Combining Algorithms for A1- and AC1-Unification Sharing Unit

Alwen Fernanto Tiu

February 1, 2001

1 Introduction

Unification is basically concerned with checking the equality between terms built over a set of symbols. The symbols we consider here consist of function symbols (such as e, f, g, a, or b) and variables (such as x, y, z). In syntactic unification, we deal only with syntactical equality between symbol. Given a pair of terms $(f(a,x), f(y,a))$ we can replace x and y with a to make both terms syntactically equal. In *equational unification*, in addition to syntactical equality of function symbols, we also take into account their semantic properties. For example if we know that f is commutative, i.e. $f(x,y) = f(y,x)$, then we can also replace x with y to make the terms $(f(a,x), f(y,a))$ equal modulo commutativity.

Unification theory has many applications in areas such automated reasoning and term rewriting. In automated reasoning for example, one often needs to check whether two expressions can be made equal (or *unifiable*), i.e., by substituting the variables in the expressions with suitable terms, and find such substitution (*unifier*) if it is so. The function symbols contained in these expressions usually have some semantic properties such as commutativity (C) or associativity-commutativity (AC). Algorithms for C-unification and AC-unification have been well developed separately. However, these algorithms can handle only one function symbol, i.e., the equational theory underlying these expressions defines only the properties of one function symbol. In practice the expressions we have to deal with contain more than one function symbol with different properties. For example in logic we sometimes have to deal with expressions like $a \wedge (b \vee c)$ or $a \vee (x \wedge y)$ where the operators $\vee$ and $\wedge$ are associative and commutative. The question now is whether the existing algorithm for AC-unification for both function symbols can be combined together. This problem has been known as the combination problem in unification theory.

In the following sections we will study an instance of the combination problem, that is the problem of combining the associative unification with unit ($A1$) and associative-commutative unification with unit ($AC1$). Both theories share the same unit element. The application of this combined theory can be found in logic which incorporates noncommutativity such as the one in [Guglielmi]. In Section 2, the basic definitions and concepts underlying unification theory are introduced. In Section 3, we will discuss two combination procedures by [Baader92] and [Ringeissen92]. These methods come with certain requirements on the theories to be combined. In Section 4 we will see that both $A1$ and $AC1$ unification algorithm satisfy these requirements, and thus can be combined.

2 Basic Notions

2.1 Syntactical Unification

Let F be a finite set of function symbols, called *signature*, and V be a countably infinite set of variables. The set of terms built over F and V is denoted by $T(F,V)$. Elements of V will be denoted by x, y, z, and elements of $T(F,V)$ by l, r, s, t, u, v. The set of variables occuring in a term t will be denoted by $\mathrm{Vars}(t)$, and this will be extended to sets of variables, equations, and sets of equations.

A *substitution* is a mapping from variables to terms which is almost everywhere equal to identity, and will generally be represented by σ, θ, η, or ρ. The identity substitution is represented by Id. The application of a substitution σ to a term t, denoted $t\sigma$, is defined by induction on the structure of terms:

$$t\sigma \; = \begin{cases} x\sigma & \text{if } t = x, \\ f(t_1\sigma, ..., t_n\sigma) & \text{if } t = f(t_1, ..., t_n). \end{cases}$$

For a substitution σ, the *domain* is the set of variables

$$\mathrm{Dom}(\sigma) \; = \{x \,|\, x\sigma \neq x\},$$

and the *range* is the set of terms

$$\mathrm{Ran}(\sigma) \; = \bigcup \{x\sigma \,|\, x \in \mathrm{Dom}(\sigma)\},$$

and the set of variables occuring in the range is $\mathrm{VRan}(\sigma) \; = \mathrm{Vars}(\mathrm{Ran}(\sigma))$. A substitution will normally be represented by a set of bindings of variables in its domain, e.g. $\{x_1/s_1, ..., x_n/s_n\}$.

The *restriction* of a substitution θ to a set of variables X, denoted by $\theta_{|X}$, is the substitution which is equal to the identitiy everywhere except over $X \cap \mathrm{Dom}(\sigma)$, where it is equal to σ. *Composition* of two substitution is written $\sigma\theta$, and is defined by $t\sigma\theta = (t\sigma)\theta$. The composition $\sigma\theta$ can be constructed as follows:

- apply θ to every term in $\mathrm{Ran}(\sigma)$ to obtain σ_1 ;
- remove from θ any binding x/t, where $x \in Dom(\sigma)$, to obtain θ_1 ;
- remove from σ_1 any trivial binding x/x, to obtain σ_2;
- take the union of the two sets of bindings σ_2 and θ_1.

Two substitutions are equal, denoted $\sigma = \theta$, if $x\sigma = x\theta$ for every variable x. We say that σ is more general than θ, denoted $\sigma \preceq \theta$, if there exists an η such that $\theta = \sigma\eta$.

Definition 2.1 *A substitution σ is a* unifier *of two terms s and t if $s\sigma = t\sigma$; it is a* most general unifier *(*mgu*), if for every unifier θ of s and t, $\sigma \preceq \theta$. A* unification problem *for two terms s and t is represented by $s \doteq t$.*

2.2 Equational Unification

Equational unification is concerned with the problem of making terms equal modulo an equational theory E by applying a suitable substitution. An *equational theory* is defined by a set of identities E which is a subset of $T(F,V) \times T(F,V)$. We will write $s =_E t$ to denote that the term s is equal modulo E to the term t. In the following sections, we will often refer to the following equational theories:

$$A1 = \{f(x, f(y,z)) \approx f(f(x,y),z),\ f(x,e) \approx x,\ f(e,x) \approx x\}$$

of associative function f with a unit e, and

$$AC1 = A1 \cup \{f(x,y) \approx f(y,x)\}$$

of associative-commutative function f with a unit e. Under these equational theories, for example, we have $f(a, f(b,e)) =_{A1} f(f(a,b),e) =_{A1} f(a,b)$ and $f(a, f(b,x)) =_{AC1} f(x,\ f(b,a))$.

For a given set of identities E, we denote by $\mathrm{Sig}(E)$ the set of all function symbols occuring in E. We sometimes denote the equational theory with E_f where f is the function symbol being defined.

Definition 2.2 *Let E be an equational theory and F a signature containing* $\mathrm{Sig}(E)$*. An $E-$*unification problem *over F is a finite set of equations*

$$\Gamma = \{s_1 \doteq_E t_1, ..., s_n \doteq_E t_n\}.$$

An E-unifier is a substitution σ such that $s_1\sigma =_E t_1\sigma, ..., s_n\sigma =_E t_n\sigma$. The set of all E-unifiers of Γ is denoted by $\mathrm{U}_E(\Gamma)$, and Γ is E-unifiable iff $\mathrm{U}_E(\Gamma) \neq \emptyset$.

Definition 2.3 *Let E be an equational theory and X a set of variables. The substitution σ is* more general modulo E on X *than the substitution θ iff there exists a substitution λ such that $x\theta =_E x\sigma\lambda$ for all $x \in X$. In this case we write $\sigma \preceq_E^X \theta$ and say that θ is an E-instance of σ on X.*

When comparing E-unifiers of a problem Γ, the set X is the set of all variables occuring in Γ. Unlike the case of syntactic unification, unifiable E-unification problems need not have a most general E-unifier. For example, the unification problem $\Gamma = \{f(x,y) \doteq_C f(a,b)\}$ of the theory $C = \{f(x,y) \approx f(y,x)\}$ has two C-unifiers $\sigma_1 = \{x/a,\ y/b\}$ and $\sigma_2 = \{x/b,\ y/a\}$. On $\mathrm{Var}(\Gamma) = \{x,y\}$, any unifier of Γ is equal to either σ_1 or σ_2, and σ_1 and σ_2 are not comparable with respect to $\preceq_C^{\{x,y\}}$.

Definition 2.4 *Let Γ be an E-unification problem over F and let $X = \mathrm{Vars}(\Gamma)$ be the set of all variables occuring in Γ. A* complete set of E-unifiers *of Γ is a set CSU_E of substitutions such that*

- $\mathrm{CSU}_E \subseteq \mathrm{U}_E(\Gamma)$, *i.e., each element of CSU_E is an E-unifier of Γ,*
- *for each $\theta \in \mathrm{U}_E(\Gamma)$ there exists $\sigma \in \mathrm{CSU_E}$ such that $\sigma \preceq_E^X \theta$.*

The set CSU_E is a minimal complete set of E-unifiers *of Γ iff it is a complete set that satisfies*

- *two distinct elements of CSU_E are incomparable w.r.t. $\preceq_E^X$, i.e., for all $\sigma, \sigma' \in \mathrm{CSU}_E$, $\sigma \preceq_E^X \sigma'$ implies $\sigma = \sigma'$.*

The substitution σ is a most general E-unifier of Γ iff $\{\sigma\}$ is a (minimal) complete set of E-unifiers of Γ.

Definition 2.5 *Let E be an equational theory, and let Γ be an E-unification problem over F. The problem Γ has type unitary (finitary, infinitary) iff it has a minimal complete set of E-unifiers of cardinality 1 (finite cardinality, infinite cardinality). If Γ does not have a minimal complete set of E-unifiers, then it is of type zero. We abbreviate type unitary by 1, type finitary by ω, type infinitary by ∞, and type zero by 0, and order these types as follows: $1 < \omega < \infty < 0$. The unification type of E w.r.t. the signature F is the maximal type of an E-equational problem over F.*

The unification properties of an equational theory (like decidability, unification type, etc.) may depend on the set of function symbols involved in the unification problem. We therefore need to distinguish the following types of unification problem.

Definition 2.6 *Let E be an equational theory and Γ an E-unification problem over F.*

- *Γ is an* elementary *E-unification problem iff $F = \mathrm{Sig}(E)$.*
- *Γ is an E-unification problem* with constants *iff $F \setminus \mathrm{Sig}(E)$ is a set of constant symbols.*
- *In a* general *E-unification problem, $F \setminus \mathrm{Sig}(E)$ may contain arbitratry function symbols.*

In addition to the three types of unification problem above, we may want to impose certain constraints to the problem. This is necessary, as we will see later in Section 3, when we want to combine unification algorithms.

Definition 2.7 *An E-unification problem with* linear constant restrictions (lcr) *consists of an E-unification problem with constants, Γ, and a linear ordering $<$ on the variables and free constants occuring in Γ. A substitution σ is an E-unifier of $(\Gamma, <)$ iff it is an E-unifier of Γ that satisfies: $x < c$ implies c does not occur in $x\sigma$ for all variables x and free constants c in Γ.*

3 Combination of Unification Algorithms

In applications of unification theory, we are often faced with the unification problem over some equational theory involving more than one (non-constant) function symbol, each having its own defining identities. For example, the equational theory

$$E = \{\, f(x, y) \approx f(y, x),\ f(x, f(y, z)) \approx f(f(x, y), z),\ g(x, e) \approx e \,\}$$

can be seen as a combination of

$$E_1 = \{f(x, y) \approx f(y, x),\ f(x, f(y, z)) \approx f(f(x, y), z)\}$$

which is an associative-commutative theory, and

$$E_2 = \{\, g(x, e) \approx e \,\}$$

which is an idempotent theory. In case the algorithms for the composing theories exist already, we might wonder if it is possible to combine them to get the algorithm for the combined theory. More precisely, the combination problem for unification algorithms can be stated as follows: "given $E_1, \ldots, E_n$ equational theories, how can algorithms for unification modulo $E_i\,(i = 1, \ldots, n)$ be combined to an algorithm for unification modulo $E_1 \cup \ldots \cup E_n$?".

Unfortunately, it is in general not possible to have a combination algorithm that work in all cases [Baader92a]. There is, however, a subset of unification algorithms which is compositional. Early results on combination problem showed that it is possible to combine unification algorithms for theories that have pairwise disjoint signatures [Baader92]. This was later extended by [Ringeissen92] to the case where the theories share only constants. The latest result allows the combining theories to share constructors [Baader99b]. We will cover here only the first two results as they are sufficient for our problem of combining A1- and AC1-unification algorithms sharing unit.

3.1 Combination of Disjoint Equational Theories

The description of the combination method for disjoint equational theories in this section is based on [Baader92]. The combination algorithm consists of two main steps: decompose the original unification problem into several subproblems in the combining theories, and combine the unifiers obtained in the subtheories into the unifier for the original unification problem.

The following theorems are the two main results on the combination of disjoint unification equational theories [Baader92].

Theorem 3.1 *Let $E_1, \ldots, E_n$ be equational theories over disjoint signatures such that solvability of E_i-unification problems with linear constant restriction is decidable for $i = 1, ..., n$. Then solvability of elementary unification problem is decidable for the combined theory $E_1 \cup ... \cup E_n$.*

Theorem 3.2 *Let $E_1, \ldots, E_n$ be equational theories over disjoint signatures such that all E_i-unification problems with linear constant restriction have finite complete sets of solutions ($i = 1, \ldots, n$). Then the combined theory $E_1 \cup \ldots \cup E_n$ is finitary.*

As stated in Theorem 3.1, the decidability of the unification in the combined theory is restricted to only elementary unification problems. This is not, however, a real restriction, since given any equational theory E we can always decompose a general unification problem (or an unification problem with constants) Γ as an elementary unification problem of the union $E \cup E_\Omega$, where E_Ω is a "dummy" equational theory containing syntactical identities of all free function symbols in Γ. For example, a general A_f-unification problem $\{f(g(x), y) \doteq_{A_f} f(z, a)\}$ can be seen as an elementary unification problem for the theory $A_f \cup E_\Omega$, where $E_\Omega = \{g(x) \approx g(x),\, a \approx a\}$. Since E_Ω is only syntactic equality, it is obvious that the unification problem with linear constant restriction is decidable. Therefore, the result of Theorem 3.1 extends also to the solvability of the general unification problem for the combined theory.

3.1.1 The Decomposition Algorithm

We now show step by step how to decompose a given combination problem into several subproblems and solve them in separate theories. To simplify the presentation of the algorithm, we will restrict ourselves to the combination problem of two theories. The combination of more than two theories can be treated analogously. Furthermore, we restrict ourselves to elementary E-unification, since the general case and the case with constants can be reduced to the elementary one.

We will use the following unification problem throughout this section to accompany the formal description of the decomposition algorithm. Let us consider a theory $E = AC1_{f,e_1} \cup A1_{g,e_2}$, where e_1 and e_2 are units for $AC1$ and $A1$, respectively. For the moment, we assume that e_1 and e_2 are different. We will see the case where $e_1 = e_2$ when we discuss the combination problem of the equational theories sharing constants. Let $\Gamma_0 = \{g(f(x, x), z) \doteq f(y, y),\, g(x, y) \doteq z'\}$ be an elementary unification problems over E. This unification problem is solvable. In particular, the substitution $\sigma = \{x/y,\, z/e_2,\, z'/g(y, y)\}$ is one of the solutions.

A first step to decompose the problem Γ_0 is to "purify" the equations in Γ_0, i.e., we want that each equation in Γ_0 consists only of the signature of one particular theory, in our case it is either $A1$ or $AC1$. For this, we introduce several definitions.

Let E_1, E_2 be two equational theories built over the disjoint signatures F_1, F_2 and let $E = E_1 \cup E_2$ denote their union. A term $t \in T(F_1 \cup F_2, V)$ is called *i-term* iff it is in the form $t = f(t_1, \ldots, t_n)$ for $f \in F_i$ ($i = 1, 2$). A subterm s of a 1-term t is called *alien subterm* of t iff it is a 2-term such that every proper *superterm* of s in t is a 1-term. Alien subterms of 2-terms are defined analogously. An i-term t is called *pure* iff $t \in T(F_i, V)$. An equation $s \doteq_E t$

is pure iff there exists an i, $1 \leq i \leq 2$, such that s and t are pure $i-$terms or variables; this equation is then called an i-equation. Note that an equation of the form $x \doteq_E y$ is both a 1- and a 2-equation.

Consider the equational theory E and the unification problem Γ_0 defined above. We have $E_1 = AC1_{f,e_1}$ and $E_2 = A1_{g,e_2}$. The term $g(f(x,x),z)$ is an 2-term which has $f(x,x)$ as its only alien subterm. The equation $g(f(x,x),z) \doteq f(y,y)$ is not pure. It can be replaced by three pure equations as follows. We replace the alien subterm $f(x,x)$ of $g(f(x,x),z)$ by a new variable x_1. This yields two equations: $g(x_1,z) \doteq f(y,y)$ and $x_1 \doteq f(y,y)$. The first equation is still not pure because on the left hand side we have a 2-term while on the right hand side we have a 1-term. We then replace the term on the right hand side in this equation with a new variable x_2 and produce two pure equations: $g(x_1,z) \doteq x_2$ and $x_2 \doteq f(y,y)$.

We are now ready to describe the decomposition algorithm. The input for the algorithm is an elementary E-unification problem $\Gamma = \{s_1 \doteq_E t_1, \ldots, s_n \doteq_E t_n\}$, where $s_1, \ldots, s_n, t_1, \ldots, t_n \in T(F_1 \cup F_2, V)$. The complete algorithm is as follows.

Step 1 *Variable abstraction.*

$$\frac{\Gamma \cup \{s[x] \doteq t,\, x \doteq u\}}{\Gamma \cup \{s \doteq t\}}$$

where u is an alien subterm of s, x is a new variable, and $s[x]$ denotes the term s with the subterm u replaced by x. This step is repeated until all the terms occuring in the system are pure. It is easy to see that this can be achieved after a finite number of iterations. Applying this step to Γ_0 we have:

$$\frac{\{g(x_1,z) \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}{\{g(f(x,x),z) \doteq f(y,y),\, g(x,y) \doteq z'\}}$$

We now obtain a new system consisting only of pure terms. But there may still exist non-pure equations consisting of a 1-term on one side and a 2-term on the other side.

Step 2 *Split non-pure equation.*

$$\frac{\Gamma \cup \{x \doteq s,\, x \doteq t\}}{\Gamma \cup \{s \doteq t\}}$$

where $s \doteq t$ is a non-pure equation, $s \in T(F_1, V) \setminus V$, $t \in T(F_2, V) \setminus V$, and x is a new variable. As the previous step, this step is performed until all the equations are pure. Continuing our previous example, we have:

$$\frac{\{g(x_1,z) \doteq x_2,\, x_2 \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}{\{g(x_1,z) \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}$$

After this step, we might want to split the the problem into several subproblems in the subtheories and solve them separately. Unfortunately this won't work for the following reason. Taking the example in Step 2, we split the unification problem $\{g(x_1, z) \doteq x_2,\, x_2 \doteq f(y, y),\, x_1 \doteq f(x, x),\, g(x, y) \doteq z'\}$ into two subproblems:

$$\Gamma_a = \{x_1 \doteq_{AC1} f(x, x),\, x_2 \doteq_{AC1} f(y, y)\}$$

and

$$\Gamma_b = \{g(x_1, z) \doteq_{A1} x_2,\, g(x, y) \doteq_{A1} z'\}.$$

It is obvious that $\theta_1 = \{x_1/e_1,\, x_2/e_1,\, x/e_1,\, y/e_1\}$ is a solution for Γ_a and $\theta_2 = \{x_1/e_2,\, x_2/e_2,\, z/e_2,\, z'/g(x, y)\}$ is a solution for Γ_b. However, these two solutions cannot be combined to give a solution for Γ_0 for obvious reason: x_1 is instantiated with two different constants. The problem is that the two subproblems still share variables and these variables are instantiated with incompatible terms. We can avoid the incompatibility of the instantiation of shared variables by ensuring that only one of the subproblem can have a shared variable instantiated. This is achieved through labelling the variable with an apropriate subscript, either 1 or 2, denoting to which theory it belongs (E_1 or E_2). The variable with the label different from its equational theory is treated as a constant.

The labelling solves the problem of incompatible instantiation of shared variables but introduces another problem. The unification problem Γ_a above is solvable only if both x_1 and x_2 can be instantiated, or in other words, they are treated just like normal variables. This means that x_1 and x_2 cannot be instantiated in subproblem Γ_b, i.e., they should be treated as free constants in Γ_b. But then the equation $g(x_1, z) \doteq x_2$ is no longer solvable. This problem can be solved if we identify the variable x_1 with x_2, or formally speaking, we apply an identifying substitution $\{x_2/x_1\}$ to both Γ_a and Γ_b to produce $\Gamma'_a = \{x_1 \doteq_{AC1} f(x, x),\, x_1 \doteq_{AC1} f(y, y)\}$ and $\Gamma_b = \{g(x_1, z) \doteq_{A1} x_1,\, g(x, y) \doteq_{A1} z'\}$. In general, we have to consider all possible identifications of variables in the unification problem.

We now define precisely what is an identifying substitution for a given $A1$-unification problem with constant Γ. Consider a partition of the variables of Γ:

$$P = \{P_1, \ldots, P_n\}$$

where P_i is a non-empty subset of $\mathrm{Vars}(\Gamma)$, $\bigcup P_i = \mathrm{Vars}(\Gamma)$ and $\bigcap P_i = \phi$. For each P_i we choose one of its element x_{P_i} as its representative. The identifying substitution σ correspond to the partition P is defined as follows:

$$\sigma = \{x/x_{P_i} \,|\, x \in P_i, \text{ for all } x \in \mathrm{Vars}(\Gamma)\}.$$

Since the set of all possible partitions of $\mathrm{Vars}(\Gamma)$ is finite, there are only finitely many such identifying substitutions for Γ.

Step 3 *Variable identification.*

$$\frac{\Gamma\sigma_1 \ \ldots \ \Gamma\sigma_n}{\Gamma}$$

where $\{\sigma_1, \ldots, \sigma_n\}$ is the set of all possible identifying substitutions correspond Γ. Continuing our previous example, we consider one particular identiyfing substitution: $\{x_2/x_1\}$.

$$\frac{\{g(x_1, z) \doteq x_1,\, x_1 \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}{\{g(x_1, z) \doteq x_2,\, x_2 \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}$$

Step 4 *Choose theory indices.*

$$\frac{\Gamma_{L_1} \ldots . \Gamma_{L_n}}{\Gamma}$$

where $L_i : \mathrm{Vars}(\Gamma) \mapsto \{1, 2\}$ is a labelling function. In this step we consider all possible labelling of variables. There are only finitely many of them since the set $\mathrm{Vars}(\Gamma)$ is finite. Each variable is labelled with a theory index (either 1 or 2), indicating to which equational theory it belongs ($AC1$ or $A1$, in our example).

Back to our example, let us consider the following labelling function:

$$\begin{array}{ll} L(x_1) = 1, & L(y) = 1, \\ L(x) = 2, & L(z) = 2, \\ L(z') = 2. & \end{array} \tag{1}$$

This means that the variables x_1 and y will be treated as variables in the $AC1$ subproblem, but the same variables will be treated as free constants in the $A1$ subproblem. The reverse applies to the variables x, z and z'. It may seems that we can split the unification problem after this step, but it turns out that we still need to consider the following problem.

Suppose we split the unification problem obtained after Step 4 into two subproblems: $\Gamma_c = \{x_1 \doteq_{AC1} f(x,x),\, x_1 \doteq f(y,y)\}$ and $\Gamma_d = \{g(x_1, z) \doteq_{A1} x_1,\, g(x,y) \doteq_{A1} z'\}$. Now consider two substitutions: $\sigma_1 = \{x_1/f(y,y),\, x/y\}$ and $\sigma_2 = \{z/e_2,\, y/g(x,y),\, z'/g(x, g(x,y))\}$, which are the solutions for Γ_c and Γ_d, respectively. It is impossible to combine σ_1 and σ_2 into a solution for Γ_0, because we have a cyclic dependency between the two substitution, i.e., in σ_1, the variable x is substituted by a term containing y and in σ_2, the variable y is substituted by a term containing x. To sort out this cyclic dependency, we must put a constraint into the subproblems that a certain variable cannot be substituted by a term containing a certain constant. In other words, we introduce an ordering to the set of all variables and free constants in the given equational problem. It turns out that a linear order is sufficient [Baader92]. In our example, we can have an ordering like this:

$$y < x < z < z' < x_1 \tag{2}$$

which means that y cannot be substituted by a term containing x, etc. and thus, the substitution σ_2 is no longer a solution for Γ_d.

Step 5 *Choose ordering .*

$$\frac{\Gamma_{<_1} \ldots\ldots \Gamma_{<_n}}{\Gamma}.$$

This step doesn't change a given system, but adds information concerning the linear ordering on variables. Again, we consider all possible ordering and produce several identical system with different ordering information. Symbols $<_1$ to $<_n$ above corresponds to all possible linear ordering on the variables of Γ.

Step 6 *Split systems.*

$$\frac{(\Gamma_1, \Gamma_2)}{\Gamma}.$$

We now split the system Γ into a pair of systems (Γ_1, Γ_2) such that Γ_1 contains only 1-equations and Γ_2 contains only 2-equations, carrying along the ordering and labelling information. These two systems can now be considered as unification problems with linear constant restriction. Taking our previous example, with the labelling function defined in (1) and the linear ordering 2, we split the unification problem into two subproblems in $AC1$ and $A1$ as follows:

$$\frac{(\{x_1 \doteq f(y,y),\, x_1 \doteq f(x,x)\}, \{g(x_1,z) \doteq x_1, g(x,y) \doteq z'\})}{\{g(x_1,z) \doteq x_1,\, x_1 \doteq f(y,y),\, x_1 \doteq f(x,x),\, g(x,y) \doteq z'\}}$$

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

Proposition 3.1 *The input system Γ_0 is solvable iff there exists a pair (Γ_1, Γ_2) in the output set such that Γ_1 and Γ_2 are solvable.*

The complete proof of this proposition is quite lengthy and will not be given here. Interested readers can consult the original paper by [Baader92].

3.1.2 Combining the Unifiers

We now consider the problem of combining the unifiers produced by the pair of equational systems after Step 6 of the decomposition algorithm. The combined solution σ of the original problem Γ_0 is defined inductively on the linear ordering $<$ chosen in Step 4 of the decomposition algorithm, i.e., we consider all the variables appearing in the ordering starting from the least one. Assume that σ_1 is a solution of Γ_1 and σ_2 is a solution of Γ_2. Without loss of generality, we can assume that the range of σ_i contains only variables with label different than i, or new variables.

Let x be the least variable w.r.t the linear ordering $<$, and let its label be i. Since the solution σ_i of Γ_i satisfies the constant restriction induced by $<$, the term $x\sigma_i$ does not contain any variables with label $j \neq i$ (because any such variables are treated as constants in Γ_i and are all bigger than x w.r.t. $<$). Thus we can simply define $x\sigma := x\sigma_i$.

Let x be an arbitrary variable with label i, and let $y_1, \ldots, y_m$ be the variables with label $j \neq i$ occuring in $x\sigma_i$. We assume that for any variable $z < x$, $z\sigma$ is already defined. Since σ_i satisfies the constant restriction induced by $<$, the variables $y_1, ..., y_m$ must be smaller than x. By our inductive hypotheses, $y_1\sigma, ..., y_m\sigma$ must be already defined. We can define $x\sigma$ now by replacing all y_k ($k = 1 \ldots m$) in $x\sigma_i$ by $y_k\sigma$. Because we have assumed that the other variables occuring in $x\sigma_i$ are new variables, we thus have $x\sigma := x\sigma_i\sigma$.

Finally, let x be a variable of the input system which has been replaced by the variable y during the variable identification step. This means that $y\sigma$ is already defined, and we can simply set $x\sigma := y\sigma$. For all variables z not occuring in the input system we define $z\sigma := z$.

This construction can now be used to generate the complete set of unifiers for the elementary $E_1 \cup E_2$-unification problem. For a given input system Γ_0, let $\{(\Gamma_1^1, \Gamma_2^1), \ldots, (\Gamma_1^n, \Gamma_2^n)\}$ be the output of the decomposition algorithm. For $i = 1, ..., n$ and $j = 1, 2$, let $M_{i,j}$ be a complete set of solutions of the E_j-unification problem with linear constant restriction Γ_j^i.

Proposition 3.2 *The set of substitutions*

$$\bigcup_{i=1}^{n} \{\sigma \mid \sigma \textit{ is the combined solution obtained from } \sigma_1 \in M_{i,1} \textit{ and } \sigma_2 \in M_{i,2}\}$$

is a complete set of $(E_1 \cup E_2)$*-unifiers of the input system* Γ_0.

A detailed proof for this proposition can be found in [Baader92].

3.2 Combination of Equational Theories Sharing Constants

In [Ringeissen92], the problem of combining algorithms of equational theories sharing constants was solved by a simple extension to the decomposition algorithm above. The extension was by taking into account all possible instantiations of variables with shared constants. In addition to the decomposition algorithm, we add the following step before Step 3:

Step 2b Assign to shared constants.

$$\frac{\Gamma\rho_1 \ldots \Gamma\rho_n}{\Gamma}$$

where $\rho_1, \ldots, \rho_n \in \{\rho \mid \mathrm{Dom}(\rho) \subseteq \mathrm{Vars}(\Gamma) \text{ and } \mathrm{Ran}(\rho) \subseteq SC\}$ and SC is the set of all shared constants.

The reason for this instantiation is because we forget some solution otherwise, as shown in the following example.

Example 3.1 *Let $E_1 = AC1_{f,e}$ and $E_2 = A1_{g,e}$ be two equational theories sharing the unit e. Let $E = E_1 \cup E_2$ be the combined theory. The E-equational problem $\Gamma = \{f(x,x) \doteq_E g(y,y)\}$ is solvable, i.e., we just instantiate both x and y with the unit e. But if we run the decomposition algorithm, after Step 2 we will get two equations: $\{f(x,x) \doteq_{E_1} z\}$ and $\{z \doteq_{E_2} g(y,y)\}$, where z is a new variable. The variable z here has to be treated as a free constant in one of the equations, for example in the second one. It is clear that there is no solution for this equation. If we instead instantiate z to e, the two equations would be $\{f(x,x) \doteq_{E_1} e\}$ and $\{g(y,y) \doteq_{E_2} e\}$ which are cleary solvable by instantiating both x and y with the unit e.*

As shown in [Ringeissen92], the Proposition 3.2 still applies for this extended algorithm with some modification, i.e., we compose every solution σ of $\Gamma\rho_i$ with ρ_i to obtain the solution for Γ.

4 Combining A1- and AC1-Unification

We now study the problem of combining algorithms for A1- and AC1-unification sharing unit. More precisely, we want to solve the *unification problem with constant* for the combined theory $A1 \cup AC1$ sharing the unit element. Since the decomposition algorithm described in the previous section works only for the elementary unification problem for the combined theory, we have to translate the problem first to the elementary one. This can be done, as explained before, by adding a "dummy" equational theory containing syntactical identities for all free constants occuring in the equational problem. This translation is trivial and will not be discussed further. We will instead look into details of $A1$ and $AC1$ unification. As we will see later, both equational theories satisfy the requirement of the combination method discussed previously, namely the unification problem with linear constant restriction must be solvable in both theory.

4.1 AC1-Unification

The $AC1$-unification problem has been known to be decidable for the three type of unification problem. The elementary type is decidable in polynomial time, while the unification problem with constant and general unification problem are NP-complete. The elementary $AC1$-unification problem is of type unitary, while the other two are of type finitary [Baader99a].

As shown in [Baader98], the problem of finding the solution for a given $AC1$-unification problem with constants can be reduced to solving linear diophantine equations in the non-negative integers. This method extends also to the $AC1$-unification problem with linear constant restriction, i.e., by adding another diophantine equation corresponds to the linear restriction. Detailed explanation of this extension can be found [Baader91]. Thus, the problem of solving the $AC1$-unification problem with linear constant restriction is decidable.

4.2 A1-Unification

The solution to the A1-unification problem is based on the solution for the A-unification problem. In [Makanin77], a decision procedure was given for determining whether an A-unification problem with constants is solvable or not. A simple algorithm for generating the minimal complete set of unifiers for the A-unification problems was given in [Plotkin72]. Plotkin also showed that all the three types of A-unification problems are infinitary. Since the set of all A-unification problems is subset of the set of all $A1$-unification problems, it is obvious that the $A1$-unification is of type infinitary.

In [Tiu00], an algorithm was given to reduce an $A1$-unification problem with constants to a set of A-unification problems. The idea of the reduction was first mentioned in [Makanin77], that is by guessing non-deterministically in advance which variables that should be instantiated with the unit element. Formally speaking, given an $A1$-unification problem with constants Γ, the set of all reducts of Γ is defined as follows:

$$r(\Gamma) = \{\Gamma\sigma \mid \Gamma\sigma \neq_{A1} e, \, \mathrm{Dom}(\sigma) \subseteq \mathrm{Vars}(\Gamma) \,\text{and}\, \mathrm{Ran}(\sigma) = \{e\}\,\}$$

where e is the unit element in $A1$ theory. Then, by removing all the unit elements in $\sigma\Gamma$, we have an A-unification problem with constants. As it was shown in [Tiu00], this reduction preserves the solvability, i.e., the original $A1$-unification problem with constants is solvable iff there exits at least one of its reducts that is solvable. As a consequence of the decidability of the A-unification, the $A1$-unification problem is decidable. Plotkin's algorithm can also be extended in the similar way to generate a complete set of unifiers for the $A1$-unification.

This extension of A-unification algorithm to the $A1$-unification algorithm works also for the case with linear constant restriction. Since the A-unification with linear constant restriction is known to be decidable [Baader91], we have that $A1$-unification with linear constant restriction is also decidable. Detailed explanation and proofs concerning $A1$-unification can be found in [Tiu00].

5 Conclusion

We have seen one particular instance of the combination problem in unification theory which is the problem of combining algorithms for $A1$- and $AC1$-unification sharing unit. The combination procedure that we have studied here put a restriction on the theories to be combined. The restriction is that the unification problem with linear constant restriction, a slight generalization of the unification problem with constant, must be solvable in the theories to be combined. Both theories $A1$ and $AC1$ fulfill this restriction. However, $A1$ unification is of type infinitary, which means we can not get a terminating combined algorithm that computes the complete set of solutions.

The instance of the combination problem we discuss here is restricted to the unification with constant (or propositional case, in the context of logic and automated reasoning). This is not a restriction since the extension to the general unification (first-order case) is straightforward. A further improvement would be to consider the theories sharing non-constant function symbols. Results in this direction can be found in [Baader99b, Domenjoud94].

This paper discussed only the general idea leaving out the details of implementation. For general data structures and algorithm in the syntactic unification, the interested reader can consult [Baader99a, Baader98]. For $A1$-unification, [Abdulrab92, Abdulrab93] gave the details of the implementation of Makanin's algorithm in LISP. Ideas for the implementation of $AC1$-unification can be found in [Baader98].

References

[Abdulrab92] H. Abdulrab (1992), "Implementation of Makanin's Algorithm", Lecture Notes in Computer Science, Vol. 572, pp. 61-84, Springer-Verlag.

[Abdulrab93] H. Abdulrab (1993), "LOP: Toward a New Implementation of Makanin's Algorithm", Lecture Notes in Computer Science, Vol. 677, pp.133-149, Springer-Verlag.

[Baader99a] F. Baader, W. Snyder (1999), "Unification Theory", Chapter 1, Handbook of Automated Reasoning, ed. Alan Robinson and Andrei Voronkov, Elsevier Science Publishers, B.V. 1999.

[Baader99b] F. Baader, C. Tinelli (1999), "Deciding the Word Problem in the Union of Equational Theories Sharing Constructors", Lecture Notes in Computer Science, Vol. 1631, pp. 175-189, Springer-Verlag.

[Baader99c] F. Baader, K.U. Schulz (1999), "Unification Theory - An Introduction".

[Baader98] F. Baader, T. Nipkow (1998), "Term Rewriting and All That". Cambridge University Press 1998.

[Baader92] F. Baader, K.U.Schulz (1992), "Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures", Lecture Notes in Artificial Intelligence, Vol. 607, pp. 50-65, Springer-Verlag.

[Baader91] F. Baader, K.U.Schulz (1991), "General A- and AX-Unification via Optimized Combination Procedures," Proceedings of the Second International Workshop on Word Equations and Related Topics, Rouen 1991.

[Benanav85] D. Benanav, D. Kapur, and P. Narendran (1985), "Complexity of Matching Problem", Lecture Notes in Computer Science, Vol. 202, pp. 417-429, Springer-Verlag.

[Boudet90] A. Boudet, E. Contejan, and H. Devie (1990), "A New AC-Unification Algorithm with A New Algorithm for Solving Diophantine Equations", in Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science, Philadelphia, USA, pp. 141-150.

[Domenjoud94] E. Domenjoud, F. Klay, C. Ringeissen, "Combination Techniques for Non-Disjoint Equational Theories".

[Guglielmi] A. Guglielmi (1999), "A Calculus of Order and Interaction", Dresden University of Technology Technical Report WV-99-04.

[Jaffar90] J. Jaffar (1990), "Minimal and Complete Word Unification", Journal of the ACM 37(1), 47-85

[Makanin77] G.S. Makanin (1977), "The Problem of Solvability of Equations in A Free Semigroup", Math. Sbornik 103, 147-236. English Translation of Math. USSR Sbornik 32, 1977.

[Plandowski99] W. Plandowski (1999), " Satisfiability of Word Equations with Constants is in PSPACE", in P. Beame, ed., Proceedings of the Fortieth Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, New York City, NY.

[Plotkin72] G. Plotkin (1972), "Building in Equational Theories," Machine Intelligence, 7, 73-90.

[Ringeissen92] C. Ringeissen (1992), "Unification in A Combination of Equational Theories with Shared Constants and Its Application to Primal Algebras", in A. Voronkov, ed., Proceedings of the Conference on Logic Programming and Automated Reasoning, Lecture Notes in Artificial Intelligence, Springer-Verlag, St. Petersburg, Russia, pp. 261-272.

[Schulz93] K.U. Schulz (1993), "Word Unification and Transformation of Generalized Equations", Journal of Automated Reasoning 11, 149-184.

[Tiu00] Alwen Tiu (2000), "A1-Unification". TU Dresden, Germany.