SEKI-PROJEKT

SEKI
MEMO

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1 W Germany

Unification in Abelian Semigroups

Alexander Herold, Jörg H. Siekmann

MEMO SEKI-85-III-KL

Unification in Abelian Semigroups

A. Herold, J. Siekmann

MEMO SEKI-85-III-KL

UNIFICATION IN ABELIAN SEMIGROUPS

Alexander Herold
Jörg H. Siekmann
Universität Kaiserslautern
Fachbereich Informatik
Postfach 30 49
D-6750 Kaiserslautern

ABSTRACT:

Unification in equational theories, i.e. solving of equations in varieties, is a basic operation in Computational Logic, in Artificial Intelligence (AI) and in many applications of Computer Science. In particular the unification of terms in the presence of an associative and commutative function, i.e. solving of equations in Abelian Semigroups, turned out to be of practical relevance for Term Rewriting Systems, Automated Theorem Provers and many AI-programming languages. The observation that unification under associativity and commutativity reduces to the solution of certain linear diophantine equations is the basis for a complete and minimal unification algorithm. The set of most general unifiers is closely related to the notion of a basis for the linear solution space of these equations.

These results are extended to unification in free term algebras combined with Abelian Semigroups.

CONTENTS

1. UNIFICATION IN EQUATIONAL THEORIES

Unification theory is concerned with problems of the following kind: Let f and g be function symbols, a and b constants and let x and y be variables and consider two ***first order terms*** built from these symbols, for example:

$$t_1 = f(x\ g(a\ b))$$

$$t_2 = f(g(y\ b)\ x)$$

The problem is whether or not there exist terms which can be substituted for the variables x and y such that the two terms thus obtained from t_1 and t_2 become equal: in the example g(a b) and a are two such terms. We shall write

$$\sigma_1 = \{x \leftarrow g(a\ b),\ y \leftarrow a\}$$

for such a unifying substitution: σ_1 is a ***unifier*** of t_1 and t_2 since $\sigma_1 t_1 = \sigma_1 t_2$.
In addition to the above ***decision problem*** there is also the problem of finding a ***unification algorithm*** which enumerates the unifiers for a given pair t_1 and t_2.
Consider a variation of the above problem, which arises when we assume that f is commutative:

(C) $\qquad f(x\ y) = f(y\ x)$

Now σ_1 is still a unifying substitution and moreover $\sigma_2 = \{y \leftarrow a\}$ is also a unifier for t_1 and t_2 since

$$\sigma_2 t_1 = f(x\ g(a\ b)) =_C f(g(a\ b)\ x) = \sigma_2 t_2.$$

But σ_2 is ***more general*** than σ_1, since σ_1 is an instance of σ_2 obtained as the ***composition*** $\lambda \cdot \sigma_2$ with $\lambda = \{x \leftarrow g(a\ b)\}$; hence a unification algorithm only needs to compute σ_2.
In some cases there is a single and essentially unique least upper bound on the generality lattice of unifiers, called the ***most general unifier***.
Under commutativity however, there are pairs of terms which have more than one most general unifier, but they always have at most ***finitely many***. This is in contrast for example to the above situation of free terms , where every pair

has at most ***one*** most general unifying substitution.
The problem becomes entirely different when we assume that the function denoted by f is associative:

$$(A) \qquad f(x\ f(y\ z)) = f(f(x\ y)\ z)$$

In that case σ_1 is still a unifying substitution, but

$$\sigma_3 = \{x \leftarrow f(g(a\ b)\ g(a\ b)),\ y \leftarrow a\}$$

is also a unifier:

$$\sigma_3 t_1 = f(f(g(a\ b)\ g(a\ b))\ g(a\ b)) =_A f(g(a\ b)\ f(g(a\ b)\ g(a\ b))) = \sigma_3 t_2.$$

But $\sigma_4 = \{x \leftarrow f(g(a\ b)\ f(g(a\ b)\ g(a\ b))),\ y \leftarrow a\}$ is again a unifying substitution and by iteration of this process it is not difficult to see that there are ***infinitely many*** unifiers, all of which are most general.
Finally, if we assume that both axioms (A) and (C) hold for f then the situation changes yet again and for any pair of terms there are at most ***finitely many most general unifiers under (A) and (C)*** which is the subject of this paper.
Central to unification theory are the notions of a ***set of most general unifiers*** $\mu U\Sigma$ (traditionally: the set of base vectors spanning the solution space) and the ***hierarchy of unification problems*** based on $\mu U\Sigma$:

(i) a theory T is ***unitary*** if $\mu U\Sigma$ always exists and has at most one element;
(ii) a theory T is ***finitary*** if $\mu U\Sigma$ always exists and is finite;
(iii) a theory T is ***infinitary*** if $\mu U\Sigma$ always exists and $\mu U\Sigma$ is infinite for at least one pair of terms;
(iv) a theory T is of ***nullary*** otherwise.

We denote a ***unification problem*** under a theory T by

$$\langle s = t \rangle_T.$$

The field of unification theory, term rewriting systems and applications are surveyed in [HO 80] [Si 84] and most recently in [Si 86].

1.1 Associative Commutative Unification.

Terms under associativity and commutativity closely resemble the datastructure multisets (sets which may contain multiple occurrences of the same element), which is used in the matching of patterns (pattern directed invocation) in many programming languages of Artificial Intelligence (AI) (e.g. [Hw 72], [RD 72]). This pattern matching problem for multisets (often called ***bags*** in the AI-literature) was investigated by M. Stickel in [St 75], [St 76], who observed that this problem can be reduced to the problem of solving homogeneous linear diophantine equations over the positive integers ($\mathbb{N}$) with the additional proviso that only ***positive*** linear combinations of the solution set are admissible. His results were finally published in [St 81].
Certain equational axioms may force an ***automated theorem prover*** [Lo 78] [CL 73][WO 84] to go astray. G. Plotkin [Pl 72] showed how to build these troublesome axioms into the unification algorithm of a resolution [Ro 65] and paramodulation [RW 69] based theorem prover. Building upon this work, Livesey and Siekmann [LS 76] [LS 78] investigated the axioms of associativity (A) and commutativity (C), since they so frequently occur in applications of automated theorem proving. Independently of M. Stickel they also observed the close relationship between the AC-unification problem and the solving of linear diophantine equations. They proposed however a very different reduction (among other differences a reduction to ***inhomogeneous*** linear diophantine equations) which appears to have some advantages over the combinatorics of the "variable-abstraction" process in the STICKEL algorithm.
These results were never properly published, since an important problem remained open: the extension of the AC-unification algorithm to the whole class of first order terms. The suggestions for such an extension in [St 76] as well as the naive sketch of an extension proposed in [LS 76] turned out to be missing a crucial point [Boyer]: the subformulas of a term to be AC-unified can become longer, i.e. have more symbols, than the original term (see section 3.3 for an example) and hence the termination of the extended AC-unification procedure became a major problem, which remained open for many years. It was finally positively solved by F. Fages [Fa 83] [Fa 84] [Fa 85] using an ingenious complexity measure on AC-terms.
G. Huet [Hu 78] , A. Fortenbacher [Fo 83], D. Lankford [La 85] and W. Büttner [Bü 85] give efficient methods to solve homogeneous linear equations where only positive linear combinations are admissible, a problem originally investigated in [G 1873]. This is an important component of every AC-unification algorithm. A comparison of the algorithms of Huet and Fortenbacher and an extension of these algorithms to the case of

---

[Boyer]: This problem was first brought to our attention by Bob Boyer, now at University of Texas, in a private communication (while refereeing [LS 78]). We gratefully acknowledge this crucial hint as well as his many helpful suggestions.

inhomogeneous equations can be found in [GH 85].
J.M. Hullot [Ht 80], F. Fages [Fa 84] and Fortenbacher [Fo 83] [Fo 85] discuss computational improvements of the original STICKEL-algorithm. Recently another approach to AC-unification was proposed in [Ki 85].
G.E. Peterson and M.E. Stickel [PS 81] present a generalisation of the KNUTH-BENDIX completion algorithm for term rewriting systems [KB 70] based inter alia on AC-unification. The practical advantage of a special purpose AC-unification algorithm is particularily well demonstrated for term rewriting systems in [St 84].
Apart from interest in a practical and fast algorithm, which computes the set of unifiers there is the main theoretical observation that the set of most general unifiers is always ***finite*** for AC-unification problems. This fact was independently discovered in [St 75] and [LS 76]. However, since the set of most general unifiers (mgu) corresponds to the set of nonnegative solutions of certain linear diophantine equations, the finiteness of the set of mgu's follows immediately from a theorem of Dickson [Di 13] as demonstrated in section 2.2.

This paper improves on the original work of [LS 76] [LS 78] and also extends the algorithm to the whole class of first order terms using a modification of the FAGES-complexity measure in the proof of termination.

1.2 Definitions and Notation

Unification theory rests upon the notions of universal algebra (see e.g. [Gr 79] [BS 81]) with the familiar concept of an algebra $\mathcal{A} = (\mathbf{A}, \mathbf{F})$ where $\mathbf{A}$ is the ***carrier*** and $\mathbf{F}$ is a family of ***operators*** given with their arities. For a given ***congruence relation*** ϱ the quotient algebra modulo ϱ is written as $\mathcal{A}_{/\varrho} = (\mathbf{A}_{/\varrho}, \mathbf{F})$.
Assuming that there is at least one constant (operator of arity 0) in $\mathbf{F}$ and a denumerable set of variables $\mathbf{V}$, we define $\mathbf{T}$, the set of first order terms, over $\mathbf{F}$ and $\mathbf{V}$, as the least set with (i) $\mathbf{V} \subseteq \mathbf{T}$, and if arity(f) = 0 for $f \in \mathbf{F}$ then $f \in \mathbf{T}$ and (ii) if $t_1, \ldots, t_n \in \mathbf{T}$ and arity(f) = n then $f(t_1 \ldots t_n) \in \mathbf{T}$.
Let $\mathbf{V}(s)$ be the set of variables occuring in term s; a term s is ***ground*** if $\mathbf{V}(s) = \emptyset$.
As usual $\mathcal{T}$ denotes the algebra with carrier $\mathbf{T}$ and the operators are the term constructors corresponding to each operator of $\mathbf{F}$. $\mathcal{T}$ is called the absolutely free (term) algebra, i.e. it just gives an algebraic structure to $\mathbf{T}$. If the carrier is ground it is called the initial algebra [GT 78] or Herbrand Universe [Lo 78].
A ***substitution*** $\sigma: \mathbf{T} \to \mathbf{T}$ is is an endomorphism on $\mathcal{T}$, which is identical almost everywhere on $\mathbf{V}$ and hence can be represented as a finite set of pairs $\sigma = \{ x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n \}$. The restriction $\sigma|_V$ of a substitution σ to a set of

variables is defined as $\sigma|_V x = \sigma x$ if $x \in V$ and $\sigma|_V x = x$ otherwise.

Σ is the set of substitutions on T and ε the identity. The application of a substitution σ to a term $t \in T$ is written as σt. The composition of substitutions is defined as the usual composition of mappings : $(\sigma \cdot \tau)t = \sigma(\tau t)$ for $t \in T$.

Define	$DOM\sigma = \{x \in V : \sigma x \neq x\}$	(domain of σ)
	$COD\sigma = \{\sigma x : x \in DOM\sigma\}$	(codomain of σ)
	$VCOD\sigma = V(COD\sigma)$	(variables in codomain of σ)

If $VCOD\sigma = \emptyset$ then σ is a ***ground*** substitution.
A set of substitutions $\Sigma \subseteq \Sigma$ is said to be ***based on W away from*** $Z \supseteq W$ iff the following two conditions are satisfied

(i)	$DOM\sigma = W$	for all $\sigma \in \Sigma$
(ii)	$VCOD\sigma \cap Z = \emptyset$	for all $\sigma \in \Sigma$

For substitutions based on some W we have $DOM\sigma \cap VCOD\sigma = \emptyset$, which is equivalent to the idempotence of σ, i.e. $\sigma\cdot\sigma = \sigma$. We shall use this property in the proofs later on.
An ***equation*** $s = t$ is a pair of terms. For a set of equations T, the ***equational theory presented by*** T (in short: the equational theory T) is defined as the finest congruence $=_T$ on T containing all pairs $\sigma s = \sigma t$ for $s = t$ in T and σ in Σ.
(i.e. the Σ-invariant congruence relation generated by T).
Two terms s,t are T-equal if $s =_T t$. We extend T-equality in T to the set of substitutions Σ by:

$$\sigma =_T \tau \quad \text{iff} \quad \forall x \in V \quad \sigma x =_T \tau x .$$

If T-equality of substitutions is restricted to a set of variables W we write

$$\sigma =_T \tau\ [W] \quad \text{iff} \quad \forall x \in W \quad \sigma x =_T \tau x$$

and say σ and τ are ***T-equal in W***.
A substitution τ is ***more general*** than σ on W (or σ ***is a T-instance of*** τ on W):

$$\sigma \leq_T \tau\ [W] \text{ iff } \exists \lambda \in \Sigma \quad \sigma =_T \lambda\tau\ [W].$$

Two substitutions σ,τ are called T-equivalent on W

$$\sigma \equiv_T \tau\ [W] \text{ iff } \sigma \leq_T \tau\ [W] \text{ and } \tau \leq_T \sigma\ [W].$$

Given two terms s,t and an equational theory T a unification problem for T is denoted as

$$\langle s = t \rangle_T$$

We say $\langle s = t \rangle_T$ is T-unifiable iff there exists a substitution $\sigma \in \Sigma$ such that $\sigma s =_T \sigma t$ and we call σ a T-unifier of s and t. For the set of all T-unifiers of s and t we write $U\Sigma_T(s, t)$. Without loss of generality we assume unifiers σ of s and t to be idempotent (if not we can always find an equivalent one which is). For a given unification problem $\langle s = t \rangle_T$, it is unnecessary to compute the whole set of unifiers $U\Sigma_T(s, t)$, which is always recursively enumerable for a decidable theory T, but rather a smaller set useful in representing $U\Sigma_T$. Therefore we define $cU\Sigma_T(s, t)$, the ***complete set of unifiers of s and t*** on $W - \mathbf{V}(s,t)$ as:

(i) $cU\Sigma_T \subseteq U\Sigma_T$ (correcteness)

(ii) $\forall \delta \in U\Sigma_T \quad \exists \sigma \in cU\Sigma_T: \quad \delta \leq_T \sigma \, [W]$ (completeness)

The ***set of most general unifiers*** $\mu U\Sigma_T(s, t)$ is defined by (i), (ii) and

(iii) $\forall \sigma, \tau \in \mu U\Sigma_T: \quad \sigma \leq_T \tau \, [W]$ implies $\sigma = \delta$ (minimality).

For technical reasons we have the requirement as defined above: For a set of variables Z with $W \subseteq Z$

(iv) $\mu U\Sigma_T$ (resp. $cU\Sigma_T$) is based on W away from Z

If conditions (i) - (iv) are fulfilled we say $\mu U\Sigma_T$ is a ***set of most general unifiers away from*** Z [PL72].

The set $\mu U\Sigma_T$ does not always exist [FH 83] [Sc 86]; if it does then it is unique up to the equivalence $=_T$ (see [Hu 76] [FH 83]). For that reason it is sufficient to generate just one $\mu U\Sigma_T$ as a representative of the equivalence class $[\mu U\Sigma_T]_{=T}$. Sometimes it turned out to be useful to extend the relation $\leq_T [W]$ used in the definition of completeness and minimality to $\leq_T [X]$ with $W \subseteq X \subseteq Z$. This procedure is justified by the so called "Fortsetzungslemma" (extension lemma) and will often be used in the completeness proofs of our algorithms:

1.2 ***Lemma*** 1: For two idempotent substitutions θ_1, θ_2 let Z and $W \subseteq Z$ be sets of variables be with $DOM\theta_2 = W$ and $VCOD\theta_2 \cap Z = \emptyset$. Then $\theta_1 \leq_T \theta_2 [W]$ iff $\theta_1 \leq_T \theta_2 [Z]$.

Proof: Let $V = Z \setminus W$ be the extension of the validity domain. By assumption there exists λ_W with $\theta_1 =_T \lambda_W\theta_2 [W]$. Since $VCOD\theta_2 \cap Z = \emptyset$ we can find λ_W such that for all $x \in V$ $\lambda_W x = x$. Define $\lambda_V = \{ x \leftarrow \theta_1 x \mid x \in V \} = \theta_1|_V$ and let $\lambda = \lambda_V\lambda_W$. Then for $x \in W$ we have $\lambda\theta_2 x =_T \lambda_V\lambda_W\theta_2 x = \lambda_V\theta_1 x = \theta_1 x = \theta_1 x$ by definition of λ_V and λ_W and the idempotence of θ_1. For $x \in V$ $\lambda\theta_2 x = \lambda_V\lambda_W\theta_2 x =_T \lambda_V x = \theta_1 x$ since $\lambda_W x = x$ for $x \in V$ and $DOM\theta_2 \cap V = \emptyset$. Hence $\theta_1 =_T \lambda\theta_2 [V \cup W = Z]$, i.e. $\theta_1 \leq_T \theta_2 [Z]$. The other direction is trivial. ■

Another technical lemma about composition of substitutions from the right useful in some proofs later on is:

1.2 ***Lemma*** 2: For idempotent substitutions δ, σ, τ and a set of variables V with $DOM\tau = \sigma(V)$ and $VCOD\tau \cap (\sigma(V) \cup VCOD\sigma \cup V) = \emptyset$:

$DOM\tau\sigma = V \cup DOM\sigma$ and

if $\delta \leq_T \tau [\sigma(V)]$ then $\delta\sigma \leq_T \tau\sigma [DOM\tau\sigma]$.

Proof: With the previous lemma we get $\delta \leq_T \tau [\sigma(V \cup DOM\sigma)]$ and hence $\delta\sigma \leq_T \tau\sigma [V \cup DOM\sigma]$ and $DOM\tau\sigma = V \cup DOM\sigma$.

2. UNIFICATION IN ABELIAN MONOIDS

Let a family of operators $\mathbf{F}_*$ consist of denumerably many constants $\mathbf{C}$ (0-ary functions, written as a b c ... a_1 b_2 ...), one distinguished constant $1 \in \mathbf{C}$ (the unit) and one binary function symbol $*$. Let $\mathbf{V}$ be a denumerable set of variables (as before denoted by x y z ... x_1 y_2 ...).
Let $\mathbf{T}_*$ be the set of terms over $\mathbf{F}_*$ and $\mathbf{V}$ (in infix notation) and $\mathcal{T}_*$ the corresponding term algebra. With the equational theory

$$AC1 = \{x*y = y*x,\ x*1 = x,\ (x*y)*z = x*(y*z)\}$$

define $=_{AC1}$ as the Σ-invariant congruence relation generated by AC1 in $\mathcal{T}_*$. In this section we are interested in solving equations in the quotient algebra $\mathcal{T}_{*/=AC1}$ modulo $=_{AC1}$, i.e. the interest is in AC1-unification problems $\langle s = t \rangle_{AC1}$ where $s, t \in \mathbf{T}_*$.
For ease of presentation we drop the $*$'s and the parentheses in this section and represent the elements of $\mathbf{T}_*$ as strings over $\mathbf{C} \cup \mathbf{V}$.
An AC1-unification problem $\langle s = t \rangle_{AC1}$ is said to be ***normalized*** to $\langle s' = t' \rangle_{AC1}$ iff the common symbols in s and t are eliminated pair by pair and s', t' either do not contain the unit 1. If one of the strings becomes empty it is set to 1.
Since the semigroup of abelian strings is isomorphic to the free commutative monoid and free monoids are left and right reducable the set of unifiers $U\Sigma_{AC1}(s, t)$ is the same as $U\Sigma_{AC1}(s', t')$. Hence we always assume that AC1-unification problems are normalized.

2.1 The Unification Algorithm

To introduce the algorithm from the point of view of an intelligent human with a blackboard and chalk we use the following example:

(E1) $\langle xxxyyyaabew = wzcde1 \rangle_{AC1}$

where $x,y,z,w \in \mathbf{V}$ and $a,b,c,d,e,1 \in \mathbf{C}$.
A more convenient notation for the normalized problem is:

$$\langle x^3y^2a^2b = zcd \rangle_{AC1}.$$

(E1) has among others $\delta = \{x \leftarrow abcd, y \leftarrow a^3bcd, z \leftarrow a^{11}b^6c^4d^4\}$ as a unifier. Also the following infinite chain of substitutions unifies (E1):

$$\delta_n = \{x \leftarrow abcd, y \leftarrow a^nbcd, z \leftarrow a^{5+2n}b^6c^4d^4\} \text{ for } n = 1, 2, 3, \ldots .$$

However for this chain of unifiers there exists an upper bound:

$$\delta_{max} = \{ (y \leftarrow ucd)(z \leftarrow x^3u^2a^2bcd) \}, \text{ with } \delta_n \leq_{AC1} \delta_{max}[W] \text{ for } n = 1,2,3,\ldots$$

where u is a new variable not in $W = V(x^3y^2a^2bew, wzde1)$.
The reader may want to construct the corresponding λ_n with

$$\delta_n =_{AC1} \lambda_n \cdot \delta_{max}[W] \text{ for } W = \{x, y, z, w\}$$

for himself, since this demonstrates the main observation of this paper: the composition of substitutions corresponds to the addition of integers.

2.1.1. First demonstration of the Main Idea

Returning to example (E.1) we ask for the general form of a most general unifying substitution δ: firstly δ will only move the variables already occurring in the two strings, in this case x,y and z respectively. Hence it will have the general form:

$$\delta = \{x \leftarrow t_1, y \leftarrow t_2, z \leftarrow t_3\} \text{ for certain } t_i \in T_*, \ i = 1, 2, 3.$$

Secondly δ will only substitute constants already occurring in the unification problem (since otherwise it can not be most general). And finally δ may substitute variables already occurring in the two strings and/or it may substitute 'new' variables.
For simplicity let δ introduce only 'new' variables; i.e. $\mu U\Sigma$ is the set of most general unifiers away from Z as defined in section 1.2. Hence a unifier for (E1) has the general form:

$$\begin{aligned} \delta = \{ & x \leftarrow u^{m_{11}} v^{m_{21}} a^{n_{11}} b^{n_{21}} c^{n_{31}} d^{n_{41}}, \\ & y \leftarrow u^{m_{12}} v^{m_{22}} a^{n_{12}} b^{n_{22}} c^{n_{32}} d^{n_{42}}, \\ & z \leftarrow u^{m_{13}} v^{m_{23}} a^{n_{13}} b^{n_{23}} c^{n_{33}} d^{n_{43}} \} \end{aligned} \qquad \text{(E2)}$$

for appropriate $m_{ik}, n_{ik} \in \mathbb{N}_0$ (assuming $s^0 = 1$), where u, v are new variables.
The unification problem is now reduced to the problem of finding appropriate values for the m_{ik}, n_{ik}. Before presenting a method of how to find such values

there is the question of how many new variables are needed: in the above example just two new variables u and v are sufficient. The general answer is, that we need as many new variables as the dimension (number of independent solution vectors) of the solution space for the corresponding homogeneous equation system, to be presented below.

In order to determine appropriate values for the m_{ik}, n_{ik} we observe the following: for each symbol ***the number of occurrences of that symbol in δt_1 must be the same as the number of occurrences in δt_2***

This fact completely determines the m_{ik}, n_{ik} since we can now set up a diophantine equation for each symbol:

(E3)

$$\begin{array}{lllll} \text{for u:} & 3\,m_{11} + 2\,m_{12} & & = m_{13} & \\ \text{for v:} & 3\,m_{21} + 2\,m_{22} & & = m_{23} & \\ \text{for a:} & 3\,n_{11} + 2\,n_{12} & + 2 & = n_{13} & \\ \text{for b:} & 3\,n_{21} + 2\,n_{22} & + 1 & = n_{23} & \text{for } m_{ik}, n_{ik} \in \mathbf{N}_0 \\ \text{for c:} & 3\,n_{31} + 2\,n_{32} & & = n_{33} + 1 & \\ \text{for d:} & 3\,n_{41} + 2\,n_{42} & & = n_{43} + 1 & \end{array}$$

That is for each variable we obtain a homogeneous equation and for each constant an inhomogeneous one. But note that the homogeneous part is the same for all equations. The following values for the m_{ik}, n_{ik} are an example for a solution of the above equation system:

(E4)

$$\begin{array}{llll} \text{for u:} & m_{11} = 1, & m_{12} = 1, & m_{13} = 5, \quad \text{say } \chi_u = (1,1,5) \\ \text{for v:} & m_{21} = 1, & m_{22} = 0, & m_{23} = 3, \quad \text{say } \chi_v = (1,0,3) \\ \text{for a:} & n_{11} = 0, & n_{12} = 1, & n_{13} = 4, \quad \text{say } \chi_a = (0,1,4) \\ \text{for b:} & n_{21} = 1, & n_{22} = 0, & n_{23} = 4, \quad \text{say } \chi_b = (1,0,4) \\ \text{for c:} & n_{31} = 1, & n_{32} = 1, & n_{33} = 4, \quad \text{say } \chi_c = (1,1,4) \\ \text{for d:} & n_{41} = 0, & n_{42} = 2, & n_{43} = 3, \quad \text{say } \chi_d = (0,2,3). \end{array}$$

Substituting these values into (E2) above, we obtain an actual unifier:

$$\delta = \{ x \leftarrow uvbc,\ y \leftarrow uacd^2,\ z \leftarrow u^5v^3a^4b^4c^4d^3 \}$$

How do we obtain ***all*** unifiers and how do we obtain the most general ones? In section 2.2 it is shown that every solution of a linear homogeneous diophantine equation in $\mathbf{N}_0$ is a ***positive*** linear combination of certain base vectors

spanning the solution space for this homogeneous equation. It is also shown that every solution of an inhomogeneous linear diophantine equation in $\mathbf{N}_0$ is a combination of a special solution which is minimal in some sense and a solution for the homogeneous part.
The base vectors spanning the solution space for the homogeneous equations in (E3) are:

(E5) $$m_1 = (0,1,2) = (m_{11}, m_{12}, m_{13})$$
$$m_2 = (1,0,3) = (m_{21}, m_{22}, m_{23})$$

and a special solution for each inhomogeneous equation is:

(E6)
$$\text{for a: } n_a = (0,0,2) = (n_{11}, n_{12}, n_{13})$$
$$\text{for b: } n_b = (0,0,1) = (n_{21}, n_{22}, n_{23})$$
$$\text{for c: } n_c = (0,1,1) = (n_{31}, n_{32}, n_{33})$$
$$\text{for d: } n_d = (0,1,1) = (n_{41}, n_{42}, n_{43})$$

By taking for each constant one special solution we can construct a ***most general unifier*** σ: taking the vectors n_a, n_b, n_c, n_d, as well as two new variables z_1, z_2 and substituting the appropriate values of the m_{ik} and n_{ik} into (E2) we obtain :

$$\sigma = \{ x \leftarrow z_2,\ y \leftarrow z_1cd,\ z \leftarrow z_2z_1a^2bcd \}.$$

Why is the previously constructed unifier δ an instance of σ? As mentioned above the solutions (E4) can be represented using the base solutions (E5) and the special solutions (E6):

(E7)
$$x_u = 1\,m_1 + 1\,m_2 = (1,1,5)$$
$$x_v = 0\,m_1 + 1\,m_2 = (1,0,3)$$
$$x_a = n_a + 1\,m_1 + 0\,m_2 = (0,1,4)$$
$$x_b = n_b + 0\,m_1 + 1\,m_2 = (1,0,4)$$
$$x_c = n_c + 0\,m_1 + 1\,m_2 = (1,1,4)$$
$$x_d = n_d + 1\,m_1 + 0\,m_2 = (0,2,3)$$

Now just as the particular solutions x_u to x_d are obtained from the m_1, m_2, n_a ,..., n_d the particular unifier δ is obtained from σ as:

$$\delta = \lambda \cdot \sigma [\{ x,y,z \}]$$

with $\lambda = \{ z_1 \leftarrow uad,\ z_2 \leftarrow uvbc \}$, where λ can be computed directly from the linear combination of vectors in (E7).

Hence any unifier can be obtained from a most general unifier in just the same way as any solution of the equation system (E3) is obtained from some special solutions of the inhomogeneous equations and from the basis solutions of the underlying homogeneous equations. However, one cannot simulate the subtraction of vectors by composition of substitutions, and also negative powers of constants are not defined. Thus we have to solve the equations in positive integers and only positive linear combinations of these vectors are allowed. This leads to the following algebraic problems:

(P1) Given a linear diophantine homogeneous equation system, does there exist a *finite* base of independent positive solution vectors such that every solution to this equation system is a *positive* linear combination of the base vectors?

And secondly:

(P2) Can the positive solutions to an inhomogeneous equation system be obtained as a *positive* linear combination of *one* special solution with the set of base vectors for the corresponding homogeneous equation?

In section 2.2 it is shown that the answer is essentially positive: the only complications are that we need a larger (as compared to the solutions in the rational numbers $\mathbf{Q}$), but still finite set of base vectors for (P1) and we have to consider more than one special solution (but still only finitely many) in order to solve (P2).

2.1.2. The AC1 - Algorithm

Following the outline of the previous paragraph, we shall now give a fomal presentation of the algorithm. The input is an AC1 - unification problem represented as:

(G1) $\langle v_1^{p_1} v_2^{p_2} \ldots v_K^{p_K} c_1^{q_1} \ldots c_l^{q_l} = v_{K+1}^{p_{K+1}} \ldots v_M^{p_M} c_{l+1}^{q_{l+1}} \ldots c_L^{q_L} \rangle_{AC1}$

where $\{v_1, \ldots, v_M\} \subseteq \mathbf{V}$, $\{c_1, \ldots, c_L\} \subseteq \mathbf{C}$ and $p_i, q_k \in \mathbf{N}$.
The output of the algorithm is a finite set of unifiers each of which is represented as a matrix (m_{ik}, n_{ik}) of non negative integers as follows. Let N be the dimension of the solution space of the homogeneous equation and let $u_1, \ldots, u_N$ be the new variables to be substituted. A unifier for (G1) is then represented as:

(G2)

	v_1	v_2	...	v_M
u_1	m_{11}	m_{12}	...	m_{1M}
u_2	m_{21}	m_{22}	...	m_{2M}
.	.	.		.
.	.	.		.
.	.	.		.
u_N	m_{N1}	m_{N2}	...	m_{NM}
c_1	n_{11}	n_{12}	...	n_{1M}
c_2	n_{21}	n_{22}	...	n_{2M}
.	.	.		.
.	.	.		.
.	.	.		.
c_L	n_{L1}	n_{L2}	...	n_{LM}

Note that this is just the matrix of (E2) such that each column represents a component of the unifier:

$$\delta = \{ v_i \leftarrow u_1^{m_{1i}} \ldots u_N^{m_{Ni}} c_1^{n_{1i}} \ldots c_L^{n_{Li}} ; 1 \le i \le M \}$$

The following two auxiliary functions DIOHOM and DIOINHOM set up and solve a homogeneous and an inhomogeneous equation respectively. They are separated out since they are the fundamental algorithms for our unification process whose efficiency may be improved independently (see [GH 85]).
The homogeneous equation arising from (G1) (say for u_i) is

(G3HOM) $$p_1 m_{i1} + p_2 m_{i2} + \ldots + p_K m_{iK} = p_{K+1} m_{iK+1} + \ldots + p_M m_{iM}$$

and the inhomogeneous equation (say for c_j, $1 \le j \le L$ and w.l.o.g. $1 \le j$) is:

(G3INHOM) $$p_1 n_{j1} + p_2 n_{j2} + \ldots + p_K n_{jK} = p_{K+1} n_{jK+1} + \ldots + p_M n_{jM} + q_j.$$

FUNCTION DIOHOM

INPUT: An AC1 - Problem like (G1).

STEP 1: Compute the homogeneous equation as in (G3HOM)
STEP 2: Solve (G3HOM) by the currently most efficient algorithm (see section 2.2).

OUTPUT: The set of independent M - dimensional base vectors $\{m_1, m_2, ..., m_N\}$ spanning the solution space or $\{0\}$ if (G3HOM) has only the trivial solution

ENDOF DIOHOM.

The next function computes the set of minimal solutions of an inhomogeneous equation as described in section 2.2: instead of just one we need several special solutions of the inhomogeneous equation.

FUNCTION DIOINHOM

INPUT: 1. An AC1 - Problem like (G1).
2. A constant c_j with $1 \leq j \leq L$

STEP 1: Compute the inhomogeneous equation as in (G3INHOM) for c_j
STEP 2: Solve (G3INHOM) by the currently most efficient algorithm (see section 2.2).

OUTPUT: The set of minimal M-dimensional solutions $S_{Cj} = \{n_1, ..., n_k\}$ or $\emptyset$ if (G3INHOM) has no solutions

ENDOF DIOINHOM.

This gives the final algorithm:

FUNCTION AC1-UNIFY

INPUT: An AC1-Problem like (G1).

STEP 0: G1 := NORMALIZE(G1)
STEP 1: $(m_1, m_2, ..., m_N)$ = DIOHOM(G1)
STEP 2: if L = 0 then $\mu U\Sigma_0$:= CONSTRUCT$(m_1, ..., m_N)$
else
STEP 3: PRODUCT := DIOINHOM(G1, c_1) × ... × DIOINHOM(G1, c_L)
STEP 4: $\mu U\Sigma_0$:= $\emptyset$;
forall $(n_1, n_2, ..., n_L) \in$ PRODUCT do
$\mu U\Sigma_0$:= $\mu U\Sigma_0 \cup$ {CONSTRUCT$(m_1, ..., m_N, n_1, ..., n_L)$}
od

OUTPUT: The set of most general unifiers $\mu U\Sigma_0$

ENDOF AC1-UNIFY.

The AC1-unification algorithm uses two auxiliary functions: the function ×, which computes the Cartesian product of two sets, and the function CONSTRUCT. Function CONSTRUCT takes N + L vectors as input,

$$\begin{bmatrix} m_1 \\ . \\ m_N \\ n_1 \\ . \\ n_2 \end{bmatrix}$$

transforms them into the matrix (see G2) and then computes the usual "set of pairs" - representation of a unifier.
Of course we still have to show that the name $\mu U\Sigma_0$ of the output set is justified, i.e. that indeed we are computing the most general set of unifiers. While this is the subject of paragraph 2.3 we shall first look at another example.

2.1.3. Another Example

Let $\langle x^2yac = b^2zc1 \rangle_{AC1}$ be an AC1-Problem, which we shall follow through the algorithm AC1-UNIFY.

STEP 0: $\langle x^2ya = b^2z \rangle_{AC1}$ is the normalized form

STEP 1: The homogeneous equation is

$$2m_{11} + m_{12} - m_{13} = 0;$$

The solution space is spanned by

$$m_1 = (0, 1, 1)$$

$$m_2 = (1, 0, 2)$$

STEP 2: There are two constants, hence L = 2.

The equations are

$$2n_{31} + n_{32} - n_{33} = -1 \text{ for } a$$

$$2n_{41} + n_{42} - n_{43} = 2 \text{ for } b$$

$S_a = \{ (0,0,1) \}$

$S_b = \{ (1,0,0), (0,2,0) \}$, hence

PRODUCT $= S_a \times S_b$

$= \{ ((0,0,1), (1,0,0)), ((0,0,1), (0,2,0)) \}$

STEP 3: Since there are two elements in PRODUCT, i. e. N = L, we have two matrices (m_1, m_2, n_1, n_2) and (m_1, m_2, n_1', n_2'), where (n_1, n_2) and (n_1', n_2') are the elements in PRODUCT. The actual values of the transformed matrices are (note that we need two new variables u and v):

	x	y	z	
u	0	1	1	m_1
v	1	0	2	m_2
a	0	0	1	n_1
b	1	0	0	n_2

	x	y	z	
u	0	1	1	m_1
v	1	0	2	m_2
a	0	0	1	n_1'
b	0	2	0	n_2'

This gives the two unifiers $\mu U\Sigma_0 = \{\sigma_1, \sigma_2\}$ with:

$$\sigma_1 = \{ x \leftarrow v_1 b, \quad y \leftarrow u_1, \quad z \leftarrow u_1 v_1^2 a \}$$

$$\sigma_2 = \{ x \leftarrow v_2, \quad y \leftarrow u_2 b^2, \quad z \leftarrow u_2 v_2^2 a \}.$$

2.2 Linear Equations over $\mathbb{N}_0$

Before we prove correctness, completeness and minimality of $\mu U\Sigma_0$, the set of unifiers returned by AC1-UNIFY, we will first have to show that the algorithms DIOHOM and DIOINHOM are welldefined.

Given an inhomogeneous linear diophantine equation

(E_b) $\quad a_1x_1 + \ldots + a_nx_n = b \qquad\qquad a_i, b \in \mathbf{Z}$ for $1 \le i \le n$

and the corresponding homogeneous equation

(E_0) $\quad a_1x_1 + \ldots + a_nx_n = 0 \qquad\qquad a_i \in \mathbf{Z}$ for $1 \le i \le n$

we are interested in finding all positive integer solutions of (E_b)

$$S_b = \{ \mathbf{y} = (y_1, \ldots, y_n) \mid a_1y_1 + \ldots + a_ny_n = b \text{ and } y_i \ge 0 \}$$

respectively all nontrivial positive integer solutions of (E_0)

$$S_0 = \{ \mathbf{y} = (y_1, \ldots, y_n) \mid a_1y_1 + \ldots + a_ny_n = 0 \text{ and } y_i \ge 0 \} \setminus \{(0, \ldots, 0)\}.$$

Let G_0 be the set of all solutions of (E_0) in $\mathbf{Z}$ then G_0 is a subgroup of $G = \mathbf{Z}^n$ the free Abelian group on n generators. Then $S_0 = G_0 \cap (F \setminus \mathbf{0})$ where $F = \mathbf{N}^n$ is the free Abelian semigroup on n generators and $\mathbf{0}$ the unit in F. Hence by Corollary 9.19 in [CP 67] S_0 (if non-empty) is a finitely generated subsemigroup of F. The basis of S_0 is the set of all minimal elements in S_0 with respect to the order

$$\mathbf{x} = (x_1, \ldots, x_n) \le \mathbf{y} = (y_1, \ldots, y_n) \text{ iff } x_i \le y_i \text{ for } 1 \le i \le n.$$

The set M of minimal elements of a set S with respect to $\le$ is defined such that for all $s \in S$ there exists $m \in M$ with $m \le s$ and for all $m \in M$ if there exists $s \in S$ with $s \le m$ then $m = s$.
Let $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ be the basis of S_0 then $S_0 = \{\mathbf{x} \mid \mathbf{x} = b_1\mathbf{b}_1 + \ldots + b_k\mathbf{b}_k, b_i \ge 0\}$.
Hence we have the following result which was first shown in [G 1873]:

2.2 ***Theorem*** 1: The set of positive integer solutions S_0 of a homogeneous linear diophantine equation is positively generated by the finite set of minimal (with respect to $\le$) elements of S_0.

In order to obtain the solutions for the inhomogeneous equation (E_b) let M_b be the set of the minimal solutions of (E_b) with respect to $\le$. By Theorem 9.18 of [CP 67] which is a consequence of a theorem of Dickson [Di 13] we have the important result that M_b is finite.

In summary, the following theorem shows how to compute S_b in principle:

2.2 ***Theorem*** 2: The set of positive integer solutions S_b of an inhomogeneous linear diophantine equation is

$$S_b = \{ y \mid y = x_b + x_0 \text{ with } x_b \in M_b \text{ and } x_0 \in S_0 \cup \{0\} \}.$$

Proof: It is easy to see that $x_b + x_0$ is in S_b. Conversely let $y \in S_b$ then if y is in M_b we are done with $x = 0$. Suppose y is not in M_b then there exists by definition x_b in M_b with $x_b \leq y$. Therefore we have $y - x_b \geq 0$ and hence $y - x_b = x_0 \in S_0$ or $y = x_b + x_0$. ■

As computation is not done "in principle" there is the important problem to find the most efficient algorithm computing B and M_b. Currently three algorithms are known which compute the basis of the solutions of a homogeneous equation [Hu 78][Fo 83] and [La 85]. In [GH 85] the algorithms of Huet and Fortenbacher are compared and extended to compute the minimal set M_b for an inhomogeneous equation as well.

2.3. Correctness, Completeness and Minimality

We shall show that $\mu U\Sigma_0$, the output of AC1-UNIFY, is indeed the intended set of most general unifiers $\mu U\Sigma_{AC1}(s, t)$.

2.3. ***Theorem*** 1: AC1-UNIFY terminates.

Proof: This is trivial, provided the two functions DIOHOM and DIOINHOM terminate and always return a *finite* set, which was shown in the previous paragraph. ■

2.3. ***Theorem*** 2: $\mu U\Sigma_0$ is correct.

Proof: Let $\sigma = \{ v_i \leftarrow u_1^{m_{1i}} \dots u_N^{m_{Ni}} c_1^{n_{1i}} \dots c_L^{n_{Li}},\ 1 \leq i \leq M \}$ be a substitution in $\mu U\Sigma_0$. As in 2.1.2 (G1) assume

(i) $s = v_1^{p_1} v_2^{p_2} \dots v_K^{p_K} c_1^{q_1} \dots c_l^{q_l}$

(ii) $t = v_{K+1}^{p_{K+1}} \dots v_M^{p_M} c_{l+1}^{q_{l+1}} \dots c_L^{q_L}$

We have to show that σ unifies s and t, i.e. in σs and σt there occur the same number of new variables u_n, $1 \leq n \leq N$ and constants c_i, $1 \leq i \leq L$. Define $\#(p,q)$ as the number of occurrences of the term p in the term q. Now for $1 \leq n \leq N$ we have

$$\#(u_n, \sigma s) = \sum_{j=1}^{K} p_j\, m_{nj} \qquad \text{from (i)}$$

$$= \sum_{j=K+1}^{M} p_j\, m_{nj} \qquad \text{from (ii) and the fact that it is a solution of (G3HOM)}$$

$$= \#(u_n, \sigma t).$$

Similarly we have for $1 \le i \le L$ (w.l.o.g. $i \le I$)

$$\#(c_i, \sigma s) = \sum_{j=1}^{K} p_j\, n_{ij} + q_i \qquad \text{from (i)}$$

$$= \sum_{j=K+1}^{M} p_j\, n_{ij} \qquad \text{from (ii) and the fact that it is a solution of (G3INHOM)}$$

$$= \#(c_i, \sigma t).$$

Hence σ is a correct unifier, since it substitutes exactly the same number of symbols into each side of the AC1-problem. ■

2.3. ***Theorem*** 3: $\mu U\Sigma_0$ is a complete set of unifiers.

Proof: Let the given AC1-unification problem be of the form:

$$(3.1) \qquad \langle v_1^{p_1} v_2^{p_2} \ldots v_K^{p_K} c_1^{q_1} \ldots c_I^{q_I} = v_{K+1}^{p_{K+1}} \ldots v_M^{p_M} c_{I+1}^{q_{I+1}} \ldots c_L^{q_L} \rangle_{AC1}$$

as in (G1) and for simplicity assume it is normalized. Let δ be an arbitrary unifier for (3.1) and let $W = \{v_1, v_2, \ldots, v_M\}$. W.l.o.g. we assume that $VCOD\delta \cap W = \emptyset$ and $DOM\delta \subseteq W$. Let $s_1, s_2, \ldots, s_H$ be the symbols in $COD\delta$ different from the constants $c_1, \ldots, c_L$.

We represent $\delta = \{v_i \leftarrow s_1^{M_{i1}} \ldots s_H^{M_{Hi}} c_1^{N_{1i}} \ldots c_L^{N_{Li}},\ 1 \le i \le M\}$ in matrix form:

(3.3)

	v_1	v_2	...	v_M	
s_1	M_{11}	M_{12}	...	M_{1M}	$= \mathfrak{M}_1$
s_2	M_{21}	M_{22}	...	M_{2M}	$= \mathfrak{M}_2$
.	.	.		.	
.	.	.		.	
s_H	M_{H1}			M_{HM}	$= \mathfrak{M}_H$
c_1	N_{11}	N_{12}	...	N_{1M}	$= \mathfrak{N}_1$
c_2	N_{21}		...	N_{2M}	$= \mathfrak{N}_2$
.	.	.		.	
.	.	.		.	
c_L	N_{L1}	N_{L2}	...	N_{LM}	$= \mathfrak{N}_L$

We want to show that there exists $\sigma \in \mu U \Sigma_0$ and some $\lambda \in \Sigma$ such that $\delta =_{AC1} \lambda \cdot \sigma$ [W]. The homogeneous equation for (3.1) is:

$$(3.4) \qquad \sum_{j=1}^{K} p_j x_j = \sum_{j=K+1}^{M} p_j x_j$$

and the inhomogeneous equation for each c_i, $1 \le i \le L$ and $i \le I$, is

$$(3.5) \qquad \sum_{j=1}^{K} p_j y_j + q_i = \sum_{j=K+1}^{M} p_j y_j$$

As δ is a unifier for (3.1) the number of occurrences of each symbol in δs is the same as in δt; i.e. the vectors $\mathfrak{M}_1, \mathfrak{M}_2, \ldots, \mathfrak{M}_H$ solve equation (3.4). Similarily the vectors $\mathfrak{N}_i$, $1 \le i \le L$, solve equation (3.5) for each c_i.
Hence by 2.2 Theorem 1 the vectors $\mathfrak{M}_i$ can be represented as:

$$(3.6) \qquad \mathfrak{M}_h = a_{h1}\mathfrak{m}_1 + a_{h2}\mathfrak{m}_2 + \ldots + a_{hN}\mathfrak{m}_N \qquad 1 \le h \le H, \text{ for some } a_{hi} \in \mathbf{N}_0$$

where $\{ \mathfrak{m}_1, \ldots, \mathfrak{m}_N \}$ are the vectors computed by DIOHOM.
By 2.2 Theorem 2 the vectors $\mathfrak{N}_i$ can be represented as:

$$(3.7) \qquad \mathfrak{N}_i = b_{i1}\mathfrak{m}_1 + b_{i2}\mathfrak{m}_2 + \ldots + b_{iN}\mathfrak{m}_N + \mathfrak{n}_i \qquad 1 \le i \le L, \text{ for some } b_{ik} \in \mathbf{N}_0$$

where $\mathfrak{n}_i$ is an element in S_{ci} as computed in DIOINHOM.

Define $\mathfrak{a}_h = (a_{h1}, a_{h2}, \ldots, a_{hN})$ for $1 \le h \le H$

$\mathfrak{b}_i = (b_{i1}, b_{i2}, \ldots, b_{iN})$ for $1 \le i \le L$

Let $\sigma \in \mu U\Sigma_0$ be the substitution corresponding to the matrix $\sigma = (m_1, \ldots, m_N, n_1, n_2, \ldots, n_L)$, which must be computed in STEP 3. Define λ as the substitution corresponding to the matrix $\lambda = (\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_H, \mathfrak{b}_1, \ldots, \mathfrak{b}_L)$.

$\lambda =$

	u_1	u_2	...	u_N
s_1	a_{11}	...		a_{1N}
.	.			.
.	.			.
s_H	a_{H1}	...		a_{HN}
c_1	b_{11}	...		b_{1N}
.	.			.
.	.			.
c_L	b_{L1}	...		b_{LN}

$\sigma =$

	v_1	v_2	...	v_M
u_1	m_{11}	...		m_{1M}
.	.			.
.	.			.
u_N	m_{N1}	...		m_{NM}
c_1	n_{11}	...		n_{1M}
.	.			.
.	.			.
c_L	n_{L1}	...		n_{LM}

We now compute the composition of λ and σ:

$\lambda \cdot \sigma = (\mathfrak{a}_1, \ldots, \mathfrak{a}_H, \mathfrak{b}_1, \ldots, \mathfrak{b}_L) \cdot (m_1, \ldots, m_N, n_1, \ldots, n_L)$ by definition

$= \{u_k \leftarrow s_1^{a_{1k}} \ldots s_H^{a_{Hk}} c_1^{b_{1k}} \ldots c_L^{b_{Lk}} \mid 1 \le k \le N\}$

$\cdot \{v_i \leftarrow u_1^{m_{1i}} \ldots u_N^{m_{Ni}} c_1^{n_{1i}} \ldots c_L^{n_{Li}} \mid 1 \le i \le M\}$

$= \{v_i \leftarrow s_1^{\alpha_{1i}} \ldots s_p^{\alpha_{pi}} c_1^{\beta_{1i}} \ldots c_L^{\beta_{Li}} \mid 1 \le i \le M\}$ [W]

where $\alpha_{hi} = m_{1i}a_{h1} + m_{2i}a_{h2} + \ldots + m_{Ni}a_{hN} - M_{hi}$ for $1 \le i \le M$; $1 \le h \le H$;

and $\beta_{ji} = m_{1i}b_{j1} + m_{2i}b_{j2} + \ldots + m_{Ni}b_{jN} + n_{ji} = N_{ji}$ for $1 \le i \le M$; $1 \le j \le L$

$= (\mathfrak{M}_1, \mathfrak{N}_2, \ldots, \mathfrak{M}_H, \mathfrak{N}_1, \mathfrak{N}_2, \ldots, \mathfrak{N}_L)$

$= \delta$ ■

2.3. ***Theorem*** 4: $\mu U\Sigma_0$ is a minimal set of unifiers.

Proof: Let $\sigma, \tau \in \mu U\Sigma_0$ with $\sigma \not\equiv_{AC1} \tau$ [W]. Let $(m_1, \ldots, m_N, n_1, \ldots, n_L)$ and $(m_1, \ldots, m_N, n_1', \ldots, n_L')$ be the matrices corresponding to σ and τ. Since

$\sigma \neq_{AC1} \tau$ [W] there is a constant c_i, $1 \le i \le L$, such that $n_i \neq n_i'$. Suppose by contradiction there exists $\lambda \in \Sigma$ such that $\sigma =_{AC1} \lambda \cdot \tau$ [W].

Claim: If $\sigma =_{AC1} \lambda \cdot \tau$ then $n_i = n_i' + \sum_{i=1}^{N} a_i m_i$ for some $a_i \in \mathbf{N}_0$

Proof: Let $DOM\sigma = DOM\tau = \{v_1, \ldots, v_M\}$ and w.l.o.g. $DOM\lambda = VCOD\tau = \{u_1, \ldots, u_N\}$ and for $1 \le j \le M$ we have, where $\#(c_i, t_j)$ is the number of occurrences of c_i in t_j:

$$\#(c_i, \sigma v_j) = \#(c_i, \lambda\tau v_j)$$

$$= \#(c_i, \tau v_j) + \sum_{k=1}^{N} \#(c_i, \lambda u_k) \cdot \#(u_k, \tau v_j)$$

with

$$\#(c_i, \sigma v_j) = n_{ij} \qquad j = 1, \ldots, M$$

$$\#(c_i, \tau v_j) = n_{ij}' \qquad j = 1, \ldots, M$$

$$\#(c_i, \lambda u_k) = a_k \qquad k = 1, \ldots, N$$

$$\#(u_k, \tau v_j) = m_{k,j} \qquad k = 1, \ldots, N, j = 1, \ldots, M$$

and

$$n_i = (n_{i1}, \ldots, n_{iM})$$

$$n_i' = (n_{i1}', \ldots, n_{iM}')$$

$$m_k = (m_{k1}, \ldots, m_{kM}) \qquad k = 1, \ldots, N$$

Substituting these values yields

$$n_i = n_i' + \sum_{k=1}^{N} l_k m_k.$$

But now $n_i' \le n_i$ and $n_i \neq n_i'$, which is a contradiction since DIOINHOM only computes a minimal set (see paragraph 2.2). ■

2.4 AC-Unification without an Identity Element

If the Abelian semigroup does not have an identity element we compute the AC-unifiers as follows: first we apply the previous algorithm and afterwards the AC-unifiers are derived from the AC1-unifiers by a trivial process of variable elimination.

Since every AC-unifier is trivially an AC1-unifier we have the following obvious relationship where $\mathbf{C}(COD\sigma)$ is the set of constants, occurring in the

codomain of σ, s and t are normalized and different from 1.

2.4 ***Lemma*** 1: $\sigma \in U\Sigma_{AC1}(s, t)$ and $1 \notin \mathbf{C}(COD\sigma)$ iff $\sigma \in U\Sigma_{AC}(s, t)$

The set $\mu U\Sigma_{AC}$ is in general larger than $\mu U\Sigma_{AC1}$, i.e. $|\mu U\Sigma_{AC1}| \leq |\mu U\Sigma_{AC}|$. In order to obtain a set of most general AC-unifiers from $\mu U\Sigma_{AC1}$, we have to consider all AC1-instances of the AC1-mgu's, which are not AC-instances. These are easy to obtain: in the AC1-case variables can be substituted by the identity 1, i.e. in fact after normalization variables can be eliminated in the AC1-case, but not in the AC-case. Hence we systematically erase the variables in the codomain of the AC1-unifiers and add these to the set $\mu U\Sigma_{AC}$

$$\mu U\Sigma_{AC}(s, t) = \{\tau \mid \tau = (\lambda\sigma)_N \text{ with } 1 \notin COD\tau, \sigma \in \mu U\Sigma_{AC1}(s, t) \text{ and } \lambda = \{x_{i1} \leftarrow 1, \ldots, x_{in} \leftarrow 1\}\}.$$

where $\{x_{i1}, \ldots, x_{in}\}$ is a subset of $VCOD\sigma$ and $(\lambda\sigma)_N$ is the substitution derived from $\lambda\sigma$ by eliminating all 1's and restricting the domain to $W = \mathbf{V}(s, t)$.

2.4 ***Theorem*** 1: $\mu U\Sigma_{AC}(s, t)$ as defined above is a correct, complete and minimal set of AC-unifiers of s and t.

Proof: *Correctness:* Let $\tau \in \mu U\Sigma_{AC}(s, t)$ then $\tau = (\lambda\sigma)_N$. Since σ is an AC1-unifier $\lambda\sigma$ is an AC1-unifier. Hence with 2.4 Lemma 1 τ is an AC-unifier since $\tau = (\lambda\sigma)_N$ is normalized and $1 \notin COD\tau$.

Completeness: Let θ be an AC-unifier of s and t then θ is an AC1-unifier by 2.4 Lemma 1 and hence there exist a $\sigma \in \mu U\Sigma_{AC1}(s, t)$ and an appropriate normalized λ with $DOM\lambda \subseteq VCOD\sigma$ such that $\theta =_{AC1} \lambda\sigma\ [W]$ where $W = \mathbf{V}(s, t)$. Seperate λ into λ_1 and λ_2 such that $\lambda = \lambda_1\lambda_2$ with $\{1\} = COD\lambda_2$ and $1 \notin COD\lambda_1$. But then $\tau = (\lambda_2\sigma)_N$ is an AC-unifier of s and t since τ is an AC1-unifier and $1 \notin COD\tau$. Now $\tau \in \mu U\Sigma_{AC}(s, t)$ by definition of $\mu U\Sigma_{AC}$ and finally $\theta =_{AC} \lambda_1\tau\ [W]$ since $1 \notin COD\lambda_1$.

Minimality: Let $\tau_1, \tau_2 \in \mu U\Sigma_{AC}(s, t)$ with $\tau_1 \leq_{AC} \tau_2$, then with $\sigma_1, \sigma_2 \in \mu U\Sigma_{AC1}(s, t)$ $\tau_1 = (\lambda_1\sigma_1)_N$, $\tau_1 = (\lambda_2\sigma_2)_N$ and $\tau_1 =_{AC} \lambda\tau_2\ [W]$. Therefore we have $\lambda_1\sigma_1 =_{AC1} \lambda\lambda_2\sigma_2\ [W]$.

Case 1: $\sigma_1 \neq \sigma_2$: As in 2.3 Theorem 4 we construct a contradiction to the minimality of the solutions of the inhomogeneous equations.

Case 2: $\sigma_1 = \sigma_2 = \sigma$: Then we have $\lambda_1\sigma =_{AC1} \lambda\lambda_2\sigma$ **[W]** with $\lambda_1 = \{x_{i1} \leftarrow 1, \ldots, x_{in} \leftarrow 1\}$. Representing $\lambda_1\sigma$ in matrix form means that we replace the rows $i_1, \ldots, i_n$ in the matrix representation of σ by **0**. But since **0** is only trivially representable, i.e. $\mathbf{0} = 0m_1 + \ldots + 0m_2$, we can compute $\lambda\lambda_2$ by the same methods as in the completeness proof of 2.3 Theorem 3. But this yields that $\lambda_1 = \lambda\lambda_2$ and since $1 \notin \mathrm{COD}\lambda$ we have $\lambda_1 = \lambda_2$ and $\lambda = \varepsilon$. Therefore we have $\tau_1 = \tau_2$. ■

As a point of reference we define the algorithm AC-UNIFY for AC-problems without an identity as:

FUNCTION AC-UNIFY

INPUT: An AC-Problem $\langle s = t \rangle_{AC}$

STEP 0: $\mu U\Sigma_{AC1} :=$ AC1-UNIFY(s, t)

STEP 1: $\mu U\Sigma_{AC}(s, t) = \{\tau \mid \tau = (\lambda\sigma)_N$ with $1 \notin \mathrm{COD}\tau, \sigma \in \mu U\Sigma_{AC1}(s, t)$ and $\lambda = \{x_{i1} \leftarrow 1, \ldots, x_{in} \leftarrow 1\}\}$

OUTPUT: the set of most general AC-unifiers $\mu U\Sigma_{AC}(s, t)$

ENDOF AC-UNIFY.

As a final demonstration consider again the example of paragraph 2.1.3 but taken as an AC-problem:

$$\langle x^2ya = b^2z \rangle_{AC}$$

In 2.1.3 we obtained

$$\mu U\Sigma_{AC1} = \{ \{x \leftarrow v_1b,\ y \leftarrow u_1,\ z \leftarrow u_1v_1{}^2a\} \quad \{x \leftarrow v_2,\ y \leftarrow u_2b^2,\ z \leftarrow u_2v_2{}^2a\}\}.$$

Using the above definition for $\mu U\Sigma_{AC}$ we obtain

$$\begin{aligned} \mu U\Sigma_{AC} = \{ & \{x \leftarrow v_1b,\ y \leftarrow u_1,\ z \leftarrow u_1v_1{}^2a\} \\ & \{x \leftarrow b,\ y \leftarrow u_1,\ z \leftarrow u_1a\} && \text{(i.e. } \{v_1 \leftarrow 1\}) \\ & \{x \leftarrow v_2,\ y \leftarrow u_2b^2,\ z \leftarrow u_2v_2{}^2a\} \\ & \{x \leftarrow v_2,\ y \leftarrow b^2,\ z \leftarrow v_2{}^2a\} \quad \} && \text{(i.e. } \{u_2 \leftarrow 1\}). \end{aligned}$$

2.5 A Comparison with the STICKEL Algorithm

The example in 2.1.3 and 2.4 is used also by Mark Stickel [ST 81] to demonstrate his algorithm. It shows the improvement by our method: the relative simplicity by which the unifiers are computed and particularly the direct relationship between the minimal integer solutions and the most general unifiers. In order to unify $\langle x^2ya = b^2z \rangle_{AC}$ according to [ST 81] the generalization to $\langle v_1^2v_2v_3 = v_4^2v_5 \rangle_{AC}$ (the variable abstraction) with $\delta = \{v_1 \leftarrow x, v_2 \leftarrow y, v_3 \leftarrow a, v_4 \leftarrow b, v_5 \leftarrow z\}$ has to be computed. Then the (pure variable) equation corresponding to this generalized problem has to be solved, which results in 69 unifiers. After that an expensive compatiblity operation is performed between these 69 unifiers and δ resulting in the deletion of 65 of these unifiers and only four most general unifiers are left over as the final solution.

In comparison we first compute the two most general AC1-unifiers for this example directly and without any additional search. From these two AC1-unifiers the four AC-unifiers are derived.

The theoretical analysis of the two algorithms in terms of matrices (representing certain linear mappings) in [Bü 85] clearly exhibits the additional computation to be performed in the Stickel algorithm. It is also shown in [Bü 85] that our algorithm is optimal in the sense that it cannot be improved theoretically.

An additional practical advantage of our approach is that the homogeneous equation is in general much ***smaller*** in the number of variables, which leads potentionally to exponential savings: the complexity of the algorithms to solve linear diophantine equations [Hu 78][Fo 83] grows exponentially in the number of variables. The second effect of smaller homogeneous equations is that we have fewer solutions and hence the number of partitions is smaller which have to be computed for deriving all AC-unifiers from the AC1-unifers. The enumeration of the partitions in the Stickel algorithm was later improved however by Hullot [Ht 80] and Fortenbacher [Fo 85].

Finally we have shown that our algorithm is minimal, whereas it is open whether or not this is the case for the Stickel algorithm. Fortenbacher claims in [Fo 85] that his version of the Stickel algorithm is minimal.

The disadvantage of our approach is that we have to solve ***inhomogeneous equations*** as well as the homogeneous one which could potentially outweigh all of the above advantages. It would be but another instance of the wellknown fact that an algorithm with the theoretically least complexity is not always the most desirable practically. While a theoretical analysis of this problem is still pending it is demonstrated in [GH 85] that solving the inhomogeneous equations ***and*** the homogeneous equation in our case is in fact already faster (compared in runtime for a selection of typical problems) than solving the

(larger) homogeneous equation alone in the Stickel case.
A final advantage of the algorithm presented here is that it solves the unification problem for an AC1-theory as well as for an AC-theory . In applications the AC-operator more often then not has a unit (Commutative Monoids, Abelian Groups, etc.), hence we would immediately obtain fewer AC1-unifiers.

3. THE COMBINATION OF FREE TERMS WITH ABELIAN SEMIGROUPS

The previous unification algorithm is now extended to handle terms composed of several different AC-function symbols as well as uninterpreted function symbols. More precisely let the family of operators $\mathbf{F}$ consist of a set $\mathbf{C}_{\emptyset}$ of denumerable many constants, of a finite set $\mathbf{F}_{AC}$ of binary function symbols f_i, $1 \le i \le n$, and a finite set of uninterpreted function symbols $\mathbf{F}_{\emptyset}$. With $\mathbf{V}$ the set of variables as before let $\mathbf{T}$ be the set of terms built up from these symbols and let $\mathcal{T}$ be the corresponding term algebra. Using the equational theory

$$AC = \{ f_i(x\ y) = f_i(y\ x),\ f_i(x\ f_i(y\ z)) = f_i(f_i(x\ y)\ z) \mid f_i \in \mathbf{F}_{AC},\ 1 \le i \le n \}$$

we define $=_{AC}$ as the Σ-invariant congruence relation generated by AC in $\mathcal{T}$. In this section we are interested in solving equations in the quotient algebra $\mathcal{T}_{/=_{AC}}$, i.e. the interest is in AC-unification problems $\langle s = t \rangle_{AC}$ for $s, t \in \mathbf{T}$.
An AC-unification algorithm for these problems is presented below. The essential idea is as follows: for the given AC-terms the subterms not starting with an AC-function symbol are temporarily replaced by new constants, thus reducing the case at hand to a problem for the AC-algorithm of the previous section. The replaced subterms are then taken care of in a recursive call of the same process.
Using a modification of the FAGES complexity measure [Fa 84] we shall show that this process terminates and produces a complete and correct set of AC-unifiers which however is not minimal in general.

3.1 The General AC-Unification Algorithm

To simplify the notation we write "+" or "*" for the AC-function symbols f_i and informally represent a term $+(t_1 +(t_2 \dots +(t_{n-1}\ t_n)\dots))$ by its flattened (i.e. n-ary instead of binary) version $+(t_1 \dots t_n)$. We assume in the sequel that all terms and subterms are totally flattened with respect to their leading AC-function symbol.
For a given term $t = g(t_1 \dots t_n)$ the term t_k, $1 \le k \le n$, is called an ***immediate subterm*** of t and t is the ***immediate superterm*** of t_k; the ***leading function symbol*** hd(t) of t is g. If t is a constant or a variable then hd(t) = t.
A subterm r is ***alien*** in t if it is not a constant or variable, and if it does not start with an AC-function symbol or with a different leading AC-function symbol than its immediate superterm. By abuse of notation we consider s to be alien in s, provided $s \notin \mathbf{V} \cup \mathbf{C}$. For a set of terms S we denote the set of all alien

subterms of S with

$$ALIEN(S) = \{ s \mid s \text{ is an alien subterm of some term in } S \}.$$

Similarly let I-ALIEN(S) be the set of the immediate alien subterms in S. We assume that for s, t in ALIEN(S) (in I-ALIEN(S)) we have $s \neq_{AC} t$; i.e. we only take one representative of each equivalence class.. Obviously if r is an alien subterm of s then σr is an alien subterm of σs for every substitution σ. We use this fact implicitly in the termination proofs.

A term t is called ****-pure*** if it only contains an AC-function $*$, constants and variables, i.e. $t \in \mathbf{T}(\{*\} \cup \mathbf{C}_{\emptyset}, \mathbf{V})$.

For a given term we want to replace all its immediate alien subterms by new constants in order to use the unification algorithm of the previous section. For that reason we define the ***constant abstraction*** of a term s as follows: let $I\text{-}ALIEN(s) = \{s_1, ..., s_k\}$ then we call $\alpha = [s_1 \Leftarrow c_1, ..., s_k \Leftarrow c_k]$ a subterm replacement where the c_i are distinct constants that do not occur anywhere in s. Define $\underline{s} = \alpha s$, the constant abstraction of s, as the simultaneous replacement of all subterms that are T-equal to s_i by c_i for $1 \leq i \leq k$. For example for the term $s = +(g(g(x)) *(x\ a) *(a\ x)\ a\ x\ y)$, where $+$ and $*$ are AC-function symbols and g is an uninterpreted function symbol, we have $ALIEN(s) = \{ s, g(g(x)), g(x), *(x\ a) \}$, $I\text{-}ALIEN(s) = \{ g(g(x)), *(x\ a) \}$, $\alpha = [g(g(x)) \Leftarrow c_1, *(x\ a) \Leftarrow c_2]$ and $\underline{s} = \alpha s = +(c_1\ c_2\ c_2\ a\ x\ y)$. Now consider the inverse subterm replacement $\alpha^{-1} = [c_1 \Leftarrow s_1, ..., c_k \Leftarrow s_k]$. If we treat the constants $c_1, ..., c_k$ in α^{-1} as ***"special variables"*** there is no need to formally distinguish between the subterm replacement α^{-1} and the substitution $\alpha = \{ c_1 \leftarrow s_1, ..., c_k \leftarrow s_k \}$ (since both are homomorphisms, one moving some constants and the other moving some variables). We then have $\alpha\underline{s} = \alpha\alpha s =_{AC} s$.

Once we have solved the unification problem for the constant abstraction of the two terms using the previous AC-unification algorithm, we have to apply it recursively to all subterms that have been "abstracted away". Hence we define the set of all potentially unifiable subproblems SP of $s, t \in \mathbf{T}$ as the set of all pairs in I-ALIEN(s, t):

$$SP(s, t) = \{ (s', t') \mid s', t' \in I\text{-}ALIEN(s, t), hd(s') = hd(t'), s' \neq t' \}.$$

Using this terminology the general AC-unification algorithm G-AC-UNIFY can now be stated as follows: we extend the traditional Robinson unification algorithm [Ro 65] by a call to G-AC-UNIFY, the general AC-unification algorithm to be presented below which in term uses the algorithm AC-UNIFY of section 2.4.

FUNCTION AC-ROBINSON

INPUT: A unification problem $\langle s = t \rangle_{AC}$ where $s,t \in \mathbf{T}$
and w.l.o.g. s is smaller than t

STEP 1: if $s =_{AC} t$ then $U\Sigma_R := \{\varepsilon\}$
STEP 2: elseif $s \in \mathbf{V}$ then if $s \in \mathbf{V}(s)$ then FAIL else $U\Sigma_R := \{\{s \leftarrow t\}\}$
STEP 3: elseif $hd(s) \neq hd(t)$ then FAIL
STEP 4: elseif $hd(s) = hd(t) \in \mathbf{F}_{AC}$ then $U\Sigma_R :=$ G-AC-UNIFY(s, t)
STEP 5: else let $s = h(s_1 \ldots s_n)$ and $t = h(t_1 \ldots t_n)$ in
$U\Sigma_R := \{\varepsilon\}$
for i = 1 ,...., n do
subunifiers := Ø
forall $\sigma \in U\Sigma_R$ do
subunifiers := subunifiers ∪ AC-ROBINSON(σs_i, σt_i)$\cdot\sigma$
od
$U\Sigma_R$:= subunifiers
od

OUTPUT: The set of unifiers $U\Sigma_R(s, t)$ away from $W \supseteq \mathbf{V}(s, t)$

ENDOF AC-ROBINSON

The following main AC-algorithm uses an operation called the merge $\sigma * \alpha$ of two substitutions. Essentially the merge is a most general common instance of the two substitutions σ and α and is defined in 3.2 along with some properties of $*$. For a set of substitutions Σ we abbreviate $\{\sigma\cdot\delta \mid \sigma \in \Sigma\}$ by $\Sigma\cdot\delta$ and $\{\sigma * \alpha \mid \sigma \in \Sigma\}$ by $\Sigma * \alpha$.

FUNCTION G-AC-UNIFY

INPUT An AC-Problem $\langle s = t \rangle_{AC}$ with $hd(s) = hd(t) \in \mathbf{F}_{AC}$; $s, t \in \mathbf{T}$

STEP 0: if not $hd(s) = hd(t) \in \mathbf{F}_{AC}$ then FAIL

STEP 1: if I-ALIEN(s, t) = $\emptyset$ then $U\Sigma_{G\text{-}AC} := AC\text{-}UNIFY(s, t)$

STEP 2: let $\underline{s}, \underline{t}$ be the constant abstraction and α be the corresponding substitution with $\alpha\underline{s} = s$ and $\alpha\underline{t} = t$
in
$U\Sigma_{G\text{-}AC} := AC\text{-}UNIFY(\underline{s}, \underline{t}) * \alpha$

STEP 3: forall $(s', t') \in SP(s, t)$ do
forall $\sigma' \in$ AC-ROBINSON (s', t') do
$U\Sigma_{G\text{-}AC} := U\Sigma_{G\text{-}AC} \cup$ G-AC-UNIFY$(\sigma's, \sigma't)\cdot\sigma'$
od
od

OUTPUT The set of AC-unifiers $U\Sigma_{G\text{-}AC}(s, t)$ away from $W \supseteq \mathbf{V}(s, t)$

ENDOF G-AC-UNIFY

We do not explicitly consider the details of basing the unifiers on $\mathbf{V}(s, t)$ away from some set of variables $W \supseteq \mathbf{V}(s, t)$, since it would only complicate the notation. The proofs demonstrating that the unifiers are based on $\mathbf{V}(s, t)$ away from W are not difficult and we always assume that G-AC-UNIFY(s, t) and AC-ROBINSON(s, t) return a set of unifiers ***away from W*** and that the domain of the unifiers is $\mathbf{V}(s, t)$.

3.2 The Merge of Two Substitutions

In the previous algorithm we used an operation often called the merge of substitutions or unification of substitutions: two substitutions σ, τ are AC-unifiable iff there exists a substitution λ such that $\lambda\sigma =_{AC} \lambda\tau$. Then λ is called an AC-unifier of σ and τ. Define the sets $U\Sigma_{AC}(\sigma, \tau)$, $cU\Sigma_{AC}(\sigma, \tau)$ and the set of most general unifiers $\mu U\Sigma_{AC}(\sigma, \tau)$ away from $W \supseteq \mathbf{V}(\sigma, \tau)$ as before (for terms). If $\mu U\Sigma_{AC}(\sigma, \tau) = \{\lambda\}$, the substitution $\sigma * \tau := \lambda\sigma =_{AC} \lambda\tau$ is called a ***merge*** of σ and τ.

In the special situation of the algorithm G-AC-UNIFY the following restrictions hold: given an AC-unifier $\tau = \{x_1 \leftarrow t_1, \ldots, x_m \leftarrow t_m\}$ computed by AC-UNIFY(s, t) for some terms s and t containing no alien subterms, and a substitution

$\alpha = \{c_1 \leftarrow r_1, \ldots, c_n \leftarrow r_n\}$ reversing the subterm replacement of a constant abstraction we have

(i) $DOM\alpha \cap DOM\tau = \emptyset$
(ii) $VCOD\alpha \cap VCOD\tau = \emptyset$.

The last equation holds since $VCOD\tau$ only contains special variables and variables not occurring in s and t, whereas $VCOD\alpha \subseteq \mathbf{V}(s, t)$. Under these circumstances $|\mu U\Sigma_{AC}(\tau, \alpha)| = 1$ and there is a particularly efficient way to compute the merge which we shall now present. First we reduce the unification of substitutions to unification of termlists, i.e. σ unifies two termlists $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ iff $\sigma s_i =_{AC} \sigma t_i$ for $1 \leq i \leq n$ and the notion of a set of most general unifiers for two termlists carries over in the usual way.

3.2 ***Lemma*** 1: For α and τ as above
$$U\Sigma_{AC}(\alpha, \tau) = U\Sigma_{AC}((x_1, \ldots, x_m, c_1, \ldots, c_n), (t_1, \ldots, t_m, r_1, \ldots, r_n)).$$

Proof: Let $\lambda \in U\Sigma_{AC}(\alpha, \tau)$, i.e. $\lambda\alpha =_{AC} \lambda\tau$. For $x_i \in DOM\tau$ we have $\lambda x_i = \lambda\alpha x_i =_{AC} \lambda\tau x_i = \lambda t_i$ for $1 \leq i \leq m$ and for $c_j \in DOM\alpha$ $\lambda c_j = \lambda\tau c_j =_{AC} \lambda\alpha c_j = \lambda r_j$ for $1 \leq j \leq n$. Hence λ is a unifier of the termlists.
Conversely let θ be a unifier of the termlists then for $x \in DOM\tau$ (i.e. $x = x_i$) $\theta\alpha x_i = \theta x_i =_{AC} \theta t_i = \theta\tau x_i$ and for $x \in DOM\alpha$ (i.e. $x = c_j$) $\theta\tau c_j = \theta c_j =_{AC} \theta r_j = \theta\alpha c_j$. Hence for all x we have $\theta\alpha x =_{AC} \theta\tau x$, thus $\theta \in U\Sigma_{AC}(\alpha, \tau)$. ■

This lemma immediately implies that $\mu U\Sigma_{AC}(\alpha, \tau) = \mu U\Sigma_{AC}((x_1, \ldots, x_m, c_1, \ldots, c_n), (t_1, \ldots, t_m, r_1, \ldots, r_n))$.
For the equational theory AC and a variable x we have $\mu U\Sigma_{AC}(x, t) = \emptyset$ if $x \in \mathbf{V}(t)$ and $\mu U\Sigma_{AC}(x, t) = \{\{x \leftarrow t\}\}$ otherwise. Hence the most general unifier of the termlists in 3.2 Lemma 1 (if it exists) is just the composition of $\{x_i \leftarrow t_i\}$ and $\{c_j \leftarrow r_j\}$. We therefore define $\tau_0 = \tau$ and $\tau_j = \sigma_j\tau_{j-1}$ with $\sigma_j = \{c_j \leftarrow \tau_{j-1}r_j\}$ for $1 \leq j \leq n$.

3.2 ***Lemma*** 2:
(i) If the termlists $(x_1, \ldots, x_m, c_1, \ldots, c_n)$ and $(t_1, \ldots, t_m, r_1, \ldots, r_n)$ are AC-unifiable then τ_n is their single most general unifier.
(ii) If the termlists $(x_1, \ldots, x_m, c_1, \ldots, c_n)$ and $(t_1, \ldots, t_m, r_1, \ldots, r_n)$ are not AC-unifiable then there exists j, $1 \leq j \leq n$, with $c_j \in \mathbf{V}(\tau_{j-1}r_j)$.

Proof: We show (i) by induction on j: let θ be a unifier then $\theta \leq_{AC} \tau_j [V]$ with $V = \mathbf{V}(x_1, ..., x_m, c_1, ..., c_n, t_1, ..., t_n, r_1, ..., r_n)$.

Base step: Since θ unifies the termlists we have $\theta x_i =_{AC} \theta t_i = \theta\tau x_i$. Hence $\theta \leq_{AC} \tau_0 = \tau [(x_1, ..., x_m)]$. Using "Fortsetzungslemma" (1.2 Lemma 1) we get $\theta \leq_{AC} \tau_0 [V]$.

Induction step: Since $\theta \leq_{AC} \tau_j [V]$ by induction hypothesis there exixts λ_j such that $\theta =_{AC} \lambda_j\tau_j [V]$. But then $\lambda_j c_{j+1} = \lambda_j\tau_j c_{j+1} =_{AC} \theta c_{j+1} =_{AC} \theta r_{j+1} =_{AC} \lambda_j\tau_j r_{j+1}$ (since $\tau_j c_{j+1} = c_{j+1}$) i.e. λ_j unifies c_{j+1} and $\tau_j r_{j+1}$. Hence $c_{j+1} \notin \mathbf{V}(\tau_j r_{j+1})$ and $\mu U\Sigma_{AC}(c_{j+1}, \tau_j r_{j+1}) = \{\sigma_{j+1}\}$ with $\sigma_{j+1} = \{c_{j+1} \leftarrow \tau_j r_{j+1}\}$ and $\lambda_j \leq_{AC} \sigma_j [\mathbf{V}(c_{j+1}, \tau_j r_{j+1})]$. Using 1.2 Lemma 2 we have $\theta =_{AC} \lambda_j\tau_j \leq_{AC} \sigma_{j+1}\tau_j [\mathbf{V}(c_1, ..., c_{j+1}, r_1, ..., r_{j+1})]$ and again by 1.2 Lemma 1 we finally get $\theta \leq_{AC} \sigma_{j+1}\tau_j = \tau_{j+1} [V]$.

Hence τ_n exists and is the single, essantially unique, most general unifier of the termlists. ■

To summarize this paragraph we have: if the substitutions τ and α are AC-unifiable then τ_n is the single most general unifier of τ and α with $\tau * \alpha = \tau_n\alpha = \tau_n\tau = \tau_n$. If the substitutions τ and α are not AC-unifiable then there exists j, $1 \leq j \leq n$ with $c_j \in \mathbf{V}(\tau_{j-1} r_j)$. That means that there is no need to perform a full AC-unification in order to compute the merge of τ and α, but it suffices to compose the $\sigma_i$and to check if there are no cycles ($c_j \in \mathbf{V}(\tau_{j-1} r_j)$).

3.3 An Example

Given the unification problem

$$\langle +(x^2\ y\ g(x\ u)) \ =\ +(z\ g(a\ b)\ g(a\ b)) \rangle_{AC}$$

where + is an AC-function symbol, g is an uninterpreted function symbol and a, b are two constants. The immediate alien subterms for $s = +(x^2\ y\ g(x\ u))$ and $t = +(z\ g(a\ b)\ g(a\ b))$ are I-ALIEN(s, t) = { g(x u), g(a b) }. The only subproblem is therefore (s', t') = (g(x u), g(a b)) with the most general unifier $\sigma' = \{x \leftarrow a, u \leftarrow b\}$. The constant abstractions of s and t are $\underline{s} = +(x^2\ y\ c_1)$ and $\underline{t} = +(z\ c_2^2)$ with $\alpha = \{c_1 \leftarrow g(x\ u), c_2 \leftarrow g(a\ b)\}$. The set of most general unifiers for $\underline{s}$ and $\underline{t}$ (see 2.4) is:

$$\mu U\Sigma_{AC}(\underline{s}, \underline{t}) = \{ \{x \leftarrow +(z_1\ c_2),\ y \leftarrow z_2,\ z \leftarrow +(z_1\ z_2^2\ c_1)\},$$
$$\{x \leftarrow c_2,\ y \leftarrow u_2,\ z \leftarrow +(u_2^2\ c_1)\},$$
$$\{x \leftarrow v_1,\ y \leftarrow +(v_2 c_2^2),\ z \leftarrow +(v_1 v_2^2 c_1 c_2)\},$$
$$\{x \leftarrow w_1,\ y \leftarrow +(c_2^2),\ z \leftarrow +(w_1 c_1 c_2)\}\ \}.$$

Merging these unifiers with α we get:

$$\mu U\Sigma_{AC}(\underline{s}, \underline{t}) * \alpha = \{$$
$$\{ x \leftarrow +(z_1\ g(a\ b)),\ y \leftarrow z_2,\ z \leftarrow +(z_1\ z_2^2\ g(\ +(z_1\ g(a\ b))\ z_3\ )),\ u \leftarrow z_3\ \},$$
$$\{ x \leftarrow g(a\ b),\ y \leftarrow u_2,\ z \leftarrow +(u_2^2\ g(\ g(a\ b)\ u_3)),\ u \leftarrow u_3\ \},$$
$$\{ x \leftarrow v_1,\ y \leftarrow +(v_2\ g(a\ b)\ g(a\ b)),\ z \leftarrow +(v_1\ v_2^2\ g(a\ b)\ g(v_1 v_3)),\ u \leftarrow v_3\ \},$$
$$\{ x \leftarrow w_1,\ y \leftarrow +(g(a\ b)\ g(a\ b)),\ z \leftarrow +(w_1\ g(a\ b)\ g(w_1\ w_3),\ u \leftarrow w_3\ \}\ \}$$

The only unifier of (s', t') is $\sigma = \{x \leftarrow a, u \leftarrow b\}$ and hence $\sigma s = +(a^2\ y\ g(a\ b))$ and $\sigma t = +(z\ g(a\ b)\ g(a\ b))$. The set of most general AC-unifiers is $\mu U\Sigma_{AC}(\sigma s, \sigma t) = \{\tau_1, \tau_2\}$ with $\tau_1 = \{y \leftarrow +(x_1\ g(a\ b)),\ z \leftarrow +(x_1\ a^2)\}$ and $\tau_2 = \{y \leftarrow g(a\ b),\ z \leftarrow +(a^2)\}$. Hence

$$\tau_1\sigma = \{x \leftarrow a, y \leftarrow +(x_1\ g(a\ b)),\ z \leftarrow +(x_1\ a^2), u \leftarrow b\}$$
$$\tau_2\sigma = \{x \leftarrow a, y \leftarrow g(a\ b),\ z \leftarrow +(\ a^2), u \leftarrow b\}$$

are two additional most general AC-unifiers of s and t. Hence we have finally:

$$\mu U\Sigma_{AC}(s, t) = \mu U\Sigma_{AC}(\underline{s}, \underline{t}) * \alpha \cup \{\tau_1\sigma, \tau_2\sigma\}.$$

3.4 Termination

Termination of the algorithms G-AC-UNIFY and AC-ROBINSON is shown by Noetherian induction on a slightly modified form of the complexity measure of F. Fages [Fa 84], which we shall now define. Let r be a subterm of a term s then Op(r, s), the set of the leading function symbols of all immediate superterms of r in s, is:

$$Op(r, s) = \{\ hd(s') \mid \text{there exists a subterm } s' \text{ in } s \text{ which is an immediate superterm of } r\}.$$

Let Op(r, S), the set of leading function symbols of a term s in a set of terms S, be $\cup\{Op(r, s) \mid s \in S\}$ For a set of terms S let

$$\mathbf{V}_s(S) = \{ x \mid x \in \mathbf{V}(S) \text{ and } |Op(x, S)| > 1 \}$$

be the set of ***shared variables***, i.e. those variables which occur immediately under at least two different function symbols. We omit the parentheses in Op(r, {s, t}) resp. in $\mathbf{V}_s(\{s, t\})$ and write Op(r; s, t) resp. $\mathbf{V}_s(s, t)$. Given a unification problem $\langle s = t \rangle_{AC}$ the complexity of that problem is defined as

$$C(s, t) = (\nu, \tau) \quad \text{with } \nu = |\mathbf{V}_s(s, t)| \text{ and } \tau = |ALIEN(s, t)|.$$

Taking the lexicographic ordering on the complexities we obtain a Noetherian ordering. For example with s = +(x y g(x u)) and t = +(z g(a b) g(a b)) we have Op(x; s, t) = {+, g}, $\mathbf{V}_s(s, t)$ = {x}, ALIEN(s, t) = {s, t, g(x u), g(a b)} and hence C(s, t) = (1, 4). Note that constants are alien subterms in Fages' complexity definition but not in ours.

3.4 ***Lemma*** 1: If s', t' ∈ ALIEN(s, t) are proper subterms of s and t then
$C(s', t') < C(s, t)$.

Proof: Let $C(s, t) = (\nu, \tau)$ and $C(s', t') = (\nu', \tau')$. Since $\mathbf{V}(s', t') \subseteq \mathbf{V}(s, t)$ and $Op(x; s', t') \subseteq Op(x; s, t)$ we have $\mathbf{V}_s(s', t') \subseteq \mathbf{V}_s(s, t)$ and hence $\nu' \leq \nu$. If $\nu' = \nu$ we have to show that $\tau' < \tau$. Consider the mapping Φ: ALIEN(s', t') ⟶ ALIEN(s, t). For r' ∈ ALIEN(s', t') define $\Phi(r') :- r \in ALIEN(s, t)$ with $r =_{AC} r'$. Then Φ is an inclusion and since at least s or t is not contained in ALIEN(s', t') we have $\tau' < \tau$. ■

In order to prove the termination of our main algorithm we have to show that the unifiers produced by the algorithm decrease the complexity of the original terms, i.e. if σ unifies some immediate alien subterms of s and t then $C(\sigma s, \sigma t) < C(s, t)$.
We say a substitution σ is **monotone for s and t** iff $C(\sigma s, \sigma t) \leq C(s, t)$ and **strictly monotone for s and t** iff $C(\sigma s, \sigma t) < C(s, t)$. In the following lemmata we show the monotony of certain substitutions.
We call a substitution σ **alien for s and t** iff $\sigma = \{x \leftarrow r\}$ with $x \in \mathbf{V}(s, t)$, $r \in ALIEN(s, t)$ and $x \notin \mathbf{V}(r)$.

3.4 ***Lemma*** 2: If a substitution σ is alien for two terms s and t then σ is monotone for s and t.

Proof: Let $C(\sigma s, \sigma t) = (\nu_\sigma, \tau_\sigma)$, $C(s, t) = (\nu, \tau)$ and $\sigma = \{x \leftarrow r\}$. Since

$\mathbf{V}_s(\sigma s, \sigma t) \subseteq \mathbf{V}_s(s, t)$ we have $\nu_\sigma \leq \nu$.
If $\nu_\sigma < \nu$ we are done. If $\nu_\sigma = \nu$ we want to show that $\tau_\sigma \leq \tau$. We construct an injective mapping Ψ from ALIEN($\sigma s, \sigma t$) to ALIEN(s, t) with $\Psi(p) = p' \in$ ALIEN(s, t) and $\sigma p' =_{AC} p$. For $p = r$ we define $\Psi(r) = r$ with $\sigma r = r$ since $r \in$ ALIEN(s, t) and $x \notin \mathbf{V}(r)$. For $p \neq r$ in ALIEN($\sigma s, \sigma t$) there exists an $p' \in$ ALIEN(s, t) with $\sigma p' =_{AC} p$; we define $\Psi(p) = p'$. Hence we have $\Psi(p) = p'$ with $\sigma p' =_{AC} p$.
The injectivity of Ψ is easy to see: $q_1 = \Psi(p_1) = \Psi(p_2) = q_2$ implies $p_1 =_{AC} \sigma q_1 = \sigma q_2 =_{AC} p_2$ and by definition of ALIEN $p_1 = p_2$ (note that for s' and t' in ALIEN(s, t) we have: if $s' =_{AC} t'$ then $s' = t'$). Hence $|\text{ALIEN}(\sigma s, \sigma t)| = \tau_\sigma \leq \tau = |\text{ALIEN}(s, t)|$. ■

For some function symbol $f \in \mathbf{F}$ a substitution σ is called **f-pure for s and t** iff $\text{DOM}\sigma \subseteq \mathbf{V}(s, t)$, $\text{VCOD}\sigma \cap \mathbf{V}(s, t) = \emptyset$ and the following two conditions are satisfied

- $f \in \text{Op}(x; s, t)$ for all $x \in \text{DOM}\sigma$ and
- $\begin{cases} \sigma x \in \mathbf{V} \cup \mathbf{C}_\emptyset & \text{if } f \in \mathbf{F}_\emptyset \\ \sigma x \text{ is a } *\text{-pure term} & \text{if } f = * \in \mathbf{F}_{AC} \text{ (i.e. } \text{COD}\sigma \subseteq \mathbf{T}(\{*\} \cup \mathbf{C}, \mathbf{V})). \end{cases}$

If there are no ambiguities, we simply speak of pure substitutions and pure terms.

3.4 ***Lemma*** 3: If a substitution σ is f-pure for two terms s and t then σ is monotone for s and t.

Proof: Let $\mathbf{C}(\sigma s, \sigma t) = (\nu_\sigma, \tau_\sigma)$, $\mathbf{C}(s, t) = (\nu, \tau)$ and σ be f-pure for s and t. We shall construct an injective mapping Φ from $\mathbf{V}_s(\sigma s, \sigma t)$ to $\mathbf{V}_s(s, t)$. Let $x \in \mathbf{V}_s(\sigma s, \sigma t)$: if $x \notin \text{VCOD}\sigma$ then $x \in \mathbf{V}_s(s, t)$ and we choose $\Phi(x) = x$. Note that $x \notin \text{DOM}\sigma$. For $x \in \text{VCOD}\sigma$ consider the set of variables $V_1 = \{y \mid x \in \mathbf{V}(\sigma y)\}$. Suppose that for all $y \in V_1$ $\sigma y \neq x$, which contradicts the definition for $f \in \mathbf{F}_\emptyset$. Let $f = * \in \mathbf{F}_{AC}$ and σ be pure then x occurs only under $*$ in σs and σt ($x \notin \mathbf{V}(s, t)$), which is a contradiction to $x \in \mathbf{V}_s(s, t)$. Now consider the set $V_2 = \{y \in V_1 \mid \sigma y = x\}$. But then there exists $y \in V_2 \cap \mathbf{V}_s(s, t)$ and we can choose $\Phi(x) = y$. Otherwise again all y would only occur under f in s and t which contradicts $x \in \mathbf{V}_s(\sigma s, \sigma t)$. Hence we

have $\sigma y = x$ for all x with $\Phi(x) = y$. The injectivity of Φ is easy to see: $y_1 = \Phi(x_1) = \Phi(x_2) = y_2$ implies $x_1 = \sigma y_1 = \sigma y_2 = x_2$. Hence $\nu_{\sigma\Box} \leq \nu$.
If $\nu_{\sigma\Box} < \nu$ the proof is done. If $\nu_{\sigma\Box} = \nu$ the mapping Φ is bijective and there exists the inverse Φ^{-1} from $\mathbf{V}_s(s, t)$ to $\mathbf{V}_s(\sigma s, \sigma t)$ with $\Phi^{-1}(x) = \sigma x = y$. We want to show that $\tau_\sigma \leq \tau$. Again we construct a mapping Ψ from ALIEN($\sigma s, \sigma t$) to ALIEN(s, t) with $\Psi(p) = p'$ and $\sigma p' =_{AC} p$.
By the bijectivity of Φ for all $x \in$ DOMσ with $\sigma x \notin \mathbf{V}$ we have $x \notin \mathbf{V}_s(s, t)$. Let p be in ALIEN($\sigma s, \sigma t$) then p is not introduced by σ since for all x with $\sigma x \notin \mathbf{V}$ Op(x, s, t) = {f} and hd(σx) = f or σx is an uninterpreted constant. Hence there must exist a subterm $p' \in$ ALIEN(s, t) with $\sigma p' =_{AC} p$ and we define $\Psi(p) = p'$.
The injectivity of Ψ is again easy to see: $q_1 = \Phi(p_1) = \Phi(p_2) = q_2$ implies $p_1 =_{AC} \sigma q_1 = \sigma q_2 =_{AC} p_2$ and as above $p_1 = p_2$. Hence we have |ALIEN($\sigma s, \sigma t$)| = $\tau_\sigma \leq \tau$ = |ALIEN(s, t)|. ■

In the termination proof we shall often use the fact that a substitution which is pure or alien for s' and t' is pure or alien for s and t as well if s' and t' are alien subterms of s and t:

3.4 ***Lemma*** 4: Let $s', t' \in$ ALIEN(s, t).
If σ is alien for s' and t' then σ is alien for s and t.
If σ is a pure substitution for s' and t' and VCOD(σ) $\cap$ $\mathbf{V}(s, t) = \emptyset$ then σ is pure for s and t.

The proof is obvious. The next lemma is the key for the termination proof.

3.4 ***Lemma*** 5: Let σ be a pure or alien substitution for s and t and let $s', t' \in$ ALIEN(s, t) be proper and distinct subterms of s or t. If σ unifies s' and t' then σ is strictly monotone.

Proof: By the 3.4 Lemma 2 and 3.4 Lemma 3 we have $\mathfrak{C}(\sigma s, \sigma t) \leq \mathfrak{C}(s, t)$. Suppose $\mathfrak{C}(\sigma s, \sigma t) = \mathfrak{C}(s, t)$, i.e. the mapping Ψ from ALIEN($\sigma s, \sigma t$) to ALIEN(s, t) constructed in the above proofs is bijective. Hence the inverse Ψ^{-1} exists with $\Psi^{-1}(p) = q$ and $\sigma p =_{AC} q$. But then $\Psi^{-1}(s') = \Psi^{-1}(t')$ since σ unifies s' and t' which contradicts the bijectivity of Ψ^{-1}. ■

Since in the algorithm the substitutions are built up by composition of alien or pure substitutions we say a substitution σ is **ap-compound for the problem** $\langle s = t \rangle_T$ (or short for s and t) iff it is a composition of alien or pure substitutions, i.e. $\sigma = \sigma_n \sigma_{n-1} \ldots \sigma_1$ where σ_i is pure or alien for $\sigma_{i-1} \ldots \sigma_1 s$ and

$\sigma_{i-1} \ldots \sigma_1 t$ for $2 \le i \le n$ and σ_1 is pure or alien for s and t. By an induction argument we have:

3.4 ***Lemma*** 6: (i) If σ is an ap-compound substitution for s and t then σ is monotone for s and t.
(ii) If in addition σ unifies two distinct and proper subterms s', t' $\in$ ALIEN(s, t) then σ is strictly monotone for s and t.

As an extension of 3.4 Lemma 4 we have if σ is ap-compound for s' and t' then σ is ap-compound for s and t provided the newly introduced variables are away from $\mathbf{V}(s, t)$. We use this fact implicitly in the proofs below.
To summarize: we introduced monotone substitutions and showed that alien and pure substitutions (the elements of the generated unifiers) and their composition are monotone. Note that the algorithm AC-UNIFY of section 2 only generate $*$-pure substitutions for $*$-pure terms.
We now show that the merge $\tau * \alpha$ is ap-compound for s and t, where τ is a unifier of the constant abstractions of s and t and α is the corresponding abstraction reverser:

3.4 ***Lemma*** 7: Let τ be a unifier of $\underline{s}$ and $\underline{t}$, where $\underline{s}$ and $\underline{t}$ are the constant abstractions of s and t, and $\alpha\underline{s} =_{AC} s$, $\alpha\underline{t} =_{AC} t$. The merge $\tau * \alpha$ is then ap-compound for s and t.

Proof: By 3.2 Lemma 2 we have $\tau * \alpha = \tau_n$ with $\tau_0 = \tau$, $\tau_i = \sigma_i \tau_{i-1}$ and $\sigma_i = \{c_i \leftarrow \tau_{i-1} r_i\}$ for $1 \le i \le n$ where $\alpha = \{c_1 \leftarrow r_1, \ldots, c_n \leftarrow r_n\}$ and τ is a unifier of the pure terms $\underline{s}$ and $\underline{t}$. We show by induction: σ_i is ap-compound for $\tau_{i-1}s$ and $\tau_{i-1}t$. First it is easy to see that τ_0 is pure for s and t. Since c_i occurs in $\mathrm{VCOD}\tau_0$ and hence in $\mathrm{VCOD}\tau_{i-1}$ it must occur in $\tau_{i-1}s$ or in $\tau_{i-1}t$. As r_i is an alien subterm of s and t r_i is in $\mathrm{ALIEN}(\tau_{i-1}s, \tau_{i-1}t)$. With $c_i \notin \mathbf{V}(\tau_{i-1}r)$ σ_i is alien for $\tau_{i-1}s$ and $\tau_{i-1}t$. ■

In order to show that G-AC-UNIFY and AC-ROBINSON terminate, we use (a stronger) induction argument on $\mathbf{C}(s, t)$, the complexity of the input problem. For each STEP and each recursive call we show:

- the generated substitutions are ap-compound
- $\mathbf{C}(s', t') < \mathbf{C}(s, t)$, where s, t are the original terms and s', t' the subterms of the recursive call. Hence the induction hypothesis applies.

3.4 ***Theorem*** 1: For all $s, t \in \mathbf{T}$: AC-ROBINSON(s, t) terminates.

Proof: In STEP 1 - STEP 3 we are done since termination is obvious and the returned substitutions are ap-compound for s and t except for the case that both s and t are variables.
STEP 4 is shown in 3.4 Theorem 2, below.
STEP 5 is shown by induction on the loop index i with $1 \leq i \leq n$.
Base Step: If s_1 and t_1 are compound terms then $s_1, t_1 \in$ ALIEN(s, t). By 3.4 Lemma 1 $C(s_1, t_1) < C(s, t)$ and hence by Noetherian induction AC-ROBINSON($\varepsilon s_1, \varepsilon t_1$) terminates and the unifiers σ_1 returned by AC-ROBINSON(s_1, t_1) are ap-compound for s_1 and t_1 and hence for s and t.
If s_1 or t_1 is not a compound term then AC-ROBINSON(s_1, t_1) terminates with FAIL or the returned substitution σ_1 is alien or h-pure for s and t.
Induction Step: Again we distinguish the cases where the terms $\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$ are compound or not. If $\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}$ or $\sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$ is not compound then AC-ROBINSON($\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}, \sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$) terminates with FAIL or the returned substitution σ_{i+1} is alien or h-pure for $\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$ and hence it is ap-compound for $\sigma_i \bullet \ldots \bullet \sigma_1 s$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t$.
Now let the terms $\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$ be compound. Since $\sigma_i \bullet \ldots \bullet \sigma_1$ is ap-compound for s and t we have with 3.4 Lemma 1 and 3.4 Lemma 5 $C(\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}, \sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}) < C(\sigma_i \bullet \ldots \bullet \sigma_1 s, \sigma_i \bullet \ldots \bullet \sigma_1 t) \leq C(s, t)$ and hence AC-ROBINSON($\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}, \sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$) terminates and yields only substitutions σ_{i+1} which are ap-compound for $\sigma_i \bullet \ldots \bullet \sigma_1 s_{i+1}$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t_{i+1}$ and hence they are ap-compound for $\sigma_i \bullet \ldots \bullet \sigma_1 s$ and $\sigma_i \bullet \ldots \bullet \sigma_1 t$. To summarize $\sigma_{i+1} \bullet \ldots \bullet \sigma_1$ is ap-compound for s and t. ■

3.4 ***Theorem*** 2: For two terms s and t with the same leading AC-function symbol G-AC-UNIFY(s, t) terminates.

Proof: In STEP 1 termination follows from 2.4 Theorem1 and the unifiers produced by AC-UNIFY(s, t) are ap-compound for s and t.
In STEP 2 $\underline{s}$ and $\underline{t}$ are the constant abstractions for s and t with I-ALIEN($\underline{s}, \underline{t}$) = Ø. Hence AC-UNIFY($\underline{s}, \underline{t}$) terminates by 2.4 Theorem 1 and by 3.4 Lemma 7 we know that for τ returned by AC-UNIFY($\underline{s}, \underline{t}$) $\tau * \alpha$ is ap-compound for s and t.
STEP 3: For $(s', t') \in SP(s, t)$ we have by 3.4 Lemma 1 $C(s', t') < C(s, t)$. Hence AC-ROBINSON(s', t') terminates. It yields only substitutions σ' which are ap-compound for s and t. But since σ' unifies s' and t' we have by

3.4 Lemma 6 (ii) $C(\sigma's, \sigma't) < C(s, t)$. Therefore by the induction hypothesis G-AC-UNIFY($\sigma's, \sigma't$) ($\sigma's$ and $\sigma't$ both start with the same function symbol as s and t) terminates and produces only substitutions σ'' which are ap-compound for $\sigma's$ and $\sigma't$. Hence $\sigma = \sigma''\bullet\sigma'$ is ap-compound for s and t. ■

3.5 Correctness and Completeness

All proofs of this chapter are by induction on the recursion depth of the unification algorithm, which is a Noetherian order as shown in the last chapter. The set of substitutions returned by the AC-unification algorithms is a correct set of unifiers of s and t :

3.5 ***Theorem*** 1: The set $U\Sigma_{G\text{-}AC}(s, t)$ returned by G-AC-UNIFY(s, t) is a correct set of unifiers for every $s,t \in \mathbf{T}$.

Proof: Consider each step in G-AC-UNIFY in turn:
STEP 1: The theorem follows from 2.4 Theorem 1 (correctness for the variable-constant-case).
STEP 2: Let $\underline{s}$, $\underline{t}$ be the constant abstractions of s and t with $\alpha\underline{s} =_{AC} s$ and $\alpha\underline{t} =_{AC} t$. By 2.4 Theorem 1 let θ be a correct AC-unifier of $\underline{s}$ and $\underline{t}$. Since $\theta*\alpha = \lambda\alpha =_{AC} \lambda\theta$ for some λ we have (using the idempotence of θ and α):

$$(\theta*\alpha)\bullet\theta =_{AC} \lambda\bullet\theta\bullet\theta = \lambda\bullet\theta =_{AC} \theta*\alpha =_{AC} \lambda\bullet\alpha = \lambda\bullet\alpha\bullet\alpha =_{AC} (\theta*\alpha)\bullet\alpha$$

Hence
$$\begin{aligned}
(\theta*\alpha)s &=_{AC} (\theta*\alpha)\bullet\alpha\underline{s} \\
&=_{AC} (\theta*\alpha)\bullet\theta\underline{s} \\
&=_{AC} (\theta*\alpha)\bullet\theta\underline{t} \qquad \text{by assumption} \\
&=_{AC} (\theta*\alpha)\bullet\alpha\underline{t} \\
&=_{AC} (\theta*\alpha)t\,.
\end{aligned}$$

STEP3: Let (s', t') be a subproblem of s and t. By induction hypothesis let σ' be a correct unifier of s' and t' and σ'' be a correct AC-unifier of $\sigma's$ and $\sigma't$. Then for $\sigma = \sigma''\bullet\sigma' \in U\Sigma_{G\text{-}AC}$ we have

$$\begin{aligned}
\sigma s &= (\sigma''\bullet\sigma')s \\
&= \sigma''(\sigma's) \\
&=_{AC} \sigma''(\sigma't) \qquad \text{by hypothesis} \\
&= \sigma t
\end{aligned}$$

■

3.5 ***Theorem*** 2: The set $U\Sigma_R(s, t)$ returned by AC-ROBINSON is a correct set of AC-unifiers for every $s,t \in \mathbf{T}$.

Proof: STEP 1 - STEP 3: Correctness is obvious.
STEP 4: This is shown in the above theorem.

STEP 5: Let $\sigma \in U\Sigma_R(s, t)$ then $\sigma = \sigma_n \bullet \ldots \bullet \sigma_1$. By induction hypothesis we know that σ_i is a unifier of $\sigma_{i-1} \bullet \ldots \bullet \sigma_1 s_i$ and $\sigma_{i-1} \bullet \ldots \bullet \sigma_1 t_i$. Hence σ is a unifier of s_i and t_i, $1 \le i \le n$, and we have:

$$\begin{aligned} \sigma s &= \sigma h(s_1, \ldots, s_n) \quad \text{for } s = h(s_1, \ldots, s_n) \text{ as defined in STEP 5} \\ &= h(\sigma s_1, \ldots, \sigma s_n) \\ &=_{AC} h(\sigma t_1, \ldots, \sigma t_n) \\ &= \sigma t \end{aligned}$$ ■

The following two theorems show that the algorithms return a complete set of unifiers.

3.5 ***Theorem*** 3: Let θ be an AC-unifier for the terms s and t . Then there exists $\sigma \in U\Sigma_R(s, t)$ (returned by AC-ROBINSON(s, t)) such that

$$\theta \le_{AC} \sigma \ [V] \quad \text{with } V = \mathbf{V}(s, t).$$

Proof: STEP 1 - STEP 3: The proof is trivial.
STEP 4: This is shown in 3.5 Theorem 4 below.
STEP 5: By induction on i, $1 \le i \le n$ we show that there exists $\sigma_i \in U\Sigma_R(\sigma_{i-1} \ldots \sigma_1 s_i, \sigma_{i-1} \ldots \sigma_1 t_i)$ such that $\theta \le_{AC} \sigma_i \ldots \sigma_1 \ [V]$. Hence there exists a $\sigma = \sigma_n \ldots \sigma_1 \in U\Sigma_R(s, t)$ such that $\theta \le_{AC} \sigma \ [V]$.
Base Step: Since θ unifies s and t it unifies s_1 and t_1. Hence by Noetherian induction there exists $\sigma_1 \in U\Sigma_R(s_1, t_1)$ with $\theta \le_{AC} \sigma_1 \ [V_1]$ where $V_1 = \mathbf{V}(s_1, t_1) \subseteq V$. Using 1.2 Lemma 1 we have $\theta \le_{AC} \sigma_1 \ [V]$.
Induction Step: Let $\theta =_{AC} \lambda_i \sigma_i \ldots \sigma_1 \ [V]$ by induction hypothesis. Then λ_i is a unifier of $\sigma_i \ldots \sigma_1 s_{i+1}$ and $\sigma_i \ldots \sigma_1 t_{i+1}$. By Noetherian induction there exists $\sigma_{i+1} \in U\Sigma_R(\sigma_i \ldots \sigma_1 s_{i+1}, \sigma_i \ldots \sigma_1 t_{i+1})$ such that $\lambda_i \le_{AC} \sigma_{i+1} \ [V_{i+1}]$ where $V_{i+1} = \mathbf{V}(\sigma_i \ldots \sigma_1 s_{i+1}, \sigma_i \ldots \sigma_1 t_{i+1})$. By 1.2 Lemma 1 and 1.2 Lemma 2 we have $\lambda_i \sigma_i \ldots \sigma_1 \le_{AC} \sigma_{i+1} \sigma_i \ldots \sigma_1 \ [\mathbf{V}(s_1, \ldots, s_{i+1}, t_1, \ldots, t_{i+1})]$. Using 1.2 Lemma 1 again we finally have $\theta =_{AC} \lambda_i \sigma_i \ldots \sigma_1 \le_{AC} \sigma_{i+1} \sigma_i \ldots \sigma_1 \ [V]$. ■

3.5 ***Theorem*** 4: Let θ be an AC-unifier of two terms s and t. Then there exists $\sigma \in U\Sigma_{G\text{-}AC}(s, t)$ (returned by G-AC-UNIFY(s, t)) such that

$$\theta \le_{AC} \sigma \ [V] \quad \text{with } V = \mathbf{V}(s, t).$$

Proof: STEP 1: If I-ALIEN(s, t) = Ø, i.e. s and t have only variables and constants as immediate subterms, completeness follows from 2.4 Theorem 1.
STEP 2: Now I-ALIEN(s, t) ≠ Ø and assume for all $(s', t') \in SP(s, t)$ it is $\theta s' \neq_{AC} \theta t'$

(else STEP 3 applies). Then by 3.5 Lemma 3 below there exists a θ' with $\theta'\underline{s} =_{AC} \theta'\underline{t}$ and $\theta \leq_{AC} \theta'*\alpha$ [V], where $\underline{s} = \alpha s$ and $\underline{t} = \alpha t$ are the constant abstractions of s and t and α is the substitution reversing the constant abstraction. Again by 2.4 Theorem 1 there exists $\theta'' \in U\Sigma_{G\text{-}AC}(\underline{s}, \underline{t})$ (returned by AC-UNIFY($\underline{s}$, $\underline{t}$)) such that

$$\theta' \leq_{AC} \theta'' \ [\underline{V}] \quad \text{where } \underline{V} = \mathbf{V}(\underline{s}, \underline{t}).$$

Using 1.2 Lemma 1 we get

$$\theta' \leq_{AC} \theta'' \ [V].$$

With 3.5 Lemma 4 below we have

$$\theta \leq_{AC} \theta''*\alpha \ [V]$$

and hence there exists $\sigma := \theta''*\alpha \in U\Sigma_{G\text{-}AC}(s, t)$ (returned by G-AC-UNIFY(s, t)) such that $\theta \leq_{AC} \sigma$ [V].

STEP 3: In this case the subproblems are considered and moreover there exists $(s', t') \in SP(s, t)$ with $\theta s' =_{AC} \theta t'$. By Noetherian induction there exists $\sigma' \in U\Sigma_{G\text{-}AC}(s', t')$ (returned by G-AC-UNIFY(s', t')) such that

$$\theta \leq_{AC} \sigma' \ [V'] \quad \text{with } V' = \mathbf{V}(s', t').$$

In other words there exists λ with $\theta =_{AC} \lambda\sigma'$ [V] using 1.2 Lemma 1.

But then λ is a unifier of σ's and σ't. By Noetherian induction there exists $\sigma'' \in U\Sigma_{G\text{-}AC}(\sigma's, \sigma't)$ (returned by G-AC-UNIFY (σ's, σ't)) with

$$\lambda \leq_{AC} \sigma'' \ [V''] \quad \text{and } V'' = \mathbf{V}(\sigma's, \sigma't).$$

Using 1.2 Lemma 1 and 1.2 Lemma 2 we obtain

$$\lambda\sigma' \leq_{AC} \sigma''\sigma' \ [V]$$

and hence with $\sigma := \sigma''\sigma'$ there exists $\sigma \in U\Sigma_{G\text{-}AC}(s, t)$ (returned by G-AC-UNIFY(s, t)) such that

$$\theta \leq_{AC} \sigma \ [V]. \qquad \blacksquare$$

While this completes the main result of this paragraph, some technical lemmata remain to be shown stating the existence of certain substitutions in STEP 2. Regarding the situation in STEP 2 we have two terms s and t starting with the same AC-function symbol, θ a unifier of s and t and for all subproblems $(s', t') \in SP(s, t)$ it is $\theta s' \neq_{AC} \theta t'$. Let $\{r_1, \ldots, r_n\}$ be the immediate alien subterms of s and t, $\alpha = \{c_1 \leftarrow r_1, \ldots, c_n \leftarrow r_n\}$ and $\alpha_\theta = \{c_1 \leftarrow \theta r_1, \ldots, c_n \leftarrow \theta r_n\} = \theta\alpha|_{DOM\alpha}$ Since $\theta r_i \neq_{AC} \theta r_j$ for $i \neq j$, $1 \leq i, j \leq n$ by assumption let $a = [r_1 \Leftarrow c_1, \ldots, r_n \Leftarrow c_n]$ and $a_\theta = [\theta r_1 \Leftarrow c_1, \ldots, \theta r_n \Leftarrow c_n]$ be the corresponding subterm replacements. If $\theta = \{x_1 \leftarrow p_1, \ldots, x_m \leftarrow p_m\}$ we then define $\theta' = \{x_1 \leftarrow a_\theta p_1, \ldots, x_m \leftarrow a_\theta p_m\}$. Furthermore we denote by $\underline{s} = \alpha s$ and $\underline{t} = \alpha t$ the constant abstractions of s and t and $V = \mathbf{V}(s, t)$ is the set of variables in s and t.

3.5 ***Lemma*** 1 : $\theta'(\alpha s) = \alpha_\theta \theta s$ and $\theta'(\alpha t) = \alpha_\theta \theta t$.

Proof: We only show the first equation for all subterms r in $\underline{s} = \alpha s$.
For $r \in \mathbf{C}$ we distinguish the cases $r \neq c_i$ and $r = c_i$ for $1 \leq i \leq n$. For the first case $\theta' r = \alpha_\theta \theta r$ is obvious. For $r = c_i$ we have $\theta' c_i = c_i$ and there exists a subterm r' in s with $r' =_{AC} r_i$ and $\alpha_\theta \theta r' =_{AC} \alpha_\theta \theta r_i = c_i$. Now let $r = x \in \mathbf{V}$. If $x \notin DOM\theta' = DOM\theta$ then $\theta' x = x = \theta x = \alpha_\theta \theta x$ and for $x \in DOM\theta$ it is $\theta' \alpha x = \alpha_\theta \theta x$. Since there occur no immediate alien subterms in $\underline{s}$ we have for all subterms r of $\underline{s}$ with $r \notin \mathbf{C} \cup \mathbf{V}$ $\theta' r = \alpha_\theta \theta r$. ■

3.5 ***Lemma*** 2 : For all terms q not containing $c_1, \ldots, c_n$: $\theta\alpha\alpha_\theta q =_{AC} \theta q$.

Proof: Suppose there exists a subterm r in q with $r =_{AC} \theta r_i$ then r is replaced by α_θ to c_i. Applying $\theta\alpha$ to that c_i we again have θr_i. If $r \neq_{AC} \theta r_i$ we have $\alpha\alpha_\theta r = r$ since q does not contain any of the c_i and hence $\theta\alpha\alpha_\theta q =_{AC} \theta q$. ■

3.5 ***Lemma*** 3 : θ' is a unifier of $\underline{s}$ and $\underline{t}$ such that

$$\theta \leq_{AC} \theta' * \alpha \; [V] \quad \text{with} \quad V = \mathbf{V}(s, t).$$

Proof: First θ' is a unifier of $\underline{s}$ and $\underline{t}$. By 3.5 Lemma 5 and the fact that θ unifies s and t we have:

$$\theta' \underline{s} = \theta'(\alpha s) = \alpha_\theta(\theta s) =_{AC} \alpha_\theta(\theta t) = \theta'(\alpha t) = \theta' \underline{t}.$$

We now show that α and θ' are unifiable, i.e. the merge exists. Using 3.2 Lemma 1 we define $h_1 = (x_1, \ldots, x_m, c_1, \ldots, c_n)$ and $h_2 = (\alpha_\theta p_1, \ldots, \alpha_\theta p_m, r_1, \ldots, r_n)$ then

$$\begin{aligned}
\theta\alpha h_2 &= (\theta\alpha\alpha_\theta p_1, \ldots, \theta\alpha\alpha_\theta p_m, \theta\alpha r_1, \ldots, \theta\alpha r_n) \\
&=_{AC} (\theta p_1, \ldots, \theta p_m, \theta r_1, \ldots, \theta r_n) \qquad \text{by 3.5 Lemma 6} \\
&= (\theta\alpha x_1, \ldots, \theta\alpha x_m, \theta\alpha c_1, \ldots, \theta\alpha c_n) \\
&= \theta\alpha h_1.
\end{aligned}$$

Hence $\theta\alpha \leq_{AC} \lambda \; [W]$ where λ is a most general unifier of α and θ' and W are the variables of α and θ'. Therefore we have with $V \subseteq W$

$$\theta = \theta\alpha \leq_{AC} \lambda\alpha = \theta' * \alpha \; [V].$$ ■

3.5 ***Lemma*** 4: Let θ'' be a unifier of $\underline{s}$ and $\underline{t}$ with $\theta' \leq_{AC} \theta'' \; [V]$ and $VCOD\theta'' \cap V = \emptyset$. Then θ'' and α are AC-unifiable and $\theta \leq_{AC} \theta'' * \alpha \; [V]$.

Proof: We assume w.l.o.g. that $DOM\theta'' = V$ (if not define $\theta''x = z$ for $x \in V \backslash DOM\theta''$ and z is a new variable which does not occur in the problem). Since $\theta' \leq_{AC} \theta''$ [V] there exists δ with $\theta' =_{AC} \delta\theta''$ [V] and

(1) $DOM\delta \cap V = \emptyset$ and $DOM\delta \subseteq VCOD\theta''$.

Furthermore using $VCOD\theta'' \cap V = \emptyset$

(2) $DOM\delta \cap \mathbf{V}(\alpha) = \emptyset$.

We now show that θ'' and α are unifiable. Using 3.2 Lemma 1 we define $g_1 = (x_1, ..., x_m, c_1, ..., c_n)$ and $g_2 = (\theta''x_1, ..., \theta''x_m, r_1, ..., r_n)$. Now we have $\delta g_1 = g_1 = h_1$ by (1) and (2) (confer for the definition of h_1 and h_2 the proof of the last lemma) and $\delta g_2 = h_2$ by (2). Since h_1 and h_2 are unifiable by $\theta\alpha$ g_1 and g_2 are unifiable by $\theta\alpha\delta$ and therefore $\theta\alpha\delta \leq_{AC} \theta''*\alpha$ [V]. Hence with $\theta\alpha\delta = \theta$ [V] we have $\theta \leq_{AC} \theta''*\alpha$ [V]. ■

4. Conclusion

We presented an AC-unification algorithm, which is not based on the 'variable abstraction' process of previous algorithms, but exploits an early idea of [LS 76]: the reduction to ***inhomogeneous*** Diophantine equations.
The algorithm is extended to handle uninterpreted function symbols. While previous suggestions for such an extension [St 81] [Ht 80] [Fa 83] [Fa 84] [Fa 85] [Fo 83] and [Fo 85] use the same 'variable abstraction' process, which was already at the heart of the STICKEL-algorithm, we propose a method quite to the contrary. Since the replacement of subterms by variables turned out to be the culprit (in terms of efficiency) in the basic STICKEL-algorithm we propose to replace subterms by 'special' constants for the reduction to proceed. Such a reduction is possible since our basic algorithm does not require the variable abstarction process in the first place.
The algorithms are presented with the aim of clarity, not for an actual implementation.
The extended algorithm is not minimal in general. Minimality can always be achieved for finitary theories with a decidable matching problem (eliminate in a second pass all those unifiers that are not instances), however it would be computationally advantageous to generate one minimal set in the first place. It is an open problem to find such a minimality condition for our algorithm which is less complex than the above elimination process. We have certain conditions which will eliminate the proliferation of redundant unifiers to some extend.

5. REFERENCES:

[Bü 85] Büttner, W., 'Unification in the Datastructure Multisets', MEMO SEKI-85-V-KL, Universität Kaiserslautern, (1985) (to appear in 'Journal of Automated Reasoning')

[BS 81] Burris, S., and Sankappanavar, H.P., 'A Course in Universal Algebra', Springer-Verlag,(1981)

[CL 73] Chang, C.-L., and Lee, R.C.-T., 'Symbolic Logic and Theorem Proving', Academic Press, (1973)

[CP 67] Clifford, A.H., and Preston, G.B., 'The Algebraic Theory of Semigroups', Volume 2, American Mathematical Society, (1967)

[Di 13] Dickson, L.E., 'Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prim Factors', Amer. J. Math. 35, 413-422, (1913)

[Fa 83] Fages, F., 'Formes canoniques dans les algèbres booléennes, et application à la démonstration automatique en logique de premier ordre', Thèse de 3ème Cycle, Université Paris VI, (1983)

[Fa 84] Fages, F., 'Associative-Commutative Unification', in Proc. of 7th CADE (ed. Shostak, R.E.), Springer-Verlag, LNCS 170, 194-208, (1984)

[Fa 85] Fages, F., 'Associative-Commutative Unification', Technical Report, INRIA, BP 105, 78153 Le Chesnay, (1985) (to appear)

[FH 83] Fages, F. and Huet, G., 'Unification and Matching in Equational Theories', Proc. of CAAP'83 (ed. Ausiello, G. and Protasi, M.), Springer-Verlag, LNCS 159, 205-220, (1983)

[Fo 83] Fortenbacher, A., 'Algebraische Unifikation', Diplomarbeit, Universität Karlsruhe, (1983)

[Fo 85] Fortenbacher, A., 'An Algebraic Apporach to Unification under Associativity and Commutativity', in Proc. of 'Rewriting Techniques and Applications' (ed. Jouannaud, J.-P.), Springer-Verlag, LNCS 202, 381-397, (1985)

[G 1873] Gordan, P., 'Über die Auflösung linearer Gleichungen mit reellen Coefficienten' , Mathematische Annalen, 23-28, (1873)

[GH 85] Guckenbiehl, T., and Herold, A., 'Solving Linear Diophantine Equations', MEMO SEKI-III-KL, Universität Kaiserslautern, (1985)

[Gr 79] Grätzer, G., 'Universal Algebra', Springer-Verlag, (1979)

[GT 78] Goguen, J. A., Thatcher, J.W. and Wagner, E. G., 'An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types', in 'Current Trends in Programming Methodology, Vol.4, Data Structuring' (ed. Yeh, R. T.), Prentice Hall, (1978)

[HO 80] Huet, G. and Oppen, D.C., 'Equations and Rewrite Rules: A Survey', in 'Formal Languages: Perspectives and Open Problems (ed Book, R.), Academic Press, (1980)

[Ht 80] Hullot, J.M., 'Compilation de formes canoniques dans des théories équationelles, Thèse de 3^{éme} Cycle, Université de Paris-Sud, (1980)

[Hu 76] Huet, G., 'Résolution d'équations dans des langages d'ordre 1, 2 ,...., ω', Thèse de doctorat d'état, Université Paris VII, (1976)

[Hu 78] Huet, G., 'An Algorithm to Generate the Basis of Solutions to Homogeneous Linear Diophantine Equations', Information Processing Letters", Vol 7, Nº. 3, 144-147, (1978)

[Hw 72] Hewitt, C., 'Description and Theoretical Analysis of PLANNER, a Language for Proving Theorems and Manipulating Models in a Robot', Ph. D. thesis, Dept. of Mathematics, MIT, (1972)

[Ki 85] Kirchner, C., 'Methodes et outils de conception systematique d'algorithmes d'unification dans les théories équationelles', Thèse de doctorat d'état, Université de Nancy 1, (1985)

[KB 70] Knuth, D., and Bendix, P., 'Simple Word Problems in Universal Algebras', in Proc. of 'Computational Problems in Abstract Algebra' (ed Leech, J.), Pergamon Press, 263-297, (1970)

[La 85] D. Lankford, 'A New Non-negative Integer Basis Algorithm for Linear Homogeneous Equations with Integer Coefficients', unpublished, (1985)

[Lo 78] Loveland, D., 'Automated Theorem Proving: A logical Basis', North-Holland, (1978)

[LS 76] Livesey, M. and Siekmann, J. , 'Unification of AC-Terms (bags) and ACI-Terms (sets)', Internal Report, University of Essex (1975) and Technical Report 3-76, Universität Karlsruhe, (1976)

[LS 78] Livesey, M. and Siekmann, J. , 'Unification of Sets and Multisets', SEKI-Technical Report, Universität Karlsruhe, (1978)

[Pl 72] Plotkin, G.D., 'Building-in Equational Theories', Machine Intelligence 7 (eds Meltzer, B. and Michie, D.), 73-90, (1972)

[PS 81] Peterson, M.S., and Stickel, M.E., 'Complete Sets of Reductions for Equational Theories with Complete Unification Algorithms', JACM 28, Nº 2, 322-264, (1981)

[RD 72] Rulifson, J.F., Derkson, J.A., and Waldinger, R.J., 'QA4: A Procedural Calculus for Intuitive Reasoning', Techn. Report, Stanford Research Institute, AI-Research Centre, TR-73, (1972)

[Ro 65] Robinson, J.A., 'A Machine-Oriented Logic Based on the Resolution Principle', JACM 12, Nº. 1, 23-41, (1965)

[RW 69] Robinson, G.A., and Wos, L., 'Paramodulation and Theorem Proving in First Order Theories with Equality', Machine Intelligence 4 (eds Meltzer, B. and Michie, D.), American Elsevier, 135-170, (1969)

[Sc 86] Schmidt-Schauß, M., 'Unification under Associativity and Idempotence is of Type Nullary', MEMO SEKI, Universität Kaiserslautern, (1986) (to appear in 'Journal of Automated Reasoning')

[Si 84] Siekmann, J., 'Universal Unification', in Proc. of 7th CADE (ed. Shostak, R. E.), Springer-Verlag, LNCS 170, 1-42, (1984)

[Si 86] Siekmann, J., 'First Order Unification Theory' (invited survey), to appear in Proc. of 'European Conference on Artificial Intelligence (ECAI)',(1986)

[St 75] Stickel, M.E., 'A Complete Unification Algorithm for Associative-Commutative Functions', in Proc. of 4th IJCAI, Tblisi, USSR, 71-82, (1975)

[St 76] Stickel, M.E., 'Unification Algorithms for Artifcial Intelligence', Ph. D. thesis, Carnegie-Mellon University, (1976)

[St 81] Stickel, M.E., 'A Unification Algorithm for Associative-Commutative Functions', JACM 28, Nº. 3, 423-434, (1981)

[St 84] Stickel, M.E., 'A Case Study of Theorem Proving by the Knuth-Bendix Method Discovering that $X^3 = X$ Implies Ring Commutativity', in Proc. of 7th CADE (ed Shostak, R.E.), Springer-Verlag, LNCS 170, 248-258, (1984)

[WO 84] Wos, L., Overbeek, R., Lusk, E., and Boyle, J., 'Automated Reasoning - Introduction and Applications', Prentice-Hall,(1984)

Aquarelle White DFS 5 mm for 31-50 sheets 541
www.bindomatic.com