RWTH Aachen
Lehr- und Forschungsgebiet Theoretische Informatik

Universität München
Centrum für Informations- und Sprachverarbeitung

General A- and AX-Unification via Optimized Combination Procedures

Franz Baader & Klaus U. Schulz

In: Pecuchet Abdulrab, editor, *Word Equations and Related Topics IWWERT91*, volume 677 of *Lecture Notes in Computer Science*, pages 23–42, Rouen, France, 1991. Springer.

General A- and AX-Unification via Optimized Combination Procedures

Franz Baader[1] and Klaus U. Schulz[2]

[1] DFKI, Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany
[2] CIS, University Munich, Leopoldstr. 139, 8000 München 40, Germany

Abstract. In a recent paper [BS91] we introduced a new unification algorithm for the combination of disjoint equational theories. Among other consequences we mentioned (1) that the algorithm provides us with a decision procedure for the solvability of general A- and AI-unification problems and (2) that Kapur and Narendran's result about the NP-decidability of the solvability of general AC- and ACI-unification problems (see [KN91]) may be obtained from our results. In [BS91] we did not give detailled proofs for these two consequences. In the present paper we will treat these problems in more detail. Moreover, we will use the two examples of general A- and AI-unification for a case study of possible optimizations of the basic combination procedure.

1 Introduction

For a long time, most applications of equational unification presupposed algorithms which solve the enumeration problem, i.e. the problem to enumerate in a compact way the set of all solutions of a given set of equations modulo an equational theory E. The recent development of constraint approaches to logic programming (see e.g., [JL87, Co90]), theorem proving (see e.g., [Bü90]) and term rewriting (see e.g., [KK89]) emphasizes the need for satisfiability checking and the decision problem received a new status. The decision problem for E is the problem to decide for a given set of equations whether they have a solution modulo E or not.

In the first part of this paper we shall study the decision problem in the context of two equational theories, namely the theories A and AI, expressing associativity respectively associativity plus idempotency of a function symbol. To be more precise, a distinction has to be made with respect to the signature which is used to built the terms of a unification problem.

In the simple case only the function symbol occurring in the equational axiom(s) is used, beside constants and variables. This type of problem will be called *unification with constants*. Associative unification problems with constants may be regarded as finite systems of equations in a free semigroup, often called word equations. Makanin's algorithm ([Ma77]) shows that the solvability of

finite systems of word equations is decidable. Decidability of AI-unification problems with constants follows easily from the fact that idempotent semigroups are locally finite (see [Mc54]).

However, applications in constraint logic programming and other areas presuppose that terms may contain additional free function symbols. The Prolog III term

$$< FF(1, < 1 > \bullet x, 2) > \bullet x \bullet < FF(x, y, y), 1 >$$

for example may be regarded as a "generalized word" corresponding to a term

$$f(1, 1 \circ x, 2) \circ x \circ f(x, y, y) \circ 1$$

where "f" is a free function symbol, "1" and "2" are constants and "$\circ$" is an associative function symbol in infix notation. Constraints over lists as used in Prolog III stand in a close relationship with *general associative unification problems*, i.e., associative unification problems with free function symbols.

The decidability of general A- and AI-unification problems was open for a long time (compare Kapur and Narendran's table of known decidability and complexity results in unification theory [KN91]). A positive answer was obtained in our paper [BS91] where we used the fact that general E-unification may be regarded as an instance of the *combination problem*. The combination problem (see e.g, [BS91, Sc89]) is concerned with the question of how to derive unification algorithms (for the enumeration problem or the decision problem) for unification in the union of equational theories over disjoint signatures from unification algorithms in the single theories. In [BS91] the following general theorem was proved:

Theorem: *Let $E_1, \ldots, E_n$ be equational theories over disjoint signatures such that solvability of E_i-unification problems with linear constant restriction is decidable for $i = 1, \ldots, n$. Then unifiability is decidable for the combined theory $E_1 \cup \ldots \cup E_n$.*

Linear constant restrictions are induced by a linear order "$<$" on the set $X \cup C$ of variables and constants, demanding that, for a unifier θ, the constant c must not occur in $\theta(x)$ if $c > x$. The combination algorithm which is used to establish this result will be described in section 2. The description — which is the one given in [BS91] — is conceptually clear, but it is not the most appropriate one for efficient implementation since it does not try to minimize the number of possible choices in the non-deterministic steps. For this reason we will discuss optimized versions of the basic combination procedure in the context of general A- and AI-unification problems in section 3.

Let us return to the theorem. Obviously, every general A-unification (resp. general AI-unification) problem with set of free function symbols Ω may be regarded as a combination of the theory A (AI) with the free theory $F_\Omega = \{f(x_1, ..., x_n) = f(x_1, ..., x_n); f \in \Omega\}$. For the latter theory unification is simply

unification in the empty theory. As a matter of fact, the solvability of Robinson unification problems with linear constant restrictions is decidable. If at least one unifier for such a problem satisfies the constant restrictions, then the most general unifier will satisfy them, and vice versa.

In order to prove that solvability of general A- (AI-) unification problems is decidable it remains to show that solvability of A- (AI-) unification problems with linear constant restrictions is decidable. These problems, and the related problems of A- (AI-) unification with *partially specified* linear constant restrictions (which arise from our optimization methods) will be discussed in sections 4 and 5.

Another problem which was solved only recently is the complexity of general AC- and ACI-unification, where AC (ACI) is the theory of one associative and commutative (and idempotent) function symbol. Kapur and Narendran [KN91] could prove that both problems are NP-decidable. In the last section we shall show that this observation follows easily from our combination result.

2 The Combination Algorithm

For the sake of convenience we shall restrict the presentation of the basic algorithm to the combination of two theories. The combination of more than two theories can be treated analogously. Before we can start with the description of the algorithm we have to introduce some notation.

Let E_1, E_2 be two equational theories built over disjoint signatures Ω_1, Ω_2, and let $E = E_1 \cup E_2$ denote their union. Since we are only interested in elementary E-unification[3], we can restrict our attention to terms built from variables and symbols of $\Omega_1 \cup \Omega_2$. The elements of Ω_1 will be called 1-*symbols* and the elements of Ω_2 2-*symbols*. A term t is called i-*term* iff it is of the form $t = f(t_1, ..., t_n)$ for an i-symbol f $(i = 1, 2)$. A subterm s of a 1-term t is called *alien subterm* of t iff it is a 2-term such that every proper superterm of s in t is a 1-term. Alien subterms of 2-terms are defined analogously. An i-term s is *pure* iff it contains only i-symbols and variables. An equation $s \doteq t$ is pure iff there exists an $i, 1 \leq i \leq 2$, such that s and t are pure i-terms or variables; this equation is then called an i-equation. Please note that according to this definition equations of the form $x \doteq y$ where x and y are variables are both 1- and 2-equations. In the following, the symbols x, y, z, with or without indices, will always stand for variables.

Example 1. Let Ω_1 consist of the binary (infix) symbol "$\circ$" and Ω_2 of the unary symbol "h", let $E_1 := \{x \circ (y \circ z) = (x \circ y) \circ z\}$ be the theory which says that "$\circ$" is associative, and let $E_2 := \{h(x) = h(x)\}$ be the free theory for "h".

[3] i.e, unification where the terms of the unification problems are built over the signature of E.

The term $y \circ h(z \circ h(x))$ is a 1-term which has $h(z \circ h(x))$ as its only alien subterm. The equation $h(x_1) \circ x_2 \doteq y$ is not pure, but it can be replaced by two pure equations as follows. We replace the alien subterm $h(x_1)$ of $h(x_1) \circ x_2$ by a new variable z. This yields the pure equation $z \circ x_2 \doteq y$. In addition, we consider the new equation $z \doteq h(x_1)$. This process of replacing alien subterms by new variables is called *variable abstraction*. It will be the first of the five steps of our combination algorithm.

Combination Algorithm — Basic Form

The input for the *combination algorithm* is an elementary E-unification problem, i.e., a system $\Gamma_0 = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$, where the terms $s_1, \ldots, t_n$ are built from variables and the function symbols occurring in $\Omega_1 \cup \Omega_2$, the signature of $E = E_1 \cup E_2$. The first two steps of the algorithm are deterministic, i.e., they transform the given system of equations into one new system.

Step 1: variable abstraction. Alien subterms are successively replaced by new variables until all terms occurring in the system are pure. To be more precise, assume that $s \doteq t$ or $t \doteq s$ is an equation in the current system, and that s contains the alien subterm s_1. Let x be a variable not occurring in the current system, and let s' be the term obtained from s by replacing s_1 by x. Then the original equation is replaced by the two equations $s' \doteq t$ and $x \doteq s_1$. This process has to be iterated until all terms occurring in the system are pure. It is easy to see that this can be achieved after finitely many iterations. Now all the terms in the system are pure, but there may still exist non-pure equations, consisting of a 1-term on one side and a 2-term on the other side.

Step 2: split non-pure equations. Each non-pure equations of the form $s \doteq t$ is replaced by two equations $x \doteq s, x \doteq t$ where the x are always new variables.

It is quite obvious that these two steps do not change solvability of the system. The result is a system which consists of pure equations. The third and the fourth step are nondeterministic, i.e., a given system is transformed into finitely many new systems. Here the idea is that the original system is solvable iff at least one of the new systems is solvable.

Step 3: variable identification. Consider all possible partitions of the set of all variables occurring in the system. Each of these partitions yields one of the new systems as follows. The variables in each class of the partition are "identified" with each other by choosing an element of the class as representative, and replacing in the system all occurrences of variables of the class by this representative.

Step 4: choose ordering and theory indices. This step doesn't change a given system, it just adds some information which will be important in the next step. For a given system, consider all possible strict linear orderings $<$ on the variables of the system, and all mappings *ind* from the set of variables into the set of theory indices $\{1, 2\}$. Each pair $(<, ind)$ yields one of the new systems obtained from the given one.

The last step is again deterministic. It splits each of the systems already obtained into a pair of pure systems.

Step 5: split systems. A given system Γ is split into two systems Γ_1 and Γ_2 such that Γ_1 contains only 1-equations and Γ_2 only 2-equations. These systems can now be considered as unification problems with linear constant restriction. In the system Γ_i, the variables with index i are still treated as variables, but the variables with alien index $j \neq i$ are treated as free constants. The linear constant restriction for Γ_i is induced by the linear ordering chosen in the previous step.

The output of the algorithm is thus a finite set of pairs (Γ_1, Γ_2) where the first component Γ_1 is an E_1-unification problem with linear constant restriction, and the second component Γ_2 is an E_2-unification problem with linear constant restriction.

Proposition 1. *The input system Γ_0 is solvable if and only if there exists a pair (Γ_1, Γ_2) in the output set such that Γ_1 and Γ_2 are solvable.*

A proof of this proposition was given in [BS91]. It was also shown how the combination procedure can be used in order to generate a complete set of unifiers for the combined theory, provided that there are algorithms which enumerate complete sets of unifiers for unification problems with constant restrictions in the single theories.

As mentioned in the introduction, the presentation given above is conceptually simple but inefficient since no attempt was made to minimize the number of possible choices in the non-deterministic steps. Let us now give an optimized version of the combination algorithm in a particular context.

3 Optimization for Combination with the Free Theory

Before we treat the cases of general A- and AI- unification let us consider the situation where an arbitrary equational theory E is combined with an instance of the free theory

$$F_\Omega := \{f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n); f \in \Omega\}$$

in order to obtain a unification algorithm for general E-unification.

We shall assume that the variables occurring in the unification problems are elements of a countable set Z which has fixed standard linear ordering $<_Z$. The following optimized combination algorithm rules out several choices which are possible in the non-deterministic steps of the non-optimized version.

Optimized Combination Algorithm — Version 1

Context: Combination of an equational theory E with the free theory F_Ω. Decision problem for general E-unification.

The input is a finite system Γ_0 of equations between terms built from variables, free constants and function symbols belonging to Ω or the signature of E. Without loss of generality we assume that Γ_0 does not contain an equation of the form $x \doteq y$ or $x \doteq a$ where x, y are variables and a is a free constant. In the following procedure, Γ_i denotes the system which is reached after step i. If all equations of Γ_i are pure, then $\Gamma_{i,F}$ denotes the subsystem of all equations which only contain free function symbols, $\Gamma_{i,E}$ denotes the subsystem with the equations containing function symbols from the signature of E. The first three steps are deterministic:

Step 1: variable abstraction. as before.
Step 2: split non-pure equations. as before.
Step 3: first variable identification. We solve the free subsystem $\Gamma_{2,F}$, treating all variables as variables and using standard unification in the free theory. If $\Gamma_{2,F}$ is unsolvable, then we stop with failure. In the following we shall assume that $\Gamma_{2,F}$ is solvable, with most general unifier (mgu) τ_1, say. Let Y_2 be the set of variables occurring in Γ_2. The maximal subsets of variables with same image under τ_1 define a partition π_1 of Y_2. Based on this partition we now identify variables in Γ_2, as described in step 3 of the non-optimized version. We obtain the system Γ_3 with set of variables Y_3.

The following steps are non-deterministic.

Step 4: second variable identification. We choose a partition π_2 of Y_3 and identify variables in Γ_3. We obtain the system Γ_4 with set of variables Y_4. If $\Gamma_{4,F}$ is unsolvable, or if the mgu τ_2 of $\Gamma_{4,F}$ identifies two distinct variables x and y of Y_4, then we stop this path with failure and backtrack. In the following we shall assume that $\Gamma_{4,F}$ is solvable with mgu τ_2 such that $\tau_2(x) \neq \tau_2(y)$ for all $x \neq y$ in Y_4.
Step 5: choose theory indices and ordering. Step 4 of the basic version is modified, imposing several restrictions on the indices *ind* and the linear orderings $<$ which may be selected. We first choose a variable indexing *ind* which satisfies the following condition: for every variable $x \in Y_4$,

– if $\tau_2(x) = f(t_1, \ldots, t_n)$ for a free function f, then $ind(x) = F$.

Let $Y_{4,F} \subseteq Y_4$ denote the set of all variables x with $ind(x) = F$, let $Y_{4,E} \subseteq Y_4$ be the remaining set of variables. We now choose a linear ordering $<_F$ on $Y_{4,F}$ which satisfies the following restriction: for all variables $x, x' \in Y_{4,F}$:

– if $\tau_2(x')$ is a subterm of $\tau_2(x)$, then $x' <_F x$.

Now $<_F$ will be extended to a linear ordering $<$ of Y_4. As a consequence of the following two conditions, this step is completely deterministic: for all $y, y_1, \ldots, y_k \in Y_{4,E}$,

– if y does not occur in any term $\tau_2(x)$, for $x \in Y_{4,F}$, then $x < y$ for all $x \in Y_{4,F}$. Otherwise, if x is the minimal element of the ordering $<_F$ such that y occurs in $\tau_2(x)$, then $y < x$. If x has an immediate predecessor x' in the ordering $<_F$, then $x' < y$,

– if $y_1 < y_2 < \ldots < y_k$ is a sequence of consecutive E-variables in $<$ (i.e., without F-variables in between), then $y_1 <_Z y_2 <_Z \ldots <_Z y_k$,

Eventually we check if the corresponding condition holds also for F-variables:

– if $x_1 < x_2 < \ldots < x_k$ is a sequence of consecutive F-variables in $<$, then $x_1 <_Z x_2 <_Z \ldots <_Z x_k$.

If the last condition is violated, then we stop with failure. In the other case the equational system is not modified, $\Gamma_5 := \Gamma_4$.

Step 6: split systems. as before. We obtain the systems $\Gamma_{6,F}$ and $\Gamma_{6,E}$ with linear constant restrictions induced by $<$.

Since, by construction of $<$, τ_2 is a solution of $\Gamma_{6,F}$, the systems of type $\Gamma_{6,E}$ may be regarded as the output of the combination procedure.

Proposition 2. *Γ_0 is solvable iff a system $\Gamma_{6,E}$ in the output set is solvable.*

Proof. We need a tool which was already introduced in [BS 91], namely unfailing completion. In the given context it suffices to apply this procedure to the theory E. This yields a possibly infinite ordered-rewriting system R which is confluent and terminating on ground terms. As in [BS 91], this system is also applied to terms containing variables from a fixed countable set of variables X_0 — for the completion these variables can simply be treated like constants. Let $T(\Omega, E, X_0)$ be the set of all terms built from function symbols in Ω, functions symbols of the signature of E and variables in X_0. Let $T_{\downarrow R}$ denote its R-irreducible elements. A substitution σ is called R-normalized on a finite set of variables Y iff $\sigma(y) \in T_{\downarrow R}$ for all variables $y \in Y$.

Since every output of the optimized version 1 is a possible output of the non-optimized procedure, correctness of the latter procedure implies correctness of the former.

Let us now prove completeness. Suppose that Γ_0 has a solution σ. We may assume that σ is also a solution of Γ_2 and that σ is R-normalized on the set Y_2 of variables occurring in Γ_2. In [BS 91] it is described how σ may be used to determine choices in the non-deterministic steps of the non-optimized version which

lead to output systems $\Gamma_F^{(n.o.)}$ (free subsystem) and $\Gamma_E^{(n.o.)}$ (E-subsystem) which have solutions $\sigma_F^{(n.o.)}$ and $\sigma_E^{(n.o.)}$ which respect the linear constant restrictions. In more detail, the following choices were made:

(a) in step 3, partition $\pi^{(n.o.)}$ is chosen in such a way that variables x and y are identified iff $\sigma(x) = \sigma(y)$,

(b) in step 4 we define $ind^{(n.o.)}(x) := E$ iff $\sigma(x) = h(t_1, \ldots, t_n)$ where the function symbol h belongs to the signature of E, and $ind^{(n.o.)}(x) := F$ otherwise,

(c) the linear ordering $<^{(n.o.)}$ which is chosen is an arbitrary extension of the partial ordering $\prec$ defined by $y \prec x$ iff $\sigma(y)$ is a strict subterm of $\sigma(x)$.

We may also assume that $<^{(n.o.)}$ satisfies the following two conditions:

(d) if $x_1 <^{(n.o.)} x_2 <^{(n.o.)} \ldots <^{(n.o.)} x_k$ is a sequence of consecutive variables with $ind^{(n.o.)}(x_1) = ind^{(n.o.)}(x_2) = \ldots = ind^{(n.o.)}(x_k)$, then $x_1 <_Z x_2 <_Z \ldots <_Z x_k$.

In fact, permutation of elements in the subsequence $x_1 <^{(n.o.)} \ldots <^{(n.o.)} x_k$ does not modify the constant restrictions imposed on the output systems.

(e) if $ind^{(n.o.)}(x) = F$, $ind^{(n.o.)}(y) = E$ and if $y <^{(n.o.)} x$ is the immediate predecessor of x, then y occurs in $\sigma_F^{(n.o.)}(x)$.

Otherwise we could just change the order of these two variables, leaving all other order relations untouched. With this modification a new constant restriction for $\Gamma_F^{(n.o.)}$ is created, demanding that y must not occur in the value of x. But $\sigma_F^{(n.o.)}$ satisfies this condition and is still a solution of the modified system. For $\Gamma_E^{(n.o.)}$, one constant restriction is erased and $\sigma_E^{(n.o.)}$ is a solution.

We shall now show that some choices in the non-deterministic steps of the optimized version lead to the same output pair $(\Gamma_{6,F}, \Gamma_{6,E}) = (\Gamma_F^{(n.o.)}, \Gamma_E^{(n.o.)})$. Let us first show that step 3 will not fail. It is not difficult to see that $\Gamma_F^{(n.o.)}$ is just a restricted version of $\Gamma_{2,F}$: if we disregard constant restrictions, then $\Gamma_F^{(n.o.)}$ may be obtained from $\Gamma_{2,F}$ by identification of variables as described in (a), then treating E-variables as constants. Thus solvability of $\Gamma_F^{(n.o.)}$ implies solvability of $\Gamma_{2,F}$. The partition π_1 of step 3 is a refinement of $\pi^{(n.o.)}$: we claim that

$$(*)\ \tau_1(x) = \tau_1(y) \text{ implies } \sigma(x) =_E \sigma(y).$$

Since σ is R-normalized on Y_2 we then get $\sigma(x) = \sigma(y)$. Let us consider (*) in more detail, since the same kind of argument will also be used later. When

we solve $\Gamma_{2,F}$ by a solved form algorithm (see e.g., [BB87, JK90]), then we add equations which are consequences of the equations in Γ_2. This follows from the fact that $=_E$ is a congruence relation and that we may derive the equations $r_1 = s_1, \ldots, r_n = s_n$ from an equation $f(r_1, \ldots, r_n) = f(s_1, \ldots, s_n)$ if f is a free function symbol. In particular every substitution which solves the equations in Γ_2 will also solve this derived equations. But the solved form contains the equations $x = \tau_1(x)$ and $y = \tau_1(y)$. Therefore $\sigma(x) =_E \sigma(\tau_1(x)) = \sigma(\tau_1(y)) =_E \sigma(y)$.

With the right choice in step 4 we will get the same variable identification as in the non-optimized procedure. This implies that σ is a solution of Γ_4. In particular σ solves $\Gamma_{4,F}$. By assumption (a), σ does not identify distinct variables of Y_4. It is easy to show that we may stop if $\tau_2(x) = \tau_2(y)$ for distinct variables $x, y \in Y_4$: with the same argument as above we see that $\sigma(x) =_E \sigma(\tau_2(x)) = \sigma(\tau_2(y)) =_E \sigma(y)$. Since σ is R-normalized it follows that $\sigma(x) = \sigma(y)$, a contradiction.

Let us now consider the restrictions of step 5. Note that $\Gamma_{4,F}$ and $\Gamma_F^{(n.o.)}$ are identical if we disregard constant restrictions in the latter system and treat E-variables as variables. If $\tau_2(x) = f(t_1, \ldots, t_n)$ for a free function symbol f, then $\sigma_F^{(n.o.)}(x)$ has top symbol f. Hence $\sigma_F^{(n.o.)}$ replaces x by a complex term. It does not treat it as a constant, therefore $ind^{(n.o.)}(x) = F$ and our choice in step 4 is conform. If $\tau_2(x')$ is a subterm of $\tau_2(x)$, then $\sigma(x') = \sigma(\tau_2(x'))$ is a strict subterm of $\sigma(\tau_2(x)) = \sigma(x)$. Thus $x' <^{(n.o.)} x$. It is now trivial to see that we may in fact choose $ind := ind^{(n.o.)}$ and "$<$" := "$<^{(n.o.)}$" in step 5. The rest is obvious. □

Remark. In version 1 of the optimized combination procedure some trivial hints were omitted which are relevant for efficient implementation. For example, the unifier τ_1 obtained in step 3 should be used when we decide solvability of $\Gamma_{4,F}$ in step 4. Another question which will not be discussed here is the question of how to determine an optimal order for the non-deterministic choices.

Partially Specified Linear Constant Restrictions

In some cases, one source of indeterminism in version 1 may be completely eliminated. The choice of a linear ordering in step 5 of version 1 of the optimized procedure can be avoided if it is possible to solve problems with *partially specified linear constant restrictions* for the theory E. Such a problem is given by an E-unification problem with constants, Γ, and a partial ordering $\leq$ on the variables and constants occurring in Γ. The problem has a solution iff there exists a linear extension of $\leq$ such that the unification problem with linear constant restriction induced by this extension has a solution.

Before we give a new version of the combination algorithm let us add a remark on the application area, in order to avoid misunderstandings. When we only ask for decidability per se, then unification problems with linear constant restrictions

and unification problems with partially specified linear constant restrictions have the same status. If solvability of the former class of problems is decidable, then the same holds for the latter class of problems, simply because every partial ordering on a finite set has only a finite number of linear extensions. The interest in partially specified linear constant restrictions relies on the fact that for various equational theories E the effort to decide solvability of unification problems with partially specified linear constant restrictions does not exceed the effort to decide solvability of unification problems with linear constant restrictions. Examples are the theories A and AI, as will be shown. Thus, the "disjunctive" treatment of several linear orderings becomes superfluous and the corresponding indeterminism is really eliminated, not only postponed.

Optimized Combination Algorithm — Version 2

Context: Decision problem for general E-unification where decidability of E-unification problems with partially specified linear constant restrictions is decidable (in "non-disjunctive manner").

We simply proceed as in version 1 until we reach the point in step 5 where the variable indexing *ind* has been chosen. Now we do not choose a linear ordering, but propagate the partial ordering $\leq$ defined by

$$y \leq x \text{ iff } \tau_2(y) \text{ is a strict subterm of } \tau_2(x)$$

to the remaining E-subsystem, defining thus a system $\Gamma_{6,E}$ with partially specified linear constant restriction.

Proposition 3. *Γ_0 is solvable iff some system $\Gamma_{6,E}$ which is reached in version 2 is solvable.*

Proof. Suppose that $\Gamma_{6,E}$ is solvable, let $<$ be a linear extension of $\leq$ such that the corresponding unification problem with linear constant restriction $\Gamma'_{6,E}$ is solvable. The proof of proposition 2 shows that we may assume without loss of generality that $<$ is a possible choice in step 5 of version 1. Correctness of version 1 implies correctness of version 2.

Now consider the completeness proof for version 1. Since the linear ordering $<$ which was chosen for step 5 is a linear extension of the partial ordering $\leq$ which is used in version 2, completeness of version 2 follows immediately. □

Let us now consider the particular case of general unification problems in the theory

$$A := \{h(h(x,y),z) = h(x,h(y,z))\}$$

expressing associativity of the binary function symbol h.

Optimized Combination Algorithm — Version 3

Context: Decision problem for general A-unification.

We may assume without loss of generality that the input problem does not contain any equation of the form $f(s_1, ..., s_m) \doteq h(t_1, t_2)$ where f is a free function symbol. Since A is collapse-free we could immediately stop with failure in such a case.

Step 1 remains as in versions 1 and 2. Obviously Γ_1 cannot have non-pure equations. Thus step 2 may be omitted. Step 3 (first variable identification) remains almost as in versions 1 and 2. We add, however, a control step: if $\tau_1(x) = f(s_1, \ldots, s_n)$ for a free function symbol f and if $\Gamma_{3,A}$ contains and equation $x \doteq h(t_1, t_2)$, then we stop with failure. In step 4 (second variable identification) we do not identify variables x and y if $\tau_1(x) = f(t_1, \ldots, t_n)$ for a free function symbol f and if $\Gamma_{3,A}$ contains an equation $y \doteq h(s_1, s_2)$, or vice versa. Step 5 of version 2 is modified, eliminating some indeterminism in the choice of a variable indexing *ind*. We define $ind(y) = A$ whenever $\Gamma_{4,A}$ contains an equation $y \doteq h(s_1, s_2)$.

Correctness and completeness of version 3 follow immediately from the fact that the theory A is collapse-free. Note that we will get A-unification problems with partially specified linear constant restrictions as output of version 3. "Non-disjunctive" decidability of such problems is possible, as we shall sketch in the next section.

In the context of general AI-unification, where

$$AI := \{h(h(x, y), z) = h(x, h(y, z)), h(x, x) = x\},$$

we may again use version 2, as will be shown in section 5. But the additional modifications which were introduced for general A-unification cannot be used here since AI contains the collapse equation $h(x, x) = x$.

4 A-Unification with Linear Constant Restriction

We first want to prove the following

Theorem 4. *A-unification with linear constant restriction is decidable.*

Let us start with a simplified representation of A-unification problems with linear constant restriction. The associativity axiom $h(h(x, y), z) = h(x, h(y, z))$ allows to write every term built from variables, free constants and the associative symbol h in the normalform

$$h(t_1, h(t_2, \ldots, h(t_{n-1}, t_n) \ldots))$$

(where every t_i is either a variable or a free constant) which corresponds to rightmost-bracketing if infix notation is used. Obviously there is a bijective correspondence between terms of this form and words

$$t_1 t_2 \ldots t_{n-1} t_n$$

where variables may be instantiated with non-empty words. This shows that every A-unification problem may be translated into a finite set of word equations, preserving unifiability in both directions. Every constant restriction of the A-unification problem translates directly into a constant restriction of the corresponding system of word equations. Thus we arrive at the following reformulation of our problem.

Theorem 5. *Unifiability of systems of word equations with linear constant restriction is decidable.*

There are two ways how we may prove this theorem. Let us first ignore matters of efficiency and just concentrate on the decidability result. In this case we may choose an even simpler formulation of the problem. A system (1) of word equations

$$U_1 \doteq V_1$$
$$U_2 \doteq V_2$$

may be rewritten into a single equivalent word equation (2)

$$U_1 a U_2 U_1 b U_2 \doteq V_1 a V_2 V_1 b V_2$$

where a and b are arbitrary distinct constants. Every unifier of (1) is a unifier of (2) and vice versa, thus both systems are also equivalent with respect to unifiability under linear constant restrictions. Therefore we may restrict considerations to single word equations.

Suppose now that a word equation WE is given with variables in $X = \{x_1, \ldots, x_n\}$, constants in C and with linear constant restriction induced by a linear ordering $<$ on $X \cup C$. We want to decide whether WE has a unifier which satisfies the constant restriction.

For technical reasons we would like to use ground unifiers only. Let us consider an alphabet $C' = C \cup \{c\}$ where c is a constant not occurring in C for this purpose. Then WE has a unifier θ over $C \cup X$ which satisfies the linear constant restriction induced by $<$ if and only if WE has a solution (i.e, a ground unifier) θ' over C' which satisfies the linear constant restriction induced by "$<$": to obtain θ' from θ we just replace all occurrences of variables by the constant c. In the converse direction, all occurrences of c are replaced by the same variable.

We are now in a position to use the following general result from [Sh90]:

Theorem: *If WE is a word equation with variables* $x_1, \ldots, x_n$ *and constants in the alphabet* C'*, and if* $L_1, \ldots, L_n$ *are regular languages over* C'*, then it is decidable whether WE has a solution* θ *such that* $\theta(x_i) \in L_i$ *for* $i = 1, \ldots, n$*.*

Let $L_i := C_i^+$, where

$$C_i := \{a \in C; a < x_i\} \cup \{c\}$$

$(i = 1, \ldots, n)$. Obviously a solution θ of WE satisfies the linear constant restriction induced by $<$ iff $\theta(x_i) \in L_i$ for $i = 1, \ldots, n$. Thus the theorem implies that unifiability of word equations with linear constant restrictions is decidable, and thus — by disjunctive treatment — also with partially specified linear constant restrictions.

The algorithm which was used in [Sh90] in order to establish the theorem mentioned above is, however, more complicated than it would be necessary for the special purpose of regular languages of the form C_i^+. In connection with the initial translation of systems of word equations into a single word equation and with the disjunctive treatment of different linear orderings for general A-unification via version 1 of the combination procedure we get a rather inefficient decision procedure.

For the readers which are familiar with Makanin's algorithm let us sketch how a better solution might look like. In particular we want to show that it is possible to avoid

- the translation of a system of word equations into a single word equation, which doubles the data size,
- the disjunctive treatment of linear extensions of the partial ordering $\leq$ obtained in step 5 of the optimized version 2.

Thus, the prerequisite which is needed for version 3 is in fact available in the case of A-unification.

First, it is not difficult to show that Makanin's algorithm for deciding the solvability of words equations may directly be used for *systems* of word equations (see e.g, [Sh90]). Second, by means of simple additional devices it is possible to modify Makanin's algorithm in such a way that we keep control over all constants which will occur in the value of a variable, following a certain path of the search tree. For this purpose, a sequence of columns of the actual generalized equation is used to represent an actual value $V(x)$ of a variable x occurring in WE. If a subcolumn of a column of $V(x)$ contains a constant base of type $a \in C$, then the constant a will occur in $\theta(x)$ for every unifier θ of WE which corresponds to a successful node which may be found below the actual node of the search tree. After any transformation step, the values (sequences) of all variables have to be updated.

We may now run the algorithm with the partial order $\leq$ used as a "filter". This means that we stop a branch of the search tree as soon as the constants which occur in the (columns which represent the) actual values $V(x)$ of the variables x of WE show that the relation $\leq_V$ contains a cycle, where

$$y \leq_V x \text{ iff } y \leq x \text{ or } y \text{ occurs in } V(x).$$

A detailled description of this algorithm for word equations with partially specified linear constant restrictions would be rather long.[4] But most constructions are straightforward, starting from a modern description of Makanin's algorithm (e.g., [Sh91]).

5 AI-Unification with Linear Constant Restriction

Let h be a binary function symbol, and let AI denote the theory of idempotent semigroups, i.e.,

$$AI := \{h(h(x,y),z) = h(x,h(y,z)), h(x,x) = x\}.$$

As mentioned in the introduction, decidability of AI-unification problems with constants is an easy consequence of the fact that idempotent semigroups are locally finite. In fact, McLean has shown as early as 1954 (see [Mc54]) that finitely generated free idempotent semigroups are finite, and he gave a formula which can be used to calculate the cardinality c_n of the free idempotent semigroup in n generators:

$$c_n = \sum_{r=1}^{n} \left(\binom{n}{r} \prod_{i=1}^{r} (r-i+1)^{2^i} \right).$$

The method used to show this finiteness result can also be employed to get a decision procedure for the word problem for free idempotent semigroups (see e.g. [BC71]).

Putting these facts together one can conclude that, for a given finite set C of constant symbols (the generators), one can compute a finite set of representatives for the $=_{AI}$ classes of all terms built over h and these constants. In addition, for two such representatives s, t one can effectively find the representative of the class of $h(s,t)$. Since the theory AI is regular, all the elements of an AI-class contain the same constants as its representative.

Theorem 6. *AI-unification with linear constant restriction is decidable.*

[4] In order to prove termination of this algorithm, a straightforward generalization of Bulitko's theorem has to be proved.

Proof. Assume that Γ is an AI-unification problem with constants, and that $<$ is a linear ordering on the constants and variables occurring in Γ. If Γ does not contain any constant, then there are no constant restrictions and we have an elementary AI-unification problem. Thus we may assume that Γ contains at least one constant. If we ignore the constant restriction induced by $<$, then it is easy to see that Γ has an AI-unifier iff it has an AI-unifier such that the images of variables occurring in Γ contain only constants from Γ. In fact, one can substitute all other constants and variables occurring in the image of a unifier by constants from Γ, and thus ends up with a unifier satisfying the above condition. This is no longer possible if one has to fulfill a constant restriction. However, if we allow for one new constant (i.e., a constant not occurring in Γ) in the image, then we may substitute all constants beside the ones from Γ and all variables occurring in the image of a solution of the problem with constant restriction by this new constant, and end up with a unifier satisfying the constant restriction.

This shows that it is enough to look for a solution in the free idempotent semigroup in finitely many generators, namely the constants occurring in Γ plus one additional new constant. As mentioned above, this semigroup is effectively given by a finite set of representatives. Thus what we can do is try all possibilities of replacing the variables in Γ by these representatives, under the restriction imposed by the linear ordering. Depending on whether we find a solution this way, the problem with linear constant restriction is solvable or not. □

As mentioned in section 3, the number of nondeterministic choices in the combination procedure can be reduced if it is possible to solve problems with partially specified linear constant restrictions. Recall that such a problem is given by a unification problem with constants, Γ, and a partial ordering $\leq$ on the variables and constants occurring in Γ. This problem has a solution iff there exists a linear extension of $\leq$ such that the unification problem with linear constant restriction induced by this extension has a solution.

It is easy to modify the procedure described in the proof of Theorem 6 to cope with such problems. In fact, when choosing a representative s to be substituted for a given variable x, this tells us that x has to be made larger than all constants from Γ occurring in s. If these additional relationships are consistent with the current partial ordering, we augment the partial ordering with them. Otherwise, we can discard this choice.

Corollary 7. *AI-Unification with partially specified linear constant restriction is decidable in "non-disjunctive" manner.*

It should however be noted that the decision methods we have just described are only practicable if the number of constants occurring in Γ is very small. This is so because the cardinalities c_n of finitely generated free idempotent semigroups are growing very fast.

6 NP-Decidability of general AC- and ACI-unification

In this section we shall prove NP-decidability of general AC- and ACI-unification problems. The result is an immediate consequence of the following two claims:

Claim 1: *For a given input problem Σ of size n_0,*[5] *the (non-optimized) combination algorithm may be organized in such a way that*

- *(1) the size n_{i+1} of the system obtained after step $i+1$ is polynomial in the size n_i of the system created at the previous step ($i = 0, \ldots, 4$).*
- *(2) the non-deterministic choices of steps 3 and 4 may be computed by means of a polynomial number of non-deterministic choices between two possibilities,*
- *(3) the complexity of the deterministic computations in step $i+1$ is polynomial in n_i ($i = 0, \ldots, 4$).*

Claim 2: *Solvability of AC and ACI-problems with linear constant restrictions may be decided by NP-algorithms.*

Proof of claim 1:

Ad (1). In step 1, note that one replacement of an alien subterms introduces two occurrences of a new variable, but does not introduce any new subterm. The number of such steps does not exceed the number of (occurrences of) subterms in the input symbol, thus it does not exceed n_0. Thus $n_1 \leq 3n_0$. Obviously $n_2 \leq 2n_1$. In the remaining steps, the size of the system(s) does not grow.

Ad (2). In step 3, the number of variables is $n \leq n_2$. Suppose that we have variables $\{x_1, ..., x_n\}$. In order to compute a partition of this set we shall successively create a partition P_i of $\{x_1, ..., x_i\}$ for $i = 1, ..., n$. Of course $P_1 := \{\{x_1\}\}$. Suppose that some $P_i = \{p_{i,1}, ..., p_{i,k}\}$ has been computed for an $i < n$. Obviously $k \leq i$. To compute P_{i+1} we may either add x_{i+1} to $p_{i,j}$ and stop, or we may continue, considering $p_{i,j+1}$, (starting with $j = 1$). If we reach the point where $p_{i,k}$ is considered, we may either add x_{i+1} to $p_{i,k}$ or we create a new equivalence class $p_{i+1,k+1} := \{x_{i+1}\}$. Obviously the final partition P_n is reached after not more than n_2^2 nondeterministic choices between two alternatives. It is also clear that every partition of $\{x_1, ..., x_n\}$ is a possible outcome of the procedure.

Suppose that the variables $\{x_1, ..., x_m\}$ are left in the input system for step 4. First we choose nondeterministically a variable which is subtracted from the

[5] The size of a unification problem is the number n of occurrences of symbols in the terms of the equations occurring in the problem.

set. This takes at most m non-deterministic decisions between two possibilities. The variable which is chosen is minimal with respect to the linear order to be constructed. The next variable is chosen in the remaining system. By iteration, the linear order is constructed in less than m^2 non-deterministic choices between two possibilities. Indexing of variables takes other m non-deterministic steps with two possibilities.

Ad (3). This is easy for step 1 since every term is only treated once, replacing all alien subterms. The remaining steps are obvious.

Proof of claim 2:

AC-Unification with Linear Constant Restriction. It is a well-known fact that solving AC-unification problems with constants can be reduced to solving systems of linear equations over the nonnegative integers (see e.g., [St81, Fa84]). As an easy consequence one can show that solvability of AC-unification problems with linear constant restriction can be expressed as an integer programming problem, thus establishing NP-decidability. Instead of giving a formal presentation of this reduction, we shall illustrate it by an example.

Let h be a binary AC-symbol, x, y be variables, and c, d be constants. We consider the AC-unification problem with constants

$$\begin{aligned}\Gamma = \{&h(x, h(x, h(c, h(c, c)))) \doteq h(y, h(y, h(y, h(y, d)))),\\ &h(x, h(x, h(x, h(y, y)))) \doteq h(x, c)\},\end{aligned}$$

and the constant restriction induced by $c < x < d < y$. As mentioned before, it is enough to look for solutions which introduce the constants c, d occurring in Γ and one additional constant, say e. For each of these three constants, we introduce a system of linear equations. The variables occurring in these equations stand for the number of occurrences of the respective constant in the image of x and y, respectively, of possible solutions of Γ. The coefficients of these variables in the equations are the number of occurrences of x and y, respectively, in Γ. Thus we get the three systems

$$\begin{pmatrix}2&0\\3&2\end{pmatrix}\begin{pmatrix}x_c\\y_c\end{pmatrix}+\begin{pmatrix}3\\0\end{pmatrix}=\begin{pmatrix}0&4\\1&0\end{pmatrix}\begin{pmatrix}x_c\\y_c\end{pmatrix}+\begin{pmatrix}0\\1\end{pmatrix}$$

$$\begin{pmatrix}2&0\\3&2\end{pmatrix}\begin{pmatrix}x_d\\y_d\end{pmatrix}=\begin{pmatrix}0&4\\1&0\end{pmatrix}\begin{pmatrix}x_d\\y_d\end{pmatrix}+\begin{pmatrix}1\\0\end{pmatrix}$$

$$\begin{pmatrix}2&0\\3&2\end{pmatrix}\begin{pmatrix}x_e\\y_e\end{pmatrix}=\begin{pmatrix}0&4\\1&0\end{pmatrix}\begin{pmatrix}x_e\\y_e\end{pmatrix}$$

In addition, since we do not have a unit element for h, the variables x, y have to be substituted by nonempty terms. This is expressed by the inequalities

$$x_c + x_d + x_e > 0 \quad \text{and} \quad y_c + y_d + y_e > 0.$$

The AC-unification problem with constants, Γ, has a solution iff the above systems of equations have nonegative integer solutions satisfying the restriction imposed by these inequalities.

Now it should be obvious how to express the constant restriction with the help of some additional equations: If a constant must not occur in the image of a variable, the corresponding variable in the system of linear equations has to be zero. In our example, we get the additional equation

$$x_d = 0$$

because $x < d$ means that d must not occur in the image of x.

ACI-Unification with Linear Constant Restriction. Kapur and Narendran [KN91] have shown that solvability of ACI-unification problems with constants can be decided by a (deterministic) polynomial algorithm. This is done by transforming the conditions for solvability of such a unification problem into a set of propositional Horn clauses. This set of Horn clauses is satisfiable iff the original problem was unifiable. Since the size of this set of clauses is quadratic in the size of the unification problem, the fact that satisfiability of propositional Horn clauses is decidable in linear time (see [DG84]) shows that one ends up with a quadratic decision procedure for ACI-unifiability with constants.

Now we shall demonstrate that this result can easily be generalized to solvability of ACI-unification with constant restriction. To this purpose we briefly review Kapur and Narendran's transformation. Let Γ be an ACI-unification problem with constants, and let X and C be the variables and constants, respectively, occurring in Γ. Let C' be C augmented by one additional constant c_0. As explained in the proof of Theorem 6, it is enough to look for solutions introducing only these constants. For each pair $(x, c) \in X \times C'$, we take a propositional variable $P_{x,c}$. The intended semantics of this variable is that it is true iff c does not occur in the image of x for the substitution under consideration.

Now consider an equation $s \doteq t$. For all constants a occurring in s but not in t, one introduces a Horn clause

$$\bigwedge_{x \in V(t)} P_{x,a} \Longrightarrow \text{false},$$

where $V(t)$ denotes the variables occurring in t. The obvious meaning of this clause is that, in order to get a solution, a must be introduced by some variable of t, since it already occurs in s. The analogous formulae are built for the constants occurring in t but not in s. For the additional constant c_0, we have to add formulae saying that if this constant is not introduced on one side of the equation, it must not be introduced on the other side. Thus we have for all variables y occurring in s the formula

$$\bigwedge_{x \in V(t)} P_{x,c_0} \Longrightarrow P_{y,c_0}.$$

Of course, one also must take the analogous formulae where the role of s and t are exchanged. Finally, the fact that all variables $x \in X$ must be replaced by a nonempty term is expressed by the formulae

$$\bigwedge_{c \in C'} P_{x,c} \Longrightarrow \text{false}.$$

If we take these formulae for all the equations in Γ then we have obtained a set of Horn clauses which is satisfiable iff Γ has a unifier.

Obviously, this encoding is very convenient for expressing constant restrictions. The fact that c must not occur in the image of x can simply be expressed by the fact

$$\text{true} \Longrightarrow P_{x,c}.$$

To sum up, we have thus shown that solvability of ACI-unification problems with linear constant restriction can be decided by a (deterministic) quadratic algorithm.

Conclusion

We have studied general unification problems for the equational theories A, AI, AC and ACI. With the combination algorithm which was introduced in [BS91] it is possible to reduce general unification problems to Robinson unification plus unification problems with linear constant restrictions in the single theories. The latter class of problems was shown to be decidable for the four theories. From the proof for the theories AC and ACI and from the structure of the combination algorithm we obtained the result that solvability of general AC- and ACI-unification problems may be decided with NP-algorithms. For the theories A and AI we showed how to use the information obtained from the solution of the free subsystem which is separated by the combination algorithm in order to optimize this procedure, eliminating possible choices in the non-deterministic steps. The notion of a unification problem with partially specified linear constant restriction arose from our optimization technique and we demonstrated that this class of problems is decidable in non-disjunctive manner for the theories A and AI.

References

[BB87] K.H. Bläsius, H.-J. Bürckert, "Deduktionssysteme," Oldenbourg Verlag, München Wien (1987).

[BS91] F. Baader, K.U. Schulz, "Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures," DFKI-Research Report RR-91-33, to appear in the *Proceedings of the 11th International Conference on Automated Deduction*, *LNCS* (1992).

[BC71] J.A. Brzozowski, K. Culik, A. Gabrielian, "Classification of Noncounting Events," *J. Computer and System Science* **5**, 1971.

[Bü90] H.-J. Bürckert, "A Resolution Principle for Clauses with Constraints," *Proceedings of the 10th International Conference on Automated Deduction, LNCS* **449**, 1990.

[Co90] A. Colmerauer, "An Introduction to PROLOG III," *C. ACM* **33**, 1990.

[DG84] W.F. Dowling, J. Gallier, "Linear Time Algorithms for Testing Satisfiability of Propositional Horn Formula," *J. Logic Programming* **3**, 1984.

[Fa84] F. Fages, "Associative-Commutative Unification," *Proceedings of the 7th International Conference on Automated Deduction, LNCS* **170**, 1984.

[JK90] J.P. Jouannaud, C. Kirchner, "Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification," Preprint, 1990. To appear in the Festschrift to Alan Robinson's birthday.

[JL87] J. Jaffar, J.L. Lassez, "Constraint Logic Programming," *Proceedings of 14th POPL Conference*, Munich, 1987.

[KN91] D. Kapur, P. Narendran, "Complexity of Unification Problems with Associative-Commutative Operators," Preprint, 1991. To appear in *J. Automated Reasoning*.

[KK89] C. Kirchner, H. Kirchner, "Constrained Equational Reasoning," *Proceedings of SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, ACM Press, 1989.

[Ma77] G.S. Makanin, "The Problem of Solvability of Equations in a Free Semigroup," *Mat. USSR Sbornik* **32**, 1977.

[Mc54] D. McLean, "Idempotent Semigroups," *Am. Math. Mon.* **61**, 1954.

[Sc89] M. Schmidt-Schauß, "Combination of Unification Algorithms," *J. Symbolic Computation* **8**, 1989.

[Sh90] K.U. Schulz, "Makanin's Algorithm – Two Improvements and a Generalization," *Proceedings of the First International Workshop on Word Equations and Related Topics IWWERT '90*, Tübingen 1990, Springer LNCS 572.

[Sh91] K.U. Schulz, "Word Unification and Transformation of Generalized Equations," CIS-Report 91-46, University of Munich, 1991 (see also this issue).

[St81] M. Stickel, "A Unification Algorithm for Associative-Commutative Functions," *J. ACM* **28**, 1981.

This article was processed using the LaTeX macro package with LLNCS style