instance of a set of substitutions *C0-1^*»i is sometimes also denoted by 61*62*-•.*0^; this operation is commutative and associative.

A substitution component v * t is called *circular* if v occurs in t.

A set of substitutions 0 « {0-}^^. is called *contradictory* if there are two substitutions 6|, 6 in G such that there is a substitution component vj -* tj. in 6j and a substitution component vj * tj_m in 0_{ffl} such that VJ - vj and the terms $t_{J\cdot I}$ and t_{Jm} are not unifiable.

Algorithms and Theorems

We now give the algorithm to compute the roost general unifier o of a set of substitutions $I0.)^n_{1\ i*I}$.

Wo assume that in each substitution of a set of substitutions the substitution-components are ordered according to a fixed enumeration of all variables and there are no circular components.

Substitution Unification Algorithm.

Step I : k:= 0; o_0 := e.
Step II : Compute $\{\theta_i o_k\}^n_{i=1}$; delete components of the form v → v from $\{\theta_i o_k\}^n_{i=1}$. If there is a circular substitution component in one of the substitutions $\theta_i o_k$ then there is no unifier and the algorithm halts. If $\{\theta_i o_k\}^n_{i=1}$ is a singleton then o_k is a most general unifier and the algorithm halts. In all other cases goto step 3.
Step III: Take the first substitution component from all $\theta_i o_k$; if they are all equal then take next ones otherwise take from this set those substitution components where the variables are first in the above mentioned enumeration on the variables, the resulting set is, say, $\{v \to t_j\}^m_{j=1}$. We now distinguish three cases.

Case I : m = 1. Then: k := k+1; o_k := o_{k-1}. $\{v \to t_1\}$; goto step 2.
Case II : The set contains substitution components that come from some but not all of the sets $\{\theta_i o_k\}^n_{i=1}$. If the set $\{v \to t_j\}^m_{j=1}$ is contradictory, then stop: not unifiable otherwise let λ be the most general unifier of $\{t_j\}^m_{j=1}$ then k := k+1; o_k = o_{k-1}. $\{v \to t_1\}$ λ; goto step 2.
Case III: The set contains substitution components that come from all $\theta_i o_k$. If the set $\{v \to t_j\}^m_{j=1}$ is contradictory then halt: not unifiable otherwise let λ be the most general unifier of $\{t_j\}^m_{j=1}$ then k := k+1; o_k := o_{k-1}.λ; goto step 2.

Examples

1. θ_1 = {x→t, y→a}; θ_2 = {x→f(y), z→f(y)};
   θ_3 = {x→z, t→z, s→b}. Ordering x,y,z,t,s.
   k = 0: o_0 = e.
   k = 1: x → t, x → f(y), x → z.
   compute m.g.u. of {t,f(y),z}: t → f(y), z → f(y)
   o_1 = {t→f(y), z→f(y)}
   $\theta_1 o_1$ = {x→f(y), y→a, z→f(y), t→f(y)}
   $\theta_2 o_1$ = {x→f(y), z→f(y), t→f(y)}
   $\theta_3 o_1$ = {x→f(y), z→f(y), t→f(y), s→b}
   o_2 = o_1.{y→a} = {t→f(a), z→f(a), y→a}
   $\theta_1 o_2$ = $\theta_2 o_2$ = {x→f(a), y→a, z→f(a), t→f(a)}
   $\theta_3 o_2$ = {x→f(a), y→a, z→f(a), t→f(a), s→b}
   o_3 = o_2.{s→b} = {t→f(a), z→f(a), y→a, s→b}
   o_3 is the most general unifier of $\{\theta_1, \theta_2, \theta_3\}$

2. θ_1 = {x→y, y→f(a)}; θ_2 = {x→f(a), y→f(a)}
   o_1 = {y→f(a)} is the most general unifier.

3. θ_1 = {x→y, y→f(a)}; θ_2 = {x→b, y→f(a)}
   o_1 = {y→b} is the most general unifier and θ_2 is is the most general instance.

4. θ_1 = {x→y}; θ_2 = {x→b, y→f(a)}
   o_1 = {y→b} $\theta_1 o_1$ = {x→b, y→b}; $\theta_2 o_1$ = {x→b, y→f(a)} because {y→b} and {y→f(a)} are contradictory, θ_1 and θ_2 are not unifiable. (since b is not variable)

Remarks

1. The most general unifier of a set of substitutions as computed by the above algorithm is always in normal form.
2. The most general instance of a set of substitutions is not always a unifier of the substitutions: example 3: θ_2 = {x→b, y→f(a)} is the most

general instance but $\theta_1\theta_2 = \{x \to f(a), y \to f(a)\} \neq \theta_2\theta_2$.

To compute the most general unifier of expressions we can also use the algorithm for computing the m.g.u. of substitutions if we consider the disagreement set to be substitutions and look at the class of disagreement sets. The class of disagreement sets of a nonempty set of expressions A consists of the class of sets of well-formed subexpressions from the expressions in A by deleting the parts which are common to all expressions in A (scanning starts from the left).

Example

$\{R(x,y), R(g(f(x,y)), h(b)), R(a,t)\}$
the disagreement sets are: $\{x,g(f(x,y)),a\}$ **and** $\{y,h(b),t\}$.

Algorithm 2 to make a set of substitutions out of a disagreement set of a set ofexpressions if possible.

Step I : If the disagreement set does not contain any variable then halt: there is no unifier.
Step IX: Choose one of the variables from the disagreement set $(e_1, \ldots, e_n\}$ say e^ and make the following set of (n-1) substitutions
$\{e_i \to e_1\}$ $\{e_i \to e_{i-1}\}$ $\{e_i \to e_{i+1}\}$ ... $\{e_i \to e_n\}$.
If there is any substitution component circular in this set then halt: there is no unifier.

THEOREM I. *Let* $O = \{\theta_i\}_{i=1}^n$ *be a non-empty set of substitutions. If* O *is unifiable then there exists a moet general unifier o of Qand* o *is determined by the substitution unification algorithm.* The proof goes similarly to that of the unification theorem.

Example

$\theta_1 = \{x \to f(y), z \to a\}; \theta_2 = \{x \to f(g(s)), t \to b\}$
$\tau = \{y \to g(a), z \to a, t \to b, u \to g(a), s \to a\}$ **is a unifier.**
$\sigma_0 = \varepsilon$; $\lambda_0 = \tau$
$\sigma_1 = \{y \to g(s)\}$; $\lambda_1 = \{t \to b, z \to a, u \to g(a), s \to a\}$
$\sigma_2 = \{y \to g(s), z \to a\}$; $\lambda_2 = \{t \to b, u \to g(a), s \to a\}$
σ_3 = m.g.u. = $\{y \to g(s), z \to a, t \to b\}$; $\lambda_3 = \{u \to g(a), s \to a\}$

THEOREM 2. *Given a unifiable set of substitutions* ^6i*?»I *oue*^ *that «£ is in normal form for all* i. *The most aeneral instance of* $\{\theta_i\}_{i=1}^n$ *is also a unifier of* $\{\theta_i\}_{i=1}^n$.

PROOF. Let the most general unifier of $\{\theta_i\}_{i=1}^n$ be o then $\theta_1\sigma = \ldots = \theta_n\sigma$ is the most general instance. Because of the normal form of the substitutions θ_i. $\theta_i = \theta_i$ holds. Because of the associativity of the composition of substitutions we have: $\theta_i\theta_i\sigma = \theta_i(\theta_i\sigma)$ and therefore $\theta_1\theta_1\sigma = \theta_2\theta_2\sigma = \theta_2\theta_1\sigma$. Hence, $\theta_1\sigma$ is a unifier. □

THEOREM 3. *For a given set of expressions which is unifiable we can compute the most general unifier also by applying the substitution unification algorithm on the set of substitutions formed by algorithm 2 from the disagreement sets. Then the* m.g.u. *of the expressions is the most general instance of these substitutions.*

(Substitutions from the second algorithm are in normal form since every substitution consists of but one substitution somponent)
The proof proceeds in two steps: first we prove that unifiabi1ity occurs in the same c ireurnstances; secondly we can prove that we get the same unifier up to isomorphisms.

Example

Are P(y,f(z),z) **and** P(x,x,t) **unifiable?**
Disagreement sets: $\{y,x\}$, $\{f(z),x\}$ $\{z,t\}$; **corresponding set of subsitutions** $\theta = \{\{y \to x, x \to f(z)\}, \{z \to t\}\}$.
Ordering of variables: x,y,z,t.
$\sigma_1 = \{x \to f(z)\}$; $\theta\sigma_1 = \{\{u \to f(z), y \to f(z)\}, \{x \to f(z)\}, \{x \to f(z), z \to t\}\}$.
$\sigma_2 = \{x \to f(z), y \to f(z)\}$
σ_3 = m.g.u. = $\{x \to f(t), y \to f(t), z \to t\}$. $\theta\sigma_3 = \{\{x \to f(t), y \to f(t), z \to t\}\}$.

Some Theorems Relating the Different Unifications

THEOREM 4. *Given a set of expressions* $\{E_i\}_{i=1}^n$ *with a most general unifier* θ *then for a set of n substitutions* $\{\theta_i\}_{i=1}^n$ *the following holds. The set of instances* $\{E_i\theta_i\}_{i=1}^n$ *is unifiable if the set of substitutions* $\{\theta\} \cup \{\theta_i\}_{i=1}^n$ *is unifiable. The most general unifier* σ *of* $\{\theta\} \cup \{\theta_i\}_{i=1}^n$ *is also a unifier of* $\{E_i\theta_i\}_{i=1}^n$.

PROOF. If $\{\theta\} \cup \{\theta_i\}_{i=1}^n$ **is unifiable then there exists a substitution** σ **the most general unifier, such that** $\theta\sigma = \theta_1\sigma = \ldots = \theta_n\sigma$; θ **is the most general unifier of** $(E_i)_{i=1}^n$ **so** $E_1\theta = \ldots = E_n\theta$ **also** $E_1\theta\sigma = = \ldots = E_n\theta\sigma$; **replacing** $\theta\sigma$ **by respectively** $\theta_1\sigma, \ldots, \theta_n\sigma$ **gives** $E_1\theta_1\sigma = \ldots = E_n\theta_n\sigma$ **which means that** σ **is a unifier of the set** $\{E_i\theta_i\}_{i=1}^n$. □

Remark

The computed unifier is not necessarily the most general one because variables which occur in Ej can be deleted by substituion fl.; therefore they do not occur in the computed unifier. This is one of the reasons that the converse of the theorem does not hold in general.

Counterexamples

1. $\{P(x), P(a)\}$ **unifiable** $\theta = \{x \to a\}$. **Let** $\theta_1 = \varepsilon$ **and** $\theta_2 = \{x \to b\}$ **then** $\{E_i\theta_i\}_{i=1}^n = \{P(x), P(a)\}$ **unifiable but** θ_1, θ_2 **and** θ **are not unifiable.**

2. $\{P(x,y), P(y,t)\}$ **unifiable** $\theta = \{x \to t, y \to t\}$. **Let** $\theta_1 = \{y \to a\}$ **and** $\theta_2 = \{y \to c\}$ **then** $\{E_i\theta_i\}_{i=1}^n = \{P(x,a), P(c,t)\}$ **unifiable but** θ_1, θ_2 **and** θ **are not unifiable.**

However, the converse theorem holds if we restrict the substitutions 0^ to variables that occur in E- and if the expressions E^ do not have any variables in common.

THEOREM *). *Let* $\{E_i\}_{i=1}^n$ *be a set of expressions which have no varxables in common and are unifiable with a moet general unifier* o. *Let* $\{\theta_i\}_{i=1}^n$ *be a set of substitutions such that a variable occuring in* 0^ *does not occur in* θ_j, $1 \leq i, j \leq n$ *and* θ_i *has as left-hand sides of its components only variables occuring in* E^. *If the set of expressions* $\{E_i\theta_i\}_{i=1}^n$ *is unifiable then the set of substitutions* $\{\sigma\} \cup \{\sigma_i\}_{i=1}^n$ *is unifiable.*

The proof proceeds by contradiction along the different steps of the substitution unification algorithm.

Example

Given *E^* * P(y_f f(z),z) and E- - P(x,x,t) then the most general unifier of E| and E2 is

σ = {x→f(t), y→f(t), z→t}.
If we now apply the substitution τ = {x→f(a)} to E_2 we can ask if P(y,f(z),z) and P(x,x,t). τ = P(f(a), f(a),t) are still unifiable. This implies the question: are the substitution σ and τ unifiable.
σ_1 = m.g.u. (f(t),f(a)) = {t→a}
σ_2 = {t→a, y→f(a)}
σ_3 = the most general unifier = {t→a, y→f(a), z→a} and this is also the most general unifier of P(y,f(z),z) and P(x,x,t). τ = P(f(a), f(a),t).

4. Application of Substitution Unification to Theorem Proving

Readers unfamiliar with the notions used in resolution theorem proving can find the necessary definitions in [5 and 2].
The application of our approach is strongly connected with on the one hand the way the resolution principle is used and on the other hand with the representation of clauses in the computer memory. If the clauses are represented in the usual way as lists, there is no profit in using the algorithms stated in §3. The improvement in efficiency comes in when we represent clauses by two pointers pointing to the parent clauses and by a pointer to the substitution applied in this resolution step [1]. For this latter representation, using the unification algorithm for expressions requires a search through the tree of pointers to make the actual clause since we want to know whether the literals are unifiable with the literals in some other clause. The advantage of using the substitution unification algorithm here can be seen right away.

We first compute the so-called classification matrix for the input-clauses [4]. The literals in the input clauses are numbered, let us say, from 1 to m and the matrix consists of substitutions. The (l,k)th entry is 0 if the lth literal is not unifiable with the k-th literal; the entry is the most general unifier of the l-th and k-th literal otherwise.

We can state the following corollary of theorem 4 and 5 from 3. (unification for a resolution step).

If L *and* K *are unifiable literals with* m.g.u. σ *(*L *and* K *do not have any variables in common) then* $L\theta_1$ *and* $K\theta_2$ *are unifiable iff* θ_1, θ_2 *and* σ *are unifiable; the most general unifier of* $\{\theta_1, \theta_2, \sigma\}$ *is also a unifier of* $L\theta_1$ *and* $K\theta_2$. *(In* θ_1, θ_2 *occur only variables occuring in* L *and* K *respectively and* θ_1, θ_2 *do not have variables in common)*

Implementing this we have to take care of the conditions stated in parentheses; we will see how in an example stated below.

In case of factoring the situation is a bit different because there the substitutions θ_1 and θ_2 have variables in common therefore we have to treat factoring separately. The following situation occurs. Two literals L and K are unifiable with m.g.u. σ; K and L do not have any variables in common because they come from different clauses. Substitutions θ_1 and θ_2 are applied to K and L respectively and then the clauses containing $K\theta_1$ and $L\theta_2$ are resolved on literals different from $K\theta_1$ and $L\theta_2$, involving unifier λ. The question to be answered now is: are $K\theta_1\lambda$ and $L\theta_2\lambda$ still unifiable given L and K are unifiable with m.g.u. σ.

THEOREM 1. $K\theta_1\lambda$ *and* $L\theta_2\lambda$ *are unifiable in the above case iff* $\sigma, \theta_1, \theta_2$ *and* λ *are unifiable. The* m.g.u. *of* $\{\sigma, \theta_1.\theta_2, \lambda\}$ *is a unifier of* $K\theta_1\lambda$ *and* $L\theta_2\lambda$.

PROOF. If $\{\sigma,\theta_1,\theta_2,\lambda\}$ unifiable then $K\theta_1\lambda$ and $L\theta_2\lambda$ are unifiable. (follows directly from theorem 4, 3). If $\{\sigma,\theta_1,\theta_2,\lambda\}$ are not unifiable then because θ_1,θ_2 and λ are unifiable, σ is not unifiable with either θ_1,θ_2 or λ.

Case I : $x \to t_1 \in \sigma$; $x \to t_2 \in \theta_1 \vee\theta_2 \vee\lambda$ with $x \in L \cup K$.
If $x \in L$ then $t_1 \in K$, in $L\theta_2\lambda$ x is replaced by t_2 so K and $L\theta_2\lambda$ are not unfiable so $K\theta_1\lambda$ and $L\theta_2\lambda$ not unifiable.
Case II: $x \to y \in \sigma$; $y \to f(x) \in \theta_1 \vee \theta_2 \vee \lambda$, $x \in L$, $y \in K$
$y \to f(x) \notin \theta_1 \vee \theta_2 \Rightarrow y \to f(x) \in \lambda$; so y in K is replaced by f(x) and $K\lambda$ and L not unifiable; if x, $y \in L$ the same follows: $y \to f(x) \in \theta_2 \vee \lambda$ etc. □

All together we can see that we are not especially interested in most general unifiers but we are interested in what happens to all variables in the input clauses during the deduction and that is what we have to keep track of. (see example).

Another application strikes the eye when we look at the theorem proving problem in a problem solving way: we start with some topclause denoting the disjunction of its literals, the goal we have to reach is to resolve away all literals (subgoals) (and - branches), each of which can be resolved in different ways (or - branches). What makes things complicated is that the subgoals are not independent [3,4]. Firstly we can assume that all new variables introduced in different branches are different from each other so that the only link between subgoals is the variables they have in common. This means that we have only to keep track of substitution components that effect those variables.

We want to prove that the most general instance of $\theta_1,\ldots,\theta_n$ is the ultimate solution to problem $L_1 \ldots L_n$ if $\theta_1,\ldots,\theta_n$ are solutions to $L_1,\ldots,L_n$ respectively. The fact that literal L (subgoal) is solvable with substitution θ (variables of left sides of the components all occur in L) means that there exists a refutation of L say C_0 = L, $C_1,\ldots,C_m$ = [] where C_i is the resolvent of C_{i-1} with a clause D_{i-1} (not necessarily a input-clause) furthermore D_0 has to contain a literal K opposite in sign to L with |K| and |L| unifiable.

From this refutation we can find a derivation of an expression E = Kσ with |E| and L unifiable with unifier θ : $C_1' = D_0$, $C_2',\ldots,C_m' = K\sigma$.

THEOREM 2. *If there exists a refutation of literal* L *(unrestricted resolution) involving substitution* θ *on* L, *then there exists a deduction of a literal* K *which has sign opposite to* L *and where* |K| *and* |L| *are unifiable.*

The proof proceeds by induction on the length of the refutation.

Example

Given a set V and a multiplication operator * which is associative and left and right solutions are in V then we want to prove the existence of the identity element. To translate this into predicate calculus we use the predicate P(x,y,z) which can be interpreted as x * y = z.

Problem input clauses:
1. P(g(x,y),x,y) '∀x∀y∃z: z*x=y.
2. P(x,h(x,y),y) '∀x∀y∃z: x*z=y.
3. ¬P(x,y,z)¬P(y,u,y) P(z,u,z) '∀x∀y∀z∀u[x*y=z∧y*u=y⇒z*u=z]
4. ¬P(j(x),x,j(x)) 'negation of the theorem: ∃y∀x: x*y=x.

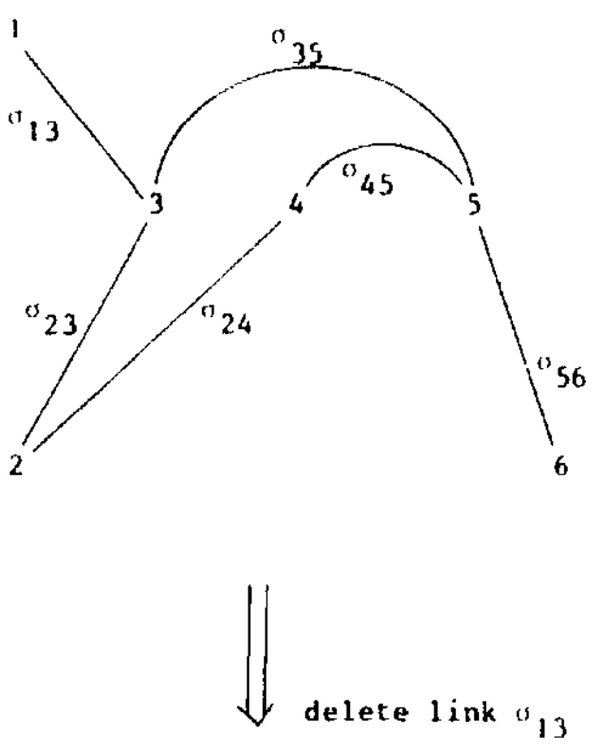
We number the literals in the problem 1 to 6.
(clause 3 containing lit 3,4 and 5)
We first compute the classification matrix:

	1	2	3	4	5	6
1	0	0	σ_{13}	0	0	0
2	0	0	σ_{23}	σ_{24}	0	0
3	σ_{13}	σ_{23}	0	0	σ_{35}	0
4	0	σ_{24}	0	0	σ_{45}	0
5	0	0	σ_{35}	σ_{45}	0	σ_{56}
6	0	0	0	0	σ_{56}	0

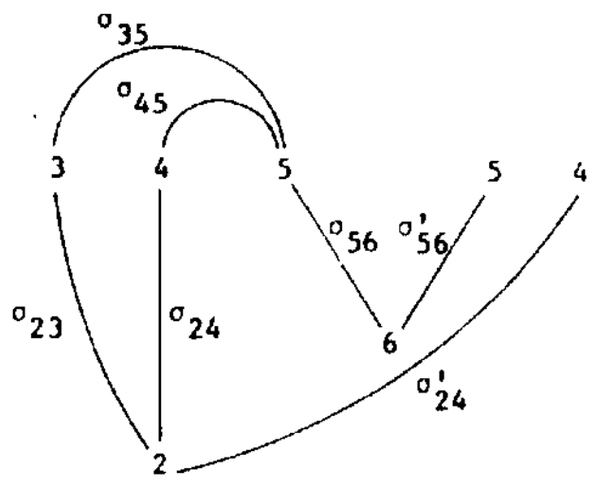
0 means not resolvable (identical to: opposite in sign and the absolute values are not unifiable). σ_{13} means literal 1 and literal 3 are resolvable involving substitution σ_{13} where variables occurring, in literal 1 are indexed by 1 and variables occurring in literal 3 by 3 so making the standard izing apart quite easily. The following substitutions are involved in the matrix:

$\sigma_{13} = \{x_3 \to g(x_1,y_1),\ y_3 \to x_1,\ z_3 \to y_1\}$
$\sigma_{23} = \{x_3 \to x_2,\ y_3 \to h(x_2,y_2),\ z_3 \to y_2\}$
$\sigma_{24} = \{y_3 \to x_1,\ y_1 \to x_1,\ u_3 \to h(x_1,x_1)\}$
$\sigma_{35} = \{x_3 \to z_5,\ y_3 \to u_5,\ z_3 \to z_5\}$
$\sigma_{45} = \{y_4 \to z_5,\ u_4 \to u_5\}$
$\sigma_{56} = \{z_5 \to j(x_6),\ u_5 \to x_6\}$

We work this example out using connection graphs [4]. If we use connection graphs it is almost necessary to use the substitution unification algorithm because the arcs between the clauses (literals) are in fact unifiers and new arcs can be computed from old arcs using this algorithm. The initial graph is made from the classification matrix:



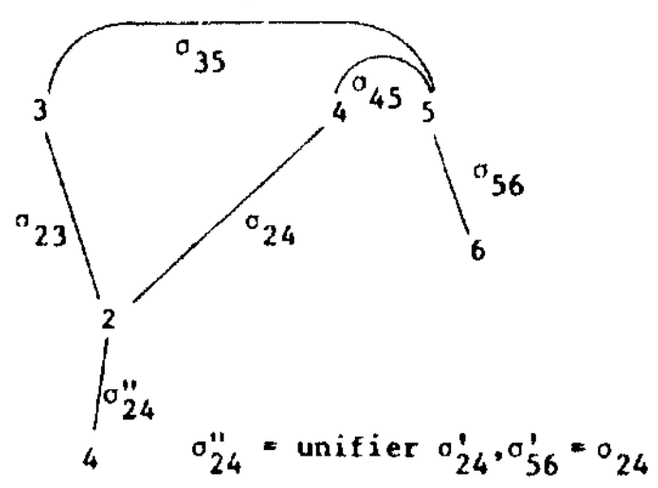
⇓ delete link σ_{13}



σ'_{56} = unifier σ_{56}, $\sigma_{13} = \sigma_{56}$

σ'_{24} = unifier σ_{24}, $\sigma_{13} = \sigma_{24}$

⇓ delete link σ'_{56}



σ''_{24} = unifier $\sigma'_{24}, \sigma'_{56} = \sigma_{24}$

⟹ delete σ''_{24} □

Note that $\sigma'_{24} = \sigma_{24}$ because we are only interested in those substitutions concerning variables in lit 2 and lit 4,5.

Acknowledgement

Much of the novel ideas in the paper developed from discussions with members of the Computational Logic Department of the University of Edinburgh during a stay of the author. In particular I am indebted to R. KOWALSKI who also gave very useful comments on a first draft.

References

[1] BOYER, B. & J. MOORE, *The sharing of structure in theorem proving programs.* Machine Intelligence 7, pp. 101-116 (eds. Metzer B., Michie D.), Edinburgh University Press, 1072.

[2] KOK, G. 4 J. VAN VAALEN, *An automatic theorem prover** Mathematisch Centrum report NR 22,1971.

[3] KOWALSKI, R. & D. KUEHNER, *Linear resolution with selection function.* Artificial Intelligence 2, pp.227-260, 1971.

[4] KOWALSKI, R., *A proof procedure using connec-*

tion graphs, Memo 74 Dept. of Computational Logic, University of Edinburgh, 1974.

[5] ROBINSON, J.A.,/] *machine oriented logic based on the resolution principle*, JACM 12 , pp.2 3-41 1965.

[6] ROBINSON, J,A., *Computational logic: the unification algorithm.* Machine Intelligence 6, pp. 63-72 (eds. Meltzer B., Michie D.), Edinburgh University Press, 1971.

[7] BAXTER, L.D., *An efficient unification algorithm.* University of Waterloo, 1973.