

Faster *Basic Syntactic Mutation* with Sorts for Some Separable Equational Theories

Christopher Lynch¹ and Barbara Morawska²

¹ Department of Computer Science, Box 5815, Clarkson University,
Potsdam, NY 13699-5815, USA

`clynch@clarkson.edu`

² Chair for Automata Theory, Institute for Theoretical Computer Science,
Dresden University of Technology, Germany

`morawska@tcs.inf.tu-dresden.de`

Abstract. Sorting information arises naturally in E-unification problems. This information is used to rule out invalid solutions. We show how to use sorting information to make E-unification procedures more efficient. We illustrate our ideas using Basic Syntactic Mutation. We give classes of problems where E-unification becomes polynomial. We show how *E*-unification can be separated into a polynomial part and a more complicated part using a specialized algorithm. Our approach is motivated by a real problem arising from Cryptographic Protocol Verification.

1 Introduction

The problem of Equational Unification is the problem of finding a substitution that makes two terms equivalent modulo an equational theory [1]. This problem arises in automated theorem proving and formal methods, such as analysis of cryptographic protocols [9]. In traditional formal methods tools for cryptographic protocol analysis, cryptographic algorithms are modeled as a black box. But it is possible to represent properties of algorithms using equational theories. Such approaches are becoming more and more common [2]. Therefore, *E*-unification becomes necessary in the analysis of cryptographic protocols. This paper is inspired by a problem in [12] from cryptographic protocol analysis.

The problem of *E*-unification is undecidable in general. So it is important to find ways to restrict the class of theories to make *E*-unification more efficient. For example, given an equational theory that is saturated under Paramodulation, *E*-unification is in NP using Basic Narrowing [10] and Basic Syntactic Mutation (BSM) [7]. If the equational theory is further restricted, BSM runs in polynomial time and gives a most general unifier. This has been used to show that *E*-unification for an approximation of the commutativity of exponents in the Diffie Hellman protocol runs in polynomial time [6]. This approximation is used in practice in the NRL Protocol Analyzer [8].

This class of theories is restrictive. It would be helpful if we could use real-life properties of the theory under consideration in order to reduce the search space further. Here, we use sorts to define whether a term makes sense. And then terms

that are not well-sorted do not have to be considered in the unification procedure. Sorts are also used to instantiate some variables. For example, a delete operation only makes sense when applied to an item and a set containing that item.

Specifically, we reconsider BSM, and give conditions for which BSM with sorts runs in polynomial time. We also give conditions for which a theory can be separated into two parts such that the E -unification procedure is divided into a polynomial time part and a part that runs a specialized E -unification procedure. The polynomial part can be performed first followed by the specialized procedure, with no interaction between them.

We illustrate our ideas with an example of the key hierarchy theory from cryptographic protocol analysis. This models a protocol where member keys are stored in a tree, along with an associated group key. Member keys can be added to or deleted from the tree. It is possible to access the group key if a member key is known. We can view the tree as a set. This theory is closed under paramodulation, so BSM runs in NP. We show how to use sorts to separate it into a polynomial time procedure yielding a most general unifier, followed by a separate procedure for E -unification modulo the theory of sets.

The paper is organized as follows. In Section 2, we introduce the key hierarchy problem that motivates our idea, and give some observations to show why sorts are needed. In Section 3, we introduce the kind of sorts used in this paper, and the properties they should have. Then we show how certain violations of the properties can be rectified, so that theories that do not obey all the properties can be forced to obey them. In Section 4, we give a sorting procedure, which we use later to provide sorts to unsorted variables. Then we give Sort Expansion inference rules that use that Sort procedure to give sorts to the variables in the equational theory and the goal, and sometimes expand the terms in the goal. In Section 5, we define a separable equational theory as one where the E -unification problem for different theories can be solved separately. In Section 6, we show how the Basic Syntactic Mutation procedure is modified for sorts. In Section 7, we show how sorts are used to make E -unification polynomial for certain theories. Finally, in Section 8 we relate our work to previous work on sorts. Proofs appear in the full version at http://www.clarkson.edu/~clynch/papers/bsmsort_full.ps.

2 Problem

In [7] we have shown that for some equational theories, called *deterministic*, we have a deterministic E -unification algorithm which runs in $O(n^2)$ time. E is *subterm collapsing* if there are terms s and t with s a proper subterm of t , and s equal to t modulo E . We call E *deterministic* if E is not subterm-collapsing, no two equations in E have the same root symbols at their sides, neither t nor s is a variable and if $s \approx t \in E$, then the root symbol of s is not the same as the root symbol of t .

Here we present another set of theories, which can allow subterm collapsing axioms, but also have a very fast E -unification solver. We first explain our ideas on an example of an equational theory of key hierarchy in security protocols¹:

¹ In this example, x , y and M are variables.

$$E = \{pick(x, tree(y, f(x, M))) \approx y \\ add(x, tree(y, M)) \approx tree(y, f(x, M)) \\ delete(x, tree(y, f(x, M))) \approx tree(y, M)\}$$

Here we want to deal with the member keys in a group of users and a group key. A group key is a root of a tree of member keys, and the member keys are leaves of that tree. $tree(y, M)$ means that y is a root of a tree with leaves M . $f(x, M)$ is just a set constructor like $cons(x, M)$ but for sets, and not for lists, hence x is an element and M is a set of elements, i.e. member keys. $pick(x, tree(y, f(x, M)))$ allows us to retrieve a group key if a member key is known. add and $delete$ adds and deletes member keys in a tree.

The theory is closed under Paramodulation², hence *Basic Syntactic Mutation* (BSM) provides an NP-algorithm for E -unification in E . The first step in the BSM-algorithm is to close a given E under the Right-Hand-Side Critical Pair rule³, which in our example yields the following equational theory:

$RHS(E)$

$$= E \cup \{delete(x', tree(y, f(x', f(x, M)))) \approx add(x, tree(y, M)), \\ delete(x, tree(y, f(x, M))) \approx delete(x', tree(y, f(x', M))), \\ pick(x, tree(y, f(x, M))) \approx pick(x', tree(y, f(x', M'))), \\ pick(x, tree(tree(y', f(x', M')), f(x, M))) \approx add(x', tree(y', M')), \\ pick(x, tree(tree(y', M'), f(x, M))) \approx delete(x', tree(y', f(x', M')))\}$$

We can see that $RHS(E)$ is not a deterministic equational theory. It is subterm-collapsing (e.g. $pick(x, tree(y, f(x, M))) \approx y$), the first condition is satisfied, but the second is not ($pick(x, tree(y, f(x, M))) \approx y$) and neither is the third (e.g. $pick(x, tree(y, f(x, M))) \approx pick(x', tree(y, f(x', M')))$).

The situation is still worse if we add axioms defining sets of member keys in order to obtain the full theory for key hierarchy.

$$E^* = E \cup \{f(x, f(y, M)) \approx f(y, f(x, M)), \\ f(x, f(x, M)) \approx f(x, M)\}$$

E^* is not finitely saturated under Paramodulation.

That leads us to some observations.

1. Considering intended interpretation, we can restrict subterms appearing in *pick*-, *add*-, *delete*-, *tree*- and *f*-terms⁴. A term $pick(u, v)$ makes no sense if u is not a member-key, and v a tree, $tree(s, t)$, where s is a group-key, and t a term representing a set of member-keys with u an element of t . Similar restrictions can be imposed on *add*-, *delete*-, *tree*- and *f*-expressions.
2. A *pick*-expression represents an operation on trees which returns a group key; *add*- and *delete*- expressions operate on trees and return a tree.

² See section 6 for definitions of Paramodulation and $RHS(E)$.

³ The rule appears again on page 100.

⁴ An *f*-term is a term with root symbol f .

3. f -expressions are troublesome, because they represent sets, and in order to enable them to properly unify, we have to add identities to E , which destroy its closure under Paramodulation, and inevitably add to the complexity of E -unification⁵. On the other hand we can postpone unification of f -terms until all *pick*-, *add*-, *delete*-, and *tree*- terms are eliminated from the goal. Hence our procedure may have two phases: first – polynomial, and the second one, nondeterministic polynomial, dealing with sets. At any rate, if the goal is ground, this phase can also be made polynomial.

We can look at it semantically: there is a universe of objects (in our example the objects are group-keys, member-keys, trees and sets). Terms in our goal (if it is unifiable) should map to some objects in the universe. But terms which are not well-sorted, do not map to anything in the universe.

The second observation is also justified by this semantic point of view. There are no special objects in our universe such as *pick*, *add* and *delete*. *pick* represents a group-key, which was accessed in a given way, and *add* and *delete* refer to trees. Assuming that our goal is E -unifiable, all its variables map eventually to objects: trees, group-keys, member-keys or sets of member-keys.

3 A Sort Theory Associated with an Equational Theory

The previous section suggests that we should use some sort theory to help us in the process of E -unification by restricting the search space for a solution. Which sort theory to use for this purpose is perhaps mostly decided by semantic information, outside of the possibilities of purely syntactic analysis of equations in E . Nevertheless we can have some requirements which make a sort theory suitable for our purposes. See [13] for the definition of a sort theory.

- **Simplicity.** The first such requirement is that the sort theory is simple, i.e. for every function symbol there is at most one sort declaration in it. For example, for our equational theory we could use the following sort theory:

$$L_E = \{G(\text{pick}(x_M, \text{tree}(y_G, f(x_M, z_S)))), T(\text{add}(x_M, y_T)), \\ T(\text{delete}(x_M, \text{tree}(y_G, f(x_M, z_S)))), T(\text{tree}(x_G, y_S)), S(f(x_M, y_S))\}$$

where G represents group-keys, M represents member-keys, T represents trees and S represents sets. Notice that this sort theory is simple. The notation x_M means that x is of sort M .

- **Consistency with E .** If $u \approx v \in E$, then there are terms s, t and sort declarations in the sort theory $S(s), S(t)$, such that there is a matching substitution σ , with $s\sigma = u$ and $t\sigma = v$. Obviously, L_E is consistent with E or $RHS(E)$.

The next requirement ensures that in each sort there is at least one term E -unifying with all the other terms in this sort.

⁵ We could also apply the inferences modulo the theory of sets, but this also adds to the complexity.

– **Property (*)**

In each sort in a sort theory L , there must be at least one sort declaration with a term u , such that for each other term v in the sort declaration of the same sort, there is a substitution σ with $E \models u\sigma \approx v\sigma$.

Our sort theory L_E satisfies this requirement. It is enough to inspect the equations in E or $RHS(E)$ to see that e.g. any term in the sort T can be chosen to be the term u in the description of the above requirement.

In order to define the fourth requirement, we first define a partial order on the terms appearing in sort declarations. If $S(f(u_1, \dots, u_n), S(g(v_1, \dots, v_m)))$ are sort declarations in L , we say that $f(u_1, \dots, u_n) \geq_L g(v_1, \dots, v_m)$, if g appears at a root position in a proper subterm of $f(u_1, \dots, u_n)$.

Definition 1. *Given a sort theory L satisfying Property (*), for each sort S in L , we call a term u appearing in a sort declaration $S(u)$, such that u is minimal with respect to \geq_L and for each other term v in the same sort, there is a substitution σ with $E \models u\sigma \approx v\sigma$ a **minimal term in the sort S** .*

- **Normal Term Requirement.** We require that for each sort S in L , there is a minimal term in S which does not contain any proper subterm of the sort S , or is of the form $g(x_S)$ and $E \models g(x_S) \approx x_S$. We call such a term, a **normal term in the sort S** .

In L_E , the normal term for the sort T can only be $tree(x_G, y_S)$, because it is the smallest term with the required property in this sort. However, L_E does not satisfy the normal term requirement, because it contains the sort declaration: $G(pick(x_M, tree(y_G, f(x_M, z_S))))$, which is the only sort declaration for the sort G , but the term $pick(x_M, tree(y_G, f(x_M, z_S)))$ does contain y_G which is not the only argument in this term. A similar problem appears with the sort declaration for sets, S . The term in the sort declaration $S(f(x_M, y_S))$ contains a subterm of the same sort.

Definition 2. *Given a sort theory L and an equational theory E , if L satisfies all the above requirements, then L is called a **sort theory associated with E** .*

Now we show how to refine a simple sort theory with Property (*) but not satisfying the normal term requirement for a sort theory associated with E , in such a way that we can have a sort theory satisfying all the requirements.

The normal term requirement is natural, because if we think about sorted terms as terms mapping into a domain of objects, the sort of such terms must be well-defined. In any case we can define a function $g(x)$ such that each term of a given sort is a fix point for this function (hence $g(v) = v$). Notice that if we add the sort declaration $S(g(x_S))$ to L and an equation $g(x) \approx x$ to E , $g(x_S)$ will satisfy requirements for the normal term in the sort S .

So we can modify a simple sort theory L with Property (*) which does not satisfy the normal term condition by adding to it a new function symbol and a sort declaration. At the same time we have to modify the equational theory E in such a way that the new sort theory is associated with the modified E .

Definition 3. 1. Given a sort theory L and an equational theory E , if L has Property (*), but for a sort S there is no term which can be normal in this sort, we add to L a new sort declaration $S(g(x))$, where g is a fresh function symbol, and set $g(x)$ as normal term in the sort S .

2. If L' is a sort theory obtained in 1, and g is a new function symbol added to L in the process of the refinement, we add the equation $g(x) \approx x$ to E .

We call L' obtained in such a way, a **refinement of L** , and E' , which is obtained from E in 2, an equational theory **modified for L'** .

If L is a simple sort theory, then the refined L' is also a simple sort theory. In our example, from L_E we obtain the following refined sort theory:

$$L'_E = \{G(\text{pick}(x_M, \text{tree}(y_G, f(x_M, z_S))))), G(\text{group-key}(x_G)), T(\text{add}(x_M, y_T)), T(\text{delete}(x_M, y_T)), T(\text{tree}(x_G, y_S)), S(f(x_M, y_S)), S(\text{set}(x_S))\}$$

The equational theory E' modified for L' is the following:

$$E' = \{\text{pick}(x, \text{tree}(y, f(x, M))) \approx y \\ \text{add}(x, \text{tree}(y, M)) \approx \text{tree}(y, f(x, M)) \\ \text{delete}(x, \text{tree}(y, f(x, M))) \approx \text{tree}(y, M)\} \cup \\ \text{fixpoint equations: } \{\text{group-key}(x) \approx x, \text{set}(x) \approx x\}$$

As a consequence of Definition 3 we know that if L is a simple sort theory such that L has Property (*) with respect to some equational theory E , and if L' is a refined sort theory obtained from L , and E' is an equational theory modified for L' , then L' is a sort theory associated with E' .

4 Sort – A Procedure to Sort a Goal

We assume we have a sort theory L associated with an equational theory E . We want to use the sort theory for two tasks: to check whether the terms in the goal are not ill-sorted and to assign appropriate sorted terms to the variables in the goal in such a way, that the terms appearing there are sorted. For these purposes we use a unification procedure *Sort* which is defined by the following rules and an arbitrary selection rule⁶.

1. If the procedure is called on a set of equations, we require that on one side of each equation is a renamed subterm from a sort theory.
2. A solved equation is an equation of the form
 - $x \approx v_S$, where x is an unsorted variable from the goal or equational theory, and v is a renamed subterm from a sort declaration of a sort theory, where x doesn't appear anywhere else in the goal, or
 - an equation of the form $u_S \approx v_S$, where both terms are renamed subterms from a sort theory.
3. Apart from the usual syntactic unification rules (Tautology, Orientation, Decomposition, Clash, Cycle), we have the following:

⁶ This procedure was inspired by the one from [13].

(a) **Weakening**

$$\frac{v_S \approx f(v_1, \dots, v_n)}{v_S \approx f(s_1, \dots, s_n), f(s_1, \dots, s_n) \approx f(v_1, \dots, v_n)}$$

where $f(v_1, \dots, v_n)$ is an unsorted term, $S(f(s_1, \dots, s_n))$ is a renamed sort declaration for f -terms, and v_S is a subterm from a renamed sort declaration from the sort theory, of the sort S (possibly a variable of the sort S) which doesn't have f as its root symbol.

(b) **Sorted Fail**

$$\frac{v_S \approx f(v_1, \dots, v_n)}{FAIL}$$

if v_S is a subterm (possibly a variable) from a sort declaration from the sort theory with sort S , $G(f(s_1, \dots, s_n))$ is a sort declaration for f -terms, and $G \neq S$ or there is no sort declaration for an f -term in L .

(c) **Variable Elimination** is applied only to unsorted variables.

We will call this procedure on equations of the form $s \approx t$, where s is a term from a sort theory, and t is an unsorted term. The procedure will return a set of substitutions for variables of s and equations of the form $u \approx v$ where both u and v are sorted. $Sortsub(s \approx t)$ is the substitution determined for the variables of s by the procedure. $Sorteqs(s \approx t)$ is the set of equations with sorted terms on both sides that result from the procedure.

By inspection of the rules of *Sort*, we can make sure that the rules preserve the form of the goal passed to the procedure, as expressed in the following lemma:

Lemma 1. *Given a goal equation where one side is a subterm from a sort declaration then the rules of *Sort* apply deterministically, and the conclusion of each rule is either FAIL or a set of equations, each of which has a subterm from the sort declaration on one side.*

The next lemma states that *Sort* always halts on a goal of the required form.

Lemma 2. *Given a sort theory L and a goal G such that each equation in G has a renamed subterm of a sort declaration on one side then *sort* G halts with a set of solved equations or Fails, in only a linear number of steps.*

Sort-Expansion of E

For each equation $f(u_1, \dots, u_n) \approx g(v_1, \dots, v_m)$ in E , we apply the following:

$$\frac{f(u_1, \dots, u_n) \approx g(v_1, \dots, v_m)}{f(u_1, \dots, u_n)\sigma \approx g(v_1, \dots, v_m)\sigma}$$

where there are sort declarations $S(f(s_1, \dots, s_n))$ and $S(g(t_1, \dots, t_m))$ in L , such that $\sigma = Sortsub(f(s_1, \dots, s_n) \approx f(u_1, \dots, u_n), g(t_1, \dots, t_m) \approx g(v_1, \dots, v_m))$.

We are guaranteed that such σ exists by the fact that L is associated with E . Moreover, we know that for the variables in $f(u_1, \dots, u_n) \approx g(v_1, \dots, v_m)$, σ is a renaming assigning sorts to variables, since L is associated with E and

therefore there is a matcher τ such that $f(s_1, \dots, s_n)\tau = f(u_1, \dots, u_n)$. The original equation in E is replaced by the equation obtained in the conclusion.

For every equation $f(u_1, \dots, u_n) \approx x$ in E , which is not a *fixpoint equation*, we apply the following rule:

$$\frac{f(u_1, \dots, u_n) \approx x}{f(u_1, \dots, u_n)\sigma[x\sigma \mapsto g(v_1, \dots, v_n)] \approx g(v_1, \dots, v_n)}$$

where $S(f(s_1, \dots, s_n))$ is a sort declaration in L such that $\sigma = \text{Sortsub}(f(s_1, \dots, s_n) \approx f(u_1, \dots, u_n))$, obtained by Sort procedure, $x\sigma$ is of sort S , $g(v_1, \dots, v_n)$ is the normal term in sort S and $[x\sigma \mapsto g(v_1, \dots, v_n)]$ is a substitution of a renamed variable $x\sigma$ the normal term in the sort S . (Again, for variables in $f(u_1, \dots, u_n) \approx x$, σ is just a renaming assigning sorts to these variables.)

Notice that in this way we get rid of collapsing equations in E , i.e. equations of the form $x \approx t$ in E , except for the *fixpoint equations*. We will prove that the *fixpoint equations* do not need to be used in the E -unification inferences.

Sort-Expansion of the Goal

For each equation in the goal, apply one of the following rules:

$$\frac{\{f(u_1, \dots, u_n) \approx g(v_1, \dots, v_m)\} \cup G}{\{f(u_1, \dots, u_n)\sigma \approx g(v_1, \dots, v_m)\sigma\} \cup G\sigma \cup eqs}$$

where there are sort declarations $S(f(s_1, \dots, s_n))$ and $S(g(t_1, \dots, t_m))$ in L , $\sigma = \text{Sortsub}(f(u_1, \dots, u_n) \approx f(s_1, \dots, s_n), g(v_1, \dots, v_m) \approx g(t_1, \dots, t_m))$, and $eqs = \text{Sorteqs}(f(u_1, \dots, u_n) \approx f(s_1, \dots, s_n), g(v_1, \dots, v_m) \approx g(t_1, \dots, t_m))$, obtained by Sort procedure.

$$\frac{\{f(u_1, \dots, u_n) \approx x\} \cup G}{\{f(u_1, \dots, u_n)\sigma \approx x\sigma\} \cup G\sigma \cup eqs}$$

where there is a sort declaration $S(f(s_1, \dots, s_n))$ in L , and $\sigma = \text{Sortsub}(f(u_1, \dots, u_n) \approx f(s_1, \dots, s_n))$, $x\sigma$ is of the sort S and $eqs = \text{Sorteqs}(f(u_1, \dots, u_n) \approx f(s_1, \dots, s_n))$, obtained by Sort procedure.

If σ does not exist, we fail, and the goal does not E -unify. If it does exist, we replace the goal with the new one, after application of each rule. We do not apply any expansion rule to an equation which was already expanded or added in the sort-expansion.

We cannot get rid of collapsing equations in the goal, as we did in E , since inference rules may create new collapsing equations. Hence we will have to deal with such equations in the inference rules.

The following definition states the conditions for *Sort* to succeed, i.e. the conditions for *Sort* not to return *Fail*.

Definition 4. *Given a sort theory L , and an equational theory E , we call an equation $u \approx v$ **sortable**, if there are ground substitutions σ and σ' obeying sort restrictions such that all of the following hold*

- if u is not a variable, there is a sort declaration $S(u')$, such that the root symbol of u is the same as the root symbol of u' and if v is not a variable, there is a sort declaration $S(v')$, such that the root symbol of v is the same as the root of v' ,
- $u'\sigma \approx u\sigma$ and $v'\sigma' \approx v\sigma'$, and
- if $x \in \text{dom}(\sigma) \cap \text{dom}(\sigma')$, then $x\sigma$ and $x\sigma'$ are of the same sort.

Theorem 1. *Given a sort theory L associated with an equational theory E , and a sortable equation $f(u_1, \dots, u_n) \approx g(v_1, \dots, v_m)$ (or $f(u_1, \dots, u_n) \approx x$), there exists a set of equations, σ , containing no Fail, which is computed by $\text{Sort}(f(s_1, \dots, s_n) \approx f(u_1, \dots, u_n), g(t_1, \dots, t_m) \approx g(v_1, \dots, v_m))$ (or $\text{Sort}(f(s_1, \dots, s_n) \approx f(u_1, \dots, u_n))$ respectively).*

We have to prove that we do not change an equational theory with respect to a sorted ground domain by the sorted expansion.

Theorem 2. *Given equational theory E and a sort theory L , if E' is obtained by Sort-expansion from E and L is associated with E' , then $E \models u \approx v$, u, v are sorted terms and there is a proof of $u \approx v$ obeying sort restrictions, iff $E' \models u \approx v$.*

As an example, if we use the sort theory:

$$L'_E = \{G(\text{pick}(x_M, \text{tree}(y_G, f(x_M, z_S))))), G(\text{group-key}(x_G)), \\ T(\text{add}(x_M, y_T)), T(\text{delete}(x_M, \text{tree}(y_G, f(x_M, z_S))))), T(\text{tree}(x_G, y_S)), \\ S(f(x_M, y_S)), S(\text{set}(x_S))\}$$

to sort the following equational theory:

$$E' = \{\text{pick}(x, \text{tree}(y, f(x, M))) \approx y \\ \text{add}(x, \text{tree}(y, M)) \approx \text{tree}(y, f(x, M)) \\ \text{delete}(x, \text{tree}(y, f(x, M))) \approx \text{tree}(y, M)\} \cup \\ \text{fixpoint equations: } \{\text{group-key}(x) \approx x, \text{set}(x) \approx x\}$$

we get the following sorted equational theory:

$$\text{Sorted}(E') = \{\text{pick}(x_M, \text{tree}(\text{group-key}(y_G), f(x_M, z_S))) \approx \text{group-key}(y_G) \\ \text{add}(x_M, \text{tree}(y_G, z_S)) \approx \text{tree}(y_G, f(x_M, z_S)) \\ \text{delete}(x_M, \text{tree}(y_G, f(x_M, z_S))) \approx \text{tree}(y_G, z_S)\} \cup$$

fixpoint equations: $\{\text{group-key}(x_G) \approx x_G, \text{set}(x_S) \approx x_S\}$

5 A Separable Equational Theory

At the end of the previous section, we have shown how to sort an equational theory E' , but in the beginning of this paper we started with an equational theory E^* , which contains the troublesome equations defining unification between sets of member-keys.

Hence we have the following:

$$\begin{aligned}
 \text{Sorted}(E^*) = & \{ \text{pick}(x_M, \text{tree}(\text{group-key}(y_G), f(x_M, z_S))) \approx \text{group-key}(y) \\
 & \text{add}(x_M, \text{tree}(y_G, z_S)) \approx \text{tree}(y_G, f(x_M, z_S)) \\
 & \text{delete}(x_M, \text{tree}(y_G, f(x_M, z_S))) \approx \text{tree}(y_G, z_S) \} \cup \\
 & \{ f(x_M, f(y_M, z_S)) \approx f(y_M, f(x_M, z_S)), \\
 & f(x_M, f(x_M, y_S)) \approx f(x_M, y_S) \} \cup \\
 \text{fixpoint equations: } & \{ \text{group-key}(x_G) \approx x_G, \text{set}(x_S) \approx x_S \}
 \end{aligned}$$

As we have noticed already, the additional equations make the theory not be saturated under Paramodulation, hence prevent us from applying the techniques described in the next section.

But we can notice that the “good” equational theory E' is *separable* from the “bad” part of E^* , so that we can solve an E^* -unification problem in two steps: first by using the E' -unification procedure and then tackling the unification in sets. The general criterion for this is stated in the following definition.

Definition 5. *Given a sort theory L and an equational theory E , sorted by L , if $E' \subset E$ such that if $u \approx v$ is in E' , and u and v are of sort S , then there is no subterm in $E \setminus E'$ of sort S , then E' is called separable in E .*

With a goal-directed E -unification procedure, like the one we use in the next section, we can try to solve a goal first with the procedure for E' , where E' is separable. This procedure will return an $E \setminus E'$ -unification problem.

This is obvious, since if u is a sorted subterm, and its sort is in $E \setminus E'$, then no subterms in u can be changed by equations from E' . Especially, no variables in u can be assigned terms of the sorts of the terms from E' .

6 Basic Mutation Refined with Sorts

We now look again into the rules from [7] in order to show that the above analysis can help us to get better E -unification time. Recall that we can apply the Basic Syntactic Mutation technique only in the case when an equational theory is saturated under Paramodulation. In the presence of sorts, we could require that the theory is closed under Paramodulation obeying sorts from a given sort theory L .

Paramodulation

$$\frac{u[s'] \approx v \quad s \approx t}{u[t]\sigma \approx v\sigma}$$

where $\sigma = \text{mgu}(s, s')$, $s\sigma \not\prec t\sigma$,
 \prec is a reduction ordering on terms.

σ in the above rule should obey sort restrictions, and neither of the premises are fixpoint equations.

Since we assume our sort theory to be simple and $\sigma = mgu(s, s')$, σ not obeying sort restrictions, would mean that at least one of the terms is not well-sorted. But we assume that E is well-sorted, hence σ can be just an mgu of s and s' , with no additional regard for sorts.

In Basic Mutation without sorts, $RHS(E)$ was constructed by the RHS Critical Pair rule. In the context of sorts, we also consider sort restrictions:

Right-Hand-Side Critical Pair

$$\frac{s \approx t \quad u \approx v}{s\sigma \approx u\sigma}$$

where $s\sigma \not\approx t\sigma$, $u\sigma \not\approx v\sigma$, $\sigma \approx mgu(v, t)$
and $s\sigma \neq u\sigma$.

σ in the above rule should obey sort restrictions. As in the case of Paramodulation, we should not perform RHS Critical Pair inferences with *fixpoint equations*, which were added to E in sort-expansion. However, if $g(x_S) \approx x_S$ is a fixpoint equation, and a is a constant of sort S , then we must consider the equation $g(a) \approx a$ as a premise in Right-Hand-Side Critical Pair.

In our example (page 98), if E' is sort-expanded and closed under Paramodulation, sorted $RHS(E')$ is the following.

Sorted- $RHS(E')$

$$= E' \cup \{ \text{delete}(x'_M, \text{tree}(y_G, f(x'_M, f(x_M, z_S)))) \approx \text{add}(x_M, \text{tree}(y_G, z_S)), \\ \text{delete}(x_M, \text{tree}(y_G, f(x_M, z_S))) \approx \text{delete}(x'_M, \text{tree}(y_G, f(x'_M, z_S))), \\ \text{pick}(x_M, \text{tree}(y_G, f(x_M, z_S))) \approx \text{pick}(x'_M, \text{tree}(y_G, f(x'_M, z'_S))) \}$$

The Basic Syntactic Mutation (*BSM*) rules with sorts are presented in [7].

The following is true in general for this set of inference rules:

Theorem 3. *Given a simple sort theory, associated with an equational theory E and a well-sorted goal, the inference rules of Sorted BSM preserve well-sortedness of the terms in the goal.*

As an immediate corollary to the definition of sort-expanded equational theory, we show that any rule involving an equation from E with a variable on one side, such as Variable Mutate, is not needed.

Corollary 1. *Given an equational theory sorted by an associated sort theory, and a goal, the inference rule Variable Mutate never applies to a selected goal equation.*

Now we can state formally completeness of BSM rules with sorts for a sorted equational theory and an equational goal.

Theorem 4. *Given an equational theory E , a sort theory L , E' which is E modified for L so that L is associated with E' , such that E is closed under sorted*

Paramodulation, if $E \models G\gamma$ and $G\gamma$ is not solved, then there is a rule in the sorted BSM-rules based on sort-expanded E' , such that $G \longrightarrow G'$ and there is $\gamma' \leq_E \gamma|_{\text{Var}(G)}$ and $E \models G'\gamma'$.

7 Redundancy of Rules and Determinism

We will show that sort restrictions can make some rules in the Sorted BSM procedure redundant. By deleting those rules from the set of our inference rules we obtain a Sorted BSM E -unification procedure, which in some cases is deterministic and polynomial.

BSM solves E -unification in non-deterministic polynomial time for theories saturated under Paramodulation. We will show that Sorted Deterministic BSM will solve it in polynomial time for our key hierarchy example.

Definition 6. A rule R :

$$\frac{s \approx t}{s_1 \approx t_1, \dots, s_n \approx t_n}$$

is redundant in the set of rules S , such that S does not contain R , iff there is a rule R' in S applicable to $s \approx t$, such that $s_1 \approx t_1, \dots, s_n \approx t_n$ is an instance of the conclusion of R' applied to $s \approx t$.

It is easy to check if some rules in the set of rules for BSM with sorts, for each function symbol in a given signature and a given equational theory are redundant. Deleting redundant rules may result in eliminating all choices between applications of different rules or choices of different equations from the equational theory for inferences, and hence in a deterministic procedure. By Corollary 1, we already know that there is no conflict between Mutate and Variable Mutate in Basic Mutation with Sorts, assuming that we have a sort theory associated with a sorted equational theory saturated under Paramodulation.

7.1 Collapsing Goal Equations

First we define a rule called Sort Imitation that is applied eagerly. This rule replaces all rules in BSM , which apply to equations of the form $x \approx v$. This is possible, because of the normal term requirement for a sort theory L associated with E .

Sort Imitation

$$\frac{\{x_S \approx v_S\} \cup G}{\{\boxed{g(v_1, \dots, v_n)_S} \approx v_S[x \mapsto \boxed{g(v_1, \dots, v_n)}]\} \cup G[x \mapsto \boxed{g(v_1, \dots, v_n)}]}$$

where $x_S \approx v_S$ is selected (hence not solved) and $g(v_1, \dots, v_n)_S$ is a normal term in the sort S in a sort theory L associated with E . This rule does not destroy the argument for termination for an E -unifiable sorted goal.

7.2 When Decomposition Is Redundant

We will show that if $f(u_1, \dots, u_n) \approx f(v_1, \dots, v_n)$ is in an equational theory, then Decomposition is often redundant. This is stated formally, with all needed assumptions, in the following theorem:

Theorem 5. *Let E be an equational theory. If L is a sort theory associated with E , $f(u_1, \dots, u_n) \approx f(v_1, \dots, v_n) \in E$, and all shared variables in $f(u_1, \dots, u_n)$ and $f(v_1, \dots, v_n)$ are at the same positions in these two terms, then the Decomposition rule for f -terms is redundant in the set of rules of Sorted-BSM.*

7.3 Deterministic BSM with Sorts

We are now ready to state the conditions which make our E -unification procedure with sorts terminating and deterministic.

Definition 7. *Given a sort theory L , an equational theory E is called L -deterministic if:*

1. *no two equations in E have the same root symbols at their sides,*
2. *L is associated with E .*

We show that if $E'' = RHS(E')$, E' is finite and saturated by Paramodulation, and there is a sort theory L associated with E , which is E'' modified for L , then if E is also L -deterministic, then BSM can be turned into a deterministic algorithm, which means that it will halt in a linear number of steps.

We define *Sorted-BSMd* as a deterministic version of *Sorted-BSM*, based on any selection rule and the rules in Figure 1.

Theorem 6. *Let $E'' = RHS(E')$, such that E' is finite and saturated by Paramodulation, and let L be a sort theory associated with E , which is E'' modified for L . Then if E is L -deterministic, the algorithm Sort-BSMd solves a goal G deterministically in polynomial time. E with sorts is then unitary.*

8 Conclusion

We have shown how sorts are necessary in real problems of E -unification. The example motivating our work is the key hierarchy theory from Cryptographic Protocol Analysis. Reasoning modulo equational theories is currently an important topic in Cryptographic Protocol Analysis, and so our results are applicable there. But the characteristics of our example problem are general, and apply to many E -unification problems from verification, such as modeling data structures with equational theories. In our example, we showed that the E -unification problem can be divided into two stages: the first stage runs in polynomial time, and the second stage can be solved solely by reasoning modulo the theory of sets. We believe that many natural problems have this same structure.

Our use of sorts has some differences with other uses of sorts. We deal with sorts in the context of E -unification. Much of the previous work considers sorts with syntactic unification, although there is some work dealing with equational

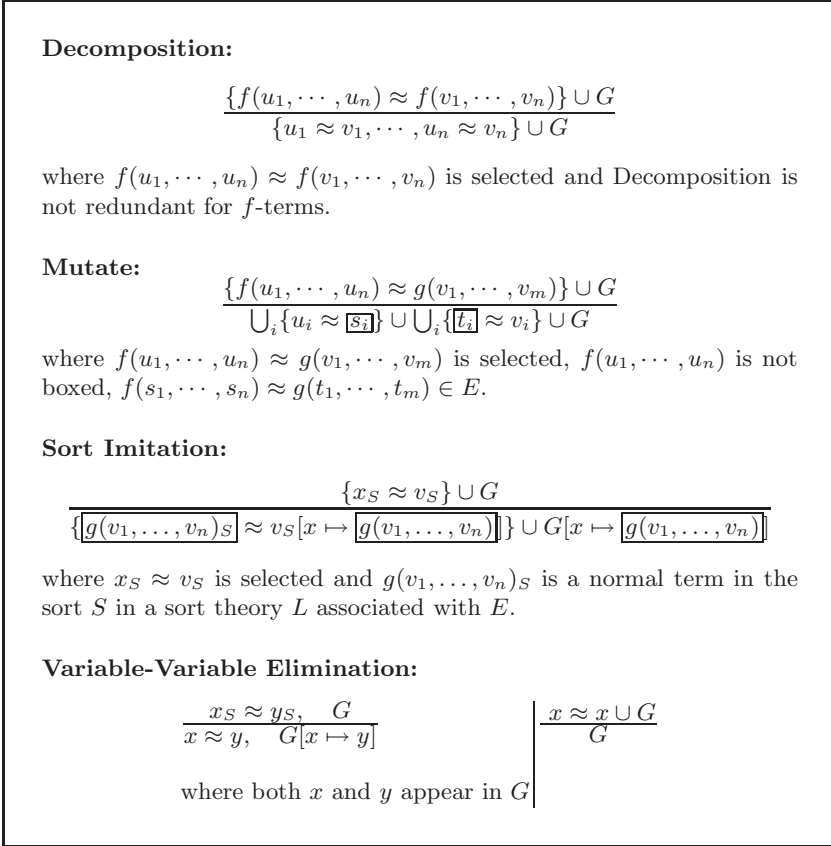


Fig. 1. Sorted *BSMd* Inference Rules.

theories[3–5, 11]. However, there are other ways where our work differs. We do not consider that the equational theory and the goal are sorted. Instead, we have procedures that give sorts to the variables in the equational theory and the goal. We require that the sort theory is simple. Much other work only requires that the sort theory is order-sorted. Deletion requires that an item can only be deleted from a set if it is in the set. We could not capture this property in an order-sorted theory. However, we do not try to deal with subsorts. We also require certain properties of the equational theory, such that the terms of the equations are instances of the sorts, and each sort must have some normal form that is E -unifiable with everything else in the sort.

The properties we require naturally occur in many real life problems. We showed in this paper that these properties imply a more efficient algorithm for Sorted E -unification. Although our approach differs from other approaches on sorts, we also have some similarities. For example, our Sort algorithm is inspired by the Syntactic Sorted Unification procedure of [13], however we use it mainly to give sorts to unsorted variables. Since we do not need to unify sorted terms, it makes the algorithm more efficient.

The results of this paper should lead to more efficient E -unification procedures, because it bases the procedure more closely on the meaning of the terms. We have also begun applying these ideas to Narrowing.

References

1. F. Baader and W. Snyder. Unification Theory. In J.A. Robinson and A. Voronkov, editors, Handbook of Automated Reasoning, volume I, pages 447–533. Elsevier Science Publishers, 2001.
2. H. Comon Intruder Theories (Ongoing Work). In Proceedings of the 7th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'04), Barcelona, Spain, March 2004, LNCS 2987, pages 1-4. Springer.
3. J. A. Goguen and J. Meseguer. Order-Sorted algebra I: equational deduction for multiple inheritance, overload, exceptions and partial operations. *Theoretical Computer Science*, 105, 217–273, 1992.
4. C. Hintermeier, C. Kirchner and H. Kirchner. Dynamically-Typed Computations for Order-Sorted Equational Presentations. In S. Abiteboul and E. Shamir (eds.): *Automata Language and Programming: 21st International Colloquium, ICALP'94*, Vol. 820 of LNCS, pp. 450–461, Springer, 1994.
5. J. P. Jouannaud and C. Kirchner. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. In J. Lassez and G. Plotkin (eds.): *Computational Logic, Essays in Honor of Alan Robinson*. MIT Press, Chap. 8, pp. 257–321, 1991.
6. C. Lynch and C. Meadows Sound Approximations to Diffie-Hellman Using Rewrite Rules. In Proceedings of the Sixth International Conference on Information and Communications Security. Malaga, Spain. October, 2004.
7. C. Lynch and B. Morawska. Basic Syntactic Mutation. In Proceedings of Conference on Automated Deduction (CADE), Vol. 2392 of LNAI, 471–485, 2002.
8. C. Meadows Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer, Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, May 1999.
9. C. Meadows. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends, IEEE Journal on Selected Areas in Communication, Vol. 21, No. 1, pp. 44-54, January 2003.
10. R. Nieuwenhuis. Decidability and Complexity Analysis by Basic Paramodulation. *Information and Computation*, 147:1-21, 1998.
11. G. Smolka, W. Nutt, J. A. Goguen and J. Meseguer. Order-Sorted Equational Computation. In H. Ait-Kaci and M. Nivat (eds.): *Resolution of Equations in Algebraic Structures*, Vol. 2 of *Rewriting Techniques*. Academic Press, Chap. 10, pp. 297–267, 1989.
12. D. Wallner, E. Harder, and Ryan C. Agee. Key Management for Multicast: Issues and Architectures, RFC 2627, June 1999.
13. C. Weidenbach. Sorted Unification and Tree Automata in Bibel W. and Schmitt P. H. , editors, Automated Deduction – A Basis for Applications, Volume 1 of Applied Logic, Chapter 9, Kluwer, pp. 291-320, 1998