stichting
mathematisch
centrum

MC

AFDELING INFORMATICA IW 24/74 DECEMBER

J. VAN VAALEN

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

Prepublication

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 68A40

ACM- Computing Reviews -category: 5.21

An extension of unification to substitutions with an application to automatic theorem proving.

by

J. van Vaalen *)

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 substitions. 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 substitution and new arcs can be generated from old arcs.

*) Mathematisch Centrum, Amsterdam.

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.

2.1 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 $n \geq 0$ followed by n terms.

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

A *substitution* is a finite (possibly empty) set of substitution components with distinct left hand sides. Therefore in a substitution $\{\nu_1 \rightarrow t_1, \nu_2 \rightarrow t_2, \ldots, \nu_k \rightarrow t_k\}$ 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 $n \geq 1$.

Let E be a finite string of symbols and $\theta = \{v_i \to t_i\}_{i=1}^{k}$ be a substitution. The *instantiation* of E by θ is the operation of simultaneously replacing each occurrence of v_1 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 $\theta = \{v_i \to t_i\}_{i=1}^{k}$ and λ be substitutions. The *composition* of θ and λ is the substitution $\theta' \cup \lambda'$ where θ' consists of the components $v_i \to t_i\lambda$ 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 $\varepsilon\theta = \theta\varepsilon = \theta$ 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.

EXAMPLES.

1. The disagreement set of {P(x), P(a), P(y)} is the set {x,a,y}
2. The disagreement set of {R(x,y); R(g(f(x,y)), h(b)), R(a,t)} is the set {x,g(f(x,y)),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.)

EXAMPLES.

1. $\{E_i\}_{i=1}^{3}$ = {P(x), P(a), P(y)}

   most general unifier σ = {x→a, y→a}

   $\{E_i\sigma\}_{i=1}^{3}$ = {P(a)}

2. $\{E_i\}_{i=1}^{3}$ = {Q(f(y),x), Q(x,z), Q(f(t), f(g(a)))}

   most general unifier σ = {x→f(g(a)), y→g(a), z→f(g(a)), t → g(a)}

   $\{E_i\sigma\}_{i=1}^{3}$ = {Q(f(g(a)), f(g(a)))}

2.2 THE ALGORITHM AND THE UNIFICATION THEOREM.

The unification algorithm and theorem are quoted from ROBINSON [5].
The *Unification algorithm* computes the most general unifier of A, where A is a finite non-empty set of expressions:

Step I : $\sigma_0 := \varepsilon$ (the empty substitution); k: = 0;

Step II : If $A\sigma_k$ is a singleton stop: σ_k is the most general unifier of A.

Step III: Compute the disagreement set of $A\sigma_k$; 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 $\sigma_{k+1} := \sigma_k \cdot \{v_k \to u_k\}$; k := k+1; go to step 2 otherwise stop then A is not unifiable.

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

PROOF. Let θ be some unifier of A. It suffices to prove that the unification algorithm applied to A will terminate in step 2 yielding a most general unifier σ of A and that for all $i \geq 0$ the equation $\sigma_i\lambda_i = \theta$ holds for some substitution λ_i.

The proof proceeds by induction on i.

(a) $i = 0$: $\lambda_0 = \theta$ and $\sigma_0 = \varepsilon$.

(b) Suppose $\sigma_i\lambda_i = \theta$ holds in step 2 of the algorithm.
Either $A\sigma_i$ is a singleton and we are done, i.e. $\sigma = \sigma_i$ or we proceed to step 3 of the algorithm. Now λ_i unifies $A\sigma_i$, λ_i also unifies the disagreement set B_i of $A\sigma_i$ and $v_i\lambda_i = u_i\lambda_i$ for all $v_i, u_i \in B_i$.
From the unifiability of A it follows that at least one element in B_i is a variable, say v_i (v_i occurs in $u_i \in B_i$, $v_i \neq u_i$ is impossible since $v_i\lambda_i = u_i\lambda_i$). Hence we return to step 2 of the algorithm with
$\sigma_{i+1} = \sigma_i \cup \{v_i \to u_i\}$ where $u_i \in B_i$ and $u_i \neq v_i$ and $\lambda_{i+1} = \lambda_i - \{v_i \to u_i\lambda_i\}$
$\lambda_i = \lambda_{i+1} \cup \{v_i \to u_i\lambda_i\} = \lambda_{i+1} \cup \{v_i \to u_i\lambda_{i+1}\} = \{v_i \to u_i\} \cdot \lambda_{i+1}$.
Hence $\theta = \sigma_{i+1}\lambda_{i+1}$.

EXAMPLES.

1. $A = \{P(x), P(y), P(z)\}$ $\theta = \{x \to a, y \to a, z \to a\} = \lambda_0$.
   $\sigma_1 = \{x \to y\}; \lambda_1 = \{y \to a, z \to a\}$
   $\sigma_2 = \{x \to z, y \to z\}; \lambda_2 = \{z \to a\}$
   σ_2 is mgu; $\sigma_2\lambda_2 = \theta$.
2. $A = \{P(x), P(y), P(z)\}$ $\theta' = \{y \to x, z \to x\} = \lambda_0'$
   $\sigma_1 = \{x \to y\}; \lambda_1' = \{y \to x, z \to x\}$
   $\sigma_2 = \{x \to z, y \to z\}; \lambda_2' = \{z \to x\}$
   σ_2 is mgu; $\sigma_2\lambda_2' = \theta'$.

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 [6 and 2].

3. AN EXTENSION OF UNIFICATION TO SUBSTITUTIONS.

3.1 DEFINITIONS.

Substitution unifier. Let $\Theta = \{\theta_i\}_{i=1}^{k}$ be a set of substitutions and σ a substitution; σ is said to unify Θ or to be a unifier of Θ if $\theta_i\sigma = \theta_{i+1}\sigma$ for all $1 \leq i < k$. 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 θ_1 and θ_2 are substitutions then the composition $\theta_1\theta_2$ is called an instance of θ_1.

Most general unifier. A unifier σ of a set of substitutions $\Theta = \{\theta_i\}_{i=1}^{k}$ is called a most general unifier if for all unifiers τ of Θ there exists a substitution λ such that $\tau = \sigma\lambda$; the instance $\theta_1\sigma$ is called the most general instance of Θ.

Normal form. A substitution $\theta = \{v_i \rightarrow t_i\}_{i=1}^{k}$ is in normal form if for all i,j: $1 \leq i, j \leq k$, v_i does not occur in t_j.

NOTE.

1. By definition it holds that all v_i's in a substition 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 substitions $\{\theta_i\}_{i=1}^{k}$ is sometimes also denoted by $\theta_1*\theta_2*\cdots*\theta_k$; this operation is commutative and associative.

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

A set of substitutions $\Theta = \{\theta_i\}_{i=1}^{k}$ is called *contradictory* if there are two substitutions θ_1, θ_m in Θ such that there is a substitution component $v_{j_1} \rightarrow t_{j_1}$ in θ_1 and a substitution component $v_{j_m} \rightarrow t_{j_m}$ in θ_m such that $v_{j_1} = v_{j_m}$ and the terms t_{j_1} and t_{j_m} are not unifiable.

3.2 ALGORITHMS AND THEOREMS

We now give the algorithm to compute the most general unifier σ of a set of substitutions $\{\theta_i\}_{i=1}^{n}$. We assume that in each substitution of a set of substitutions the substitution-components are ordered according to a fixed enumeration of all variables.

Substitution Unification Algorithm.

Step I . $k = 0$; $\sigma_0 := \varepsilon$.

Step II. Compute $\{\theta_i\sigma_k\}_{i=1}^{n}$; delete components of the form $v \to v$ from $\{\theta_i\sigma_k\}_{i=1}^{n}$. If there is a circular substitution component in one of the substitutions $\theta_i\sigma_k$ then there is no unifier and the algorithm halts.
If $\{\theta_i\sigma_k\}_{i=1}^{n}$ is a singleton then σ_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 $\theta_i\sigma_k$; 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, $\{v \to t_j\}_{j=1}^{m}$. We now distinguish three cases.

Case I . $m = 1$. Then: $k := k+1$; $\sigma_k := \sigma_{k-1} \cdot \{v \to t_1\}$; goto step 2.

Case II. The set contains substitution components that come from some but not all of the sets $\{\theta_i\sigma_k\}_{i=1}^{n}$. If the set $\{v \to t_j\}_{j=1}^{m}$ is contradictory then stop: not unifiable otherwise let λ be the most general unifier of $\{t_j\}_{j=1}^{m}$ then $k := k+1$; $\sigma_k = \sigma_{k-1} \cdot \{v \to t_1\}\ \lambda$; goto step 2.

Case III. The set contains substitution components that come from all $\theta_i\sigma_k$. If the set $\{v \to t_j\}_{j=1}^{m}$ is contradictory then halt: not unifiable otherwise let λ be the most general unifier of $\{t_j\}_{j=1}^{m}$ then $k := k + 1$; $\sigma_k := \sigma_{k-1} \cdot \lambda$; goto step 2.

EXAMPLES

1. $\theta_1 = \{x \to t, y \to a\}$; $\theta_2 = \{x \to f(y), z \to f(y)\}$; $\theta_3 = \{x \to z, t \to z, s \to b\}$.
   Ordering: x,y,z,t,s.
   $k = 0$: $\sigma_0 = \varepsilon$.
   $k = 1$: $x \to t$, $x \to f(y)$, $x \to z$.

compute m.g.u. of {t,f(y),z}: $t \to f(y)$, $z \to f(y)$

$\sigma_1 = \{t \to f(y), z \to f(y)\}$

$\theta_1\sigma_1 = \{x \to f(y), y \to a, z \to f(y), t \to f(y)\}$

$\theta_2\sigma_1 = \{x \to f(y), z \to f(y), t \to f(y)\}$

$\theta_3\sigma_1 = \{x \to f(y), z \to f(y), t \to f(y), s \to b\}$

$\sigma_2 = \sigma_1 \cdot \{y \to a\} = \{t \to f(a), z \to f(a), y \to a\}$

$\theta_1\sigma_2 = \theta_2\sigma_2 = \{x \to f(a), y \to a, z \to f(a), t \to f(a)\}$

$\theta_3\sigma_2 = \{x \to f(a), y \to a, z \to f(a), t \to f(a), s \to b\}$

$\sigma_3 = \sigma_2 \cdot \{s \to b\} = \{t \to f(a), z \to f(a), y \to a, s \to b\}$

σ_3 is the most general unifier of $\{\theta_1, \theta_2, \theta_3\}$

2. $\theta_1 = \{x \to y, y \to f(a)\}$; $\theta_2 = \{x \to f(a), y \to f(a)\}$
   $\sigma_1 = \{y \to f(a)\}$ is the most general unifier.

3. $\theta_1 = \{x \to y, y \to f(a)\}$; $\theta_2 = \{x \to b, y \to f(a)\}$
   $\theta_1 = \{y \to b\}$ is the most general unifier and θ_2 is the most general instance.

4. $\theta_1 = \{x \to y\}$; $\theta_2 = \{x \to b, y \to f(a)\}$
   $\sigma_1 = \{y \to b\}$ $\theta_1\sigma_1 = \{x \to b, y \to b\}$; $\theta_2\sigma_1 = \{x \to b, y \to f(a)\}$
   because $\{y \to b\}$ and $\{y \to f(a)\}$ 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: $\theta_2 = \{x \to b, y \to f(a)\}$ 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 more disagreement sets, in the sense of going further right until the end of the expressions, at the same time.

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 of expressions if possible.

Step I: If the disagreement set does not contain any variable then halt: there is no unifier.

Step II: Choose one of the variables from the disagreement set $\{e_1,\ldots,e_n\}$ say e_i and make the following set of (n-1) substitutions
$\{e_i \to e_1\} \ldots \{e_i \to e_{i-1}\}\ \{e_i \to e_{i+1}\} \ldots \{e_i \to e_n\}$.
If there is any substitution component circular in this set then halt: there is no unifier.

THEOREM 1. Let $\Theta = \{\theta_i\}_{i=1}^{n}$ *be a non-empty set of substitutions.* If Θ *is unifiable then there exists a most general unifier* σ of Θ *and* σ *is determined by the substitution unification algorithm.*

PROOF. (Similar to that of the unification theorem in 2.2) Let τ be a unifier of Θ. It suffices to prove that under the hypotheses of the theorem the algorithm will terminate at step 2 when applied to Θ and for all $i \geq 0$
$\sigma_i\lambda_i = \tau$ holds for some λ_i.
The proof proceeds by induction on i.
(a) i = 0 then $\sigma_0 = \varepsilon$ and $\lambda_0 = \tau$.
(b) $i \geq 0$ and $\sigma_i\lambda_i = \tau$ holds for some λ_i at step 2, then either $\{\theta_j\sigma_i\}_{j=1}^{n}$ is a singleton and we are done or we proceed to step 3; λ_i now unifies $\Theta\sigma_i$ and we can have one of the three cases:
Case I. There is just one $v \to t$ in $\{\theta_j\sigma_i\}_{j=1}^{n}$ so λ_i has to unify $\{\varepsilon,\{v\to t\}\}$ and therefore $\{v\to t\ \lambda_i\} \in \lambda_i$. Let $\sigma_{i+1} = \sigma_i \cdot \{v\to t_i\}$
$\lambda_{i+1} = \lambda_i - \{v\to t\ \lambda_i\}$; $\lambda_i = \lambda_{i+1} \cup \{v\to t\ \lambda_i\} = \lambda_{i+1} \cup \{v\to t\ \lambda_{i+1}\} = \{v\to t\}\cdot\lambda_{i+1}$
and hence $\tau = \sigma_{i+1}\cdot\lambda_{i+1}$.
Case II. Similar to case I because λ_i now has to unify $\{\varepsilon,\{v\to t_i\}_{i=1}^{m}\}$ therefore $\sigma'_{i+1} = \sigma_i \cdot \{v\to t_1\}$ and $\lambda'_{i+1} = \lambda_i \cup \{v\to t, \lambda_i\}$ and $\sigma'_{i+1}\cdot\lambda'_{i+1} = \tau$.
Furthermore, λ_i has to unify $\{v\to t_i\}_{i=1}^{m}$ so λ_i has to unify $\{t_i\}_{i=1}^{m}$ let the most general unifier be η then $\sigma_{i+1} = \sigma'_{i+1}\cdot\eta$ and $\lambda_{i+1} = \lambda'_{i+1} - \eta\cdot\lambda'_{i+1}$
$\lambda'_{i+1} = \lambda_{i+1} \cup.\eta\ \lambda'_{i+1} = \lambda_{i+1} \cup \eta.\lambda_{i+1} = \eta.\lambda_{i+1}$. which gives $\sigma_{i+1}\cdot\lambda_{i+1} = \tau$.
Case III. Analogous to the second part of case II. □

AN 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.

$\sigma_0 = \varepsilon; \lambda_0 = \tau$

$\sigma_1 = \{y \to g(s)\}; \lambda_1 = \{t \to b, z \to a, u \to g(a), s \to a\}$

$\sigma_2 = \{y \to g(s), z \to a\}; \lambda_2 = \{t \to b, u \to g(a), s \to a\}$

$\sigma_3 = \text{m.g.u.} = \{y \to g(s), z \to a, t \to b\}; \lambda_3 = \{u \to g(a), s \to a\}$

THEOREM 2. *Given a unifiable set of substitutions* $\{\theta_i\}_{i=1}^n$ *such that* θ_i *is in normal form for all* i. *The most general 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 σ then $\theta_1\sigma = \ldots = \theta_n\sigma$ is the most general instance. Because of the normal form of the substitutions $\theta_i \cdot \theta_i = \theta_i$ 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.*

PROOF. (Substitutions from the second algorithm are in normal form since every substitution consists of but one substitution component)
The proof is in two steps: first we prove that unifiability occurs in the same circumstances, secondly we prove that we get the same unifier up to isomorphisms.

Unifiability. Let D_1^k be the disagreement set which we encounter in the algorithm computing the most general unifier of expressions and σ_k the substitution lastly made in the algorithm. Let D_0^k be the corresponding disagreement set, which we encounter in algorithm 2, previous to the application of σ_k, then we have for all k: $D_1^k = D_0^k \cdot \sigma_k$.
Now we can distinguish the different situations where the set D_1^k is not unifiable: D_1^k contains no variables or D_1^k contains a variable x and a

term containing x. If D_0^k does not contain variables or contains x and t(x) then algorithm 2 gives already the non-unifiability.
Otherwise either D_0^k contains variables and σ_k does substitute non-variables which means that algorithm 2 delivers two contradictory substitutions which leads to non-unifiability, or D_0^k contains two variables and σ_k does substitute a term which contains x for the other variable which means that algorithm 2 delivers two substitutions like $\{x \mapsto y\}$ $\{y \mapsto t(x)\}$ which leads also to non-unifiability.

Unifier. For the simple case that there is just one disagreement set we obtain the same unifier since we unify the set of (n-1) terms and add $x \mapsto t_1\sigma$ to get the most general instance of the substitutions which is the m.g.u. of the set of expressions; generalization is straightforward. □

AN 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 substitutions θ = {{y↦x},{x↦f(z)}, {z↦t}}.
Ordering of variables: x,y,z,t.
σ_1 = {x↦f(z)}; $\theta\sigma_1$ = {{u↦f(z),y↦f(z)}, {x↦f(z)},{x↦f(z),z↦t}}.
σ_2 = {x↦f(z), y↦f(z)}
σ_3 = m.g.u. = {x↦f(t), y↦f(t), z↦t}. $\theta\sigma_3$ = {{x↦f(t), y↦f(t), z↦t}}.

3.3. SOME THEOREMS RELATING THE DIFFERENT UNIFICATIONS.

THEOREM 1. *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 E_i can be deleted by substitution θ_i; therefore they don't 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 θ = {x↦a}. Let $\theta_1 = \varepsilon$ and θ_2 = {x↦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 θ = {x↦t, y↦t}. Let θ_1 = {y↦a} and θ_2 = {y↦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 θ_i to variables that occur in E_i and if the expressions E_i don't have any variables in common.

THEOREM 2. *Let* $\{E_i\}_{i=1}^n$ *be a set of expressions which have no variables in common and are unifiable with a most general unifier* σ. *Let* $\{\theta_i\}_{i=1}^n$ *be a set of substitutions such that a variable occuring in* θ_i *does not occur in* θ_j, $1 \le i, j \le n$ *and* θ_i *has as left-hand sides of its components only variables occuring in* E_i. *If the set of expressions* $\{E_i\theta_i\}_{i=1}^n$ *is unifiable then the set of substitutions* $\{\sigma\} \cup \{\theta_i\}_{i=1}^n$ *is unifiable.*

PROOF. Assume that $\{\sigma\} \cup \{\theta_i\}_{i=1}^n$ is not unifiable. Since no variables from θ_i occurs in θ_j, $1 \le i, j \le n$ the set $\{\theta_i\}_{i=1}^n$ is unifiable. Therefore there must be a θ_j such that θ_j and σ are not unifiable. Now the substitution unification algorithm applied on (θ_j,σ) halts either in step 2 or in step 3.
Step II. If we stop at step II then there is a substitution component x↦y in θ_j or σ resp. and a substitution component y↦t(x) in σ_k (the substitution in the algorithm) then either y↦t(x) is already in σ or θ_j respectively.
If we consider $\{E_i\theta_i\}_{i=1}^n$ then the disagreement set consists of {y,x} and in the next step of the unification algorithm for expressions the disagreement set consists of {t(x), x} so $\{E_i\theta_i\}_{i=1}^n$ is not unifiable; or there is a substitution component z↦x in σ; if we consider $\{E_i\theta_i\}_{i=1}^n$ then the disagreement set consists of {y,z} and in the next step {z,t(z)} so $\{E_i\theta_i\}_{i=1}^n$ is not unifiable.

Step III. If we stop at step III then there are substitution components $x \mapsto t_1$ in θ_j and $x \mapsto t_2$ in σ with t_1 and t_2 not unifiable. If we consider the disagreement sets we get t_1, t_2 somewhere as disagreement set in the unification algorithm for expressions on $\{E_i\theta_i\}_{i=1}^{n}$, so $\{E_i\theta_i\}_{i=1}^{m}$ is not unifiable. □

EXAMPLE.
Given $E_1 = P(y,f(z),z)$ and $E_2 = P(x,x,t)$ then the most general unifier of E_1 and E_2 is $\sigma = \{x \mapsto f(t), y \mapsto f(t), z \mapsto t\}$.
If we now apply the substitution $\tau = \{x \mapsto f(a)\}$ to E_2 we can ask if $P(y,f(z),z)$ and $P(x,x,t)$. $\tau = P(f(a), f(a),t)$ are still unifiable.
This implies the question: are the substitution σ and τ unifiable.
σ_1 = m.g.u. $(f(t),f(a)) = \{t \mapsto a\}$
$\sigma_2 = \{t \mapsto a, y \mapsto f(a)\}$
σ_3 = the most general unifier = $\{t \mapsto a, y \mapsto f(a), z \mapsto a\}$
and this is also the most general unifier of $P(y,f(z),z)$ and $P(x,x,t).\tau = P(f(a), f(a),t)$.

4. APPLICATION OF SUBSTITUTION UNIFICATION TO THEOREM PROVING.

For readers unfamiliar with the notions used in resolution theorem proving we give the necessary definitions in the appendix.
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 (1,k)th entry is 0 if the 1th literal is not unifiable with the k th literal; the entry is the most general unifier of the 1 th and k th literal otherwise.

We can state the following corollary of theorem 1 and 2 from 3.3. (unification for a resolution step).

If L *and* K *are unifiable literals with* m.g.u. σ (L *and* K *don't 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 *don't 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 don't 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 stated case iff* σ,θ_1,θ_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 theorem1,3.3). 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 unifiable 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 examples.)

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.

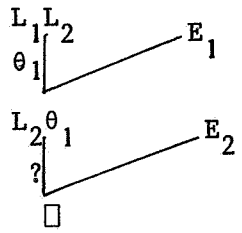
We want to prove that the most general instance of $\theta_1,\ldots,\theta_n$ is the ultimate solution to problem $L_1 \ldots L_n$ if $\theta_1,\ldots,\theta_n$ are solutions to $L_1,\ldots,L_n$ respectively. The fact that literal L (subgoal) is solvable with substitution θ (variables of left sides of the components all occur in L) means that there exists a refutation of L say $C_0 = L, C_1,\ldots,C_m = \square$ 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 $E = K\sigma$ 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.*

PROOF. Refutation of L is a sequence $C_0, C_1,\ldots,C_m$ where $C_0 = L$, C_i is resolvent of clause C_{i-1} with some clause D_{i-1} $1 \leq i < m$ and $C_m = \square$.

This means that in D_0 there is a literal K' which is opposite in sign to L and where |K'| and |L| are unifiable involving substitution σ.
Now we can make a derivation of $K'\lambda = K\ C_1', \ldots, C_i', \ldots, C_m' = K'\lambda$. with $C_1' = D_0$ and $C_i'\sigma - K\lambda'\sigma = C_i$. All the resolution steps go through because C_i is an instance of C_i.
We know that K and L have opposite sign and have to prove that |K| and |L| are unifiable.
In case of a top-clause with two literals $L_1\ L_2$ we have (solvable with θ_1, θ_2).



We know $|L_2|$ and $|E_2|$ are unifiable with m.g.u.θ_2. $|L_2\theta_1|$ and $|E_2|$ now are unifiable iff θ_1 and θ_2 are unifiable. The most general instance of θ_1, θ_2 now is the solution of $L_1\ L_2$.
This is again a corrollary of theorem 1 and 2 section 3.3, because L_2, E_2 don't have variables in common and θ_1 does not contain any variable from E_2. □

EXAMPLE.
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∀x∀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 classiffication 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 standardizing apart quite easily. The following substitutions are involved in the matrix:

$\sigma_{13} = \{x_3 \rightarrow g(x_1,y_1),\ y_3 \rightarrow x_1,\ z_3 \rightarrow y_1\}$

$\sigma_{23} = \{x_3 \rightarrow x_2,\ y_3 \rightarrow h(x_2,y_2),\ z_3 \rightarrow y_2\}$

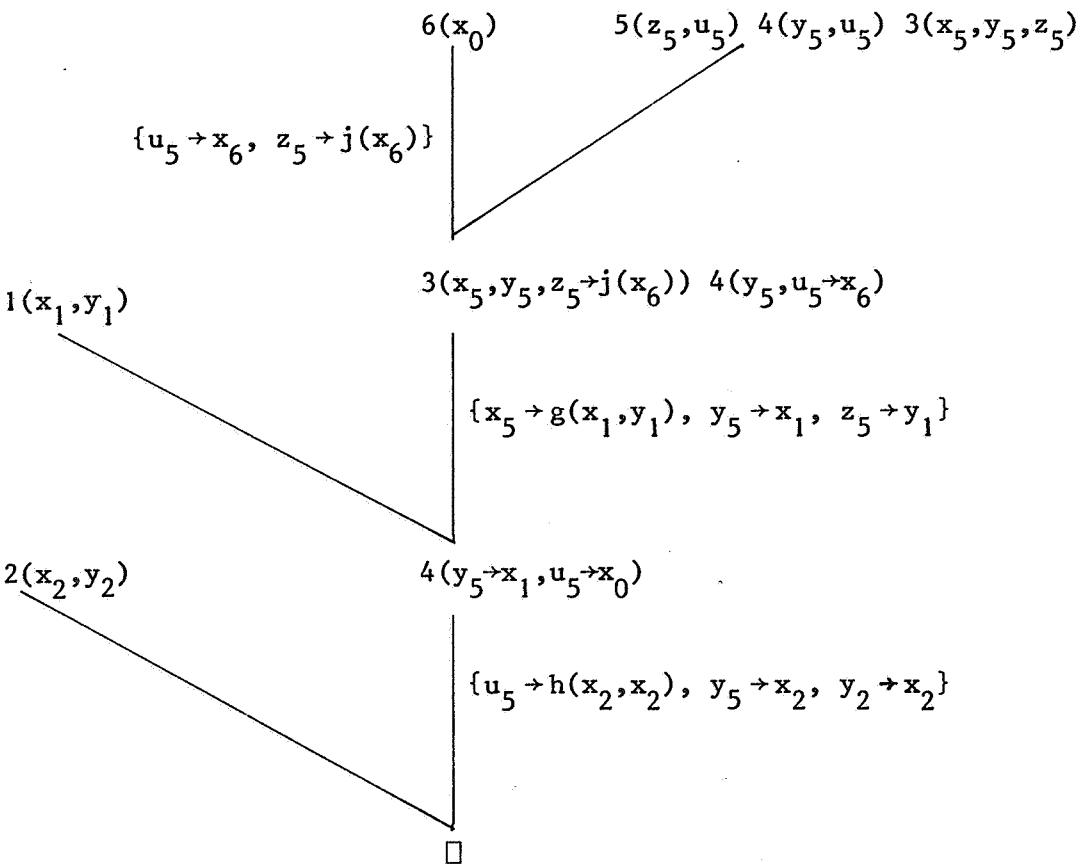
$\sigma_{24} = \{y_3 \rightarrow x_1,\ y_1 \rightarrow x_1,\ u_3 \rightarrow h(x_1,x_1)\}$

$\sigma_{35} = \{x_3 \rightarrow z_5,\ y_3 \rightarrow u_5,\ z_3 \rightarrow z_5\}$

$\sigma_{45} = \{y_4 \rightarrow z_5,\ u_4 \rightarrow u_5\}$

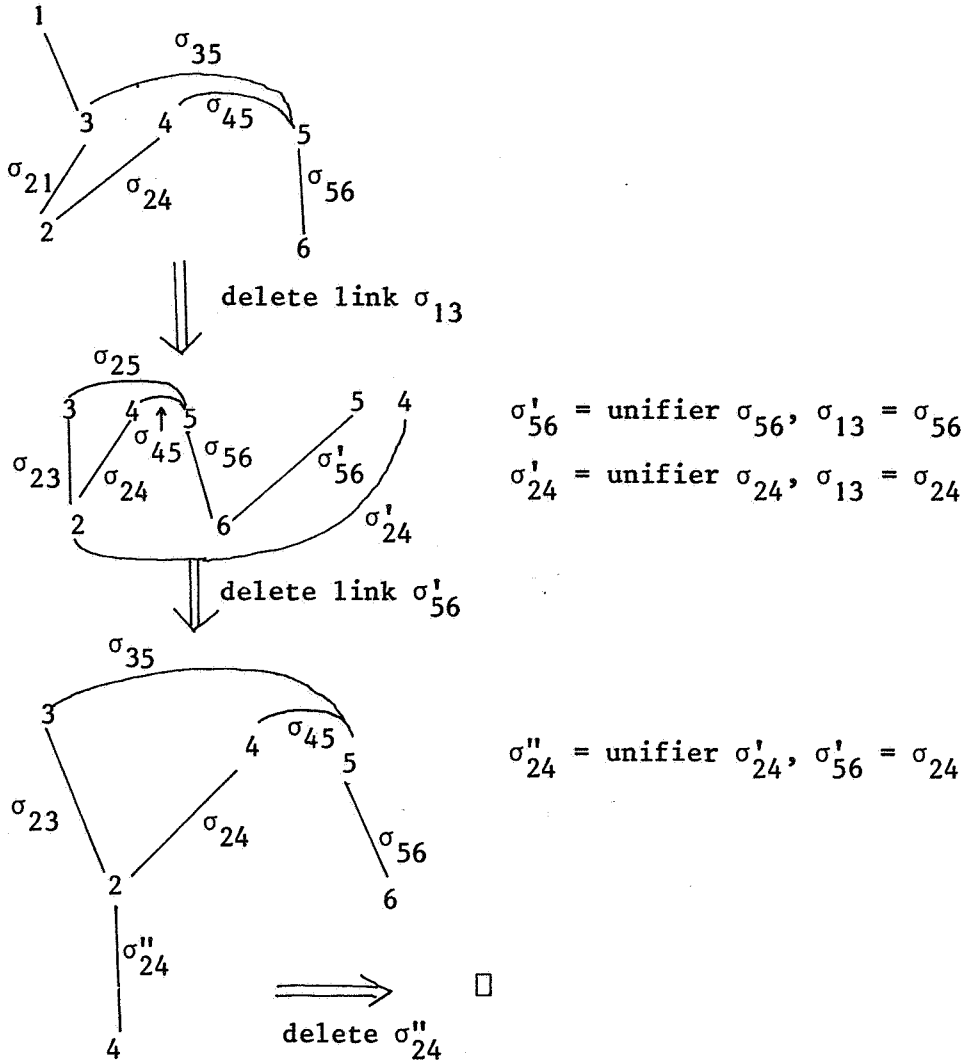
$\sigma_{56} = \{z_5 \rightarrow j(x_6),\ u_5 \rightarrow x_6\}$

Now we try and find a refutation using SLN resolution [3] with as search strategy diagonal search where f is the level and g is the number of literals; literal 6 is the top clause. Each literal will be represented by its number and its variables with or without substitutions.



Perhaps we are interested in 'the answer', what is substituted for x_6: $x_6 \rightarrow h(x_2,x_2)$.
Notice that using this set of clauses and diagonal search no more clauses are generated.

Now we have the same example 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:



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.

APPENDIX. *Some definitions in theorem proving* [5].

A *literal* is a predicate letter (of order n) followed by n terms and possibly preceded by a negation sign. Ex. P(x);¬Q(a,f(g(x,h(y,a)))).

A *clause* is a set of literals denoting the disjunction of those literals.

For convenience the empty set of literals (denoted by □)considered to be a (special case of a) clause called the empty clause and having the truth value *false*.

The goal of the resolution theorem proving algorithm is to derive the empty clause (contradiction) from a given set of clauses (consisting of the axioms of the theory and the negation of the theorem we want to prove). The set of clauses denotes the conjunction of those clauses and is a straightforward translation of the problem stated in first order predicate logic to the clausal form [5].

The order in which the new clauses are formed depends on the so-called search strategy.

The inference rule used is the *resolution rule*.

Two clauses A and B are *resolvable* if A contains a literal L and B contains a literal K such that K and L are opposite in sign and there exists a substitution σ such that $|K|\sigma = |L|\sigma$ ($|K|$ denotes the literal K regardless of the negation symbol) the new clause is called the *resolvent* of A and B: $(A-L)\sigma \cup (B-K)\sigma$.

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, 1972.

2. KOK, G. & 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. KOWALSI, R., *A proof procedure using connection graphs*, Memo 74 Dept. of Computational Logic, University of Edinburh, 1974.

5. ROBINSON, J.A., *A machine oriented logic based on the resolution principle*, JACM 12, pp. 23-41, 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.