Combination Techniques for Non-Disjoint Equational Theories*

Eric Domenjoud, Francis Klay, Christophe Ringeissen

CRIN-CNRS & INRIA-Lorraine
BP 239, 54506 Vandœuvre-lès-Nancy Cedex (France)
e-mail: {Eric.Domenjoud,Francis.Klay,Christophe.Ringeissen}@loria.fr

Introduction

Modularity is a central problem in automated deduction in equational theories. The problem may be stated as follows: given two equational theories E_1 and E_2 and algorithms for solving a problem P in each theory, can one build automatically an algorithm for solving the problem P in $E_1 \cup E_2$. Almost all results known up to now are restricted to the case where the signatures of E_1 and E_2 are disjoint, in which case, we speak of disjoint theories. Many authors studied the problem of combining unification algorithms for disjoint theories [6, 14, 13] and the best result is due to F. Baader and K. Schulz [1] who described a general technique for combining decision procedures for unifiability and, by the way, solved the problem of combining unification algorithms for non-finitary theories. This result was extended in two different ways: F. Baader and K. Schulz themselves [2] extended it to take into account some restricted forms of disunification problems, and Ch. Ringeissen [10] described an extension where theories may share constants. This last result was, up to now, the only modularity result for non-disjoint equational theories. Some authors also considered the problem of combining matching algorithms for disjoint equational theories [8, 12] which is again more complicated than just putting together two matching algorithms.

We consider here the combination of non-disjoint theories, and give sufficient conditions under which algorithms for solving the word problem, the matching problem and the unification problem in the union may be built automatically from algorithms for each theory. The problem is obviously unsolvable in general, and we restrict our attention to theories sharing constructors which have to be defined in a suitable way. This restriction is however still not sufficient in general. Indeed, some properties like simplicity, which turns out to be very useful for unification, are lost when combining theories sharing constructors. A theory is simple if a term is never equivalent to one of its strict subterms. If we consider the theories presented respectively by $fgx = hx$ and $khx = gx$, which share the constructors h and g according to our definition, we have $kfgx = gx$ modulo their union which is thus not simple. Each of these theories is however obviously simple.

Due to lack of space, almost all proofs are omitted and may be found in the extended version.

* This work has been partially supported by the Esprit Basic Research Working Group-6028 CCL and the GDR Programmation of CNRS.

1 Preliminaries

We first recall some basic definitions and notations about terms, substitutions and equational theories.

$T(F, X)$ denotes the set of terms built over the countable set of function symbols F and the infinite countable set of variables X. A term is ground if it contains no variable. $T(F)$ denotes the set of ground terms built over F. Given a term t and a position p in t, $t(p)$ denotes the symbol at position p in t. The root position is denoted by ϵ so that $t(\epsilon)$ is the top-symbol (or the root) of t. We write $t[s_1, \ldots, s_n]_{p_1,\ldots,p_n}$ to indicate that s_i's ($i = 1, \ldots, n$) are the subterms of t at positions p_i's and if the positions are irrelevant or clear from the context, we simply write $t[s_1, \ldots, s_n]$ to indicate that s_i's are subterms of t. $t[s_1, \ldots, s_n]_{p_1,\ldots,p_n}$ also denotes the term t whose subterm at positions $p_1, \ldots, p_n$ have been replaced with s_i's. $\mathcal{V}(t)$ is the set of variables occurring in t.

A substitution is a mapping from X to $T(F, X)$ which is the identity almost everywhere. It extends to a unique morphism form $T(F, X)$ to $T(F, X)$. If σ is a substitution, the set of variables for which $\sigma(x) \neq x$ is called the domain of σ and is denoted by $\mathrm{Dom}(\sigma)$. The range of σ is defined as $\mathrm{Ran}(\sigma) = \{\sigma(x) \mid x \in \mathrm{Dom}(x)\}$. If the domain of σ is $\{x_1, \ldots, x_n\}$, σ is also written as $\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$. The application of σ to a term t is written in postfix notation as $t\sigma$. If $t = s\sigma$, t is an instance of s. Similarly, the composition of substitutions is written from left to right. If $\mu = \rho\sigma$ for some substitution σ, μ is an instance of ρ and σ is idempotent if $\sigma\sigma = \sigma$.

An axiom is an unordered pair of terms written $l = r$. An equational presentation is a pair (F, A) where F is a finite countable set of function symbols and A is a finite set of axioms $l = r$ with $l, r \in T(F, X)$. For any equational presentation $E = (F, A)$, $=_E$ denotes the equational theory generated by A on $T(F, X)$, that is the smallest congruence containing all instances of axioms in A. F is called the signature of E. By abuse of terminology, we do not distinguish between a theory and its presentation so that we speak of the theory E. Two terms t and s such that $t =_E s$ are said E-equal or equal modulo E. A theory E is consistent if there exists at least one model of E with more than one element. E is collapsing if there exists a non-variable term t and a variable x with $t =_E x$. E is regular if for any terms t and s, $t =_E s \Rightarrow \mathcal{V}(t) = \mathcal{V}(s)$. The union of two equational theories (F_1, A_1) and (F_2, A_2) is the theory $(F_1 \cup F_2, A_1 \cup A_2)$.

A strict ordering $>$ is well founded (or noetherian) if there exists no infinite decreasing sequence $x_1 > x_2 > \cdots$. An ordering $>$ on $T(F, X)$ is monotonic if for any function symbol f and any terms t and s, $t > s \Rightarrow f(\ldots, t, \ldots) > f(\ldots, s, \ldots)$.

Definition 1 (E-normal forms and constructors). Let $E = (F, A)$ be a consistent theory, X be a infinite countable set of variables and $>$ be a monotonic well founded ordering on $T(F \cup X)$ (where variables are treated as constants) such that any congruence class of E in $T(F \cup X)$ contains a unique minimal element with respect to $>$. The minimal element of the class of t modulo E will be denoted by $t{\downarrow_E}$ and will be called the E-normal form of t. A symbol $h \in F$ is a constructor of E if and only if

$$\forall t_1, \ldots, t_n \in T(F \cup X) : h(t_1, \ldots, t_n){\downarrow_E} = h(t_1{\downarrow_E}, \ldots, t_n{\downarrow_E})$$

A constructor h of E occurs at the top in E if there exist terms $t, s_1, \ldots, s_n \in T(F, X)$ with $t(\epsilon) \neq h$ such that $t =_E h(s_1, \ldots, s_n)$.

Two equational theories $E_1 = (F_1, A_1)$ and $E_2 = (F_2, A_2)$ share constructors if all symbols in $F_1 \cap F_2$ are constructors of $(F_1 \cup F_2, A_1)$ and $(F_1 \cup F_2, A_2)$.

A constructor h is shared at the top if it occurs at the top in (F_1, A_1) or (F_2, A_2).

Note: The E-normal forms and the constructors depend highly on the ordering $>$. For instance, if we consider the theory $fx = gx$ then depending on the ordering, either f or g is a constructor. To be completely formal, we should speak of $(E, >)$-normal form and $>$-constructors. Since we always consider the same ordering $>$, we omit to mention it.

Example 1. Let us consider the theories $E_i = (\{+_i, h\}, AC(+_i) \cup \{h(x) +_i h(y) = h(x +_i y)\})$ where $AC(+_i)$ denotes the axioms of associativity and commutativity for $+_i$. We consider the rpo ordering induced by the precedence $+_2 > +_1 > h > \cdots > x_2 > x_1$ with left to right status for $+_i$. h is then a constructor of E_1, E_2 and $E_1 \cup E_2$. Since h occurs at the top in E_1 and E_2, it is shared at the top.

One may note that if E is presented by a convergent rewriting system, our definition of constructors meets the standard one.

The fundamental property of constructors which interests us is the following:

Proposition 2. *Let h be a constructor of a theory E then for any terms t_i's and s_j's,*

$$h(t_1, \ldots, t_n) =_E h(s_1, \ldots, s_n) \iff \forall i : t_i =_E s_i$$

Now let g be another constructor of E (with respect to the same ordering) distinct from h, then for any terms t_i's and s_j's,

$$h(t_1, \ldots, t_n) \neq_E g(s_1, \ldots, s_n)$$

In all the rest of the paper, we only consider consistent theories sharing constructors. The set of shared constructors is denoted by SF. The following theorem justifies our interest in theories sharing constructors.

Theorem 3. *If theories E_1 and E_2 share constructors then $E_1 \cup E_2$ is a conservative extension of E_1 and E_2 and any constructor of E_1 and E_2 is a constructor of $E_1 \cup E_2$.*

Definition 4 (Pure terms and aliens). A term $t \in T(F_i, X)$ is said i-pure (i is 1 or 2). A term $t \in T(SF, X)$ is called a shared term. A strict subterm s of a term t is an *alien* if $s(\epsilon) \in F_i \backslash F_j$ $(i \neq j)$ and all symbols on the path form the root of t to s belong to F_j.

Note that from this definition, a term which is pure may however have aliens if its top-symbol is shared. This seems odd, but turns out to be useful.

Definition 5. A variable-abstraction π is a mapping from $T(F, X)$ to X such that $\pi(t) = \pi(s) \iff t =_E s$. The i-abstraction of a term t, denoted by t^{π_i}, is defined inductively by:

- If $t = f(t_1, \ldots, t_n)$ and $f \in F_i$ then $t^{\pi_i} = f(t_1^{\pi_i}, \ldots, t_n^{\pi_i})$
- else $t^{\pi_i} = \pi(t)$

2 Word problem

The first problem we are faced with when considering the union of theories sharing constructors, is to decide equality in the union. The problem of deciding whether two terms are equal modulo a theory is also called the word problem. It would be nice to have a modularity result stating that the word problem is decidable in the union if it is decidable in each theory. We were not able to establish such a result in general, but we have nevertheless a modularity result for a slightly more general problem which, in some cases, reduces to a word problem. Since two terms are equal modulo E if and only if their E-normal forms are identical, the idea for deciding whether two terms t and s are equal is to compute for each of them some reduced form which *looks like* its E-normal form.

Definition 6 (Layers reduced forms). A term $t \in T(F_1 \cup F_2, X)$ is in layers reduced form if and only if all its aliens are in layers reduced form, t is not equal modulo $E_1 \cup E_2$ to one of its aliens, and either $t(\epsilon) \in SF \cup X$, or t is not equal modulo $E_1 \cup E_2$ to a variable or a term whose top-symbol is a shared constructor.

Layers reduced forms enjoy the following properties:

Proposition 7. *If t is a term in layers reduced form, then $t^{\pi_i} =_{E_i} (t{\downarrow_E})^{\pi_i}$*
If t and s are in layers reduced form then $s =_{E_1 \cup E_2} t \iff s^{\pi_i} =_{E_i} t^{\pi_i}$

As a corollary, we get

Proposition 8. *$E_1 \cup E_2$-equality of terms in layers reduced form is decidable if E_i-equality is decidable for $i = 1, 2$.*

However, the decidability of equality in each theory might not be sufficient to compute a layers reduced form for any term. Therefore we introduce a new kind of problems.

Definition 9 (Symbol matching). Let $E = (F, A)$ be a consistent theory and h be a symbol in F. The symbol matching problem on h modulo E consists in deciding for any term $t \in T(F, X)$ whether there exist terms t_i's such that $t =_E h(t_1, \ldots, t_n)$.

Remarks:

1. Since matching modulo an arbitrary theory E is semi-decidable, t_i's may be effectively computed as soon as we know that they exist. Some general unification procedure [5, 4] may be used for this purpose.
2. the symbol matching problem reduces to a word problem if h is a constant.
3. If the symbol h does not occur at the top in E, then the symbol matching problem on h is trivially unsatisfiable.

Definition 10. Let t be a i-pure term. The term $t{\Downarrow_{E_i}}$ is recursively defined by:

- if $t =_{E_i} x$ for some variable x then $t{\Downarrow_{E_i}} = x$
- else if for some shared constructor h, and i-pure terms $t_1, \ldots, t_n$, $t =_{E_i} h(t_1, \ldots, t_n)$ then $t{\Downarrow_{E_i}} = h(t_1{\Downarrow_{E_i}}, \ldots, t_n{\Downarrow_{E_i}})$.
- else $t{\Downarrow_{E_i}} = t$.

Proposition 11. *For any term $t \in T(F_i, X)$, $t\Downarrow_{E_i}$ is in layers reduced form. Moreover, if E_i-equality is decidable and for any shared constructor h, the symbol matching problem on h is decidable modulo E_i, then $t\Downarrow_{E_i}$ may be computed in finite time.*

We may now give an operational definition of layers reduced forms.

Definition 12. For any term t, the term $t\Downarrow$ is defined recursively by:

- if t is i-pure then $t\Downarrow = t\Downarrow_{E_i}$
- else $t = C[t_1, \ldots, t_n]$ where t_k's are aliens of t. Let $t' = C[t_1\Downarrow, \ldots, t_n\Downarrow] = C'[s_1, \ldots, s_m]$ where s_k's are aliens of t' and variables of t' that do not occur in an alien. Now let $\overline{t}$ be the i-pure term (i may be 1 or 2) obtained by replacing each s_k by a variable, where $E_1 \cup E_2$-equal s_k's are replaced by the same variable, and let ρ be a substitution assigning to each of these variables one of the s_k's it replaces. Now $t\Downarrow = (\overline{t}\Downarrow_{E_i})\rho$.

Proposition 13. *For any term t, $t\Downarrow$ is a term in layers reduced form equal to t modulo $E_1 \cup E_2$. Moreover, if E_1-equality and E_2-equality is decidable and for any shared symbol h, the symbol matching problem on h is decidable modulo E_1 and E_2, then $t\Downarrow$ may be computed in finite time.*

We get then the modularity theorem:

Theorem 14. *Let E_1 and E_2 be two theories sharing constructors. Assume that the word problem is decidable modulo E_1 and E_2 and for any shared constructor h, the symbol matching problem on h is decidable modulo E_1 and E_2. Then the word problem is decidable modulo $E_1 \cup E_2$ and for any shared constructor h, the symbol matching problem on h is decidable modulo $E_1 \cup E_2$*

Corollary 15. *If all constructors shared at the top are constants, and equality is decidable modulo E_1 and E_2 then equality modulo $E_1 \cup E_2$ is decidable.*

3 Matching

The second problem we are interested in is the combination of matching algorithms for theories sharing constructors. The idea for solving this problem is to perform an abstraction of terms, i.e. replace aliens with fresh variables, and then solve pure match-equations in each theory. Unfortunately, purification of match-equations may introduce new variables x in right hand sides and the related solved equations $x =^? s$. The specificity of matching problems is then lost since we have to deal with unification. However, this unification can be turned into matching if purification is only performed in left hand sides of match-equations and solutions of match-equations are ground (when variables of right hand sides are considered as constants) i.e. theories are regular. In all the remaining of this section, we only consider regular theories and we assume that a complete matching algorithm is known for E_1 and E_2.

The combination technique for matching algorithms relies on the computation of a layers reduced form (see section 2) of right hand sides of match-equations. Since we

assume that matching is decidable in each theory, from proposition 13, these layers reduced forms may be computed. By replacing then aliens with free constants (two $E_1 \cup E_2$-equal aliens are replaced with the same constant), a matching algorithm modulo E_i may be used to solve the problem $s \leq^? t$ if s is i-pure and t is in layers reduced form.

The transformation rules for matching problems modulo $E_1 \cup E_2$ are given in figure 1. In rule MatchSolve, $CSS_{E_i}(s \leq^? t\Downarrow)$ denotes a complete set of solutions modulo E_i of $s \leq^? t\Downarrow$.

LeftPurif

$$\frac{\Gamma \wedge s[u] \leq^? t}{\Gamma \wedge s[x] \leq^? t \wedge x =^? u} \quad \textbf{if } u \text{ is an alien of } s \text{ and } x \text{ is a fresh variable}$$

Merge

$$\frac{\Gamma \wedge x \leq^? t \wedge x =^? s}{\Gamma \wedge x \leq^? t \wedge s \leq^? t}$$

MatchSolve

$$\frac{\Gamma \wedge s \leq^? t}{\Gamma \wedge \bigwedge_{k \in K} x_k \leq^? t_k} \quad \textbf{if } s \in T(F_i, X) \backslash X, \{x_k \mapsto t_k\}_{k \in K} \in CSS_{E_i}(s \leq^? t\Downarrow)$$

Delete

$$\frac{\Gamma \wedge x \leq^? t \wedge x \leq^? t'}{\Gamma \wedge x \leq^? t} \quad \textbf{if } t =_E t'$$

Fail

$$\frac{\Gamma \wedge x \leq^? t \wedge x \leq^? t'}{\perp} \quad \textbf{if } t \neq_E t'$$

Fig. 1. Set of rules $\mathcal{RM}$ for matching in the union of regular theories

3.1 Soundness

The following lemma states how to solve a match-equation with an i-pure left hand side, thanks to the E_i-matching algorithm.

Lemma 16. *If s is i-pure, σ is a substitution in E-normal form and t a term in layers reduced form, then $s\sigma =_E t \iff s\sigma^{\pi_i} =_{E_i} t^{\pi_i}$.*

Therefore, we can conclude that a complete set of solutions modulo $E_1 \cup E_2$ of $s \leq^? t\Downarrow$ is obtained from $CSS_{E_i}(s \leq^? (t\Downarrow)^{\pi_i})$. Since equational theories are assumed regular, solutions are ground and so they may be written out as match-equations. Note that MatchSolve must be applied in a non-deterministic way in order to preserve all solutions.

3.2 Completeness

It is easy to check that the normal forms w.r.t $\mathcal{RM}$ are the conjunctions of solved match-equations $\wedge_{k \in K} x_k \leq^? t_k$ or $\bot$ or $\top$. Assume $\star$ is not a conjunction of solved match-equations. If there exists an equation $x =^? s$, then either $x \leq^? t$ is a match-equation in $\star$ and Merge applies or $s[x] \leq^? t$ with $s \notin X$ and either LeftPurif or Matchsolve applies. Otherwise, if there exists a match-equation $s \leq^? t$ with $s \notin X$, then either LeftPurif or MatchSolve applies. Otherwise, there exists two solved match-equations $x \leq^? t$ and $x \leq^? t'$ in $\star$ and either Delete or Fail applies. Thus, a transformation rule in $\mathcal{RM}$ can always be applied to $\star$.

3.3 Termination

Proposition 17. *$\mathcal{RM}$ terminates for any control.*

Sketch of proof. For any problem $\star$, we consider the following complexity measures:

- TS_{mul} is the multiset of theory sizes of non-variable left hand sides of match-equations and non-variable right hand sides of equations in $\star$. The theory size of $t = C[t_1, \ldots, t_n]$ where t_i's are all aliens of t is defined by $TS(t) = 1 + \sum_{i=1}^{n} TS(t_i)$.
- NEQ is the number of equations in $\star$.
- $NMEQ$ is the number of match-equations in $\star$.

These measures are combined lexicographically. The situation is summarized in the table below:

rules	TS_{mul}	NEQ	$NMEQ$
LeftPurif	$\downarrow$		
Merge	$=$	$\downarrow$	
MatchSolve	$\downarrow$		
Delete	$=$	$=$	$\downarrow$

□

We get then the modularity theorem:

Theorem 18. *If E_1 and E_2 are regular theories sharing constructors and a complete and finite algorithm is known for matching modulo E_1 and E_2 then a complete and finite algorithm may be built for matching modulo $E_1 \cup E_2$.*

4 Unification

The last problem we are interested in is the unification problem. As we shall see, this problem is more difficult in the case of theories sharing constructors than in the case of disjoint theories. In order to be able to establish a modularity theorem, we had to restrict our attention to finitary theories in which no non-constant constructor is shared at the top. One should notice that from definition 1, this forbids collapsing theories but not non-regular ones. For the sake of simplicity, we present the algorithm only for regular theories because the non-regular case requires many additional definitions. However, the algorithm remains almost the same since most

of the treatment is encoded in the unification algorithm of each theory. We just have to assume the existence of a more powerful unification algorithm in each theory: namely an algorithm for unification with free function symbols. The formalism we take for designing our combination algorithm is mostly borrowed from A. Boudet [3] and F. Baader & K. Schulz [1] with some modifications due to the special nature of the problem we address.

4.1 Preprocessing and data structure

The algorithm we describe is devoted to the transformation of a conjunction $\star$ of equations into a finite set of solved forms. As in the case of disjoint equational theories, the first step of the algorithm consists in purifying the problem. We first abstract all aliens of terms in $\star$ by fresh variables and add the corresponding equations to the problem. Repeated application of this operation obviously terminates and yields equations between pure terms which are either shared terms or terms with an unshared symbol at the top. Remind that from definition 4, a pure term may have aliens. We could however have equations $t =^? s$ where t is a 1-pure term and s is 2-pure term and none of them is a shared term. Such equations are said impure. In our case, since the theories are not collapsing, if such an equation has a solution σ, there must exist a term u with a shared constructor at the top such that $t\sigma =_E u$ and $s\sigma =_E u$. This means that u is a constant which occurs at the top in E_1 and E_2. The set of such constants will be denoted by SF_0. The equation may thus be split into $t =^? c \wedge s =^? c$ where c is chosen non-deterministically in SF_0. In order to remain complete, we have to collect all problems generated this way. A very similar situation is the case where the problem contains two equations $x =^? t$ and $x =^? s$ where t is a 1-pure term, s is a 2-pure term and none of them is a shared term. Again, a constant c is chosen non-deterministically in SF_0. In order to make the process terminate, we have to replace x with c everywhere in the problem and add the equation $x =^? c$. Once the purification process terminates, in each generated problem $\star'$, we may distinguish three kinds of equations: equations between shared terms, and equations $s =^? t$ between i-pure terms ($i = 1$ or 2) where s and t are either shared terms or terms with an unshared symbol at the top, and s or t is not a shared term. Since shared terms are built only from constructors, the subproblem consisting of the conjunction of all equations between shared terms may be solved in the empty theory, yielding either failure or the minimal idempotent solution σ_0. In the former case, $\star'$ has no solution. In the latter one, we apply σ_0 to the remaining equations and keep it apart from the problem.

4.2 The combination algorithm

Once a problem has been preprocessed it may be written as $\langle \sigma_0; \star_1 \wedge \star_2 \rangle$ where

- σ_0 is an idempotent substitution from X to $T(SF, X)$
- $\star_1$ (resp. $\star_2$) contains the equations $s =^? t$ between 1-pure (resp. 2-pure) terms where s or t is not a shared term.
- No variable in the domain of σ_0 occurs in an equation in $\star_1 \wedge \star_2$.

– For each equation $x =^? t \in \star_1$ (resp. $\star_2$), $\star_2$ (resp. $\star_1$) does not contain an equation $x =^? s$.

Each $\star_i$ will be solved using a unification algorithm for E_i. The following proposition states that doing so we do not loose any solution.

Proposition 19. *Let $s =^? t$ be a i-pure equation and σ a substitution in E-normal form. Then $s\sigma =_E t\sigma \iff s\sigma^{\pi_i} =_{E_i} t\sigma^{\pi_i}$.*

At this point, the main difference between the disjoint and the non-disjoint case is that a variable may get instantiated in both theories without generating a conflict. We avoid this situation by keeping in $\star_i$ only equations of the same form as the ones generated by preprocessing. Solving in one theory could instantiate a variable x by a term of the form $C[t_1, \ldots, t_n]$ where $C[\,]$ contains only constructors and variables and t_i's have unshared symbols at the top. In this case, we abstract t_i's by fresh variables x_i's, replace x everywhere by $C[x_1, \ldots, x_n]$ and add the equations $x_i =^? t_i$ to the problem.

Applying repeatedly resolution modulo E_1 and E_2 will eventually produce problems $\star_1$ and $\star_2$ which are both solved. It could however happen that $\star_1 \wedge \star_2$ is not solved if a compound cycle occurs. A compound cycle is a subproblem of the form $x_1 =^? t_1[x_2] \wedge \cdots \wedge x_{2n-1} =^? t_{2n-1}[x_{2n}] \wedge x_{2n} =^? t_{2n}[x_1]$ where $x_{2i-1} =^? t_{2i-1}[x_{2i}] \in \star_1$, $x_{2i} =^? t_{2i}[x_{2i+1}] \in \star_2$ and no $t_i[x_{i+1}]$ is reduced to x_{i+1} (by convention, $x_{2n+1} = x_1$). Since we are actually only interested in x_i's, we write such a cycle as $\mathcal{C}[x_1, \ldots, x_{2n}]$.

In the disjoint case, if theories are regular, such cycles have no solution. In our case, a compound cycle could be broken by instantiating some x_i with a constant in SF_0. This leads to the rule Cycle in figure 2.

We assume that we know for each theory E_i, a complete and finite unification algorithm which takes as input a problem of the form (P, V) where P is a conjunction of equations and V is a finite set of variables. Solutions of (P, V) are solutions of P which instantiate all variables in V either by a variable or by a constant in SF_0. Such an algorithm may be built if a complete and finite algorithm is known for unification with free constants (For the case of non-regular theories, we would require a unification algorithm with free function symbols). $CSS_{E_i}(P, V)$ denotes the complete set of solutions returned by the algorithm.

At any time during the unification algorithm, we may distinguish three classes of variables in the problem:

1. *Initial variables* which are the variables occurring in the problem before preprocessing.
2. *Abstraction variables* which are variables coming from an abstraction, either during preprocessing or during the algorithm itself.
3. *Introduced variables* which are variables introduced by the unification algorithms for each theory.

We make the very natural assumption that the unification algorithm for each theory may recognize initial, abstraction and introduced variables and never assigns an introduced variable to a non-introduced one or an abstraction variable to an initial one. With this assumption, our combination algorithm will always make an

introduced variable appear in at most one $\star_i$. We may thus also suppose that the domain of each solution does not contain an introduced variable. This does not compromise the soundness of our algorithm.

The combination algorithm is described by the two rules given in figure 2. In the rule $\mathsf{UnifSolve}_i$, ρ_{SF} is obtained by abstracting aliens in the range of ρ by fresh variables. ρ_{F_i} is the substitution such that $x\rho = x\rho_{SF}\rho_{F_i}$ for all $x \in \mathrm{Dom}(\rho)$. $\hat{\rho}_{F_i}$ denotes the conjunction $\wedge_{x \in \mathrm{Dom}(\rho_{F_i})} x =^? x\rho_{F_i}$.

$\mathsf{UnifSolve}_i$

$$\frac{\langle \sigma_0; \Gamma_i \wedge \Gamma_j \rangle}{\langle \sigma_0 \rho_{SF}; \hat{\rho}_{F_i} \wedge \Gamma_j \rho_{SF} \rangle}$$

if

- Γ_i is unsolved
- V_j $(j \neq i)$ is the set of variables instantiated in Γ_j.
- $\rho \in CSS_{E_i}(\Gamma_i, V_j)$

Cycle

$$\frac{\langle \sigma_0; \Gamma \wedge \mathcal{C}[x_1, \ldots, x_{2n}] \rangle}{\langle \sigma_0; \Gamma \wedge \mathcal{C}[x_1, \ldots, x_{2n}] \rangle \{x_i \mapsto c\}}$$

if

- Γ_1 and Γ_2 are solved
- $\mathcal{C}[x_1, \ldots, x_{2n}]$ is a compound cycle
- $c \in SF_0$

Note: Both these rules are non-deterministic. In $\mathsf{UnifSolve}_i$ we must consider each solution in $CSS_{E_i}(\Gamma_i, V_j)$. In Cycle, we must consider all possible choices of the index i and the constant c.

Fig. 2. Set of rules $\mathcal{RU}$ for unification

4.3 Completeness

We have to check that the normal forms w.r.t. $\mathcal{RU}$ are solved forms.

If either $\star_1$ or $\star_2$ is not solved then $\mathsf{UnifSolve}_i$ applies. Otherwise, if $\star_1 \wedge \star_2$ is not solved then it contains a compound cycle $\mathcal{C}[x_1, \ldots, x_{2n}]$ and Cycle applies.

Consequently, a transformation rule in $\mathcal{RU}$ can always be applied to a problem $\star$ if $\star$ is not solved.

4.4 Termination

Proposition 20. *$\mathcal{RU}$ terminates for any control.*

Sketch of proof. For each problem $\langle \sigma_0; \star_1 \wedge \star_2 \rangle$, we consider the following complexity measures which are combined lexicographically.

- NIV is the number of initial variables occurring in $\star_1 \wedge \star_2$
- UIV is the number of initial variables which occur not instantiated in $\star_1 \wedge \star_2$
- NAV is the number of abstraction variables occurring in $\star_1 \wedge \star_2$
- USP is the number of unsolved subproblems $\star_i$

Three cases must be distinguished for the rule $\mathsf{UnifSolve}_i$:

1. If some initial variable is identified with another initial variable or instantiated by a constant in SF_0 or some variable is instantiated by a term whose top-symbol belongs to $SF \backslash SF_0$ then NIV decreases. Indeed, in the last case, this variable is necessarily an initial variable which was not previously instantiated. It is then replaced everywhere by an abstraction of its value. Note also that it is the first case in which NAV may increase.
2. else, if some initial variable is newly instantiated then UIV decreases.
3. else, if an abstraction variable is identified with an initial or abstraction variable or instantiated by a constant in SF_0 then NAV decreases.
4. else, the substitution returned by the unification algorithm for E_i is of the form $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where x_i's are either initial or abstraction variables and each t_i has an unshared symbol at the top. In this case, the substitution ρ_{SF} is the identity. Thus if $\star_j$ $(j \neq i)$ was previously solved, it remains solved so that USP decreases. Furthermore no new abstraction variable is introduced.

For Cycle, since the variable which gets instantiated by a shared constant was previously instantiated, it is either an initial variable in which case NIV decreases, or an abstraction variable, in which case NAV decreases.

The situation is summarized in the table below.

rules	NIV	UIV	NAV	USP
$\mathsf{UnifSolve}_i(1)$	$\downarrow$			
$\mathsf{UnifSolve}_i(2)$	$=$	$\downarrow$		
$\mathsf{UnifSolve}_i(3)$	$=$	$=$	$\downarrow$	
$\mathsf{UnifSolve}_i(4)$	$=$	$=$	$=$	$\downarrow$
Cycle(1)	$\downarrow$			
Cycle(2)	$=$	$=$	$\downarrow$	

□

Theorem 21. *If E_1 and E_2 are regular theories sharing constructors, all constructors shared at the top are constants and a finite and complete algorithm is known for unification with free function symbols modulo E_1 and E_2, then a complete and finite algorithm may be built for unification modulo $E_1 \cup E_2$.*

5 Undecidability results

We exhibit now two families of theories for which no uniform algorithm exists for deciding unifiability. Each theory in these families is the union of two theories sharing constructors in which unification with free function symbols is decidable. However, in each case, one of the conditions of theorem 21 is not satisfied. This shows that weakening these conditions is a difficult problem. The first family shows that there

exists no uniform combination technique for combining unification for simple linear finitary theories sharing non-constant constructors at the top. The second one shows that there exists no general technique for combining unification algorithms for simple linear infinitary theories sharing no constructor at the top. We recall that a theory is simple if no term is equivalent to one of its strict subterms. For each case, undecidability is proved by showing that unification allows to solve a Post correspondence problem:

Definition 22 (Post correspondence problem). Let $\mathcal{A}$ and $\mathcal{C}$ be finite disjoint alphabets, and φ and φ' be two morphisms from $\mathcal{A}^*$ to $\mathcal{C}^*$. The Post correspondence problem for φ and φ' consists in finding a non-empty sequence $\alpha \in \mathcal{A}^+$ such that $\varphi(\alpha) = \varphi'(\alpha)$.

Theorem 23 (Post 1947 [9]). *There exists no uniform algorithm for solving the Post correspondence problem. The problem remains undecidable when φ and φ' are injective.*

In the following, we shall consider families of theories built from $\mathcal{A}$, $\mathcal{C}$, φ and φ'.

5.1 Undecidability of unification in the union of finitary theories sharing non-constant constructors at the top

We consider the theory $E_{\varphi,\varphi'}$ presented by the convergent rewriting system

$$\begin{array}{c}\{f(a_i x, y) \to \omega_i f(x, a_i y) \mid i = 1, \ldots, n\} \cup \{f(\bot, y) \to h(y)\} \\ \cup \{f'(a_i x, y) \to \omega'_i f'(x, a_i y) \mid i = 1, \ldots, n\} \cup \{f'(\bot, y) \to h(y)\}\end{array}$$

where φ and φ' are injective and ω_i and ω'_i denote respectively $\varphi(a_i)$ and $\varphi'(a_i)$. Due to injectivity, $\varphi(a_i)$ and $\varphi'(a_i)$ are non-empty words.

The following proposition in conjunction with theorem 23 shows that there exists no uniform decision algorithm for unifiability modulo $E_{\varphi,\varphi'}$.

Proposition 24. *If unifiability is decidable modulo $E_{\varphi,\varphi'}$ then the Post correspondence problem for φ and φ' is decidable.*

Sketch of proof. Any solution of the unification problem

$$P \equiv \bigvee_i f(a_i x, \bot) =^? f'(a_i x, \bot)$$

is of the form $x \mapsto \alpha\bot$ where $\alpha \in \mathcal{A}^*$ and for some a_i, $\varphi(a_i\alpha) = \varphi'(a_i\alpha)$. Thus P has a solution if and only if the Post correspondence problem for φ and φ' has a solution. □

$E_{\varphi,\varphi'}$ is the union of E and E' presented by the convergent rewriting systems

$$\{f(a_i x, y) \to \omega_i f(x, a_i y) \mid i = 1, \ldots, n\} \cup \{f(\bot, y) \to h(y)\}$$

and

$$\{f'(a_i x, y) \to \omega'_i f'(x, a_i y) \mid i = 1, \ldots, n\} \cup \{f'(\bot, y) \to h(y)\}$$

Moreover, all shared symbols are clearly constructors of both E and E'.

Proposition 25. *E (resp. E') is simple and there exists a complete and finite algorithm for unification modulo E (resp. E') with free function symbols.*

As a corollary, we get

Corollary 26. *There exists no uniform technique for combining unification algorithms for simple linear finitary theories sharing constructors at the top.*

5.2 Undecidability of unification in the union of infinitary theories sharing no symbol at the top

We consider now the theory $E_{\varphi,\varphi'}$ presented by the convergent rewriting system

$$\{f(a_i x, y) \to f(x, \overline{\omega_i} y) \mid i = 1, \ldots, n\} \cup \{f'(a_i x, y) \to f'(x, \overline{\omega'_i} y) \mid i = 1, \ldots, n\}$$

where for any word λ, $\overline{\lambda}$ denotes the word obtained by reversing λ. We do not need anymore to suppose that φ and φ' are injective.

The following proposition in conjunction with theorem 23 shows that there exists no uniform decision algorithm for unifiability modulo $E_{\varphi,\varphi'}$.

Proposition 27. *If unifiability is decidable modulo $E_{\varphi,\varphi'}$ then the Post correspondence problem for φ and φ' is decidable.*

Sketch of proof. Any solution of the unification problem

$$P \equiv \bigvee_i f(a_i x, \bot) =^? f(\bot, y) \wedge f'(a_i x, \bot) =^? f'(\bot, y)$$

is of the form $\{x \mapsto \alpha\bot, y \mapsto \overline{\varphi(a_i\alpha)}\bot\}$ where $\alpha \in \mathcal{A}^*$ and $\varphi(a_i\alpha) = \varphi'(a_i\alpha)$. Thus, P has a solution if and only if the Post correspondence problem for φ and φ' has a solution. □

$E_{\varphi,\varphi'}$ is the union of E and E' presented by the convergent rewriting systems

$$\{f(a_i x, y) \to f(x, \overline{\omega_i} y) \mid i = 1, \ldots, n\} \text{ and } \{f'(a_i x, y) \to f'(x, \overline{\omega'_i} y) \mid i = 1, \ldots, n\}$$

Moreover, all shared symbols are clearly constructors of both E and E'.

Proposition 28. *E (resp. E') is simple, and unification modulo E (resp. E') with free function symbols is infinitary and decidable.*

As a corollary, we get

Corollary 29. *There exists no uniform technique for combining unification algorithms for simple linear infinitary theories sharing constructors, even if no constructor is shared at the top.*

Conclusion and perspectives

We have established modularity results for the word problem, the matching problem and the unification problem in theories sharing constructors. For the case of unification, the result seems rather weak but the undecidability results given in section 5 show that the problem is hard. However it seems possible to extend our results in various directions. As already mentioned, we are actually able to handle non-regular theories sharing constructors, provided that only constants are shared at the top. Even this restriction may be somehow relaxed if finitely many contexts built from constructors are shared at the top [11]. In this case, theory conflicts that occur in the combination algorithm may be solved by introducing explicitely a shared term taken from a finite set. Another extension, which in practice would be very useful, is to combine collapsing theories sharing constructors. This seems very difficult since we are then unable to bound the terms allowing to solve a theory conflict.

Another problem consists in combining decision algorithms for unification. This means that we do not assume that we know for each theory a unification algorithm but only a decision algorithm for unifiability. The combination becomes possible with the very strong assumption that in some sense, an equality step in each theory looks only at finitely many shared symbols [11].

Concerning modularity, one may note that our definition of constructors is actually not really modular in the sense that they are defined by the mean of an ordering on the whole combined theory. It would be nice to be able to define for each theory complete sets of constructors so that theories sharing constructors are defined in a syntactic way. That's to say: (F_1, A_1) and (F_2, A_2) share constructors if $F_1 \cap F_2 \subseteq C_1 \cap C_2$ where C_i is a complete set of constructors of (F_i, A_i). One possible idea would be to define constructors of (F, A) by the mean of a simplification ordering on $T(F)$ such that any congruence class of (F, A) contains a minimal element. If for any F' containing F, we are able to extend such an ordering to a simplification ordering on $T(F')$ such that any congruence class of (F', A) still contains a minimal element, then using a result of Kurihara & Ohuchi [7] we get a simplification ordering for the union of theories such that constructors are preserved.

At last, we could imagine to share non-free constructors that might for instance be defined as constructors modulo another theory. For example, this would allow to share a symbol which is commutative in both theories.

Acknowledgements

We would like to thank the PROTHEO group at Nancy for many fruitful discussions, especially Hélène Kirchner, Claude Kirchner and Michaël Rusinowitch.

References

1. Franz Baader and Klaus Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In *Proceedings 11th International Conference on Automated Deduction, Saratoga Springs (N.Y., USA)*, pages 50–65, 1992.

2. Franz Baader and Klaus U. Schulz. Combination techniques and decision problems for disunification. In Claude Kirchner, editor, *Rewriting Techniques and Applications, 5th International Conference, RTA-93*, LNCS 690, pages 301–315, Montreal, Canada, June 16–18, 1993. Springer-Verlag.
3. A. Boudet. *Unification dans les mélanges de théories équationelles.* Thèse de Doctorat d'Université, Université de Paris-Sud, Orsay (France), February 1990.
4. D. Dougherty and P. Johann. An improved general E-unification method. In M. E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 261–275. Springer-Verlag, July 1990.
5. J. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science*, 67(2-3):203–260, October 1989.
6. Claude Kirchner. *Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles.* Thèse de Doctorat d'Etat, Université de Nancy I, 1985.
7. M. Kurihara and A. Ohuchi. Modularity of simple termination of term rewriting systems with shared constructors. *Theoretical Computer Science*, 103(2):273–282, 1992.
8. T. Nipkow. Combining matching algorithms: The regular case. In N. Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications, Chapel Hill (N.C., USA)*, volume 355 of *Lecture Notes in Computer Science*, pages 343–358. Springer-Verlag, April 1989.
9. E. Post. Recursive unsolvability of a problem of thue. *The Journal of Symbolic Logic*, 12:1–11, 1947.
10. Ch. Ringeissen. Unification in a combination of equational theories with shared constants and its application to primal algebras. In *Proceedings of the 1st International Conference on Logic Programming and Automated Reasoning, St. Petersburg (Russia)*, volume 624 of *Lecture Notes in Artificial Intelligence*, pages 261–272. Springer-Verlag, 1992.
11. Ch. Ringeissen. *Combinaison de Résolutions de Contraintes.* Thèse de Doctorat d'Université, Université de Nancy I, December 1993.
12. Ch. Ringeissen. Combination of matching algorithms. In P. Enjalbert, E. W. Mayr, and K. W. Wagner, editors, *Proceedings 11th Annual Symposium on Theoretical Aspects of Computer Science, Caen (France)*, volume 775 of *Lecture Notes in Computer Science*, pages 187–198. Springer-Verlag, February 1994.
13. M. Schmidt-Schauß. Combination of unification algorithms. *Journal of Symbolic Computation*, 8(1 & 2):51–100, 1989. Special issue on unification. Part two.
14. K. Yelick. Unification in combinations of collapse-free regular theories. *Journal of Symbolic Computation*, 3(1 & 2):153–182, April 1987.