Unification Theory – An Introduction[*]

Franz Baader
Lehr- und Forschungsgebiet Theoretische Informatik, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
e-mail: baader@informatik.rwth-aachen.de

Klaus U. Schulz
CIS, Universität München
Oettingenstraße 67, 80538 München, Germany
e-mail: schulz@cis.uni-muenchen.de

1 Preface

Equational unification is a generalization of syntactic unification in which semantic properties of function symbols are taken into account. For example, assume that the function symbol "+" is known to be commutative. Given the unification problem $x + y \doteq a + b$ (where x and y are variables, and a and b are constants), an algorithm for syntactic unification would return the substitution $\{x \mapsto a, y \mapsto b\}$ as the only (and most general) unifier: to make $x + y$ and $a + b$ syntactically equal, one must replace the variable x by a and y by b. However, commutativity of "+" implies that $\{x \mapsto b, y \mapsto b\}$ also is a unifier in the sense that the terms obtained by its application, namely $b + a$ and $a + b$, are equal modulo commutativity of "+". More generally, equational unification is concerned with the problem of how to make terms equal modulo a given equational theory, which specifies semantic properties of the function symbols that occur in the terms to be unified.

In this chapter, we first motivate equational unification by its applications in theorem proving and term rewriting. In addition to applications that require the computation of unifiers, we will also mention constraint-based approaches, in which only solvability of unification problems (i.e., the existence of unifiers) must be tested. Then we extend the definitions known from syntactic unification (such as most general unifier) to the case of equational unification. It turns out that, for equational unification, one must be more careful when introducing these notions. In the third section, we will mention some unification results for specific equational theories. In the fourth, and central, section of this chapter,

[*] This work is a preliminary version of the chapter on unification theory in a volume on automated deduction produced by the participants of the nationwide German research programme on automated deduction (SSP "Deduktion"). It was supported by a DFG grant (SSP "Deduktion") and by the EC Working Group CCL II.

we treat the important problem of how to combine unification algorithms. This problem occurs, for example, if we have a unification algorithm that can treat the commutative symbol "+" and another algorithm that can treat the associative symbol "×", and we want to unify terms that contain both symbols. In Section 6, we will reconsider unification from a more logical and algebraic point of view. This will yield an interesting logical characterization of the theories to which the combination approach introduced in the previous section applies. Finally, we conclude with a short section in which other interesting topics in the field of equational unification are mentioned, which could not be treated in more detail in this chapter.

2 Motivation

It is a well-known phenomenon that general purpose methods that can treat a wide range of problems are usually less efficient on a specific problem than special purpose methods designed for solving this very problem. The integration of such special purpose methods into general purpose approaches thus combines the advantages of the general purpose method (such as flexibility and wide applicability) with the advantages of the special purpose method (such as efficiency). An instance of this phenomenon can be observed in automated deduction, where general purpose theorem provers are known to go astray when faced with equational axioms that specify simple "semantic" properties of function symbols, such as associativity or commutativity. Already in 1967 J.A. Robinson [53] proposed that substantial progress ("a new plateau") could be achieved by removing these troublesome axioms from the data base, and building them directly into the inference rules of the deductive machinery. One way of achieving this goal was proposed by G. Plotkin in his pioneering paper [50]. He showed that (under certain assumptions) equational axioms can be removed from the set of input clauses of a resolution-based theorem prover if the usual syntactic (Robinson-style) unification is replaced by equational unification. Plotkin's approach of building-in equational theories is thus only applicable if there exist special purpose unification algorithms for equational theories that axiomatize common semantic properties of function symbols. In the area of equational unification, such theories have been thoroughly studied, and this has lead to an impressive collection of results on and efficient algorithms for unification in equational theories (see Section 4 for some examples).

The usefulness of equational unification has independently been discovered in the area of term rewriting, where certain equational axioms like commutativity cause problems because they cannot be oriented into terminating rewrite rules. Rewriting modulo equational theories [49, 36] takes such problematic axioms completely out of the rewriting process: in principle, these axioms generate a congruence relation on terms, and rewriting is performed modulo this congruence. Consequently, in the computation of critical pairs during Knuth-Bendix completion, syntactic unification must be replaced by unification modulo this congruence, that is, equational unification.

The traditional applications of syntactic unification in term rewriting and resolution-based theorem proving depend on an algorithm that computes a most general (syntactic) unifier of the terms to be unified. In Plotkin's approach and in Knuth-Bendix completion modulo equational theories, the rôle of the most general unifier is taken on by a complete sets of unifiers, that is, a set of unifiers that "represents" all solutions of the unification problem (see Section 3 for a formal definition). Unfortunately, such complete sets may become very large or even infinite. In the context of term rewriting, this may lead to a huge or even infinite number of critical pairs and possibly new rules, and in the theorem proving application it may lead to a combinatorial explosion of the search space or even an infinitely branching search space. It should be noted, however, that this may still be more efficient than "blindly" searching for all solutions of the unification problem with a general purpose theorem prover, and more promising than terminating the Knuth-Bendix completion procedure unsuccessfully because a non-orientable equational axiom was encountered.

Constraint-based approaches to automated deduction and term rewriting [15, 47, 41] require decision procedure for equational unification, that is, algorithms that decide whether a given equational unification problem is solvable or not. To be more precise, these approaches avoid the problems caused by large complete sets of unifiers since equational unification is no longer used as a mechanism for generating instantiations of variables (by applying the computed unifiers), but rather as a filter that prohibits instantiations that do not satisfy the (unification) constraints. In each step of the deductive process, constraints are added that describe under which conditions the step is possible. No additional branching of the search space is introduced this way. At suitable points in the deductive process, the accumulated constraints are tested for solvability, to find out whether there are still admissible instances. For unification constraints, this test can be realized with the help of a decision procedure for equational unification.

3 Basic notions

An *equational theory* over the signature Σ is defined by a set E of *identities* between Σ-terms, i.e., a subset of $T(\Sigma, V) \times T(\Sigma, V)$, where V is a countably infinite set of variables. From a logical point of view, such an identity is a universally quantified equational axiom. With $=_E$ we denote the equational theory defined by E, that is, the least congruence relation on $\mathcal{T}(\Sigma, V)$ that is closed under substitution and contains E. $\mathcal{T}(\Sigma, V)/{=_E}$ denotes the quotient term algebra modulo $=_E$, which is the free algebra with generators V in the class of all models of E. In the following, we will usually forget about the formal distinction between an equational theory $=_E$ and its defining set of identities E, i.e., we will also call E an equational theory. An equational theory E is *non-trivial* iff $x \neq_E y$ for distinct variables $x, y \in V$. It is easy to see that this is equivalent to saying that E has a model of cardinality greater than 1.

Example 3.1 Let $\Sigma = \{f\}$ for the binary function symbol f. The theory $A_f := \{f(f(x,y),z) = f(x,f(y,z))\}$ expresses associativity of f, $C_f := \{f(x,y) = f(y,x)\}$ expresses commutativity of f, and $AC_f := A_f \cup C_f$ states that f must be associative and commutative. From a logical point of view, A_f corresponds to the formula $\forall x.\forall y.\forall z.f(f(x,y),z) = f(x,f(y,z))$. The class of all models of A_f is the class of all semigroups. For a set of variables V, the quotient term algebra $\mathcal{T}(\Sigma,V)/{=_{A_f}}$ is isomorphic to V^+, the free semigroup with generators V, which consists of all words over the alphabet V and interprets the function symbol f as concatenation of words.

If the function symbol f is clear from the context or irrelevant, then these theories will be denoted A, C, and AC.

In equational unification, as in the case of the usual (Robinson-style) syntactic unification, to "unify" terms means to make them equal by applying a suitable substitution. Instead of using syntactic equality, equational unification employs equality modulo a given equational theory E. Syntactic unification can be obtained as a special case if one takes the empty set of identities, which has syntactic equality of terms as its equational theory, i.e., $s =_\emptyset t$ iff $s = t$.

Definition 3.2 *Let Σ be a signature and E be an equational theory over Σ. An* E-unification problem over Σ *is a finite set of equations between Σ-terms, that is, a set of the form $\Gamma := \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ where $s_1, \ldots, t_n \in T(\Sigma, V)$. The Σ-substitution σ is an* E-unifier *(or* solution*) of Γ iff $s_i\sigma =_E t_i\sigma$ holds for all $i, 1 \leq i \leq n$. The set of all E-unifiers of Γ is denoted by $U_E^\Sigma(\Gamma)$.*

For example, let $\Sigma = \{a, b, f\}$, where a, b are nullary function symbols (i.e., constant symbols) and f is a binary function symbol. The C_f-unification problem $\Gamma := \{f(x,y) \doteq f(a,b)\}$ has two C_f-unifiers:

$$\{x \mapsto a,\ y \mapsto b\} \text{ and } \{x \mapsto b,\ y \mapsto a\}.$$

If we consider Γ as an $\emptyset$-unification problem, then only the first of these two substitutions is an $\emptyset$-unifier (i.e., syntactic unifier).

For a given E-unification problem over Σ, the signature Σ determines which function symbols may occur in the terms to be unified and in the unifiers. The above example shows that Σ may be larger than $\mathit{sig}(E)$, the set of all function symbols that occur in an identity of E: considered as a C_f-unification problem, Γ contains the additional constant symbols a, b, and considered as an $\emptyset$-unification problem, Γ also contains an additional binary function symbol. These additional symbols are called *free* symbols since their interpretation is not constrained by the equational theory. The next definition classifies unification problems according to which symbols may be contained in $\Sigma \setminus \mathit{sig}(E)$.

Definition 3.3 *Let Σ be a signature, and let E be an equational theory and Γ an E-unification problem over Σ.*

1. *Γ is called an* elementary *E-unification problem iff $\Sigma \setminus sig(E) = \emptyset$.*

2. *Γ is called an E-unification problem* with constants *iff $\Sigma \setminus sig(E)$ contains only constant symbols.*

3. *Γ is always a* general *E-unification problem, i.e., for general E-unification problems, $\Sigma \setminus sig(E)$ may contain function symbols of arbitrary arity.*

For syntactic unification, we have $E = \emptyset$, and thus there are no elementary unification problems, and unification problems with constants are rather trivial. Thus, only general unification problems are of interest.[1] For other equational theories E, all three types of unification may be of interest, and they may lead to different results:

- There exists an equational theory for which elementary unification is decidable, but unification with constants is undecidable (see [14]).
- From the development of the first algorithm for AC-unification with constants [61, 45], it took almost a decade until the termination of an algorithm for general AC-unification was shown [24, 25].

Thus, it is usually important to specify, for which type of unification a particular result holds, and to find out which kind of unification problems occur in an application. In the applications in term rewriting and theorem proving mentioned above, one usually obtains general E-unification problems. The next example demonstrates this for the case of building-in equational theories into resolution-based theorem provers.

Example 3.4 When talking about A_f-unification, one may first think of unifying modulo A_f terms built by using just the symbol f and variables, or equivalently, of unifying words over the alphabet V of all variables. However, suppose that a resolution-based theorem prover—which has built-in the theory A_f—receives the formula

$$\exists x. \, (\forall y. f(x,y) = y \ \wedge \ \forall y. \exists z. f(z,y) = x)$$

as an axiom. In a first step, this formula must be Skolemized, i.e., the existential quantifiers have to be replaced by new function symbols. In our example, we need a nullary symbol e and a unary symbol i in the Skolemized form

$$\forall y. f(e,y) = y \ \wedge \ \forall y. f(i(y),y) = e$$

of the axiom. Hence, even if we start with formulae containing only terms built over f, the theorem prover eventually has to handle terms containing additional free symbols.

[1] Sometimes it is more convenient to represent a syntactic unification problem Γ as an elementary unification problem with respect to the "free theory" that is defined by the "dummy" identities $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$ for all function symbols f occurring in Γ.

For many interesting equational theories (such as the theory AC), designing an algorithm for unification with constants is a lot easier than directly coming up with an algorithm for the general case. Hence, the question arises whether there is a general method that allows to "lift" an algorithm for E-unification with constants to an algorithm for general E-unification. It turns out that this is a special case of the more general problem of how to combine unification algorithms for equational theories over disjoint signature, to which we devote the central section of this chapter (see Section 5).

Complete sets of E-unifiers

For a given E-unification problem Γ over Σ, the set of *all* E-unifiers is usually infinite. For syntactic unification, the most general unifier σ of Γ yields a compact representation of $U_{\emptyset}^{\Sigma}(\Gamma)$ since all unifiers can be obtained as instances of σ. For arbitrary equational theories, a single unifier is often not sufficient to represent the set of all unifiers: the rôle of the most general unifier is taken on by the complete set of unifiers. Before we can define this notion, we must make precise what kind of instantiation relation is used in the case of E-unification. The instantiation preorder $\leq_E^W$, which is parameterized by a finite set of variables W, is defined as follows:

$$\sigma \leq_E^W \theta \;\text{ iff }\; \begin{array}{l}\text{there exists a substitution } \lambda \text{ such that } x\theta =_E x\sigma\lambda \\ \text{for all variables } x \in W.\end{array}$$

If $\sigma \leq_E^W \theta$ then θ is called an *E-instance* of σ on W, and σ is said to be *more general* than θ on W. Obviously, if σ is an E-unifier of Γ, and θ is an E-instance of σ on the set of variables occurring in Γ, then θ is also an E-unifier of Γ.

Definition 3.5 *Let Γ be an E-unification problem over Σ. A* complete set of E-unifiers of Γ *is a set $cU_E^{\Sigma}(\Gamma)$ that satisfies the following conditions:*

1. *each $\sigma \in cU_E^{\Sigma}(\Gamma)$ is an E-unifier of Γ, and*
2. *for all $\theta \in U_E^{\Sigma}(\Gamma)$ there exists $\sigma \in cU_E^{\Sigma}(\Gamma)$ such that $\sigma \leq_E^W \theta$, where W is the set of variables occurring in Γ.*

For reasons of efficiency, such a complete set should be as small as possible. For example, in term rewriting modulo equational theories each element of the complete set leads to a critical pair, which must be tested for confluence and may lead to a new rule, and in resolution-based theorem proving each element of the complete set opens a new branch of the search tree. Thus, one is interested in *minimal complete sets $\mu U_E^{\Sigma}(\Gamma)$ of E-unifiers of* Γ, that is, complete sets satisfying the additional condition

3. *For all $\sigma, \theta \in \mu U_E(\Gamma)$, $\sigma \leq_E^W \theta$ implies $\sigma = \theta$.*

If $\mu U_E^\Sigma(\Gamma) = \{\sigma\}$ has cardinality 1, then the element σ of this set is called a *most general E-unifier of* Γ.

Example 3.6 Consider the A_f-unification problem with constants

$$\Gamma := \{f(x,a) \doteq f(a,x)\}.$$

For $n \geq 1$, let σ_n be a substitution that maps x to the term $f(a, \ldots f(a,a) \ldots)$ having n occurrences of the free constant a. Each substitution σ_n solves Γ modulo A_f, and each A_f-unifier of Γ is equal modulo A_f to a substitution σ_n for some $n \geq 1$. It follows that $\{\sigma_n \mid n = 1, 2, \ldots\}$ is a complete set of A_f-unifiers of Γ. Since the terms $x\sigma_n$ are ground terms, and since $x\sigma_n \neq_{A_f} x\sigma_m$ for $n \neq m$, it is easy to see that the set $\{\sigma_n \mid n = 1, 2, \ldots\}$ is minimal as well.

A minimal complete set of E-unifiers may not always exist, but if it does then it is unique up to the equivalence relation $\equiv_E^W$ induced by the preorder $\leq_E^W$ (see [26]):

$$\sigma \equiv_E^W \theta \quad \text{iff} \quad \sigma \leq_E^W \theta \ \wedge \ \theta \leq_E^W \sigma$$

For this reason, the *unification type* of an equational theory E can be defined with respect to the cardinality and existence of minimal complete sets. More precisely, an equational theory has three different kinds of unification types, depending on which type of unification problems are considered. The definition below is formulated for elementary unification. The unification types for unification with constants (general unification) are obtained by replacing "elementary E-unification problem" by "E-unification problem with constants" ("general E-unification problem").

Elementary unification type 1 *(unitary):*
A minimal complete set $\mu U_E(\Gamma)$ exists for all elementary E-unification problems Γ, and it always has cardinality ≤ 1.

Elementary unification type ω *(finitary):*
A minimal complete set $\mu U_E(\Gamma)$ exists for all elementary E-unification problems Γ, and it is always finite.

Elementary unification type ∞ *(infinitary):*
A minimal complete set $\mu U_E(\Gamma)$ exists for all elementary E-unification problems Γ, and there exists at least one elementary E-unification problem for which this set is infinite.

Elementary unification type 0 *(zero):*
There exists at least one elementary E-unification problem Γ that does not have a minimal complete set of E-unifiers.

Examples 3.7 We give an example for each unification type:

- The empty theory $\emptyset$ is unitary (for general unification) since every solvable unification problem has a most general syntactic unifier, which yields a minimal complete set of cardinality 1 (see [52]). Unsolvable unification problems always have the empty set as minimal complete set of unifiers.

- Commutativity C_f is finitary for all three kinds of unification (see, e.g., [59]). The problem $\{f(x,y) \doteq f(a,b)\}$ is an example of a C_f-unification problem with constants that has a minimal complete set of C_f-unifiers of cardinality > 1.

- Plotkin [50] has shown that a general A_f-unification problem always has a minimal complete set of A_f-unifiers, and he described an algorithm that enumerates such a set (see Section 4 below). Example 3.6 shows that such sets may already be infinite for A_f-unification with constants, and one can show that even the elementary A_f unification problem $\{f(x,y) \doteq f(y,x)\}$ has an infinite minimal complete set of A_f-unifiers.[2]

- The theory $AI_f := A_f \cup \{f(x,x) = x\}$ of idempotent semigroups is of type zero. This was shown in [54] for unification with constants, and in [1] for elementary unification: the elementary AI_f-unification problem $\{f(x,f(y,x)) \doteq f(x,f(z,x))\}$ does not have a minimal complete set of AI_f-unifiers.

For these examples, the unification type does not depended on which kind of unification (elementary, general, or with constants) is considered. In general, however, they may lead to different unification types. For example, there exist theories that are unitary with respect to elementary unification, but only finitary with respect to unification with constants. One such example is the theory of Abelian monoids, i.e., $AC1 := AC_f \cup \{f(1,x) = x\}$, where 1 is a constant symbol (see, e.g., [30]).

Readers that are more familiar with syntactic unification may have wondered why our definition of complete and minimal complete sets of unifiers differs from the usual definition of most general unifiers for syntactic unification in that the instantiation preorder $\leq_E^W$ is restricted to the set W of all variables occurring in the unification problem. If we changed the definition of minimal complete sets of unifiers such that substitutions are compared on all variables, then we would obtain different unification types. For example, the theory *AC1* would no longer be unitary for elementary unification [2]. Since the restricted instantiation preorder is sufficient for the above mentioned applications of E-unification, there is no reason for using an instantiation preorder that yields worse unification types.

Another difference to the usual presentation of syntactic unification is that our unification problems consist of sets of equations rather than of one single equation. For general E-unification, there is no difference between solving a system of equations or a single equation: the E-unification problem $\Gamma := \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ can be encoded in the general E-unification problem $\{f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n)\}$, where f is an n-ary free function symbol that does not occur in Γ. However, for elementary E-unification and E-unification with constants, there may be significant differences. For example, there exists an equational theory E such that all elementary E-unification problems of

[2] This can, for example, be seen when analyzing the behaviour of Plotkin's algorithm on this input.

cardinality 1 (i.e., single equations) have minimal complete sets of E-unifiers, but E is of elementary unification type 0 since there exists an elementary E-unification problems of cardinality 2 that does not have a minimal complete set of E-unifiers [16].

Unification algorithms

As mentioned in Section 2, there are two different types of applications of E-unification. The first, more "traditional" type of applications simply replaces the most general unifier by a complete set of unifiers. Usually, these approaches are only applied to finitary theories, i.e., theories that allow for finite complete sets of unifiers. In these applications, one needs an algorithm that is able to compute a finite complete set of E-unifiers for a given E-unification problem. We will call such an algorithm an *E-unification algorithm.* For reasons of efficiency, the algorithm should in fact compute a minimal complete set.

For non-finitary theories, one is sometimes interested in an enumeration of a complete set of unifiers. We call a procedure that is able to enumerate a (minimal) complete set of E-unifiers a *(minimal) E-unification procedure.* It should be noted that the set of all E-unifiers is a complete set, and that it is always possible to enumerate this set since equality modulo E is semi-decidable: simply dove-tail an enumeration of all substitution with a semi-decision procedure that tests whether a given substitution is a unifier. Thus, if one designs an E-unification procedure, one must take care that this procedure is more efficient and less redundant than the trivial E-unification procedure sketched above.

Constraint-based applications of E-unification need to know whether a given E-unification problem has a solution or not. An algorithm that is able to decide this problem is called a *decision procedure* for E-unification. In constraint-based applications, one usually has the situation that the constraint sets are increasing: after a set of constraints Γ has been tested for solvability, an inference step adds a new constraint, and then the augmented set of constraints Γ' must again be tested for solvability. Hence, it is desirable to have an *incremental decision procedure*, i.e., a procedure that is able to reuse (some of) the work done during testing Γ for solvability when faced with the larger set Γ'.

Note that an E-unification algorithm always yields a decision procedure since a given input problem has a solution iff the complete set that is returned by the algorithm is non-empty. An E-unification procedure (even a minimal one) need not yield a decision procedure since it may run forever even though there are no solutions.

4 Three example theories

In unification theory, the unification properties of a larger number of different equational theories have been investigated. Usually, one was interested in finding answers to (some of) the following questions:

- Is solvability of E-unification problems decidable? What is the complexity of this decision problem? How can one design practical algorithms for the decision problem? Can one find incremental decision procedures?

- What is the unification type of E? If E is finitary: is there an (efficient) E-unification algorithm? How large are the minimal complete sets? What is the complexity of computing these sets? For non-finitary E: is there a minimal E-unification procedure? Are there interesting special cases for which this procedure is guaranteed to terminate?

As mentioned above, the answers to these questions may differ, depending on which kind of unification problem (elementary, with constants, or general) is considered. In many cases, it turned out to be easier to design unification algorithms or decision procedure first for E-unification with constants, and then use general combination methods (see Section 5) to obtain an algorithm or decision procedure for general E-unification.

Most of the results for specific theories and references to the literature can be found in survey articles on unification [60, 35, 9]. A table of complexity results for unification is contained in [37]. Instead of recounting these results, we consider three equational theories as examples:

$$\begin{array}{rcl} A_f & := & \{f(f(x,y),z) = f(x,f(y,z))\}, \\ AC_f & := & A_f \cup \{f(x,y) = f(y,x)\}, \\ ACI_f & := & AC_f \cup \{f(x,x) = x\}. \end{array}$$

The goal is not to provide an extensive list of references (which can be found in the above mentioned overviews) or to give a detailed description of a unification algorithm or decision procedure, but rather to convey an intuition on how unification in these theories works. For the first theory, we sketch a minimal unification procedure, for the second a unification algorithm, and for the third a decision procedure.

Unification under associativity

The theory A_f axiomatizes associativity of the binary function symbol f. Since the corresponding free algebra is the free semigroup, whose elements are words, A_f-unification problems with constants are often called word equations. The following is a brief synopsis of the known results:

Unification type: infinitary for all three kinds of unification [50],

Unification procedure: there exists a minimal unification procedure for general A_f-unification [50, 43]. Using a decision procedure (see below) as a subprocedure it is possible to modify this procedure such that it always terminates on unification problems that have a finite minimal complete set of A_f-unifiers [34, 57].

Decision problem: A_f-unification is decidable both for unification with constants [46] and for general A_f-unification [4],

Complexity: the decision problem is known to be NP-hard [10], but the current best bound on the time complexity of Makanin's algorithm is nondeterministic triple-exponential [42].

After a long series of attacks and partial solutions, the decision problem was finally solved by G.S. Makanin [46]. His description of the decision procedure for A_f-unification with constants is rather long and complex (70 pages of very tersely written material), but this appears to be due to the inherent complexity of the problem. Subsequent descriptions of the algorithm [48, 34, 57] have simplified and clarified the presentation, but in essence they still coincide with Makanin's original version, from which they inherit the complexity. The decision problem for general A_f-unification was solved with the help of general combination techniques (see Section 5).

In the following, we briefly describe *Plotkin's minimal A_f-unification procedure.* To make the presentation simpler, we restrict ourselves to A_f-unification with constants, and consider only single equations (i.e., A_f-unification problems with constants of cardinality 1). Let V be a set of variables, and C be a set of (free) constants. Modulo A_f, terms built using the binary symbol f, variables from V, and constants from C can be seen as non-empty words over the alphabet $V \cup C$. In unification problems reached by applying Plotkin's procedure, we will sometimes also encounter the empty word ε.

The procedure builds a (possibly infinite) tree, which can be seen as a search tree for A_f-unifiers of the unification problem at the root of the tree. The tree is inductively defined as follows. Let $\{u_0 \doteq v_0\}$ be the original unification problem, where $u_0, v_0 \in (V \cup C)^+$.

Initialization: Create a tree that consists just of a root, which is labelled by the pair $(u_0 \doteq v_0 \mid id)$ where id stands for the identity substitution.

Leaf extension: Assume that the leaf node k under consideration has label $(u \doteq v \mid \sigma)$, where u, v are (possibly empty) words over $V \cup C$. We distinguish the following cases:

1. Assume that one of u, v is the empty word. If $u = \varepsilon = v$, then k is a success-node, and σ is a unifier. Otherwise, k is a failure-node.

2. Assume that $u = au'$ starts with the constant a and $v = bv'$ starts with the constant b. If $a = b$, then create one new leaf node that is a direct successor of k and is labelled by $(u' \doteq v' \mid \sigma)$. Otherwise, k is a failure-node.

3. Assume that $u = au'$ starts with the constant a and $v = yv'$ starts with the variable y. Let $\tau_1 := \{y \mapsto a\}$ and $\tau_2 := \{y \mapsto ay\}$. Create two new leaf nodes that are direct successors of k. Label one of them with $(u'\tau_1 \doteq v'\tau_1 \mid \sigma\tau_1)$ and the other with $(u'\tau_2 \doteq y(v'\tau_2) \mid \sigma\tau_2)$.

4. The case where u starts with a variable and v with a constant is treated symmetrically.

5. Assume that $u = xu'$ and $v = xv'$ start with the same variable x. Create a new leaf node that is a direct successor of k and is labelled with $(u' \doteq v' \mid \sigma)$.

6. Assume that $u = xu'$ starts with the variable x and $v = yv'$ starts with the variable $y \neq x$. Let $\tau_1 := \{y \mapsto xy\}$, $\tau_2 := \{y \mapsto x\}$, and $\tau_3 := \{x \mapsto yx\}$. Create three new leaf nodes that are direct successors of k. Label the first with $(u'\tau_1 \doteq y(v'\tau_1) \mid \sigma\tau_1)$, the second with $(u'\tau_2 \doteq v'\tau_2 \mid \sigma\tau_2)$, and the third with $(x(u'\tau_3) \doteq v'\tau_3 \mid \sigma\tau_3)$.

The substitutions that are labels of success-nodes in this tree form a minimal complete set of A_f-unifiers of the A_f-unification problem with constants $\{u_0 \doteq v_0\}$ [58, 57]. Since the tree may have infinite paths, it must be generated in a breadth-first manner.

To get an intuition of how the procedure works, let us consider the third case of the leaf extension step in more detail. If θ is a unifier of the word equation $au' \doteq yv'$, then $a(u'\theta) = (au')\theta = (yv')\theta = (y\theta)(v'\theta)$. Hence, $y\theta$ must start with the constant a. Now, either $y\theta = a$ or $y\theta$ starts with a and continues with a non-empty suffix. This corresponds to the two cases considered in the corresponding leaf extension step. In the second case, the variable y is replaced by ay, that is, y is reused and now stands for the non-empty suffix obtained by removing the leading a from $y\theta$. The equations in the label of the new leaves are obtained by applying τ_1 or τ_2 to $au' \doteq yv'$, and removing the two leading a's.

As an example, let us first consider the unification problem $\{ax \doteq xa\}$:

$$\begin{tikzcd}
ax \doteq xa \mid id \arrow[r] \arrow[d] & a \doteq a \mid \{x \mapsto a\} \arrow[r] & \varepsilon \doteq \varepsilon \mid \{x \mapsto a\} \\
ax \doteq xa \mid \{x \mapsto ax\} \arrow[r] \arrow[d] & a \doteq a \mid \{x \mapsto aa\} \arrow[r] & \varepsilon \doteq \varepsilon \mid \{x \mapsto aa\} \\
ax \doteq xa \mid \{x \mapsto aax\} \arrow[r] \arrow[d] & a \doteq a \mid \{x \mapsto aaa\} \arrow[r] & \varepsilon \doteq \varepsilon \mid \{x \mapsto aaa\} \\
\ldots & &
\end{tikzcd}$$

We see that the unification problem $\{ax \doteq xa\}$ reproduces itself, and that there are infinitely many success-nodes, which yield the unifiers $\{x \mapsto a^n\}$ for $n \geq 1$. The unification problem $\{ax \doteq xb\}$ produces infinitely many failure nodes and no success-node:

$$\begin{tikzcd}
ax \doteq xb \mid id \arrow[r] \arrow[d] & a \doteq b \mid \{x \mapsto a\} & \texttt{failure} \\
ax \doteq xb \mid \{x \mapsto ax\} \arrow[r] \arrow[d] & a \doteq b \mid \{x \mapsto aa\} & \texttt{failure} \\
ax \doteq xb \mid \{x \mapsto aax\} \arrow[r] \arrow[d] & a \doteq b \mid \{x \mapsto aaa\} & \texttt{failure} \\
\ldots & &
\end{tikzcd}$$

In both examples, the tree generated by Plotkin's procedure is infinite, but only finitely many different unification problems occur as labels of nodes in the tree.

It is easy to see that this phenomenon always occurs if one starts with a unification problem $\{u_0 \doteq v_0\}$ such that *every variable occurs at most twice* in the word u_0v_0. In fact, in this situation, the length of the words uv such that $u \doteq v$ occurs as a unification problem in the tree generated by Plotkin's procedure is bounded by the length of u_0v_0. Since the unification problems $u \doteq v$ occurring in the tree are built with the variables and free constants occurring in u_0v_0, there are only finitely many different unification problems of this kind. As a consequence, one obtains that *Plotkin's procedure yields a decision procedure* for A_f-unification problems in which every variable occurs at most twice. This is achieved by not extending a leaf k if it is labelled with a unification problem that occurs in the label of a node above k. In our example $\{ax \doteq xb\}$, the node with label $(ax \doteq xb \mid \{x \mapsto ax\})$ need not be extended, and one can immediately decide that the problem does not have a solution. In the example $\{ax \doteq xa\}$, it is easy to determine from a finite part of the tree how all unifiers look like. However, this is not always possible, even for the cases where Plotkin's procedure equipped with the above described cycle test yields a decision procedure (see [32] for an example of an A_f-unification problem where the set of unifiers cannot be represented in a "parameterized way").

If variables occur more than twice, then the tree may become infinite without any repetitions of unification problems. As an example, the interested reader may consider what happens if Plotkin's procedure is applied to the unification problem $\{axx \doteq xxb\}$.

Unification under associativity and commutativity

The theory AC_f, which axiomatizes associativity and commutativity of a binary function symbol, is the theory most frequently used in rewriting modulo equational theories and in theorem proving with built-in theories. It turns out that it is more convenient to investigate first unification in the theory $AC1_f := AC_f \cup \{f(x,1) = x\}$, which extends AC_f by an identity that says that the constant symbol 1 is a unit element for f. Let us start with a brief synopsis of the results known about these two theories:

Unification type: *AC1* is unitary for elementary unification, and finitary for unification with constants and for general unification, whereas AC is finitary for all three types of unification [61, 45, 24, 25]. The number of unifiers in a minimal complete set of AC_f-unifiers may be doubly exponential in the size of a given (elementary) AC_f-unification problem [22].

Unification algorithm: there exist AC_f and $AC1_f$-unification algorithms for all three types of unification [61, 45, 40, 27, 17, 30, 31, 44, 12].

Complexity of the decision problem: The decision problem for AC_f and $AC1_f$-unification with constants is NP-complete, and solvability of general AC_f and $AC1_f$-unification problems can also be decided by NP-algorithms [37].

In the following, we sketch a unification algorithm. To keep things simple, we will mainly restrict our attention to elementary AC_f and $AC1_f$-unification. We start with an algorithm that computes a most general $AC1_f$-unifier for every elementary $AC1_f$-unification problems. The algorithm for AC_f takes this most general unifier and derives a complete set of AC_f-unifiers from it.

Let $\Sigma := \{f, 1\}$ where f is binary and 1 is a constant symbol, and let V be a countably infinite set of variables. For a term $t \in T(\Sigma, V)$ and a variable $x \in V$, we denote with $|t|_x$ the number of occurrences of x in t. These numbers can be used to characterize equality modulo $AC1_f$:

$$s =_{AC1_f} t \text{ iff } |s|_x = |t|_x \text{ for all } x \in V.$$

It is easy to see that 1 is the only element of its $=_{AC1_f}$-equivalence class. Any term in $T(\Sigma, V) \setminus \{1\}$ is equivalent to a term in $T(\{f\}, V)$, and for terms $s, t \in T(\{f\}, V)$ we have $s =_{AC1_f} t$ iff $s =_{AC_f} t$.

An equation $s \doteq t$ between terms in $T(\Sigma, V)$ can be translated into a homogeneous *linear Diophantine equation.* Let $\{x_1, \ldots, x_n\}$ be the set of variables occurring in s or t. The equation $s \doteq t$ corresponds to the equation

$$(*) \quad a_1x_1 + \ldots + a_nx_n = b_1x_1 + \ldots + b_nx_n,$$

where $a_i := |s|_{x_i}$ and $b_i := |t|_{x_i}$ for $i = 1, \ldots, n$. If $(c_1, \ldots, c_n) \in \mathbb{N}^n$ is a solution of $(*)$, i.e., an n-tuple of nonnegative integers such that $a_1c_1 + \ldots + a_nc_n = b_1c_1 + \ldots + b_nc_n$, then the following is an $AC1_f$-unifier of $s \doteq t$:

$$\{x_1 \mapsto z^{c_1}, \ldots, x_n \mapsto z^{c_n}\},$$

where z is a variable and z^{c_i} abbreviates the term $f(z, f(z, \cdots f(z, z) \cdots))$ that has c_i occurrences of z.[3] Conversely, if σ is an $AC1_f$-unifier of $s \doteq t$ and $z \in V$ is a variable, then $(|x_1\sigma|_z, \ldots, |x_n\sigma|_z)$ is a solution of $(*)$ in $\mathbb{N}^n$. If one has more than one equation in the unification problem, then one obtains a system of linear Diophantine equations of the form $(*)$, which must be solved simultaneously.

Obviously, the tuple $(0, \ldots, 0)$ is a (trivial) solution of every homogeneous linear Diophantine equation of the form $(*)$, which corresponds to the trivial

[3] We have $z^0 = 1$ (for $c_i = 0$) and $z^1 = z$ (for $c_i = 1$).

$AC1_f$-unifier of $s \doteq t$, i.e., the substitution that replaces every variable occurring in s or t by 1. Thus, every elementary $AC1_f$-unification problem has a unifier.

The most general $AC1_f$-unifier of a given elementary $AC1_f$-unification problem can be obtained from the set of minimal non-trivial solutions of the corresponding system of linear Diophantine equations. A solution $(c_1, \ldots, c_n)$ of a system of homogeneous linear Diophantine equations is a *minimal non-trivial solution* iff it is not the trivial solution $(0, \ldots, 0)$ and every solution $(d_1, \ldots, d_n) \neq (c_1, \ldots, c_n)$ satisfying $d_1 \leq c_1, \ldots, d_n \leq c_n$ is trivial. One can show that the set of all minimal non-trivial solutions of a given system of homogeneous linear Diophantine equations is always finite. Algorithms for computing these solutions are, for example, described in [28, 33, 19, 12, 51, 21, 20]. If this set is empty, then the trivial unifier that replaces every variable occurring in the unification problem by 1 is the most general unifier. Otherwise, let $\{(c_{1,1}, \ldots, c_{1,n}), \ldots, (c_{k,1}, \ldots, c_{k,n})\}$ be the set of minimal non-trivial solutions of the system of linear Diophantine equations corresponding to the elementary $AC1_f$-unification problem Γ. Then the following is a *most general $AC1_f$-unifier* of Γ:

$$\{x_1 \mapsto z_1^{c_{1,1}} \cdots z_k^{c_{k,1}}, \ldots, x_n \mapsto z_1^{c_{1,n}} \cdots z_k^{c_{k,n}}\},$$

where $z_1^{c_{1,i}} \cdots z_k^{c_{k,i}}$ abbreviates the term $f(z_1^{c_{1,i}}, f(\cdots f(z_{k-1}^{c_{k-1,i}}, z_k^{c_{k,i}}) \cdots))$.

As an example, let us consider the $AC1_f$-unification problem

$$\Gamma := \{f(x_1, x_2) \doteq f(x_3, x_4)\}.$$

The corresponding linear Diophantine equation is $x_1 + x_2 = x_3 + x_4$, which has the minimal non-trivial solutions $\{(1,0,1,0), (1,0,0,1), (0,1,1,0), (0,1,0,1)\}$. Hence, the most general $AC1_f$-unifier[4] of Γ is

$$\sigma := \{x_1 \mapsto f(z_1, z_2),\ x_2 \mapsto f(z_3, z_4),\ x_3 \mapsto f(z_1, z_3),\ x_4 \mapsto f(z_2, z_4)\}.$$

Assume that Γ is an *AC_f-unification problem*, i.e., in contrast to an $AC1_f$-unification problem, the terms in the problem and the terms introduced by the unifiers cannot be the unit element 1. Γ may, however, also be seen as a $AC1_f$-unification problem, and thus we can compute the most general $AC1_f$-unifier of Γ. Obviously, every AC_f-unifier of Γ is also an $AC1_f$-unifier of Γ, and thus an instance of the most general $AC1_f$-unifier. However, the instantiation may depend on the presence of the unit element 1. In our example, $\tau := \{x_1 \mapsto z_1, x_2 \mapsto z_4, x_3 \mapsto z_1, x_4 \mapsto z_4\}$ is an AC_f-unifier of Γ, which is an $AC1_f$-instance of the most general $AC1_f$-unifier σ: $\tau = \sigma\lambda$ for the Σ-substitution $\lambda := \{z_2 \mapsto 1, z_3 \mapsto 1\}$. However, λ is not an admissible substitution for the smaller signature $\{f\}$. Consequently, τ is not an AC_f-instance of σ.

In order to obtain a *minimal complete set of AC_f-unifiers* of an AC_f-unification problem Γ, one must look at all possible ways of applying such

[4] In the representation of this unifier, we have used the fact that 1 is an unit element for f; for example, $x_1\sigma = f(z_1, z_2)$ instead of $f(z_1, f(z_2, f(1, 1)))$.

erasing substitutions λ to the most general $AC1_f$-unifier of Γ. To be more precise, assume that the most general $AC1_f$-unifier σ of Γ introduces the variables $z_1, \ldots, z_k$. For every subset Z of $\{z_1, \ldots, z_k\}$, we define $\lambda_Z := \{z \mapsto 1 \mid z \in Z\}$. Obviously, each substitution $\sigma\lambda_Z$ is an $AC1_f$-unifiers of Γ. This substitution can also be seen as an AC_f-unifier iff $x_i\sigma\lambda_Z \neq_{AC1_f} 1$ for all variables x_i occurring in Γ. In this case, we say that λ_Z is *admissible* for σ. It can be shown that the set

$$\{\sigma\lambda_Z \mid Z \subseteq \{z_1, \ldots, z_k\} \text{ and } \lambda_Z \text{ admissible for } \sigma\}$$

is a minimal complete set of AC_f-unifiers of Γ. In our example, this set contains 7 elements, corresponding to the subsets $\emptyset$, $\{z_1\}$, $\{z_2\}$, $\{z_3\}$, $\{z_4\}$, $\{z_1, z_4\}$, and $\{z_2, z_3\}$ of $Z = \{z_1, z_2, z_3, z_4\}$.

There are two different ways of extending this approach to AC_f-unification with constants. Stickel [61] first treats free constants like variables, computes the minimal complete set of unifiers for the elementary problem obtained this way, and uses this set to read off a minimal complete set for the original problem. Thus, in Stickel's approach one still solves homogeneous linear Diophantine equations. In contrast, Livesey and Siekmann handle constants with the help of inhomogeneous equations (see [31] for details). General AC_f-unification can again be treated using general combination techniques (see Section 5).

Unification under associativity, commutativity, and idempotence

The theory ACI_f axiomatizes associativity, commutativity and idempotence of the binary symbol f. For example, logical conjunction and disjunction, and union and intersection of sets satisfy these axioms. Let us first summarize the results from unification theory concerning this equational theory:

Unification type: ACI_f is finitary for all three types of unification [45, 18, 3, 38]. As for AC_f, the number of unifiers in a minimal complete set of ACI_f-unifiers may be doubly exponential in the size of a given (elementary) ACI_f-unification problem [38].

Unification algorithm: there exist unification algorithms for all three types of ACI_f-unification [3, 38].[5]

Complexity of the decision problem: For elementary ACI_f-unification and for ACI_f-unification with constants, the decision problem is polynomial, and solvability of general ACI_f-unification problems is NP-complete [37].

In the following, we sketch the polynomial decision procedure for ACI_f-unification with constants. Let $\Sigma := \{f\}$ where f is binary, and let V and C be countably infinite sets of variables and (free) constants, respectively. For a term $t \in T(\Sigma \cup C, V)$, we denote with $V(t)$ the set of variables occurring in t,

[5] To be more precise, [3] describes an algorithm for $ACI1_f$-unification with constants (where an additional unit element is present). The transition from $ACI1_f$-unification to ACI_f-unification can be achieved in the same way as for AC_f.

and with $C(t)$ the set of free constants occurring in t. These sets can be used to characterize equality modulo ACI_f:

$$s =_{ACI_f} t \text{ iff } V(s) = V(t) \text{ and } C(s) = C(t).$$

Let $\Gamma := \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ be an ACI_f-unification problem with constants, let $x_1, \ldots, x_k \in V$ be the variables and $c_1, \ldots, c_\ell \in C$ the free constants occurring in Γ. Without loss of generality, we assume that $\ell \geq 1$: if Γ is an elementary ACI_f-unification problem, then we take an arbitrary constant $c_1 \in C$. It is easy to see that Γ is solvable iff it has an ACI_f-unifier σ such that

$$(*) \quad V(x_i\sigma) = \emptyset \quad \text{and} \quad C(x_i\sigma) \subseteq \{c_1, \ldots, c_\ell\} \quad \text{for } i = 1, \ldots, k.$$

In fact, if τ is an arbitrary ACI_f-unifier of Γ, then σ can be obtained from τ by replacing in the terms $x_i\tau$ all free constants in $C \setminus \{c_1, \ldots, c_\ell\}$ and all variables by c_1.

The existence of a solution σ satisfying $(*)$ can now be expressed with the help of propositional Horn formulae. To this purpose, we introduce a propositional variable $P_{x,c}$ for every pair $(x, c) \in \{x_1, \ldots, x_k\} \times \{c_1, \ldots, c_\ell\}$, which has the intended meaning "$c \notin C(x\sigma)$". These variables are used to build the following formulae:

- For each $x \in \{x_1, \ldots, x_k\}$ the formula

  $$\left(\bigwedge_{i=1}^{\ell} P_{x,c_i} \right) \Rightarrow \texttt{false},$$

  which expresses that each variable must be replaced by a non-empty term, i.e., $C(x\sigma) \neq \emptyset$.

- For each equation $s \doteq t \in \Gamma$ and each $c \in C(s) \setminus C(t)$ the formula

  $$\left(\bigwedge_{x \in V(t)} P_{x,c} \right) \Rightarrow \texttt{false}.$$

  Since c occurs in s, and thus also in $s\sigma$, it must also occur in $t\sigma$. Because c does not occur in t, this is only possible if it is introduced by σ, i.e., if $c \in C(x\sigma)$ for some $x \in V(t)$.

- Symmetrically, we have for each equation $s \doteq t \in \Gamma$ and each $c \in C(t) \setminus C(s)$ the formula

  $$\left(\bigwedge_{x \in V(s)} P_{x,c} \right) \Rightarrow \texttt{false}.$$

- For each equation $s \doteq t \in \Gamma$ and each $c \in C \setminus (C(s) \cup C(t))$ the formula

  $$\left(\bigwedge_{x \in V(s)} P_{x,c} \right) \Leftrightarrow \left(\bigwedge_{x \in V(t)} P_{x,c} \right),$$

which expresses that a constant that does not occur in $s \doteq t$ may be introduced on the left-hand side iff it is introduced on the right-hand side. Actually, this formula is not a Horn formula, but it can easily be expressed by a set of $|V(s)| \cdot |V(t)|$ Horn formulae.

It is easy to see that the set of these Horn formulae is satisfiable iff Γ has a solution (satisfying $(*)$). Since the size of this set of Horn formulae is polynomial in the size of Γ, and since satisfiability of sets of propositional Horn formulae can be decided in linear time [23], this shows that solvability of ACI_f-unification problems with constants can be decided in polynomial time.

For example, consider the equation $f(a,x) \doteq f(y,b)$, where a, b are constants and x, y are variables. The corresponding set of propositional Horn formulae consists of

- $(P_{x,a} \wedge P_{x,b}) \Rightarrow$ `false` and $(P_{y,a} \wedge P_{y,b}) \Rightarrow$ `false`,
- $P_{y,a} \Rightarrow$ `false` and $P_{x,b} \Rightarrow$ `false`.

Because of the second two formulae, an evaluation that satisfies these formulae must assign `false` to $P_{y,a}$ and to $P_{x,b}$, that is, in a solution σ of $f(a,x) \doteq f(y,b)$, the constant a must occur in $y\sigma$ and the constant b must occur in $x\sigma$. With this assignment, the first two formulae are trivially satisfied. For this reason, $P_{x,a}$ and $P_{y,b}$ can be assigned arbitrary Boolean values. For example, if we assign the value `true` to both, then this corresponds to the substitution $\{x \mapsto b, y \mapsto a\}$, which is an ACI_f-unifier of the equation satisfying $(*)$.

5 Combination of unification algorithms

The unification algorithms and decision procedures for the theories AC and ACI sketched above cannot treat general unification problems, that is, problems containing additional free function symbols. The interpretation of these free function symbols is not constrained by the theory under consideration. Thus, an AC- or ACI-unification problem that contains only these free symbols can be solved by an algorithm for syntactic unification. This suggest that an algorithm for general AC- or ACI-unification could be obtained by appropriately combining an algorithm for AC- or ACI-unification with constants with an algorithm for syntactic unification. More generally, the additional function symbols not contained in the signature of AC may be non-free themselves, that is, their properties may be defined by another equational theory. For example, in many application one must deal with AC-unification problems that contain more than one AC-symbol. As an example, consider the theory of Boolean rings: both addition + and multiplication × are associative-commutative. If one tries to handle this theory via term rewriting, one usually employs rewriting modulo associativity and commutativity of addition and multiplication, i.e., modulo the theory $AC_+ \cup AC_\times$. In this setting, the computation of critical pairs depends on an algorithm for general $(AC_+ \cup AC_\times)$-unification. The question is thus how to obtain such an algorithm from algorithms for AC-unification with constants and for syntactic unification. The research on this question resulted in unification algorithms that can treat unification problems containing several AC-symbols and free symbols [61, 62, 24, 31].

It turned out that the approaches used in the context of combining AC-unification with syntactic unification can also be employed in the more general setting of the so-called *combination problem for unification:*

> Assume that $E_1, \ldots, E_n$ are equational theories over pairwise disjoint signatures.[6] How can algorithms for unification modulo E_i $(i = 1, \ldots, n)$ be combined to an algorithm for unification modulo $E_1 \cup \cdots \cup E_n$.

The first solutions to this more general problem [40, 63, 29, 64, 13] were restricted to theories that satisfy certain restrictions (such as collapse-freeness or regularity[7]) on the syntactic form of their defining identities. These restrictions made sure that the theories behaved similar to AC and syntactic equality. The theory ACI could not be treated by these methods since it is not collapse-free.

The problem was finally solved in a very general form by Schmidt-Schauß [55]. His approach imposes no restriction on the syntactic form of the identities. The only requirements on the single theories E_i to be combined are of an algorithmic nature: E_i-unification problems with constants must be finitary solvable in E_i, and so-called "constant elimination problems" (see [55] for a definition) must be finitary solvable in E_i. It is easy to see that ACI satisfies these properties. The major drawback of this method is that it can combine only unification algorithms (i.e., algorithms computing complete sets of unifiers). Decision procedures cannot be treated with this approach. For example, it is not possible to show decidability of general A-unification with the help of this method.

A general combination method

We shall now describe the combination method introduced in [4]. In contrast to the method by Schmidt-Schauß, it can be used both for combining unification algorithms and for combining decision procedures. As for the method of Schmidt-Schauß, algorithms for unification with constants for the single theories E_i are not really sufficient for the combination approach to apply. However, instead of splitting the algorithmic problem into two parts (unification with constants and constant elimination), we restrict our attention to only one type of problem, which is a generalization of unification with constants:

[6] Note that without this disjointness condition there cannot be a general combination method since, for non-disjoint signatures, $(E_1 \cup \ldots \cup E_n)$-unification may be undecidable (non-finitary) even though E_i-unification is decidable (finitary) (for $i = 1, \ldots, n$).

[7] A theory E is called collapse-free if it does not contain an identity of the form $x = t$ where x is a variable and t is a non-variable term, and it is called regular if the left- and right-hand sides of the identities contain the same variables.

Definition 5.1 *An E-*unification problem with linear constant restrictions *consists of an E-unification problem with constants Γ and a linear ordering $<$ on the variables and free constants occurring in Γ. An E-*unifier *(or* solution*) of this problem is an E-unifier σ of Γ that satisfies*

$$x < c \quad \Rightarrow \quad c \text{ does not occur in } x\sigma$$

for all variables x and free constants c occurring in Γ.

For example, let $E := \{h(x) = h(x)\}$ and consider the E-unification problem with constants $\Gamma := \{h(x) \doteq h(c)\}$. Since E contains only a trivial identity for h, E-unification is simply syntactic unification, and thus any solution of Γ must substitute x by c. If we add the constant restriction $x < c$ to Γ, then the obtained E-unification problem with linear constant restrictions does not have a solution.

In order to make clear which are the free constants and which the variables of a given E-unification problem with linear constant restrictions, we will write such a problem as a 4-tuple $\langle \Gamma, X, C, < \rangle$ where X denotes the set of variables and C the set of free constants, and $<$ is a linear ordering on $X \cup C$. Complete sets of solutions of E-unification problems with linear constant restrictions are defined as in the case of E-unification with constants.

Let $E_1, \ldots, E_n$ be non-trivial equational theories over disjoint signatures. Our first goal is to reduce solvability of a given *elementary* $(E_1 \cup \cdots \cup E_n)$-unification problem Γ_0 to solvability of E_i-unification problems with linear constant restrictions. To this purpose, Γ_0 is first transformed into an equivalent problem in decomposed form. An (elementary) $(E_1 \cup \cdots \cup E_n)$-unification problem Γ is in *decomposed form* iff $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_n$ where each Γ_i is an elementary E_i-unification problem. The transformation simply introduces new variables for alien subterms and adds appropriate new equations (see [4] for details).

Example 5.2 Let $E_1 := \{(x \circ y) \circ z = x \circ (y \circ z)\}$ for a binary (infix) symbol $\circ$ and $E_2 := \{h(x) \doteq h(x)\}$ for a unary symbol h, and consider the $(E_1 \cup E_2)$-unification problem

$$\Gamma_0 := \{h(x) \circ y \doteq y \circ h(z_1 \circ z_2)\}.$$

On the left-hand side, $h(x)$ is an alien subterm of the term $h(x) \circ y$. This term can be replaced by a new variable x_1, provided that we add a new equation $x_1 \doteq h(x)$. Similarly, the alien subterms on the right-hand side can be replaced by variables. This results in the decomposed problem

$$\Gamma := \quad \{x_1 \circ y \doteq y \circ x_2,\ x_3 \doteq z_1 \circ z_2\} \quad \cup \quad \{x_1 \doteq h(x),\ x_2 \doteq h(x_3)\}.$$

Unfortunately, it is not the case that a decomposed $(E_1 \cup \cdots \cup E_n)$-unification problem $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_n$ is solvable if all the subproblems Γ_i are separately solvable modulo E_i. The reason is that there is an interaction between the problems Γ_i via shared variables. Thus, a solution of Γ_1 might replace a shared variable by one term, and a solution of Γ_2 might replace this variable by a completely different term, and it is not clear how these two incompatible solutions could be combined to a solution of the whole problem. To avoid incompatible solutions of the single problems, we must (nondeterministically) add some additional information:

1. First, we must choose a *partition* $\Pi = \{P_1, \ldots, P_k\}$ of the set of variables occurring in Γ. For each of the classes P_i, let $x_i \in P_i$ be a representative of this class, and let $X_\Pi := \{x_1, \ldots, x_k\}$ be the set of these representatives. The substitution that replaces, for all $i = 1, \ldots, k$, each element of P_i by its representative x_i is denoted by σ_Π. The unification problem $\Gamma\sigma_\Pi = \Gamma_1\sigma_\Pi \cup \cdots \cup \Gamma_n\sigma_\Pi$ is obtained from $\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_n$ by applying σ_Π to the terms occurring in Γ. Obviously, X_Π is the set of variables occurring in $\Gamma\sigma_\Pi$.

2. Second, we choose a *labelling function* $L : X_\Pi \to \{1, \ldots, n\}$, which assigns a theory label to each variable occurring in $\Gamma\sigma_\Pi$. We denote the set of variables with label i by X_i, and define $C_i := X_\Pi \setminus X_i$.

3. Finally, we choose a *linear ordering* $<$ on X_Π.

With the help of L and $<$, each of the elementary E_i-unification problems $\Gamma_i\sigma_\Pi$ is turned into the E_i-unification problem with linear constant restrictions $\langle \Gamma_i\sigma_\Pi, X_i, C_i, < \rangle$, i.e., the variables with label i are still treated as variables, but the variables with different label are treated as (free) constants.

Proposition 5.3 *Let $\Gamma := \Gamma_1 \cup \cdots \cup \Gamma_n$ be an (elementary) $(E_1 \cup \cdots \cup E_n)$-unification problem in decomposed form. Then the following statements are equivalent:*

1. *Γ is solvable, i.e., there exists an $(E_1 \cup \cdots \cup E_n)$-unifier of Γ.*

2. *There exists a partition Π, a labelling function $L : X_\Pi \to \{1, \ldots, n\}$, and a linear ordering $<$ on X_Π such that, for all $i = 1, \ldots, n$, the E_i-unification problem with linear constant restrictions $\langle \Gamma_i\sigma_\Pi, X_i, C_i, < \rangle$ is solvable.*

A proof of this proposition can be found in [4, 8]. Obviously, any $(E_1 \cup \cdots \cup E_n)$-unification problem can be transformed into a decomposed $(E_1 \cup \cdots \cup E_n)$-unification problem in polynomial time, and an appropriate partition Π, a labelling function L and linear ordering $<$ can be guessed in *nondeterministic* polynomial time. This yield the following combination result for decision procedures:

Theorem 5.4 *Let $E_1, \ldots, E_n$ be non-trivial equational theories over disjoint signatures.*

1. *If solvability of E_i-unification problems with linear constant restrictions is decidable for $i = 1, \ldots, n$, then solvability of elementary $(E_1 \cup \cdots \cup E_n)$-unification problems is decidable.*

2. *If solvability of E_i-unification problems with linear constant restrictions is decidable by an NP-algorithm for $i = 1, \ldots, n$, then solvability of elementary $(E_1 \cup \cdots \cup E_n)$-unification problems is decidable by an NP-algorithm.*

As an example, let us reconsider the decomposed unification problem

$$\Gamma := \quad \{x_1 \circ y \doteq y \circ x_2,\ x_3 \doteq z_1 \circ z_2\} \quad \cup \quad \{x_1 \doteq h(x),\ x_2 \doteq h(x_3)\}$$

computed in Example 5.2. Let Π be the partition where x_1 and x_2 constitute one class, and all the other classes are singletons, and assume that x_1 is the representative of the class $\{x_1, x_2\}$. Hence, $X_\Pi = \{x, x_1, x_3, y, z_1, z_2\}$, $\sigma_\Pi = \{x_2 \mapsto x_1\}$, and $\Gamma\sigma_\Pi = \Gamma_1\sigma_\Pi \cup \Gamma_2\sigma_\Pi$ where

$$\Gamma_1\sigma_\Pi = \{x_1 \circ y \doteq y \circ x_1,\ x_3 \doteq z_1 \circ z_2\} \text{ and } \Gamma_2\sigma_\Pi = \{x_1 \doteq h(x),\ x_1 \doteq h(x_3)\}.$$

If we choose the labelling $L(x_3) = L(y) = 1$ and $L(x_1) = L(x) = L(z_1) = L(z_2) = 2$, and the linear ordering $z_1 < z_2 < x_3 < x < x_1 < y$, then we have $X_1 = C_2 = \{x_3, y\}$ and $X_2 = C_1 = \{x_1, x, z_1, z_2\}$. The substitution $\sigma_1 := \{x_3 \mapsto z_1 \circ z_2,\ y \mapsto x_1\}$ solves the E_1-unification problem with linear constant restrictions $\langle \Gamma_1\sigma_\Pi, X_1, C_1, < \rangle$, and $\sigma_2 := \{x_1 \mapsto h(x_3),\ x \mapsto x_3\}$ solves the E_2-unification problem with linear constant restrictions $\langle \Gamma_2\sigma_\Pi, X_2, C_2, < \rangle$. By Proposition 5.3, this implies that the decomposed problem Γ (and thus also the original problem Γ_0) has a solution. The solutions σ_1 and σ_2 can be combined into a solution σ of Γ by induction on the linear order $<$:

$$\begin{aligned} \sigma \ = \ & \{z_1 \mapsto z_1,\ z_2 \mapsto z_2,\ x_3 \mapsto z_1 \circ z_2,\ x \mapsto z_1 \circ z_2, \\ & x_1 \mapsto h(z_1 \circ z_2),\ x_2 \mapsto h(z_1 \circ z_2),\ y \mapsto h(z_1 \circ z_2)\}. \end{aligned}$$

A formal definition of the combined solution can be found in [4, 8]. Given complete sets of solutions of the E_i-unification problems with linear constant restrictions induced by a chosen triple $(\Pi, L, <)$, one obtains a set of solutions of the original problem by constructing all possible combined solutions. It can be shown (see [4, 8]) that the union of these sets for all possible triples $(\Pi, L, <)$ yields a complete set of solutions of the original problem. Since there are only finitely many different triples $(\Pi, L, <)$, we thus obtain the following combination result for unification algorithms:

Theorem 5.5 *Let $E_1, \ldots, E_n$ be non-trivial equational theories over disjoint signatures that are finitary for E_i-unification with linear constant restrictions. Then $E_1 \cup \cdots \cup E_n$ is finitary for elementary unification.*

This theorem is effective in the sense that algorithms computing finite complete sets of solutions of E_i-unification problems with linear constant restrictions can be used to construct (via the above described nondeterministic combination method) an algorithm for elementary $(E_1 \cup \cdots \cup E_n)$-unification.

As examples of theories that satisfy the prerequisites of our two combination theorems, we mention free theories, *A*, *AC* and *ACI*:

1. For a finite signature Σ, the free theory F_Σ consists of the identities $f(x_1, \ldots, x_n) \doteq f(x_1, \ldots, x_n)$ where f is an n-ary symbol in Σ. Obviously, unification modulo F_Σ is just syntactic unification. It is easy to see that F_Σ-unification with linear constant restrictions is unitary and decidable in linear time. In fact, such a problem has a solution iff the most general unifier of the corresponding syntactic unification problem satisfies the constant restrictions. In this case, this unifier is also a most general solution of the problem with linear constant restrictions. Consequently, free theories can always be handled by our combination method. This shows that the results of Theorem 5.4 and 5.5, which were formulated for elementary unification in the combined theory, can be lifted to general unification by just adding an appropriate free theory in the combination process.

2. For A, decidability of unification problems with linear constant restrictions is an easy consequence of a result by Schulz [56] on a generalization of Makanin's decision procedure. Together with decidability of unification with linear constant restrictions for free theories, this implies that general A-unification is decidable by Theorem 5.4.

3. For AC, solvability of unification problems with constants can be reduced to solvability of systems of homogeneous and inhomogeneous linear Diophantine equations (see [31]). It is very easy to handle constant restrictions in this reduction: $x_i < c$ simply means that the variable x_i must be set to zero in the inhomogeneous equation corresponding to c (see [6] for more details). Consequently, solvability of AC-unification problems with linear constant restrictions can be reduced to an integer programming problem, and is thus decidable by an NP-algorithm. By Theorem 5.4 this implies that general AC-unification is NP-decidable.

   Since AC is a regular equational theory, the fact that AC is finitary for unification with constants implies that it is also finitary for unification with linear constant restrictions. In fact, for a regular equational theory, a complete set of solutions of a given unification problem with linear constant restrictions can be obtained from a complete set of solutions of the corresponding unification problem with constants by simply removing those unifiers not satisfying the constant restrictions. The reason is that any instance of a unifier not satisfying the constant restrictions also does not satisfy the constant restrictions.[8] By Theorem 5.5, we can conclude that AC is finitary with respect to general unification.

4. For ACI, the decision procedure for unification with constants described in Section 4 can easily be extended to a decision procedure for ACI-unification with linear constant restrictions: since the intuitive meaning of the propositional variable $P_{x,c}$ is "$c \not\in C(x\sigma)$", a constant restriction $x < c$ simply means that $P_{x,c}$ must be assigned the value `true`. However, we can no longer restrict our attention to unifiers that introduce only

[8] Note that this need not be the case for non-regular theories.

free constants contained in the unification problem: we must allow for an additional free constant c in C (see [6] for details). As for the case of unification with constants, this reduction yields a polynomial decision procedure for *ACI*-unification with linear constant restrictions. Thus, Theorem 5.4 implies that general *ACI*-unification is NP-decidable.

Since the theory *ACI* is regular, the same argument as for *AC* shows that *ACI* is finitary for unification with linear constant restrictions, and thus also with respect to general unification.

Optimization techniques

Both Schmidt-Schauss' method and the general combination method described above solve the problem of how to combine unification algorithms and decision procedures for unification only from a theoretical point of view. Since these methods are highly nondeterministic, significant optimizations are necessary before one can hope for a combined unification algorithms that can be used in a realistic application.

Some simple optimizations are quite straightforward. In both algorithms, it is possible to restrict all nondeterministic choices to "shared" variables, that is, variables that occur in at least two subproblems of the decomposed problem. Another simple optimization for the algorithm given in [8] relies on the observation that different linear orders need not lead to different constant restrictions. For example, assume that x, y are variables and c, d are constants. Then the ordering $x < c < d < y$ leads to the same restrictions on solutions of a unification problem as the ordering $x < d < c < y$ (both just say that x must not be replaced by a term containing c or d). This observation can be used to prune the number of different linear orderings that must be considered.

An optimized version of Schmidt-Schauss' algorithm has been described by A. Boudet [11]. Basically, Boudet's algorithm takes a mixed unification problem in decomposed form and computes unifiers for the pure subproblems in the component theories. Nondeterministic choices (of theory labels for variables, etc.) are only made to resolve conflicts that are created by incompatible instantiations made by these unifiers. In this respect, the algorithm creates only the "necessary" nondeterminism. However, for non-unitary theories, it introduces another source of nondeterminism, which is due to the fact that, for every unifier in a complete set computed during the algorithm, it opens a new branch of the search tree.

When combining decision procedures, no information on the form of the unifiers of the pure subproblems is available in general. For this reason, Boudet's optimization techniques cannot be used in this context. Two orthogonal optimization approaches for the general combination algorithm described above have been introduced in [39]. The first method, called iterative decomposition, applies to the combination of $n > 2$ theories. Unlike the general combination method described above, it does not make all the nondeterministic decisions before trying to solve the resulting unification problems with linear constant restrictions. It starts with the first theory by making only the decisions that are necessary for solving Γ_1. If this system turns out to be solvable for some specific combination of decisions, the remaining decisions necessary for the second system are made, etc. Whereas this first method determines in which order nondeterministic decision should best be made, the second one shows how to use specific algorithms for the component theories to reach certain decisions deterministically. In principle, these specific algorithms provide partial information on the form of the unifiers, and this information is used to preclude certain decisions. A synthesis of both techniques has been implemented, and run time tests show that the optimized combination method obtained this way leads to combined decision procedures that have a quite reasonable time complexity [39].

6 Unification from a logical and algebraic point of view

It is well-known that the decision problems for elementary unification and for unification with constants correspond to "natural" classes of logical decision problems. In addition, they can also be seen as decision problems for certain free algebras. In the following, we shall show that a similar logical and algebraic characterization can (simultaneously) be given for general unification and unification with linear constant restrictions.

Before we can recall the characterizations for elementary unification and for unification with constants, we must introduce some notation. Let Σ be a signature and E be an equational theory such that $sig(E) = \Sigma$. An atomic Σ-formula is an equation $s = t$ between Σ-terms. A positive Σ-matrix is built from atomic Σ-formulae using conjunction and disjunction, and a positive Σ-sentence is a quantifier-prefix followed by a positive Σ-matrix containing only variables bound by the prefix. Such a positive Σ-sentence is called existential iff its prefix contains only existential quantifiers, and it is called a positive AE Σ-sentence iff its quantifier prefix consists of a block of universal quantifiers followed by a block of existential quantifiers. The positive (positive existential, positive AE) Σ-theory of E consists of all positive (positive existential, positive AE) Σ-sentences that are true in all models of E. The positive (positive existential, positive AE) Σ-theory of the free algebra $\mathcal{T}(\Sigma, V)/{=_E}$ consists of all positive (positive existential, positive AE) sentences that are true in $\mathcal{T}(\Sigma, V)/{=_E}$.

Theorem 6.1 *Let Σ be a signature, and E be a non-trivial equational theory such that $sig(E) = \Sigma$.*

1. *Solvability of elementary E-unification problems is decidable iff the positive existential Σ-theory of E is decidable iff the positive existential Σ-theory of $\mathcal{T}(\Sigma, V)/{=_E}$ for a countably infinite set of variables V is decidable.*

2. *Solvability of E-unification problems with constants is decidable iff the*

positive AE Σ-theory of E is decidable iff the positive AE Σ-theory of $\mathcal{T}(\Sigma, V)/{=_E}$ for a countably infinite set of variables V is decidable.

The first statement should be obvious, and the second becomes clear if one Skolemizes the universal quantifiers in the positive AE sentence (which replaces the universally quantified variables by free constants).

For example, the elementary A_f-unification problem $\{f(x,y) \doteq f(y,x)\}$ can be translated into the positive existential sentence $\exists x.\exists y.f(x,y) = f(y,x)$, and the A_f-unification problem with constants $\{f(a,y) \doteq f(y,a)\}$ (where a is a constant) can be translated into the positive AE sentence $\forall x.\exists y.f(x,y) = f(y,x)$.

In [8, 5], this characterization was generalized to general unification and unification with linear constant restrictions as follows:

Theorem 6.2 *Let Σ be a signature, and E be a non-trivial equational theory such that sig(E) $= \Sigma$. Then the following statements are equivalent:*

1. *Solvability of E-unification problems with linear constant restrictions is decidable.*
2. *The positive Σ-theory of E is decidable.*
3. *The positive Σ-theory of $\mathcal{T}(\Sigma, V)/{=_E}$ for a countably infinite set of variables V is decidable.*
4. *Solvability of general E-unification problems is decidable.*

This theorem gives a nice logical and algebraic characterization of general E-unification and of the (at first sight rather technical) concept of E-unification with linear constant restrictions. From a practical point of view, it is interesting because it shows that any theory that can reasonably be integrated in a universal deductive machinery via unification can also be combined with other such theories. In fact, we have pointed out above (see Example 3.4) that such an integration usually requires algorithms for general unification, and the theorem shows that such an algorithm also makes sure that the preconditions for our combination method are satisfied.[9]

In the following, we motivate the equivalence between the decision problems for E-unification with linear constant restrictions, for the positive theory of E, and for general E-unification by sketching how these problems can be translated into each other (see [8] for details):

- Any E-unification problem with linear constant restrictions $\langle \Gamma, X, C, < \rangle$ can be translated into an an equivalent positive Σ-sentence ϕ_Γ as follows:

[9] Actually, the theorem makes this statement only for decision procedure, but in [8] it is shown that the equivalence between general unification and unification with linear constant restrictions also holds with respect to unification algorithms.

both variables and free constants are treated as variables in this formula; the matrix of ϕ_Γ is the conjunction of all equations in Γ; and in the quantifier prefix, the elements of X (variables in Γ) are universally quantified, the elements of C (free constants in Γ) are existentially quantified, and the order of the quantifications is given by the linear ordering $<$.

- Given a positive Σ-sentence ϕ with conjunctive[10] matrix, one first removes universal quantifiers by Skolemization. The obtained existential formula is translated into a unification problem in the obvious way. The resulting problem may contain Skolem function, which explains why it may be a general E-unification problem.

- The combination method described above shows how solvability of a given general E-unification problem can be reduced to solvability of E-unification problems with linear constant restrictions.

As an example, consider the free theory $F_{\{g\}} := \{g(x) = g(x)\}$, and the $F_{\{g\}}$-unification problem with constants $\{x \doteq g(c)\}$. If we add the constant restriction $x < c$, then this problem is not solvable (since any solution must substitute x by the term $g(c)$, which contains the constant c). However, under the restriction $c < x$ the problem is solvable. The following are the positive sentences and general unification problems obtained by translating these two unification problems with linear constant restrictions:

unification with lcr	*positive sentence*	*general unification*
$\{x \doteq g(c)\},\ x < c$	$\exists x.\forall y.\ x = g(y)$	$\{x \doteq g(h(x))\}$
$\{x \doteq g(c)\},\ c < x$	$\forall y.\exists x.\ x = g(y)$	$\{x \doteq g(d)\}$

For example, $\exists x.\forall y.\ x = g(y)$ is not valid in the theory of $F_{\{g\}}$ since this formula says that g must be a constant function, which obviously does not follow from $F_{\{g\}}$. Correspondingly, $\{x \doteq g(h(x))\}$ does not have a solution because it leads to an occur-check failure during syntactic unification.

Theorem 6.2 together with our combination result for decision procedures yields the following modularity result for decidability of positive theories:

Theorem 6.3 *Let $E_1, \ldots, E_n$ be non-trivial equational theories over disjoint signatures. Then the positive theory of $E_1 \cup \ldots \cup E_n$ is decidable iff the positive theories of the component theories E_i are decidable, for $i = 1, \ldots, n$.*

In its equivalent *algebraic formulation*, this theorem says that the positive theory of the combined free algebra $\mathcal{T}(\Sigma, V)/{=_E}$ for $\Sigma := \Sigma_1 \cup \cdots \cup \Sigma_n$ and $E := E_1 \cup \cdots \cup E_n$ is decidable iff the positive theories of the free algebras $\mathcal{T}(\Sigma_i, V)/{=_{E_i}}$ are decidable, for $i = 1, \ldots, n$. This algebraic combination result

[10] To make the description simpler, we do not consider disjunction her; in [8] it is shown how disjunction can be handled.

can be proved directly (see [5]), using an explicit algebraic construction of the combined free algebra from the single free algebras. In addition, the combination result is generalized in [5] to free structures, i.e., to the case of signatures that may also contain predicate symbols, and in [7] to a more general class of structures, which we called quasi-free structures. The algebra of rational trees is an example of a quasi-free structure that is not free.

7 Further topics

The purpose of this chapter was to introduce the notions that are important in unification theory, and to illustrate the results obtained in this research area by giving some examples. There are, however, other interesting topics in unification theory that we have not addressed at all. Exhaustive surveys on most aspects of equational unification and related topics can be found in [60, 35, 9]. To give an impression on what additional topics there are in this area, we mention the following three problems (references to the relevant literature can, for example, be found in [9]:

General procedures for equational unification. The unification algorithms and procedure described in Section 4 were specific to the equational theory under consideration. In contrast, a general unification procedure receives as input not only the unification problem, but also the equational theory modulo which the problem is to be solved. A special case of such a procedure is the *narrowing* procedure, which assumes that the equational theory is given by a confluent and terminating term rewriting system.

Higher-order unification and matching. Instead of considering first-order terms and equational theories, one may ask how to compute a representation of the set of all solutions of an equation between higher-order terms? The main difference to first-order unification is that the terms may now contain variables for functions, which may be replaced by substitutions.

Unification in sort theories. In many applications, the universe on which function symbols operate is not just one homogeneous set, but it is divided into different subsets, which are represented by sorts (such as the sort of *integers*, the sort of *lists*, etc.). In a simple many-sorted environment (where sorts are interpreted as arbitrary non-empty sets), unification does not behave very differently from the single-sorted case considered in this chapter. However, as soon as one considers order-sorted signatures (where one sort may be declared to be a subsort of another), things become a lot more complex.

References

[1] F. Baader. Unification in idempotent semigroups is of type zero. *J. Automated Reasoning*, 2(3):283–286, 1986.

[2] F. Baader. Unification, weak unification, upper bound, lower bound, and generalization problems. In R.V. Book, editor, *Proceedings of the 4th International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 86–97, Como, Italy, 1991. Springer-Verlag.

[3] F. Baader and W. Büttner. Unification in commutative idempotent monoids. *Theoretical Computer Science*, 56(1):345–352, 1988.

[4] F. Baader and K.U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 50–65, Saratoga Springs, NY, USA, 1992. Springer-Verlag.

[5] F. Baader and K.U. Schulz. Combination techniques and decision problems for disunification. In C. Kirchner, editor, *Proceedings of the 5th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Artificial Intelligence, Montreal, Canada, 1993. Springer-Verlag.

[6] F. Baader and K.U. Schulz. General A- and AX-unification via optimized combination procedures. In *Proceedings of the Second International Workshop on Word Equations and Related Topics*, volume 677 of *Lecture Notes in Computer Science*, pages 23–42, Rouen, France, 1993. Springer-Verlag.

[7] F. Baader and K.U. Schulz. On the combination of symbolic constraints, solution domains, and constraint solvers. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming, CP95*, volume 976 of *Lecture Notes in Artificial Intelligence*, Cassis, France, 1995. Springer Verlag.

[8] F. Baader and K.U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symbolic Computation*, 21:211–243, 1996.

[9] F. Baader and J.H. Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 41–125. Oxford University Press, Oxford, UK, 1994.

[10] D. Benanav, D. Kapur, and P. Narendran. Complexity of matching problems. In J.-P. Jouannaud, editor, *Proceedings of the 1st International Conference on Rewriting Techniques and Applications*, volume 202 of *Lecture Notes in Computer Science*, pages 417–429, Dijon, France, 1985. Springer-Verlag.

[11] A. Boudet. Combining unification algorithms. *Journal of Symbolic Computation*, 8:449–477, 1993.

[12] A. Boudet, E. Contejean, and H. Devie. A new AC-unification algorithm with a new algorithm for solving diophantine equations. In *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science*, pages 141–150, 1990.

[13] A. Boudet, J.-P. Jouannaud, and M. Schmidt-Schauß. Unification in Boolean rings and Abelian groups. *J. Symbolic Computation*, 8:449–477, 1989.

[14] H.-J. Bürckert. Some relationships between unification, restricted unification and matching. In J.H. Siekmann, editor, *Proceedings of the 8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, Oxford, UK, 1986. Springer-Verlag.

[15] H.-J. Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*, volume 568 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1991.

[16] H.-J. Bürckert, A. Herold, and M. Schmidt-Schauß. On equational theories, unification, and decidability. *J. Symbolic Computation*, 8(3,4):3–49, 1989.

[17] W. Büttner. Unification in the data structure multiset. *J. Automated Reasoning*, 2(1):75–88, 1986.

[18] W. Büttner. Unification in the data structure sets. In J.H. Siekmann, editor, *Proceedings of the 8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 470–488, Oxford, UK, 1986. Springer-Verlag.

[19] M. Clausen and A. Fortenbacher. Efficient solution of linear diophantine equations. *J. Symbolic Computation*, 8(1,2):201–216, 1989.

[20] E. Contejean. An efficient algorithm for solving systems of diophantine equations, 1992. To appear in *Information and Computation.*

[21] E. Domenjoud. *Outils pour la déduction automatique dans les théories associatives-commutatives.* Thèse de Doctorat d'Université, Université de Nancy I, 1991.

[22] E. Domenjoud. A technical note on AC-unification. The number of minimal unifiers of the equation $\alpha x_1 + \cdots + \alpha x_p = \beta y_1 + \cdots + \beta y_q$. *J. Automated Reasoning*, 8(1):39–44, 1992.

[23] W.F. Dowling and J. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programmming*, 1(3):267–284, 1984.

[24] F. Fages. Associative-commutative unification. In R.E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction*, volume 170 of *Lecture Notes in Computer Science*, pages 194–208, New York, 1984. Springer-Verlag.

[25] F. Fages. Associative-commutative unification. *J. Symbolic Computation*, 3:257–275, 1987.

[26] F. Fages and G. Huet. Complete sets of unifiers and matchers in equational theories. *Theoretical Computer Science*, 43(1):189–200, 1986.

[27] A. Fortenbacher. An algebraic approach to unification under associativity and commutativity. In J.-P. Jouannaud, editor, *Proceedings of the 1st International Conference on Rewriting Techniques and Applications*, volume 202 of *Lecture Notes in Computer Science*, pages 381–397, Dijon, France, 1985. Springer-Verlag.

[28] P. Gordan. Über die Auflösung linearer Gleichungen mit reellen Coeffizienten. *Mathematische Annalen*, pages 23–28, 1873.

[29] A. Herold. Combinations of unification algorithms. In J.H. Siekmann, editor, *Proceedings of the 8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 450–469, Oxford, UK, 1986. Springer-Verlag.

[30] A. Herold. *Combination of Unification Algorithms in Equational Theories.* Ph.D. thesis, Universität Kaiserslautern, Kaiserslautern, Germany, 1987.

[31] A. Herold and J.H. Siekmann. Unification in Abelian semigroups. *J. Automated Reasoning*, 3:247–283, 1987.

[32] Ju. I. Hmelevskij. *Equations in Free Semigroups*, volume 107 of *Proceedings of the Steklov Institute of Mathematics.* AMS, Providence, Rhode Island, 1976.

[33] G.P. Huet. An algorithm to generate the basis of solutions to homogeneous linear diophantine equations. *Information Processing Letters*, 7(3), 1978.

[34] J. Jaffar. Minimal and complete word unification. *J. of the ACM*, 37(1):47–85, 1990.

[35] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of A. Robinson.* MIT Press, Cambridge, MA, 1991.

[36] J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Computing*, 15, 1986.

[37] D. Kapur and P. Narendran. Complexity of unification problems with associative-commutative operators. *J. Automated Reasoning*, 9:261–288, 1992.

[38] D. Kapur and P. Narendran. Double exponential complexity of computing complete sets of AC-unifiers. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, pages 11–21, Santa Cruz, California, 1992.

[39] S. Kepser and J. Richts. Optimisation techniques for combining unification algorithms. LTCS-Report 96-04, Theoretische Informatik, RWTH Aachen, Germany, 1996.

[40] C. Kirchner. *Méthodes et Outils de Conception Systématique d'Algorithmes d'Unification dans les Théories Equationelles.* Thèse d'État, Université de Nancy I, France, 1985.

[41] C. Kirchner and H. Kirchner. Constrained equational reasoning. In *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, Portland, Oregon, 1989. ACM Press.

[42] A. Koscielski and L. Pacholski. Complexity of unification in free groups and free semigroups. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 824–829, Los Alamitos, 1990.

[43] A. Lentin. Equations in free monoids. In M. Nivat, editor, *Automata, Languages and Programming*, 1972.

[44] P. Lincoln and T. Christian. Adventures in associative-commutative unification. *J. Symbolic Computation*, 8(1,2):217–240, 1989.

[45] M. Livesey and J.H. Siekmann. Unification of AC-terms (bags) and ACI-terms (sets). Internal report, University of Essex, 1975. Also published as Technical Report 3-76, Universität Karlsruhe, 1976.

[46] G.S. Makanin. The problem of solvability of equations in a free semigroup. *Math. Sbornik*, 103:147–236, 1977. English translation in Math. USSR Sbornik 32, 1977.

[47] R. Nieuwenhuis and A. Rubio. AC-supperposition with constraints: No AC-unifiers needed. In A. Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 545–559, Nancy, France, 1990. Springer-Verlag.

[48] J.P. Pécuchet. *Équations avec constantes et algorithme de Makanin.* Thèse de doctorat, Laboratoire d'Informatique, University of Rouen, 1981.

[49] G. Peterson and M.E. Stickel. Complete sets of reductions for equational theories with complete unification algorithms. *J. of the ACM*, 28(2):233–264, 1981.

[50] G. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[51] L. Pottier. Minimal solutions of linear diophantine equations: Bounds and algorithms. In R.V. Book, editor, *Proceedings of the 4th International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173, Como, Italy, 1991. Springer-Verlag.

[52] J.A. Robinson. A machine oriented logic based on the resolution principle. *J. of the ACM*, 12(1):23–41, 1965.

[53] J.A. Robinson. A review of automatic theorem proving. In *Annual Symposium in Applied Mathematics*, pages 1–18, Providence, 1967. American Mathematical Society.

[54] M. Schmidt-Schauß. Unification under associativity and idempotence is of type nullary. *J. Automated Reasoning*, 2(3):277–282, 1986.

[55] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *J. Symbolic Computation*, 8(1,2):51–99, 1989.

[56] K.U. Schulz. Makanin's algorithm – two improvements and a generalization. Technical Report CIS-Report 91–39, CIS, University of Munich, 1991.

[57] K.U. Schulz. Word unification and transformation of generalized equations. *J. Automated Reasoning*, 11:149–184, 1993.

[58] J.H. Siekmann. *Unification and Matching Problems.* Memo, Essex University, 1978.

[59] J.H. Siekmann. Unification of commutative terms. In *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation, EUROSAM'79*, volume 72 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 1979.

[60] J.H. Siekmann. Unification theory: A survey. *J. Symbolic Computation*, 7(3,4):207–274, 1989.

[61] M.E. Stickel. A complete unification algorithm for associative-commutative functions. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 71–82, Tblisi, USSR, 1975.

[62] M.E. Stickel. A unification algorithm for associative commutative functions. *J. of the ACM*, 28(3):423–434, 1981.

[63] E. Tidén. Unification in combinations of collapse-free theories with disjoint sets of function symbols. In J.H. Siekmann, editor, *Proceedings of the 8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 431–449, Oxford, UK, 1986. Springer-Verlag.

[64] K. Yelick. Unification in combinations of collapse-free regular theories. *J. Symbolic Computation*, 3(1,2):153–182, 1987.