# ORDER-SORTED
# RIGID E-UNIFICATION

by

**Jean H. Gallier**
Computer Science Department
Moore School of Engineering
Univeristy of Pennsylvania
Philadelphia, PA 19104

and

**Tomás Isakowitz**
Information Systems Department
Leonard N. Stern School of Business
New York University
New York, New York 10003

**December 1991**

Center for Research on Information Systems
Information Systems Department
Leonard N. Stern School of Business
New York University

**Working Paper Series**

STERN IS-91-40

## Abstract

Rigid E-Unification is a special type of unification which arises naturally when extending Andrew's method of matings to logic with equality. We study the rigid E-Unification problem in the presence of subsorts. We present an order sorted method for the computation of order sorted rigid-E-unifiers. The method is based on an unsorted one which we refine and extend to handle sort information. Our approach is to incorporate the sort information within the method so as to leverage it.We show via examples how the order sorted method is able to detect failures due to sort conflicts at an early stage in the construction of potential rigid E Unifiers. The algorithm presented here is NP-complete, as is the unsorted one. This is significant, specially due to the complications presented by the sort information.

# 1 Introduction

Rigid E-Unification is a special type of unification that occurs when extending Andrews [And81] method of matings to include equations. It was first introduced by Gallier, Raatz and Snyder [GRS87]. Gallier, Narendran, Plaisted and Snyder [GNPS90] show that the problem is NP-complete and they present a method for finding rigid $E$-unifiers. We extend their work to order-sorted logic [Gog78, GM87b]. This is of interest because the order-sorted framework can be utilized to provide a formal framework for the treatment of such important concepts as *inheritance* and *overloading.* The results we present in this paper are significant from two different perspectives. Firstly, we improve upon the unsorted rigid E-unification method by simplifying it and secondly, we construct an inherently order-sorted method which takes sort information into consideration in each one of its phases; and produces order-sorted unifiers.

The concept of an Order-Sorted Algebra was introduced by Goguen in [Gog78]. Goguen and Meseguer [GM87b] present order-sorted algebras as the natural semantics for order-sorted logic. Order-sorted algebras are based on an approach similar to many-sorted algebra where families of functions are associated with each function symbol. Eqlog [GM84] is a programming language with built-in overloading and inheritance that has a clean mathematical semantics based on order-sorted algebra. Inheritance is achieved via subsorts. There are other similar semantic approaches to subsorts, e.g. Smolka [Smo86], Walther [Smo86] among others. The principal differences lie in the treatment of overloaded operators in the underlying algebraic structure.

A significant advantage of the order-sorted approach over the unsorted one lies in the efficiency of computations. Sort information can be embedded within the algorithms. For example, there is an order-sorted unification algorithm that is able to trim the search space by taking sort information into consideration. These order-sorted algorithms are not just simple extensions of their unsorted counterparts; they require original approaches to the issues at stalk.

The problem of rigid-E-unification arises when extending Andrews' method [And81] of matings to first order logic with equality. Extending matings to order-sorted matings implies an order-sorted version of rigid E-unification. Thus, the work we present here adapts and extends the unsorted methods to the order-sorted case.

Rigid Unification involves finding a solution $\theta$ to a term equation using only a limited resource of axioms. The number of times the axioms in $E$ are used is not restricted, what is restricted is the number of variations of such axioms. This is done by freezing the variables in $\theta(E)$ and treating them as constants as if $E$ were a set of ground equations. It can be stated as the following problem.

**Problem.** Given a finite set $E = \{u_1 \doteq v_1, \ldots, u_n \doteq v_n\}$ of equations and a pair $\langle u, v \rangle$ of terms, is there a

equations, $\theta(u) \overset{*}{\cong}_{\theta(E)} \theta(v)$, that is, $\theta(u)$ and $\theta(v)$ are congruent modulo $\theta(E)$ (by congruence closure)?

The substitution $\theta$ is called a *rigid* $(\Sigma, E)$*-unifier of* $u$ *and* $v$.

**Example 1.1** Let $E = \{g(f(z_1)) \doteq f(z_1), g(f(z_2)) \doteq q(z_2)\}$ and $u = q(z_3)$ and $v = f(z_4)$. Then any substitution $\theta$ unifying $< z_1, z_2, z_3, z_4 >$ is a rigid-E-unifier of $u$ and $v$ because

$$\theta(q(z_3)) = q(z) \;{}_{\theta(g(f(z_3))\doteq q(z_3))}\longleftarrow g(f(z)) \longrightarrow_{\theta(g(f(z_4))\doteq f(z_4))} f(z) = \theta(v)$$

where $z$ is the common value of $\theta'(z_1) = \theta'(z_2) = \ldots$.

Only a single instance of each equation in $E$ can be used, and in fact, these instances $\theta(u_1 \doteq v_1), \ldots, \theta(u_n \doteq v_n)$ must arise from the *same* substitution $\theta$. Also, once these instances have been created, the remaining variables (if any) are considered rigid, that is, treated as constants, so that it is not possible to further instantiate these instances.

**Example 1.2** Let $E = \{f(x) \doteq x\}$, consider rigid E-unifying $u = g(f(a), f(b))$ and $v = g(a, b)$. The simple solution of substituting $a$ for $x$ to rewrite $g(f(a), f(b))$ to $g(a, f(b))$ and then using $f(x) \doteq x$ again with $b$ for $x$ does not work out because we are using two different instances of $f(x) \doteq x$.

Notice that there is no way $f(a)$ can be rewritten to $a$ without binding $x$ to $a$. Similarly, in order for an equality step to be applicable to $f(b)$, $x$ has to be bound to $b$. This is precisely why the two terms are **not** rigid E-unifiable. However, if we consider $E' = \{f(x) \doteq x, f(y) \doteq y\}$ then $\theta = [x/a, y/b]$ is a rigid E-unifier of $u$ and $v$.

Hence rigid $(\Sigma, E)$-unification differs from $(\Sigma, E)$-unification in that in the latter a proof of $\theta(u) \doteq \theta(v)$ from $E$ might involve the use of different instantiations of the same equation in $E$. In the rigid case however, only the instances $\theta(E)$ (regarded as ground) can be used. It is interesting to observe that the solution to the rigid unification problem involves the use of the congruence closure, rewriting and term unification. We develop an order-sorted method for finite signatures which is also in NP. Since this type of unification forms the core of *equational matings*, it sets a precedent for the development of an extension to Andrews' method of Matings to the order-sorted equational case. Gallier, Narendran, Plaisted and Snyder in [GNPS90] provide an NP procedure to generate complete sets of unsorted rigid $E$-unifiers. Our task is to provide a method that produces order-sorted rigid $E$-unifiers (rigid $(\Sigma, E)$-unifiers where $\Sigma$ is an order-sorted signature.) We could take the following approach:

1. Run the unsorted algorithm to produce an unsorted rigid $E$-unifier $\theta$, and then
2. using sort information try to produce for each unsorted $\theta$ obtained in step 1, a family of sort assignments that results in a family of $\Sigma$-substitutions for $\theta$.

The disadvantage of this approach is that it does not make full use of the sort information. For example, if $u$ and $v$ have no common subsort, then $u$ and $v$ can not have a rigid $(\Sigma, E)$-unifier. However, the method described above would first run the $NP$ unsorted algorithm; then try to compute a family of sort assignments and finally, upon discovering that the family of sort assignments is empty, return **failure**.

The approach we take here however, differs in that the method itself is intrinsically order-sorted. We modify the unsorted method for finding rigid $E$ unifiers to a method that builds order-sorted substitutions. Since the sort information is used at each and every step of the order-sorted algorithm, it is more effective than the method described above because it is able to detect failure due to sort conflicts at an earlier stage. Our method uses an algorithm for finding order-sorted unifiers in *triangular form* presented in [Isa89] based on work by Meseguer, Goguen and Smolka [MGS89].

*Order assignments* constitute a significant component of the unsorted rigid E-unification method presented by Gallier, Narendran, Plaisted and Snyder in [GNPS90]. Without entering into too much detail, order assignments represent guesses on the ordering a ground rigid E-unifier will impose on terms. This ordering is used to guess other aspects of the solution. Although this concept is quite interesting, it complicates the method and its proof. By extending a procedure by Snyder [Sny89] that finds interreduced sets of rewrite rules equivalent to a system $E$ of equations, we manage to eliminate order assignments from the method (this works as well for the unsorted version of rigid E-unification).

Thus, there are significant differences between the unsorted and the order-sorted versions of the rigid $E$-unification method such as:

- Use of sort information at each and every step of the algorithm.
- Use of general equations to avoid hitting ill-typed terms.
- At the heart of the method we use an order-sorted unification algorithm which does not return an mgu, but a member of a complete family of $\Sigma$-unifiers. Since we are restricting ourselves to finite signatures, this family is finite. The order-sorted unification method is an extension of the one in [MGS89] as described in section 4. Even though $\Sigma$-unification with no equations is NP-complete, we manage to obtain an NP algorithm for rigid $(\Sigma, E)$-unification.
- As described above we avoid using *order assignments*. This requires a different method and different proofs which are simpler.
- We show that a rigid $E$-unifier can be obtained by a sequence of guesses. This is a consequence of the removal of order assignments.

Thus, our method solves the rigid E-unification problem for order-sorted general equation systems and also represents substantial improvements over the unsorted method.

This paper is organized as follows. In section 2 we provide some background on order-sorted algebras. We describe general equations, the particular class of equations to which our results on rigid $(\Sigma, E)$-unification do apply, in section 3. The concept of unification for order-sorted terms is reviewed in section 4 where we also present some interesting results on triangular forms for both unsorted and order-sorted unifiers. In section 5 we formally describe the rigid $E$-unification problem and give some general remarks about the method, which is developed in sections 6 through 9. Complete sets of rigid $(\Sigma, E)$-unifications are explored in section 6, and minimal sets of rigid $(\Sigma, E)$-unifications are studied in section 7. An important aspect of our method is that sets of order-sorted equations can be transformed into reduced sets of rewrite rules in polynomial time. These results are exhibited in section 8. The actual method and its correctness proof are given in section 9. Section 10 proves that the method given is in fact in NP. In section 11 we summarize our results and discuss directions for further research.

# 2 Order-Sorted Algebra

Order-Sorted Algebras are presented by Goguen and Meseguer [GM87b] as the natural semantics for Order-Sorted logic. There are other approaches, e.g. Smolka [Smo86], Walther [Smo86] among others. The principal difference lies in the treatment of overloaded operators and the underlying algebraic structure.

Order-Sorted Algebras are based on an approach similar to Many-sorted Algebra where families of functions are associated with each function symbol. The principal idea is to interpret the subsort relation as inclusion of domains. That is, if $s$ is a subsort of $s'$ then the domain of discourse $A_s$ assigned to $s$ is a subset of $A_{s'}$, the domain of $s'$. Similarly, function symbols are interpreted as functions between the domains of discourse, and certain natural relations hold between the interpretations of an overloaded function symbol.

## 2.1 Signatures

We shortly review the elements of many-sorted algebra. Given an *index* set $S$, an *S-sorted set* $A$ is just a family $(A_s)_{s\in S}$ of sets, one set $A_s$ for each $s \in S$. Similarly, given two $S$-sorted sets $A$ and $B$, an $S$-sorted function $f : A \mapsto B$ is an $S$-indexed family $(f_s : A_s \mapsto B_s)_{s\in S}$ of functions $f_s : A_s \mapsto B_s$, and an $S$-sorted relation $R$ is an $S$-indexed family $(R_s)_{s\in S}$ of relations $R_s \subseteq A_s \times B_s$. Let us assume a fixed set $S$ called the *sort set*, with a partial order $\leq$.

**Definition 2.1** A many-sorted signature is defined as a triple $(S, \Sigma, \rho)$, where $S$ is a sort set and $\rho : \Sigma \to 2^{S^* \times S}$ is a *rank function* assigning a set $\rho(f)$ of *ranks* $(w, s)$ to each symbol in $\Sigma$. The elements of the sets $\Sigma$ are called operators or function symbols. The set $\Sigma$ can be viewed as an indexed family if for every $(w,s) \in S^* \times S$ we let $\Sigma_{w,s} = \{f \in \Sigma \mid (w,s) \in \rho(f)\}$.

Note that $\Sigma_{w,s}$ and $\Sigma_{w',s'}$ are not necessarily disjoint, since a symbol in $\Sigma$ may have several ranks. Whenever convenient, we omit the function $\rho$, and view $\Sigma$ as family of sets $(\Sigma_{w,s})_{(w,s)\in S\times S^*}$.

**Definition 2.2** An order-sorted signature is a quadruple $(S, \leq, \Sigma, \rho)$, such that $(S, \Sigma, \rho)$ is a many-sorted signature and $(S, \leq)$ is a partially ordered set.
In addition the following *monotonicity condition* is imposed to rule out bizarre models :

if $f \in \Sigma_{w_1,s_1} \cap \Sigma_{w_2,s_2}$, and if $w_1 \leq w_2$ then $s_1 \leq s_2$.

When the sort set $S$ is clear, we write $(\Sigma, \rho)$ or $\Sigma$ for $(S, \Sigma, \rho)$. Similarly when the partialy ordered set is clear, we write $(\Sigma, \rho)$ or $\Sigma$ for $(S, \leq, \Sigma, \rho)$.

For function symbols, we may write $f : w \mapsto s$ when $(w,s) \in \rho(f)$ to emphasize that $f$ denotes a function with arity $w$ and co-arity $s$. An important case occurs when $w = \lambda$, the empty string; then $f$ denotes a constant of sort $s$. When $(\omega, s) \in \rho(f)$ we will also say that $f$ has arity $\omega$ and co-arity $s$.

**Example 2.3** Let the set of sorts be $S = \left\{\mathbf{zero}, \mathbf{Q}^+, \mathbf{Q}\right\}$, and let the partial order be: $\mathbf{zero} \leq \mathbf{Q}, \mathbf{Q}^+ \leq \mathbf{Q}$.
The following is an order-sorted $\Sigma$-signature which we denote by **Rationals**:

- $\Sigma_{\lambda,\mathbf{zero}} = \{\mathbf{0}\}$;
- $\Sigma_{\mathbf{Q}.\mathbf{Q},\mathbf{Q}} = \{+\}$; and
- $\Sigma_{\mathbf{Q}.\mathbf{Q}^+,\mathbf{Q}} = \{/\}$

Figure 1 graphically depicts this signature. The constant $\mathbf{0}$ is of sort **zero**. Notice that the second argument of / is of sort $\mathbf{Q}^+$, which is intended to exclude zero. Hence we are formalizing the idea of disallowing a division by zero.

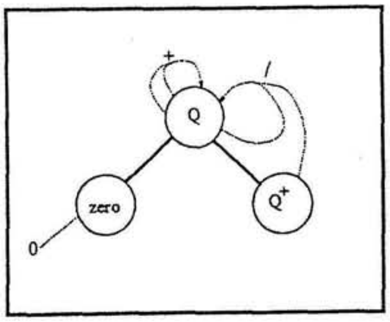


Figure 1: The **Rationals** signature

In order for a number of useful properties to hold, restrict our attention to a special class of signatures called *regular*. Essentially, regularity asserts that overloaded operations are consistent under restrictions to subsorts. Note that the ordering $\leq$ on $S$ extends to an ordering on strings of equal length in $S^*$ as follows: $s_1 \ldots s_n \leq s'_1 \ldots s'_n$ iff $s_i \leq s'_i$ for $1 \leq i \leq n$ . Similarly, $\leq$ extends to pairs in $S^* \times S$ by stating that $(w, s) \leq (w', s')$ iff $w \leq w'$ and $s \leq s'$.

**Definition 2.4** An order-sorted signature $S$ is *regular* iff for every $f \in \Sigma$, every $w^0 \in S^*$, and every $(w, s) \in \rho(f)$, if $w^0 \leq w$, then the set $\{(w', s') \in \rho(f) \mid w^0 \leq w'\}$ has a least element.

When the set of sorts is finite (or well founded), regularity is captured by a combinatorial condition (see the paper by Goguen and Meseguer [GM87b]).

**Lemma 2.5** An order-sorted signature $\Sigma$ over a finite (or well founded) sort set $S$ is regular iff for every every $f \in \Sigma$, every $w^0 \in S^*$, and every pair of ranks $(w, s)$, $(w', s') \in \rho(f)$, if $w^0 \leq w, w'$, then the set $\{(w, s), (w', s')\}$ has a lower bound $(w_l, s_l)$ such that $(w_l, s_l) \in \rho(f)$, and $w^0 \leq w_l$.

Let $\equiv\, = \;(\leq \cup \leq^{-1})^+$ be the least equivalence relation containing the partial order $\leq$. We say that two sorts $s$ and $s'$ are *connected* if $s \equiv s'$. The equivalence classes of $\equiv$

are called *connected components.* The concept of connected sorts is important for defining quotient algebras. Indeed, in order for the usual construction of the quotient of an algebra by a congruence to hold, we need a condition on signatures called coherence.

**Definition 2.6** A regular order-sorted signature is *coherent* if every connected component has a greatest element called the *top sort* of the connected component.

In this paper we limit our attention to finite coherent signatures.

## 2.2 Algebras

For any string $w = s_1, \ldots, s_n$ $(n \geq 1)$, let $A_w = A_{s_1} \times \ldots \times A_{s_n}$, with $A_\lambda = \{\lambda\}$ (a one element set).

**Definition 2.7** Let $(S, \leq, \Sigma, \rho)$ be an order-sorted signature. An order sorted $(S, \leq, \Sigma, \rho)$-algebra $\mathcal{A}$ is a pair $\langle A, I\rangle$ consisting of an $S$-sorted family $A = (A_s)_{s\in S}$ called the *carrier* of $\mathcal{A}$, and a function $I$ called the *interpretation function* of $\mathcal{A}$, where $I$ assigns to every $f \in \Sigma$ an indexed family of functions $I(f) = (f_{\mathcal{A}}^{w\mapsto s} : A_w \rightarrow A_s)_{(w,s)\in\rho(f)}$. In particular, when $w = \lambda$, $f_{\mathcal{A}}^{\lambda\mapsto s}$ is an element of $A_s$. For each sort $s$, $A_s$ is the carrier of sort $s$. Note that the carrier of sort $s$ may be empty. Moreover, the following conditions hold:

1. $A_s \subseteq A_{s'}$ whenever $s \leq s'$, and
2. If $(w, s) \in \rho(f)$ and $(w', s') \in \rho(f)$, $s \leq s'$, and $w \leq w'$, then $f_{\mathcal{A}}^{w\mapsto s} : A_w \mapsto A_s$ is equal to the restriction of $f_{\mathcal{A}}^{w'\mapsto s'} : A_{w'} \mapsto A_{s'}$ to $A_w$. That is, for any $\overline{x} \in A_w, f_{\mathcal{A}}^{w'\mapsto s'}(\overline{x}) = f_{\mathcal{A}}^{w\mapsto s}(\overline{x})$.

By abuse of notation, we may denote an algebra and its carrier by the same name unless confusions arise. For example in the the previous definition we might use $A$ for both the carrier (which is $A$) and for the algebra (which is $\mathcal{A}$). We may also drop some of the components in $(S, \leq, \Sigma, \rho)$ when talking about order-sorted algebras, or drop the superscript $(w, s)$ when referring to a function $f_{\mathcal{A}}^{w\mapsto s}$.

**Example 2.8** Consider the signature presented of example 2.3, an order-sorted $\Sigma$-algebra $\mathcal{A}$ is:

- $A_{\mathbf{Q}} = Q$ ( the set of rational numbers),
- $A_{\mathbf{Q}^+} = Q - \{0\}$ ( the set of non-zero rationals), and
- $A_{\mathbf{zero}} = \{0\}$.

The functions have their natural interpretations:

- $0_{\mathcal{A}} = 0$;
- $+_{\mathcal{A}}$ is addition of rational numbers;
- $/_{\mathcal{A}}$ is division of rational numbers.

For any $w = s_1 \ldots s_n \neq \lambda$ and $\overline{a} = (a_1, \ldots, a_n) \in A_w$, let $h_w(\overline{a}) = (h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$.

**Definition 2.9** Let $(S, \leq, \Sigma, \rho)$ be an order-sorted signature, and let $\mathcal{A}$ and $\mathcal{B}$ be $(S, \leq, \Sigma, \rho)$-order-sorted algebras. A $(S, \leq, \Sigma, \rho)$-homomorphism $h : \mathcal{A} \mapsto \mathcal{B}$ is an $S$-sorted function such that

1. for every constant $c$ of sort $s$, $h_s(c_{\mathcal{A}}) = c_{\mathcal{B}}$,
2. for every $f \in \Sigma$, every $(w, s) \in \rho(f)$, and every $\overline{a} \in A_w$,
$$h_s(f_{\mathcal{A}}^{w \mapsto s}(\overline{a})) = f_{\mathcal{B}}^{w \mapsto s}(h_w(\overline{a})),$$
3. $w \leq w'$ and $\overline{a} \in A_w$ implies $h_w(\overline{a}) = h_{w'}(\overline{a})$.

When the partialy ordered set is clear, $(S, \leq, \Sigma, \rho)$-homomorphisms are called order-sorted $\Sigma$-homomorphisms. We may also drop some of the components in $(S, \leq, \Sigma, \rho)$ when talking about order-sorted homomorphisms.

## 2.3 Order-Sorted term algebra

Following [GM87b], we now define the order-sorted $\Sigma$-term algebra $\mathcal{T}_\Sigma$ as the least family $\{\mathcal{T}_{\Sigma,s} | s \in S\}$ of sets satisfying the following conditions:

1. $\Sigma_{\lambda,s} \subseteq \mathcal{T}_{\Sigma,s}$ for $s \in S$;

2. $\mathcal{T}_{\Sigma,s} \subseteq \mathcal{T}_{\Sigma,s'}$ whenever $s \leq s'$;

3. if $f \in \Sigma_{w,s}$, and if $t_i \in \mathcal{T}_{\Sigma,w_i}$ where $w = w_1, \ldots, w_i \neq \lambda$, then the string $ft_1 \ldots t_n$ is in $\mathcal{T}_{\Sigma,s}$

In addition, the function symbols are interpreted as string constructors as follows: for $f \in \Sigma_{w,s}$, $f_{\mathcal{T}_\Sigma}^{w \mapsto s}(t_1, \ldots, t_n) = ft_1 \ldots t_n$. Regular signatures have a number of desirable properties. For example, unique sorts can be assigned to terms in $\mathcal{T}_\Sigma$ as the following theorem form [GM87b] states.

**Theorem 2.10** Let $\Sigma$ be a regular order-sorted signature. Then every term $t$ in $\mathcal{T}_\Sigma$ has a least sort denoted by $LS(t)$.

For the rest of this paper we assume that all signatures are regular. In order to define non-ground terms, we enlarge the signature $\Sigma$ with variables. The variables form an $S$-sorted set $X = \{X_s\}_{s \in S}$ which is assumed to be disjoint from $\Sigma$ such that each variable belongs to exactly one $X_s$, i.e. it has a unique sort. The extended signature is denoted by $\Sigma(X)$, it is regular provided $\Sigma$ is regular. The term algebra $\mathcal{T}_{\Sigma(X)}$ is denoted also by $\mathcal{T}_\Sigma(X)$, and it is the *free* $\Sigma$ order-sorted algebra on $X$ ([GM87a]), i.e.

**Theorem 2.11** Let $\mathcal{A}$ be an order-sorted $\Sigma$-algebra and let $\alpha : X \mapsto A$ be an $S$-sorted function (an assignment from $X$ to $A$). Then there exists a unique order-sorted $\Sigma$-homomorphism $\alpha^* : \mathcal{T}_\Sigma(X) \mapsto \mathcal{A}$ that extends $\alpha$.

## 2.4 Order-Sorted deduction

A fundamental component of deductive systems is the notion of a *substitution* which provides a tool for the instantiation of terms. Since order-sorted substitutions have to produce well typed terms, their definition has to take sort information into account. We follow [MGS89] in the defining substitutions as homomorphic extensions of well-sorted assignments, thus departing from Walther [Wal87] who defines them as being endomorphisms of a fixed term algebra.

**Definition 2.12** Given an $S$-sorted assignment $\theta : X \mapsto \mathcal{T}_\Sigma(Y)$ such that $\theta(x) = x$ almost everywhere (i.e. the set $\{x \mid \theta(x) \neq x\}$ is finite), its homomorphic $\Sigma$-extension $\theta^* : \mathcal{T}_\Sigma(X) \mapsto \mathcal{T}_\Sigma(Y)$ is an order-sorted substitution.

We will write "$\Sigma$-*substitution*" for "Order-Sorted substitution" when the signature in consideration is $\Sigma$, even though this is somewhat ambiguous because we are not specifying the set of variables involved. By allowing a slight abuse of notation, we will denote $\theta^*$ by $\theta$.

Note that since an assignment is an $S$-sorted map we have that $\theta(x) \in \mathcal{T}_\Sigma(Y)_s$ whenever $x \in X_s$. Therefore if the signature is regular, $LS(\theta(x)) \leq LS(x)$. We will denote substitutions as association lists of the form $[x_1/t_1, \ldots, x_n/t_n]$. If we drop the sort information from a signature $\Sigma$, we obtain an unsorted signature $|\Sigma|$. Clearly, every order-sorted substitution is an unsorted one, i.e. every order-sorted signature is a $|\Sigma|$-substitution. The contrary however, is false as we show in the next example.

**Example 2.13** Consider the signature **Rationals**, let $z_{\mathbf{rat}}$ be a variable of sort **rat** and let $z_{\mathbf{rat}^+}$ be a variable of sort $\mathbf{rat}^+$. Consider the mapping $\theta$ such that $\theta(z_{\mathbf{rat}^+}) = \mathbf{0}$. Although $\theta$ is an unsorted substitution, it is not a $\Sigma$-substitution because the sorts of $z_{\mathbf{rat}^+}$ and $\mathbf{0}$ are incomparable.
However, the mapping $\theta'$ such that $\theta'(z_{\mathbf{rat}}) = \mathbf{0}$ is a $\Sigma$-substitution and $LS(\theta'(z_{\mathbf{rat}})) \leq LS(z_{\mathbf{rat}})$.

We now turn our attention to order-sorted equational deduction. First, we point out that in order for an equation to make sense, the terms equated must have a common supersort. Then, we can think of the two terms as being equal in that sort. Recall that in a coherent signature each connected component of the sorts poset has a greatest element. Since the signatures considered here are coherent, it is enough to restrict equations to terms with sorts in the same connected component

**Definition 2.14** Given a coherent order-sorted signature $\Sigma$, let $u$ and $v$ be terms in $\mathcal{T}_\Sigma(Y)$ such that their least sorts are connected, and let $X$ be a superset of the set of all variables occurring in $u$ or $v$ (notice $X \subseteq Y$). Then $(\forall Y)u \doteq v$ is an *equation*. If $Y = \{y_1, \ldots, y_n\}$, we might write $\forall y_1 \ldots \forall y_n \; u \doteq v$ instead of $(\forall Y)u \doteq v$.

The concept of validity of an equation is defined using the freeness of $\mathcal{T}_\Sigma(X)$.

**Definition 2.15** An equation $(\forall X)u \doteq v$ is valid in some order-sorted $\Sigma$-algebra $\mathcal{A}$ (denoted $\mathcal{A} \models (\forall X)u \doteq v$) if and only if for every assignment $\alpha : X \mapsto A$, $\alpha^*_{LS(u)}(u) = \alpha^*_{LS(v)}(v)$.

A $\Sigma$-algebra $\mathcal{A}$ satisfies a set $E$ of equations if it satisfies every equation in $E$. A set $E$ of equations *semantically entails* an equation $(\forall X)u \doteq v$, written $E \models (\forall X)u \doteq v$, if $(\forall X)u \doteq v$ is valid in every model of $E$.

We now provide a set of deduction rules for equations involving variables. Given an order-sorted signature $\Sigma$ and a set $E$ of $\Sigma(X)$-equations, the following is a complete set of deduction rules for order-sorted equational logic ([MGS89]):

1. *reflexivity.* Each equation $(\forall X)t \doteq t$ is derivable.
2. *Symmetry.* If $(\forall X)t \doteq t'$ is derivable, then so is $(\forall X)t' \doteq t$.
3. *Transitivity.* If $(\forall X)t \doteq t'$ and $(\forall X)t' \doteq t''$ are derivable, then so is $(\forall X)t \doteq t''$.
4. *Congruence.* Given $t \in \mathcal{T}_\Sigma(X)$ and $\Sigma$-substitutions $\theta, \theta' : X \mapsto \mathcal{T}_\Sigma(Y)$ such that for each $x \in X$, the equation $(\forall Y)\theta(x) = \theta'(x)$ is derivable, then the equation $(\forall Y)\theta(t) = \theta'(t)$.
5. *Substitutivity.* If $(\forall X)t \doteq t' \in E$, and if $\theta : X \mapsto \mathcal{T}_\Sigma(Y)$ is a $\Sigma$-substitution, then $(\forall Y)\theta(t) \doteq \theta(t')$ is derivable.

We denote the derivability relation by $\vdash_\Sigma$ as usual. When the order-sorted signature is clear from the context, we might simply write $\vdash$.

**Theorem 2.16** [Soundness and Completeness Theorem [GM87b]] Given a coherent order-sorted signature $\Sigma$, a set $E$ of $\Sigma(X)$-equations, and terms $t, t' \in \mathcal{T}_\Sigma(X)$, the following are equivalent:

- $E \vdash_\Sigma t = t'$.
- $E \models_\Sigma t = t'$.

# 3 General Equations

Given the complexity of E-unification in the case of arbitrary equational theories, it makes sense to restrict the kind of equations and to study the problem under those restrictions. We focus our attention to a special class which we call *General* equations.

The study of rigid $(\Sigma, E)$-unification for equation systems which are not general, although of interest, is beyond the scope of this paper.

*General* equations are sort preserving in a very strong sense: not only are both terms involved of the same sort, but this property is stable under variable renamings.

A variable renaming $\Sigma$-substitution is a $\Sigma$-substitution $\theta : X \mapsto Y$ where $Y$ is a set of variables, i.e. $\theta(x)$ is always a variable. Notice that the sort of $\theta(x)$ has to be below that of $x$. Thus, talking about variable renamings is equivalent to talking about the set of sorts below a given one. If the signature is finite (as in our case), then, module alphabetic variants, there is only a finite number of possible variable renamings for a term $t$.

**Definition 3.1** Given an equation $e = (\forall X)t \doteq t'$ over $\Sigma$, we say that $e$ is *general* provided

1. $Var(t) = Var(t')$, and

2. for any variable renaming $\rho$, $LS(\rho(t)) = LS(\rho(t'))$.

In particular, $LS(t) = LS(t')$. A system $E = \{t_i \doteq t'_i, i \in I\}$ is said to be general if each equation is general.

Intuitively, we make sure that every instance of the equation is sort preserving. This will ensure that no ill-typed terms can be generated when rewriting. We illustrate via an example what is **not** general.

**Example 3.2** Consider the signature $\mathbf{MG}_1$ shown in figure 2.
Let $e = (\forall x : s_1)f(x) \doteq g(x)$. Although $LS(f(x)) = LS(g(x)) = s_1$, there is a problem when we apply the variable renaming $\rho(x) = z : s_4$ because $LS(f(z)) = s_3$ but $LS(g(z)) = s_2$. This shows that $e$ is not general. Thus when using $e$ to make deduction special attention to the sorts has to be drawn. For example, even though $f(z) \doteq g(z)$ is a valid consequence of $e$,

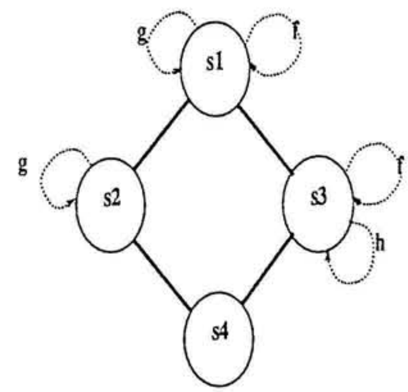


Figure 2: The $\mathbf{MG}_1$ signature

$h(f(z)) \doteq h(g(z))$ is not only invalid, but $h(g(z))$ is ill-typed. Hence replacement of equals by equals cannot be used with equations which are not general.

The previous example shows that some unsorted theorem proving methods are not sound for order-sorted deduction. However as we will see, congruence closure, can be safely applied to systems of *frozen* equations. This will be come a key issue in our algorithm for rigid $(\Sigma, E)$-unification.

**Lemma 3.3** Let $l \doteq r$ be a general equation and let $\sigma$ be a $\Sigma$-substitution, then $\sigma(l) \doteq \sigma(r)$ is also general.

**Proof:**

1. Clearly $Var(\sigma(l)) = Var(\sigma(r))$.

2. To show that renamings of $\sigma(l) \doteq \sigma(r)$ are sort preserving. Notice that the sort of such a renaming can be characterized by renamings of the original equation. This is so because one can define a renaming $\rho$ s.t.

$$LS(\sigma(l)) = LS(\rho(l)) = LS(\rho(r)) = LS(\sigma(r).$$

   This is done as follows: for $x \in Var(l)$ let $x_{\sigma(x)}$ be a variable of sort $LS(\sigma(x))$. Let $\rho(x) = x_{\sigma(x)}$. The least sort of any renaming of $\sigma(x)$ can then be realized by an appropriate renaming of $x$.

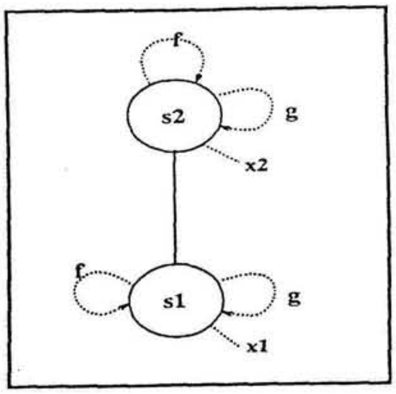


Figure 3: $E = \{(\forall(x_1) f(x_1) \doteq g(x_1)\}$ is not *most general.*

□

The class of general equations is less restrictive than the class of *most general equations* defined by Meseguer, Goguen and Smolka in [MGS89]. They require an equation to be sort preserving under arbitrary renamings (not just $\Sigma$-substitutions). For example, consider the signature of figure 3 and the equation $E = \{(\forall(x_1) f(x_1) \doteq g(x_1)\}$. Clearly $E$ is general. Since $f(x_2) \doteq g(x_2)$ is not covered by $E$, the system is not *most general.*
The focus in [MGS89] is on utilizing unsorted theorem methods which at a second pass are transformed into order-sorted ones. In that context it is important to preserve the unsorted deducibility relation. Notice that $E \not\vdash_\Sigma (\forall x_2) f(x_2) \doteq g(x_2)$.

# 4 Order-Sorted Unification

Unification basically amounts to finding values for the variables appearing in terms so as to make them equal. Given two terms $t$ and $t'$, a substitution $\theta$ is a unifier of $t$ and $t'$ if $\theta(t) = \theta(t')$. Thus a unifier can be seen as a solution of the equation $t \doteq t'$. Given a system $T$ of term equations, we say that a substitution $\theta$ is a unifier of the system $T$ if $\theta$ unifies every term equation in $T$. General unification, commonly called $E$-unification amounts to solving a system $T$ of term equations modulo a set $E$ of equations.

## 4.1 Term unification

The order-sorted unification problem has been addressed by different researchers [Kir88, MGS89, SS87, Wal87, Wal84]. Order-Sorted Unification differs from its unsorted version. In the simple case of unifying two variables $x : s_1$ and $y : s_2$ the existence of an order-sorted unifier of $x$ and $y$ depends on the sort structure. If there is no lower bound to the set $\{s_1, s_2\}$ there is no unifier. If however, the set $LBd(\{s_1, s_2\}) = \{s \in S \mid s \leq s_1 \text{ and } s \leq s_2\}$ is not empty, any element of it represents a order-sorted unifier. That is, for any $s \in LBd(\{s_1, s_2\})$, let $z_s \in X_s$ be a variable of sort $s$, then the substitution $[x/z_s, y/z_s]$ is an order-sorted unifier of $x$ and $y$.

In the unsorted case Robinson [Rob65] shows the existence of a most general unifier for a set of unifiable terms. There exist several algorithms to compute a most general unsorted unifier [Hue76, PW78, MM82]. The Martelli-Montanari approach, by abstracting over the control structure, provides a good method for proving existence of unifiers in more general settings [Sny88]. In contrast to the unsorted case, most general unifiers do not exist in the order-sorted case. Complete families of unifiers can be defined as in the case of $E$-unification.

**Definition 4.1** Given a set $T$ of terms, a set of $\Sigma$-substitutions $CSU(T)$ is a complete set of $\Sigma$-unifiers for $T$ iff

**(i)** each $\sigma \in CSU(T)$ satisfies $D(\sigma) \subseteq Var(T)$ and $D(\sigma) \cap I(\sigma) = \emptyset$ ($\sigma$ is idempotent);

**(ii)** if $\sigma \in CSU(T)$ then it is a unifier of $S$;

**(iii)** For every $\Sigma$-unifier $\theta$ of $T$, there exists $\sigma \in CSU(T)$ such that $\sigma \preceq \theta$.

**Example 4.2** Consider the signature **NMGU** shown in figure 4.
Let $z_1, \ldots, z_4$ be variables of sort $s_1, \ldots, s_4$ respectively. The $\Sigma$-substitution $\theta = [z_1/z_3, z_2/z_3]$ is an order-sorted unifier of $z_1$ and $z_2$, and so is $\theta' = [z_1/z_4, z_2/z_4]$. Notice however, that neither does $\theta$ subsume $\theta'$, nor does $\theta'$ subsume $\theta$. Furthermore, it is easy to see that there does not exist a $\Sigma$-substitution $\phi$ such that $\phi \preceq \theta$ and $\phi \preceq \theta'$. Therefore, no mgu exists for the term pair $< z_1, z_2 >$. However, $\{\theta, \theta'\}$ is a complete set of $\Sigma$-unifiers for $\{z_1, z_2\}$.

Isakowitz [Isa89] presents a non-deterministic algorithm to compute $CSU(T)$.

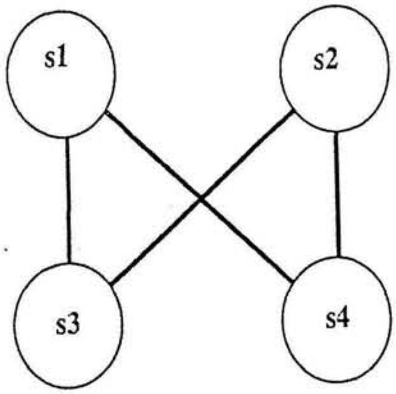


Figure 4: The **NMGU** signature

## 4.2 $E$-Unification

In this section we define the notion of Order-Sorted E-Unification ($\Sigma - E$ Unification), we briefly review and comment on some of the results presented by Meseguer and Goguen and Smolka in [MGS89]. The system of equations which are studied there are called *most general.* Our notion of *general* equational system is weaker than the notion of *most general* equations which is used in [MGS89]. Hence our results do apply to a larger class of equations.

**Definition 4.3** Given a set $E$ of equations and $\Sigma$-terms $t$ and $t'$, we say that a $\Sigma$-substitution $\theta$ is a $(\Sigma, E)$ *unifier of $t$ and $t'$* iff

$$E \vdash_{\Sigma} \theta(t) = \theta(t').$$

By considering the unsorted signature $|\Sigma|$ obtained by *forgetting* the sorts from $\Sigma$ and the unsorted system of equations $|E|$ obtained from $E$, one can compare unsorted and order-sorted $E$-unification. In [MGS89], the relationship between these is studied. A number of characterization theorems are presented which show that for *reasonable signatures*, families of order-sorted $E$-unifiers can be obtained from unsorted $E$-unifiers. The method consists in first computing an unsorted $E$-unifier and then finding sort assignments for the variables to construct order-sorted unifiers. However, such sort assignments might not always exist, in which case there is not order-sorted version of the $E$-unifier. As we shall see later, our

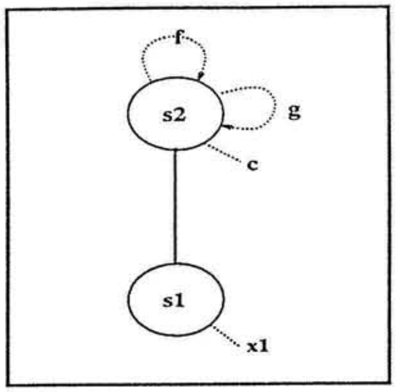


Figure 5: $f(c) \doteq g(c)$

method detects that a potential substitution can not become a $\Sigma$-unifiers earlier and can therefore present significant efficiency gains over the unsorted method.

**Example 4.4** Consider the signature of figure 4.2 and the equation $f(c) \doteq g(c)$. The $\Sigma$-terms $f(x_1)$ and $g(x_1)$ are not $(\Sigma, E)$-unifiable. However, the method described above would first discover the unsorted $E$-unifier $[c/x_1]$. Any attempt to come up with an order-sorted version of this unifier is deemed to failure.

## 4.3 Unifiers in Triangular Form

In order to show that our decision procedure for rigid order-sorted unification is in NP, we will need the fact that members of $CSU(u, v)$ can be represented concisely in triangular form (the size of this system is linear in the number of symbols in $u$ and $v$). We will denote a complete family of $\Sigma$-unifiers in triangular form by $CTU(T)$. When $T$ consists of a single pair $\langle u, v\rangle$, $CTU(S)$ is also denoted by $CTU(u, v)$.

An algorithm for finding a complete family of $\Sigma$-unifiers in triangular form for arbitrary finite coherent signatures is described by Isakowitz in [Isa89]. This method is obtained from the fast method using multiequations of Martelli and Montanari [MM82] adapted to the order-sorted case as presented by Meseguer, Goguen and Smolka in [MGS89] by utilizing a non-deterministic version of the $IP$ algorithm ([MGS89]). [1] Thus, this method is nondeterministic, and it computes elements of $CTU(T)$ in nondeterministic quasi-linear time.

In addition to the fact that complete families of triangular $\Sigma$-unifiers do exist, we will use some properties of triangular forms in the proof of the soundness of our method. We develop an abstract view of triangular forms. First, we define *triangular forms.*

**Definition 4.5** Given an idempotent $\Sigma$-substitution $\sigma$ with domain $D(\sigma) = \{x_1, \ldots, x_k\}$, a *triangular form for* $\sigma$ is a finite set $T$ of pairs $\langle x, t\rangle$ where $x \in D(\sigma)$ and $t$ is a term, such that this set $T$ can be sorted (possibly in more than one way) into a sequence $\langle\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\rangle$ satisfying the following properties: for every $i$, $1 \leq i \leq k$,

**(1)** $x_1, \ldots, x_i \cap Var(t_i) = \emptyset$, and

**(2)** $\sigma = [t_1/x_1]; \ldots; [t_k/x_k]$.

The set of variables $\{x_1, \ldots, x_k\}$ is called the *domain* of $T$. Note that in particular $x_i \notin Var(t_i)$ for every $i$, $1 \leq i \leq k$, but variables in the set $\{x_{i+1}, \ldots, x_k\}$ may occur in $t_1, \ldots, t_i$. It is easily seen that $\sigma$ is an idempotent mgu of the term system $T$.

**Example 4.6** Consider the $\Sigma$-substitution $\sigma = [f(f(x_3, x_3), f(x_3, x_3))/x_1, f(x_3, x_3)/x_2]$. The system $T = \{\langle x_1, f(x_2, x_2)\rangle, \langle x_2, f(x_3, x_3)\rangle\}$ is a triangular form of $\sigma$ since it can be ordered as $\langle\langle x_1, f(x_2, x_2)\rangle, \langle x_2, f(x_3, x_3)\rangle\rangle$ and $\sigma = [f(x_2, x_2)/x_1]; [f(x_3, x_3)/x_2]$.

The triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$ of a $\Sigma$-substitution $\sigma$ also defines a $\Sigma$-substitution, namely $\sigma_T = [t_1/x_1, \ldots, t_k/x_k]$. This $\Sigma$-substitution is usually different from $\sigma$ and not idempotent as can be seen from example 4.6.

The method for computing $\Sigma$-unifiers returns triangular forms, i.e. given $\Sigma$-terms $t$ and $t'$, the method returns either **failure** or a triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$ for a $\Sigma$-unifier $\theta$ of $t$ and $t'$. The substitution $sigma_T$ associated with this triangular form plays a crucial role in our decision procedure by providing a succinct representation of a $\Sigma$-unifiers.

[1] In fact, this result can be strengthened: our method works for *finitary* signatures while the one presented in [MGS89] works for *unitary* signatures.

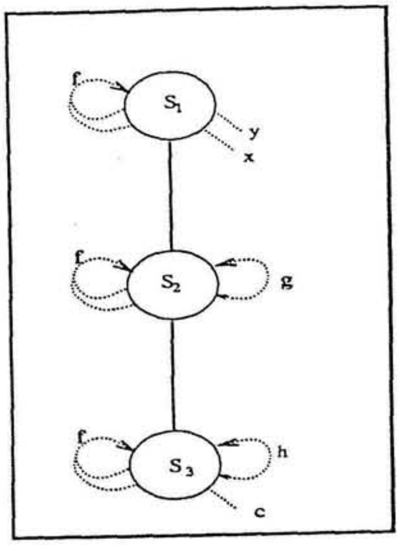


Figure 6

This reduces the complexity of the algorithm. Notice however that even though $\sigma_T$ is associated to $\theta$ (which unifies $t$ and $t'$),

1. as is well known that $\sigma_T$ might not unify $t$ and $t'$ [2]; and
2. $LS(\sigma_T(t))$ and $LS(\sigma_T(t'))$ might differ.

This last observation presents a problem to our development.

**Example 4.7** Consider the signature presented in figure 6. Given $\Sigma$-terms $t = f(x, y, z)$ and $t' = f(y, g(z), h(c))$, the $\Sigma$-substitution $\theta = [g(h(c))/x, g(h(c))/y, h(c)/z]$ is a $\Sigma$-unifier of $t$ and $t'$. The following is a $\Sigma$-substitution associated with a triangular form for $\theta$: $\sigma_T = [y/x, g(z)/y, h(c)/z]$. However,

$$\sigma_T(f(x, y, z)) = f(y, g(z), h(c)) \text{ and;}$$
$$\sigma_T(f(y, g(z), h(c))) = f(g(z), g(h(c)), h(c)).$$

[2] For example, $\sigma$ in example 4.6 is a triangular form of a unifier of $t = f(x_1, x_2)$ and $t' = f(f(x_2, x_2), f(x_3, x_3))$. However, as the reader is invited to check, $\sigma_T(t) \neq \sigma_T(t')$

Not only do $\sigma_T(t)$ and $\sigma_T(t')$ differ in structure, but also in sorts: $LS(\sigma_T(t)) = s_1$ while $LS(\sigma_T(t')) = s_2$.

In order to *force* $\sigma_T(t)$ and $\sigma_T(t')$ to have the same sort, we observe that since $t$ and $t'$ are unifiable, there has to exist a variable renaming $\rho$ such that $LS(\rho(\sigma_T(t))) = LS(\rho(\sigma_T(t'))) = LS(\theta(t))$. In fact, by reading a triangular form from right to left, such a variable assignment can be obtained. New variables are utilized to represent the renaming. In the case of the previous example, $y$ will get the sort of $g(z')$ which is $s_2$, and $x$ will also be *pushed* to have sort $s_2$.

**Definition 4.8** Given a $\Sigma$-substitution $\theta$ with triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$. Let $\rho_{k+1} = id$, and for $j = 0, \ldots k-2$, let

$$\begin{aligned} srt_{k-j} &= LS(\rho_{k-(j-1)}(t_{k-j})); \\ \rho_{k-j} &= \rho_{k-(j-1)}\,[y_{s_{k-j},k-j}/x_{k-j}] \end{aligned}$$

where each $y_i$ is a different variable of sort $srt_i$ not appearing in the original system (for $i = 1, \ldots, k$).
The *special triangular form* $T^\bullet$ is defined by $T^\bullet = \{\langle x_1, \rho_2(t_1)\rangle, \ldots, \langle x_k, \rho_{k+1}(t_k)\rangle\}$. Its associated substitution will be denoted by $\sigma_T^\bullet$.

By construction, we have the following result:

**Lemma 4.9** If $\sigma_T^\bullet$ is a special triangular form for a $\Sigma$-substitution $\sigma$, then for every $x \in Dom(\sigma)$, $LS(\sigma_T^\bullet(x)) = LS(\sigma(x))$.

From this we have the following important corollary:

**Corollary 4.10** Let $\theta$ be a $\Sigma$-unifier of the $\Sigma$-terms $t$ and $t'$, and let $\sigma_T^\bullet$ be a special triangular form for $\theta$, then $LS(\sigma_T^\bullet(t)) = LS(\sigma_T^\bullet(t'))$.

**Example 4.11** Recall from example 4.7, $\sigma_T = [y/x, g(z)/y, h(c)/z)]$. Then

$$srt_3 = LS(id(h(c)) = s_3$$

$$\rho_3(z) = y_{s_3}$$

$$srt_2 = LS(\rho_3(g(z))) = LS(g(y_{s_3})) = s_2$$

$$\rho_2(y) = y_{s_2}$$

Thus $\sigma_T^\bullet = [y_{s_2}/x, g(y_{s_3})/y, h(c)/z)]$. Let us compute $\sigma_T^\bullet(t)$ and $\sigma_T^\bullet(t')$:

$$\sigma_T^\bullet(t) = \sigma_T^\bullet(f(x, y, z)) = f(y_{s_2}, g(y_{s_3}), h(c)); \text{ and;}$$
$$\sigma_T^\bullet(t') = \sigma_T^\bullet(f(y, g(z), h(c)) = f(g(y_{s_3}), g(h(c)), h(c)).$$

We still have $\sigma_T^\bullet(t) \neq \sigma_T^\bullet(t')$. However, $LS(\sigma_T^\bullet(t)) = s_2$ and $LS(\sigma_T^\bullet(t')) = s_2$!

Special triangular forms play an important role in the algorithm for rigid $(\Sigma, E)$-unification. In what follows, all triangular forms and associated $\Sigma$-substitutions are assumed to be in this special form and will be denoted by $T$ and $\sigma_T$ instead of $T^\bullet$ and $\sigma_T^\bullet$. We now develop a series of lemmas which will be utilized in the proofs of the soundness and completeness of our rigid $(\Sigma, E)$-unification method. First, we adapt a technical lemma from [GNPS90].

**Lemma 4.12** Given a triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$ for a $\Sigma$-substitution $\sigma$ and the associated $\Sigma$-substitution $\sigma_T = [t_1/x_1, \ldots, t_k/x_k]$, for every $\Sigma$-unifier $\theta$ of $T$, $\theta = \sigma_T\,;\theta$.
**Proof:** Since $\theta$ is a $\Sigma$-unifier of $T$, we have $\theta(x_i) = \theta(t_i) = \theta(\sigma_T(x_i))$ for every $i$, $1 \leq i \leq k$. Since $\sigma_T(y) = y$ for all $y \notin \{x_1, \ldots, x_k\}$, $\theta = \sigma_T\,;\theta$ holds. □

Another important observation about $\sigma_T$ is that even though it is usually not idempotent, at least one variable in $\{x_1, \ldots, x_k\}$ does not belong to $I(\sigma_T)$ (otherwise, condition (1) of the triangular form fails).

The following results from [Isa89], which also hold in the unsorted case, shed some light on the relationship between a $\Sigma$-unifier and its triangular form. Interestingly enough, the results are developed algebraically, as opposed to concentrating on the methods to obtain triangular forms. Although $\sigma$ and $\sigma_T$ are different substitutions, the following lemma shows that composing $\sigma_T$ with itself enough times yields $\sigma$.

**Lemma 4.13** Given a term system $S$; $\sigma$ an idempotent $\Sigma$-unifier of $S$; and $T = \{< x_1, t_1 >, \ldots < x_n, t_n >\}$ a triangular form for $\sigma$, let $\sigma_T = [x_1/t_1, \ldots, x_n/t_n]$ be the $\Sigma$-substitution associated with $T$. Then ${\sigma_T}^{(n)} = \sigma$.
The proof is given in appendix A.2.

Based on the previous lemma we can state a result similar to lemma 4.12.

**Lemma 4.14** Given $T$ a triangular form of an idempotent $\Sigma$-unifier $\sigma$ of a system $S$, if $\theta$ unifies $T$, then $\theta = \sigma; \theta$.
**Proof:** By lemma 4.12, $\theta = \sigma_T; \theta$, and hence for any $i > 0$, $\theta = \sigma_T^{(i)}; \theta$. By the previous lemma $\sigma_T^{(n)} = \sigma$. Therefore, $\theta = \sigma; \theta$. □

We can now prove the following result:

**Lemma 4.15** Given $T$, a triangular form for an idempotent $\Sigma$-unifier $\sigma$ of a term system $S$; every $\Sigma$-unifier of $T$ is also a $\Sigma$-unifier of $S$.
**Proof:** Let $\theta$ be a $\Sigma$-unifier of $T$. By lemma 4.14 $\theta = \sigma; \theta$. Since $\sigma$ unifies $S$, so does $\theta$ because given any $< t, t' > \in S$, $\theta(t) = \theta(\sigma(t)) = \theta(\sigma(t')) = \theta(t')$. □

**Lemma 4.16** If $\sigma$ is an idempotent $\Sigma$-unifier of $S$ and $T$ is a triangular form for $\sigma$, then $\sigma$ unifies $T$.
The proof is given in appendix A.3.

# 5 Rigid-E-Unification

In this section we give the formal definition of rigid $(\Sigma, E)$-unification and we provide some intuition for the method we are about to develop. Our approach is based on the method given by Gallier, Narendran, Plaisted and Snyder in [GNPS90]. Our accomplishments are twofold.

Firstly, we significantly simplify the unsorted method and its correctness proofs, thereby presenting an improved unsorted rigid E-unification method. The major simplification is the removal of order assignments from the transformation which is an important compinent of the unsorted method as presented in [GNPS90]. Order assignments represent guesses of portions of the final solution. Their role in the rigid E-unification method is difficult to understand and their presence complicates the proofs. We incorporate the guessing within another component of the method: the reduction procedure. By doing so, we manage to reduce the number of components of the method, thereby simplifying it. We also modify the reduction procedure by incorporating a reduction method by Snyder [Sny89]. We then provide new soundness and completeness proofs which show the correctness of the order-sorted algorithm and also apply to the unsorted method.

Secondly, our method is intrinsically order-sorted. We utilize an order-sorted unification algorithm to ensure that at each step of our method, the sort information is taken into account. This makes for an efficient algorithm which is able to discard unfit substitutions as these are built, by identifying sort conflicts.

We begin with some formal definitions.

**Definition 5.1** Let $E \subseteq T_\Sigma(X) \times T_\Sigma(X)$ be a binary relation on terms. We define the relation $\longleftrightarrow_E$ over $T_\Sigma(X)$ as follows: Given any two terms $t_1, t_2 \in T_\Sigma(X)$, then $t_1 \longleftrightarrow_E t_2$ iff there is some variant[3] $(s,t)$ of a pair in $E \cup E^{-1}$, some tree address $\alpha$ in $t_1$, and some substitution $\sigma$, such that

$$t_1/\alpha = \sigma(s), \quad \text{and} \quad t_2 = t_1[\alpha \leftarrow \sigma(t)].$$

(In this case, we say that $\sigma$ is a *matching substitution* of $s$ onto $t_1/\alpha$. The term $t_1/\alpha$ is called a *redex*.) Note that the pair $(s,t)$ is used as a two-way rewrite rule (that is, non-oriented). In such a case, we denote the pair $(s,t)$ as $s \doteq t$ and call it an *equation*. When $t_1 \longleftrightarrow_E t_2$, we say that we have an *equality step*. When we want to fully specify an equality step, we

[3] A pair $(s,t)$ is a *variant* of a pair $(u,v) \in E$ iff there is some renaming $\rho$ with domain $Var(u) \cup Var(v)$ such that $s = \rho(u)$ and $t = \rho(v)$.

use the notation

$$t_1 \longleftrightarrow_{\alpha, s \doteq t, \sigma} t_2$$

(where some of the arguments may be omitted). A sequence of equality steps

$$u = u_0 \longleftrightarrow_E u_1 \longleftrightarrow_E \ldots \longleftrightarrow_E u_{n-1} \longleftrightarrow_E u_n = v$$

is called a *proof* of $u \stackrel{*}{\longleftrightarrow}_E v$.

**Definition 5.2** Given a finite set $E$ of equations (ground or not), we say that *$E$ is treated as a set of ground equations* iff for every pair of terms $u$, $v$ (ground or not), for every proof of $u \stackrel{*}{\longleftrightarrow}_E v$, then for every equality step $s \longleftrightarrow_{\alpha, l \doteq r, \sigma} t$ in this proof, $\sigma$ is the identity substitution and $l \doteq r \in E \cup E^{-1}$ (no renaming of the equations in $E \cup E^{-1}$ is performed). This means that variables are treated as constants. We use the notation $u \stackrel{*}{\cong}_E v$ to express the fact that $u \stackrel{*}{\longleftrightarrow}_E v$, treating $E$ as a set of ground equations. Equivalently, $u \stackrel{*}{\cong}_E v$ iff $u$ and $v$ can be shown congruent from $E$ by *congruence closure* (Kozen [Koz76],[Koz77], Nelson and Oppen [NO80], Downey, Sethi, and Tarjan [DST80]) again, treating all variables as constants — they are considered *rigid*.

The results in [Isa89] on congruence closure show that the method is sound for order-sorted deduction when the equations are general. More formally, if $u$ and $v$ are $\Sigma$-terms and $E$ is general then $u \stackrel{*}{\cong}_E v$ implies $E \vdash_\Sigma u \doteq v$. This is the reason why we require the equations to be general!

We give the definition of a rigid $(\Sigma, E)$-unifier.

**Definition 5.3** Let $E = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m)\}$ be a finite set of equations, and let $Var(E) = \bigcup_{(s \doteq t) \in E} Var(s \doteq t)$ denote the set of variables occurring in $E$.[4] Given a $\Sigma$-substitution $\theta$, we let $\theta(E) = \{\theta(s_i \doteq t_i) \mid s_i \doteq t_i \in E,\ \theta(s_i) \neq \theta(t_i)\}$. Given any two terms $u$ and $v$,[5] a $\Sigma$-substitution $\theta$ is a *rigid $(\Sigma, E)$-unifier of $u$ and $v$ modulo $E$* (for short, a *rigid $(\Sigma, E)$-unifier of $u$ and $v$*) iff $\theta(u) \stackrel{*}{\longleftrightarrow}_E \theta(v)$, treating $\theta(E)$ as a set of ground equations i.e., $\theta(u) \stackrel{*}{\cong}_{\theta(E)} \theta(v)$.

[4]It is possible that equations have variables in common.

[5]It is possible that $u$ and $v$ have variables in common with the equations in $E$.

Note that if $E$ is general then a rigid $(\Sigma, E)$-unifier is a $(\Sigma, E)$-unifier. (This follows from the soundness of congruence closure.) The converse, as shown in example 1.2, is not true. Our method for rigid $(\Sigma, E)$-unification can be described in terms of a single transformation on pairs of the form $\langle S, E\rangle$, where $S$ is a unifiable set of pairs and $E$ is a set of general equations. Starting with an initial pair $\langle \emptyset, E_0\rangle$ initialized using $E$ and $u$, $v$, one considers sequences of transformations $\langle \emptyset, E_0\rangle \Rightarrow^+ \langle S_k, E_k\rangle$ consisting of at most $k \leq m$ steps where $m$ is the number of variables in $E$. It will be shown that $u$ and $v$ have some rigid $(\Sigma, E)$-unifier iff there is some sequence of steps as above such that 1) the special equations involving the markers appear in $E_k$, and 2) $S_k$ is unifiable. Then, any $\Sigma$-unifier of $S_k$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$.

# 6 Complete Sets of Rigid $(\Sigma, E)$-Unifiers

As in the case of general $E$-unification, we are interested in complete families of rigid $(\Sigma, E)$-unifiers. The contents of this section are adapted from [GNPS90] to deal with subsorts. The missing proofs are essentially the same as in the unsorted case. We need some definitions regarding complete sets of rigid $(\Sigma, E)$-unifiers. First, we define some preorders on $\Sigma$-substitutions.

**Definition 6.1** Let $E$ be a (finite) set of equations, and $W$ a (finite) set of variables. For any two $\Sigma$-substitutions $\sigma$ and $\theta$, $\sigma =_E \theta[W]$ iff $\sigma(x) \overset{*}{\cong}_E \theta(x)$ for every $x \in W$. The relation $\sqsubseteq_E$ is defined as follows. For any two $\Sigma$-substitutions $\sigma$ and $\theta$, $\sigma \sqsubseteq_E \theta[W]$ iff $\sigma =_{\theta(E)} \theta[W]$. The set $W$ is omitted when $W = X$ (where $X$ is the set of variables), and similarly $E$ is omitted when $E = \emptyset$.

Intuitively speaking, $\sigma \sqsubseteq_E \theta$ iff $\sigma$ can be generated from $\theta$ using the equations in $\theta(E)$. Clearly, $\sqsubseteq_E$ is reflexive. However, it is not symmetric as shown by the following example.

**Example 6.2** Let $E = \{f(x) \doteq x\}$, $\sigma = [f(a)/x]$ and $\theta = [a/x]$. Then $\theta(E) = \{f(a) \doteq a\}$ and $\sigma(x) = f(a) \overset{*}{\cong}_{\theta(E)} a = \theta(x)$, and so $\sigma \sqsubseteq_E \theta$. On the other hand $\sigma(E) = \{f(f(a)) \doteq f(a)\}$, but $a$ and $f(a)$ are not congruent from $\{f(f(a)) \doteq f(a)\}$. Thus $\theta \sqsubseteq_E \sigma$ *does not* hold.

Some positive facts about the relation $\sqsubseteq_E$ are shown in the following lemma from [GNPS90]. These results easily adapt to the order-sorted case.

**Lemma 6.3** For any two $\Sigma$-substitutions $\sigma$, $\theta$,

**(i)** if $\sigma =_{\theta(E)} \theta$, then $\sigma(u) \overset{*}{\cong}_{\theta(E)} \theta(u)$ for any term $u$.

**(ii)** If $\sigma =_{\theta(E)} \theta$, then for all terms $u$, $v$, if $u \overset{*}{\cong}_{\sigma(E)} v$ then $u \overset{*}{\cong}_{\theta(E)} v$.

**(iii)** $\sqsubseteq_E$ is transitive.

**(iv)** For any two terms $u$, $v$, and any $\Sigma$-substitution $\theta$, if $u \overset{*}{\cong}_E v$ then $\theta(u) \overset{*}{\cong}_{\theta(E)} \theta(u)$.

This lemma shows that $\sqsubseteq_E$ is special relationship, a *preorder* as defined below.

**Definition 6.4** A *preorder* $\preceq$ on a set $A$ is a binary relation $\preceq \subseteq A \times A$ that is reflexive and transitive. A *partial order* $\preceq$ on a set $A$ is a preorder that is also antisymmetric. The converse of a preorder (or partial order) $\preceq$ is denoted as $\succeq$. A *strict ordering* (or *strict order*) $\prec$ on a set $A$ is a transitive and irreflexive relation. Given a preorder (or partial order) $\preceq$ on a set $A$, the strict ordering $\prec$ associated with $\preceq$ is defined such that $s \prec t$ iff $s \preceq t$ and $t \not\preceq s$. Conversely, given a strict ordering $\prec$, the partial ordering $\preceq$ associated with $\prec$ is defined such that $s \preceq t$ iff $s \prec t$ or $s = t$. The converse of a strict ordering $\prec$ is denoted as $\succ$. Given a preorder (or partial order) $\preceq$, we say that $\preceq$ is well founded iff $\succ$ is well founded.

From *(i)* and *(ii)* it follows that if $\sigma \sqsubseteq_E \theta$ and $\sigma$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$, so is $\theta$. We also need an extension of $\sqsubseteq_E$ defined as follows.

**Definition 6.5** Let $E$ be a (finite) set of equations, and $W$ a (finite) set of variables. The relation $\leq_E$ is defined as follows: for any two $\Sigma$-substitutions $\sigma$ and $\theta$, $\sigma \leq_E \theta[W]$ iff $\sigma\,;\eta \sqsubseteq_E \theta[W]$ for some $\Sigma$-substitution $\eta$ (that is, $\sigma\,;\eta =_{\theta(E)} \theta[W]$ for some $\eta$). The conventions for omitting $[W]$ and $E$ are those of definition 6.1.

Intuitively speaking, $\sigma \leq_E \theta$ iff $\sigma$ is more general than some $\Sigma$-substitution that can be generated from $\theta$ using $\theta(E)$. Clearly, $\leq_E$ is reflexive. It can also be shown that it is transitive.

Thus, $\leq_E$ is a preorder, and it is clear that it extends $\sqsubseteq_E$. When $\sigma \leq_E \theta[W]$, we say that $\sigma$ *is rigid more general than* $\theta$ *over* $W$. By the remark following lemma 6.3 and part **(iv)** of lemma 6.3, it is immediately verified that if $\sigma$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$ and $\sigma \leq_E \theta$, then $\theta$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$. However, the converse is false.

In the next definition, the concept of a complete set of $(\Sigma, E)$-unifiers is generalized to rigid $(\Sigma, E)$-unifiers.

**Definition 6.6** Given a (finite) set $E$ of equations, for any two terms $u$ and $v$, letting $V = Var(u) \cup Var(v) \cup Var(E)$, a set $U$ of $\Sigma$-substitutions is a *complete set of rigid $(\Sigma, E)$-unifiers for $u$ and $v$* iff: For every $\sigma \in U$,

**(i)** $D(\sigma) \subseteq V$ and $D(\sigma) \cap I(\sigma) = \emptyset$ (idempotence),

**(ii)** $\sigma$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$,

**(iii)** For every rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$, there is some $\sigma \in U$, such that, $\sigma \leq_E \theta[V]$.

Condition **(i)** is the *purity condition*, condition **(ii)** the *consistency condition*, and condition **(iii)** the *completeness condition*.

It should be clear that if $U$ is a complete set of rigid $E$-$\Sigma$-unifiers for $u$ and $v$, $\sigma \in U$, and $\sigma \leq_E \theta$, then $\theta$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$.

A rigid E-unification method that only uses the constant and function symbols already present in $E, u$ and $v$, is called *pure*. The substitutions generated by a pure method do not introduce new symbols. As demonstrated in [GNPS90], pure methods are of interest because their completeness proof can be simplified. Instead of having to consider arbitrary rigid $(\Sigma, E)$-unifiers, it is enough to show completeness with respect to *ground* rigid $(\Sigma, E)$-unifiers whose domains contain $V$. That is, clause **(iii)** of definition 6.6, is replaced by

**(iii')** for every ground rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$ such that $V \subseteq D(\theta)$, there is some $\sigma \in U$ such that $\sigma \leq_E \theta[V]$ (where $V = Var(E) \cup Var(u, v)$).

# 7 Minimal Rigid $(\Sigma, E)$-Unifiers

The concepts and results of this section have been adapted to the order-sorted case from [GNPS90]. Although most results look similar, they involve new techniques and subtleties related to the sorts. We prove some useful lemmas about general equations that are fundamental to our method, and we prove some new results which are interesting in themselves and do not appear in [GNPS90].

Given a finite or countably infinite order-sorted signature $\Sigma$, it is always possible to define a total simplification ordering $\preceq$ on $\mathcal{T}_\Sigma$ (the set of all ground terms). For instance, we can choose some total well-founded ordering $\preceq$ on $\Sigma$ and extend $\preceq$ to $\mathcal{T}_\Sigma$ as follows: $s \prec t$ iff either

1. $size(s) < size(t)$, or
2. $size(s) = size(t)$ and $Root(s) \prec Root(t)$, or
3. $size(s) = size(t)$, $Root(s) = Root(t)$, and letting $s = fs_1 \ldots s_n$ and $t = ft_1 \ldots t_n$, $\langle s_1, \ldots, s_n \rangle \prec_{lex} \langle t_1, \ldots, t_n \rangle$, where $\prec_{lex}$ is the lexicographic ordering induced by $\prec$.

Notice that $t \prec t'$ does not imply $LS(t) \leq LS(t')$. In the rest of this paper, we assume that $\preceq$ is a fixed simplification ordering which is total on $\mathcal{T}_\Sigma$. Given a set $E$ of equations, for any ground substitution $\theta$, we let $< \theta(E), \preceq >$ denote the set $\{\theta(l) \doteq \theta(r) \mid \theta(l) \succ \theta(r),\ l \doteq r \in E \cup E^{-1}\}$ of oriented instances of $E$. Thus, we can also view $\theta(E)$ as a set of rewrite rules. When $\preceq$ is clear from concept, we might simply write $\theta(E)$ instead of $< \theta(E), \preceq >$ . Some ambiguity might arise from not knowing when $\theta(E)$ denotes a set of rewrite rules or a set of equations. In general we mean the former.

Since we restrict ourselves to the case where $E$ is general, the equations are sort-preserving and we obtain a sort-preserving rewrite system. Thus, we do not have to worry about generating ill-typed terms when rewriting. That is why the ordering $\preceq$ can disregard sort information.

We shall use the total simplification ordering $\prec$ on $\mathcal{T}_\Sigma$ to define a well-founded partial order $\prec$ on ground $\Sigma$-substitutions. For this, it is assumed that the set of variables $X$ is totally ordered as $X = \langle x_1, x_2, \ldots, x_n, \ldots \rangle$.

**Definition 7.1** The partial order $\prec$ is defined on ground $\Sigma$-substitutions as follows. Given any two ground $\Sigma$-substitutions $\sigma$ and $\theta$ such that $D(\sigma) = D(\theta)$, letting $\langle y_1 \ldots, y_n \rangle$ be the sequence obtained by ordering the variables in $D(\sigma)$ according to their order in $X$, then $\sigma \prec \theta$ iff

$$\langle \sigma(y_1), \ldots, \sigma(y_n) \rangle \preceq_{lex} \langle \theta(y_1), \ldots, \theta(y_n) \rangle,$$

where $\preceq_{lex}$ is the lexicographic ordering on tuples induced by $\preceq$.

Since $\preceq$ is well-founded and $\prec$ is induced by the lexicographic ordering $\preceq_{lex}$ which is well-founded, $\prec$ is also well-founded. In fact, given any finite set $V$ of variables, note that $\prec$ is a total well-founded ordering for the set of ground $\Sigma$-substitutions with domain $V$.

We utilize a total simplification ordering $\preceq$ on ground terms, to define a notion minimal rigid $(\Sigma, E)$-unifiers. Following [GNPS90], we define an ordering among ground $\Sigma$-unifiers in which minimal elements do exist.

**Definition 7.2** Let $E$ be a set of general equations (over $\mathcal{T}_\Sigma(X)$) and $u, v \in \mathcal{T}_\Sigma(X)$ any two terms. For any ground rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$, let

$$S_{E,u,v,\theta} = \{\rho \mid D(\rho) = D(\theta),\ \rho(u) \overset{*}{\cong}_{\rho(E)} \rho(v),\ \rho \sqsubseteq_E \theta,\ \text{and } \rho \text{ ground}\}.$$

Obviously, $\theta \in S_{E,u,v,\theta}$, so $S_{E,u,v,\theta}$ is not empty. Since $\prec$ is total and well-founded on ground $\Sigma$-substitutions with domain $D(\theta)$, the set $S_{E,u,v,\theta}$ contains some least element $\sigma$ (w.r.t. $\prec$).

We define the notion of *rigid equivalency*.

**Definition 7.3** Given two sets $E$ and $E'$ of equations, we say that $E$ and $E'$ are *rigid equivalent* iff for every two terms $u$ and $v$, $u \overset{*}{\cong}_E v$ iff $u \overset{*}{\cong}_{E'} v$ (treating $E$ and $E'$ as sets of ground equations).

**Lemma 7.4** If $E$ and $E'$ are rigid equivalent then $S_{E,u,v,\theta} = S_{E',u,v,\theta}$.

**Proof:** Since $E$ and $E'$ are rigid equivalent, so are $\rho(E)$ and $\rho(E')$ for any $\Sigma$-substitution $\rho$. Hence for any terms $u$ and $v$, $\rho(u) \overset{*}{\cong}_{\rho(E)} \rho(v)$ iff $\rho(u) \overset{*}{\cong}_{\rho(E')} \rho(v)$. □

We shall now state a result from [GNPS90], but first we define *degenerate equations*.

**Definition 7.5** A *degenerate equation* is an equation of the form $x \doteq t$, where $x$ is a variable and $x \notin Var(t)$, and a *nondegenerate equation* is an equation that is not degenerate.

**Lemma 7.6** Let $E$ be a set of equations (over $\mathcal{T}_\Sigma(X)$) and $u, v \in \mathcal{T}_\Sigma(X)$ any two terms. For any ground rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$, if $\sigma$ is the least element of the set $S_{E,u,v,\theta}$ of definition 7.2, then the following properties hold:

1. every term of the form $\sigma(x)$ is irreducible by every oriented instance $\sigma(l) \rightarrow \sigma(r)$ of a nondegenerate equation $l \doteq r \in E \cup E^{-1}$, and
2. every proper subterm of a term of the form $\sigma(x)$ is irreducible by every oriented instance $\sigma(l) \rightarrow \sigma(r)$ of a degenerate equation $l \doteq r \in E \cup E^{-1}$.

In view of lemma 7.6, it is convenient to introduce the following definition.

**Definition 7.7** Given a set $E$ of equations, a total simplification ordering $\preceq$ on ground terms, and any two terms $u$, $v$, a ground rigid $E$-unifier $\theta$ of $u$ and $v$ is *reduced w.r.t.* $\theta(E)$ iff

1. every term of the form $\theta(x)$ is irreducible by every oriented instance $\theta(l) \rightarrow \theta(r)$ of a nondegenerate equation $l \doteq r \in E \cup E^{-1}$, and
2. every proper subterm of a term of the form $\theta(x)$ is irreducible by every oriented instance $\theta(l) \rightarrow \theta(r)$ of a degenerate equation $l \doteq r \in E \cup E^{-1}$.

We have the following lemma as a combination of lemmata 7.4, 7.6 and the existence of minimal elements in $S_{E,u,v,\theta}$.

**Lemma 7.8** Let $E$ be a set of general equations (over $\mathcal{T}_\Sigma(X)$) and $u, v \in \mathcal{T}_\Sigma(X)$ any two terms. For any ground rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$, if $\sigma$ is the least element of the set $S_{E,u,v,\theta}$ of definition 7.2, then $\sigma$ is reduced with respect to $\sigma(E')$ for any set $E'$ rigid equivalent to $E$.

Given this and the remark on *pure* methods at the of section 6, we will assume for the rest of this chapter that
rigid $(\Sigma, E)$-unifiers are ground and reduced. The next lemma shows why reduced substitutions are interesting.

**Lemma 7.9** Let $t \in \mathcal{T}_\Sigma$, $l \doteq r \in E$, and let $\theta$ be a ground $\Sigma$-substitution that is reduced w.r.t. $\theta(E)$. Suppose that $\theta(t) \rightarrow_{\beta,\theta(l \doteq r)} t''$. Let $t' = t[\beta \leftarrow r]$. Then

1. $\beta$ occurs inside $t$, i.e. $\beta \in Dom(t)$, and

2. $t' \in \mathcal{T}_\Sigma$ and $t'' = \theta(t')$.

The proof is given in appendix A.4.

This lemma is important because it shows that pieces of a rigid $(\Sigma, E)$-unifier of $u$ and $v$ can be tracked down to the terms in $\{E, u, v\}$. By an inductive argument on the length of rewrite proofs, we obtain the following corollary.

**Corollary 7.10** Consider a rewrite proof of the form

$$\theta(u_0) \xleftrightarrow{*}_{\beta_1,\theta(E)} u'_1 \xleftrightarrow{*}_{\beta_2,\theta(E)} u'_2 \xleftrightarrow{*}_{\beta_3,\theta(E)} \cdots \xleftrightarrow{*}_{\beta_n,\theta(E)} u'_n.$$

For $1 \leq i \leq n$ let $u_i = u_{i-1}[\beta_i \leftarrow l_i] = u_0[\beta_1 \leftarrow r_1, \ldots \beta_i \leftarrow r_i]$. Then

$$\theta(u_0) \xleftrightarrow{*}_{\beta_0,\theta(E)} \theta(u_1) \xleftrightarrow{*}_{\beta_1,\theta(E)} \theta(u_2) \xleftrightarrow{*}_{\beta_2,\theta(E)} \cdots \xleftrightarrow{*}_{\beta_{n-1},\theta(E)} \theta(u_n).$$

Furthermore, for $1 \leq i \leq n$, $u'_i = \theta(u_i)$ and $\beta_i \in Dom(u_i)$.

# 8 Finding Reduced Sets of Rewrite Rules

Rewrite systems are like equations except that they clearly specify a left and a right hand side. Rewriting specifies an operational semantics that can be used for equality steps. As opposed to equations, rewrite rules specify direction which can be used to define *normal forms*. These normal forms are interesting because they state a type of *finalizing* condition which we need to ensure progress at each step of the rigid E-unification method we present in section 9.

We formally define some of these concepts before presenting the results.

**Definition 8.1** Let $\longrightarrow$ be a binary relation $\longrightarrow \subseteq T_\Sigma(X) \times T_\Sigma(X)$ on terms. The relation $\longrightarrow$ is *monotonic* iff for every two terms $s, t$ and every function symbol $f$, if $s \longrightarrow t$ then $f(\ldots, s, \ldots) \longrightarrow f(\ldots, t, \ldots)$. The relation $\longrightarrow$ is *stable* (under substitution) if $s \longrightarrow t$ implies $\sigma(s) \longrightarrow \sigma(t)$ for every substitution $\sigma$.

**Definition 8.2** When a pair $(s, t) \in E$ is used as an oriented equation (from left to right), we call it a *rule* and denote it as $s \dot{\rightarrow} t$. The *reduction relation* $\longrightarrow_E$ is the smallest stable and monotonic relation that contains $E$. We can define $t_1 \longrightarrow_E t_2$ explicitly as above the only difference being that $(s, t)$ is a variant of a pair in $E$ (and not in $E \cup E^{-1}$). When $t_1 \longrightarrow_E t_2$, we say that $t_1$ *rewrites* to $t_2$, or that we have a *rewrite step*. When we want to fully specify a rewrite step, we use the following notation.

$$t_1 \longrightarrow_{\alpha, s \dot{\rightarrow} t, \sigma} t_2$$

Some of the arguments $\alpha, s \dot{\rightarrow} t$ or $\sigma$ may be omitted. This notation means that tree $t_1$ is rewritten at address $\alpha$ using rewrite rule $s \dot{\rightarrow} t$ and substitution $\sigma$ to obtain tree $t_2$.

When $Var(r) \subseteq Var(l)$, then a rule $l \dot{\rightarrow} r$ is called a *rewrite rule*; a set of such rules is called a *rewrite system*.

**Definition 8.3** Consider a ground term rewriting system $R$. $R$ is *noetherian* iff there exists no infinite sequence of terms $t_1, t_2, t_3, \ldots$ such that $t_1 \rightarrow_R t_2 \rightarrow_R t_3 \rightarrow_R \ldots$, and it is *confluent* iff whenever $t_1 \overset{*}{\longleftrightarrow}_R t_2$, there exists a term $t_3$ such that $t_1 \overset{*}{\longrightarrow}_R t_3 \ {}_R\overset{*}{\longleftarrow} t_2$. $R$ is *canonical* iff it is noetherian and confluent.

A term $t$ is *irreducible* by $R$ (or in *normal form*) if there exists no $t'$ such that $t \rightarrow_R t'$.

A system $R$ is *left-reduced* iff for every $l \dot{\rightarrow} r \in R$, $l$ is irreducible by $R - \{l \dot{\rightarrow} r\}$; $R$ is *right-reduced* iff for every $l \dot{\rightarrow} r \in R$, $r$ is irreducible by $R$. $R$ is called *reduced* iff it is left-reduced and right-reduced.

## 8.1 Ground Equations

Snyder [Sny89] presents an $O(n \log n)$ method for *compiling* ground equations into reduced sets of rewrite rules. For example, if $E = \{f^3(a) \doteq a, f^2(a) \doteq a, g(c) \doteq f(a), g(h(a)) \doteq g(c), c \doteq h(a), b \doteq m(f(a))\}$ then $R = \{f(a) \rightarrow a,\ g(c) \rightarrow a,\ m(a) \rightarrow b,\ h(a) \rightarrow c\}$ is reduced equivalent to $E$.

Snyder's method computes $R$ by first computing the congruence closure of $E$, rewriting some terms using congruent subterms and selecting representatives for each congruence class.

Since general equations are sort preserving, term rewriting modulo $E$ is sound since it does not violate sort constraints. Similarly rewriting must preserve the set of variables and satisfy the *variable renaming property* hence given a set $E$ of general equations, any equivalent set $R$ of rewrite rules produced by Snyder's algorithm is also general.

We expand the method to systems which contain variables when we regard these as frozen. Hence if the equations are order-sorted and general, so is the resulting reduced set of rewrite rules. This justifies the use of an unsorted algorithm to interreduce sets of $\Sigma$-equations. The complexity of Snyder's algorithm is $O(nlogn)$ where $n$ is the size of the system of equations in DAG format. The method is nondeterministic in that it produces some reduced set of ground rewrite rules. If we denote the reduction procedure by $\Rightarrow_{\mathcal{R}}$ we can state the following results.

**Lemma 8.4** If $E$ is a set of general equations and $E \Rightarrow_{\mathcal{R}} R'$, then $R'$ is also general. In particular all terms in $R'$ are $\Sigma$-terms.

**Theorem 8.5** [Soundness (Snyder)] For any set of ground equations $E$, if $E \Rightarrow_{\mathcal{R}} R'$, then $\xleftrightarrow{*}_{R'} \; = \; \xleftrightarrow{*}_{E}$ .

**Theorem 8.6** [Completeness (Snyder)] For any set $E$ and for any reduced ground term rewriting system $R'$ equivalent to $E$, $E \Rightarrow_{\mathcal{R}} R'$.

## 8.2 Non-ground Equations

Snyder's method handles only the ground case. We are interested in extending the reduction procedure to systems of equations containing variables, but we regard those variables as frozen, i.e. as constants over an extended signature. The method and all the results adapt themselves without difficulty to this case. We restate some of the results in these terms.

Let us recall the notion of rigid equivalence given in definition 7.3 on page 31.

Given two sets $E$ and $E'$ of equations, we say that $E$ and $E'$ are *rigid equivalent* iff for every two terms $u$ and $v$, $u \overset{*}{\cong}_E v$ iff $u \overset{*}{\cong}_{E'} v$ (treating $E$ and $E'$ as sets of ground equations).

It is clear that if $E$ and $E'$ are rigid equivalent, then for every $\Sigma$-substitution $\theta$, $\theta(E)$ and $\theta(E')$ are rigid equivalent. The soundness result now reads as follows.

**Theorem 8.7** If $E \Rightarrow_{\mathcal{R}} R'$ then viewing $R'$ as an equation system, $E$ and $R'$ are rigid equivalent.

**Definition 8.8** A strict ordering $\prec$ has the *subterm property* iff $s \prec f(\ldots, s, \ldots)$ for every term $f(\ldots, s, \ldots)$ (since we are considering symbols having a fixed rank, the deletion property is superfluous, as noted in Dershowitz [Der87]). A *simplification ordering* $\prec$ is a strict ordering that is monotonic and has the subterm property. A *reduction ordering* $\prec$ is a strict ordering that is monotonic, stable, and such that $\succ$ is well founded. With a slight abuse of language, we will also say that the converse $\succ$ of a strict ordering $\prec$ is a simplification ordering (or a reduction ordering). It is shown in Dershowitz [Der87] that there are simplification orderings that are total on ground terms.

We are interested in obtaining a reduced system which is compatible with respect to a given ordering. That is, where the rules are oriented such that if $l \rightarrow r \in R$, then $r \preceq l$. We develop this now. First we notice that although we do not know exactly how to produce a reduced system compatible with a given ordering, such a reduction does exist.

**Theorem 8.9** [Completeness with respect to $\preceq$] Let $E$ be a set of $\Sigma(X)$-equations (i.e. the terms in the equations are in $\mathcal{T}_\Sigma(X)$), and let $\preceq$ be a total simplification ordering on $\mathcal{T}_\Sigma(X)$. Then there exists a reduced set $R'$ of $\Sigma$-rewrite rules compatible with $\preceq$ such that $E \Rightarrow_{\mathcal{R}} R'$.
**Proof:** Gallier, Narendran, Plaisted, Raatz and Snyder [GNP+92] present the desired rigid equivalent set of rewrite rules $R'$. By theorem 8.6 $E \Rightarrow_{\mathcal{R}} R'$. □

We now show how to obtain total simplification orderings on terms with variables. The following definition is an extension of one appearing in [GNPS90]. There, a total simplification ordering is defined on the set of subterms of an equation system. We extend this by defining a total simplification ordering on the whole term algebra $\mathcal{T}_\Sigma(X)$. This ordering becomes crucial when showing the completeness of the method for finding rigid $(\Sigma, E)$-unifiers. In [GNPS90], portions of this ordering are guessed and then extended. Although our approach deals with an infinite ordering, our method never has to guess any portion of it. We simply need to know its existence.

**Definition 8.10** Given a ground $\Sigma$-substitution $\theta$ and a total simplification ordering $\prec$ on ground $\Sigma$-terms, the total simplification ordering $\prec_\theta$ on $\mathcal{T}_\Sigma(X)$ is defined as follows.
First, arbitrarily define a total ordering on the set of variables $X$. For example pick some enumeration of the variables, if $X = \{x_1, \ldots, x_i, \ldots\}$ define

$$x_i \preceq' x_j \text{ if } i \leq j.$$

Extend $\preceq'$ by stating that a variable is less than any non-variable term:

$$x \preceq' t \text{ whenever } x \in X \text{ and } t \notin X.$$

Now, we define $\prec'_\theta$ recursively as follows: given $\Sigma$-terms $u$ and $v$, $u \prec'_\theta v$ iff either

**(1)** $\theta(u) \prec \theta(v)$, or

**(2)** $\theta(u) = \theta(v)$, and either

**(2a)** $u$ is a variable and $u \prec' v$, or

**(2b)** $u = f(u_1, \ldots, u_n)$, $v = f(v_1, \ldots, v_n)$, and $\langle u_1, \ldots, u_n \rangle (\prec'_\theta)^{lex} \langle v_1, \ldots, v_n \rangle$, where $(\prec'_\theta)^{lex}$ is the lexicographic extension of $\prec'_\theta$.

Consider the reflexive transitive closure of $\prec'_\theta$ and denote it by $\preceq_\theta$.

We claim that $\preceq_\theta$ is a total ordering on $\mathcal{T}_\Sigma(X)$ that is monotonic and has the subterm property. The only problem is in showing that $\preceq_\theta$ is total, as the other conditions are then easily verified. The proof is given in the appendix A.5.

In view of theorem 8.9 we have the following corollary:

**Corollary 8.11** Let $E$ be a set of equations and $\theta$ a ground $\Sigma$-substitution. There exists a rigid reduced Rewrite System $R'$ compatible with $\preceq_\theta$ such that $E \Rightarrow_{\mathcal{R}} R'$. Furthermore, $R'$ can be computed in non-deterministic $nlog(n)$ time.

# 9 Finding Complete Sets of Rigid $(\Sigma, E)$-Unifiers

In this section we develop an order-sorted method to find rigid $(\Sigma, E)$-unifiers for systems $E$ of general equations. The method is intrinsically order-sorted in that each of its components is order-sorted and the central component of the method, namely the reduction of peaks, is performed in such a way that a piece of an order-sorted rigid $(\Sigma, E)$-unifier is created. We compare our approach to the one taken by Meseguer, Goguen and Smolka in [MGS89] where an unsorted algorithm is used to come up with a complete set of unsorted $E$-unifiers. Then a complete set of order-sorted $(\Sigma, E)$ unifiers is produced by using the sort information. We could take a similar approach here by using the algorithm presented by Gallier, Narendran, Plaisted and Snyder in [GNPS90]. They present an NP procedure to generate complete sets of unsorted Rigid $E$-Unifiers. We could first run the unsorted algorithm and then use the sort information to produce a complete family of order-sorted rigid $E$-unifiers. The disadvantage of this approach is that it does not make full use of the sort information. If $u$ and $v$ are rigid $(\Sigma, E)$-unifiable then $\theta(E) \vdash_\Sigma \theta(u) \doteq \theta(v)$. Since $E$ is general, so are $\theta(E), \theta(u)$ and $\theta(v)$.

Hence $LS(\theta(u)) = LS(\theta(v))$. Since $\theta$ is a $\Sigma$-substitution, $LS(\theta(u)) \leq LS(u)$ and $LS(\theta(v)) \leq LS(v)$. Therefore, unless $u$ and $v$ have a common subsort, they have no rigid $(\Sigma, E)$-unifier. The method described above would first run the $NP$ unsorted algorithm and then, upon discovering that the family of sort assignments is empty, return **failure**.

Our method is intrinsically order-sorted. We modify the unsorted method for finding rigid $E$ unifiers to a method that builds order-sorted substitutions. Since the sort information is used at each and every step of the order-sorted algorithm, it detects failure due to sort conflicts at an earlier stage. At the heart of our method is the algorithm for finding families of order-sorted unifiers in triangular form described in section 4 which produces complete families of order-sorted unifiers in triangular form. Those $\Sigma$-unifiers have two properties that are needed for our method to work: they are idempotent and variable decreasing.

We have also improved upon the unsorted algorithm of [GNPS90] by providing an alternative way of dealing with the problem of orienting the equations. We show that it is possible to simply guess an orientation. Thus we manage to remove *order assignments* from the unsorted method. This improvement also applies to the unsorted case, it substantially clarifies the method and places the role of the orientation of rewrite rules in its proper place. Without entering into too much detail, *order assignments* are guesses of finite portions of a simplification ordering on $\Sigma(X)$-terms. They provide an orientation to the equations in $E$ so that by looking at them as rewrite rules one can, via overlaps, discover pieces of a rigid $(\Sigma, E)$-unifier. By using the procedure to find reduced sets of rewrite rules equivalent to $E$ presented in section 8 and by imposing a total simplification ordering on the algebra $\mathcal{T}_\Sigma(X)$ we manage to do without guessing any portion of the ordering. We simply use the fact that such an ordering exists and that the reduction procedure is complete (corollary 8.11). Our method uses the reduction procedure of section 8 and a single transformation on certain systems defined next. Recall that we are assuming $E$ to be a set of general equations. The following definition is needed.

**Definition 9.1** Given a set $E$ of general equations and some equation $l \doteq r$, the set of equations obtained from $E$ by deleting $l \doteq r$ and $r \doteq l$ from $E$ is denoted by $(E - \{l \doteq r\})^\dagger$. Formally, we let $(E - \{l \doteq r\})^\dagger = \{u \doteq v \mid u \doteq v \in E,\ u \doteq v \neq l \doteq r, \text{ and } u \doteq v \neq r \doteq l\}$. Notice that if $E$ is general so is $(E - \{l \doteq r\})^\dagger$.

Intuitively, the method we present works on three different issues simultaneously. First one tries to find a peak-free proof of $\theta(u) \xleftrightarrow{*}_{\theta(E)} \theta(v)$ by applying some transformations to $E$ in order to obtain an equivalent system $E'$ which is reduced in which there is a valley proof $\theta(u) \xrightarrow{*}_{\theta(E')} w \;{}_{\theta(E')}\!\xleftarrow{*} \theta(v)$. Then one tries to reduce $u$ in the guessed system $E'$, or alternatively, one tries to reduce $v$ in $E'$. If a common element is obtained as a reduction from $u$ and $v$ we are done, otherwise the system $E'$ is transformed by guessing a piece of the rigid $(\Sigma, E)$-unifier of $u$ and $v$ into another equivalent system $E''$ with fewer variables. However, the proof $\theta(u) \xleftrightarrow{*}_{\theta(E'')} \theta(v)$ might not be a valley proof, hence the process restarts. The reason it terminates is because in each iteration the number of variables in the system decreases. There is an NP procedure [Koz76, Koz77] for the base case with no variables, i.e. $\theta(E) = E$.

In order to avoid having three different types of transformations (on $E$, on $u$ and on $v$) the method combines all these into one single apparatus by adding special equations involving $u$ and $v$. These allow for the reductions of $u$ and $v$ to be done as part of the transformations on the system $E$ and they also act as markers to determine when the method has been successful. We extend the signature $\Sigma$ of $E$ to include function names for these markers and the new equations. The markers are the function symbols $eq$, $T$ and $F$. The equations are $eq(u, v) \doteq F(u, v)$ and $eq(z, z) \doteq T(z)$. [6] The idea is that at some point $eq(u, v)$ and $eq(z, z)$ will unify and this will result in a rigid $(\Sigma, E)$-unifier of $u$ and $v$. We face the question of assigning sorts to the new symbols.

We explained previously that if $u$ and $v$ have no common subsort there can be no rigid $(\Sigma, E)$-unifier for $u$ and $v$. If we denote by $LBd(S)$ the set of lower bounds for the elements of a poset $S$, the last sentence states that $LBd(\{LS(u), LS(v)\})$ cannot be empty. The first step of the order-sorted method is to determine whether $LBd(\{LS(u), LS(v)\})$ is empty. If it is then it returns **failure**, otherwise a member $s$ of $LBd(\{LS(u), LS(v)\})$ is guessed. This sort $s$ is a guess of the solution's sort, i.e. $LS(\theta(u)) = LS(\theta(v))$. Notice that failure can be detected due to sorts conflict at this early stage [7]. Given $s$ one defines the order-sorted signature $\Sigma^s$ by adding to $\Sigma$ the following

[6] We use $F(u, v)$ and $T(z)$ instead of $F$ and $T$ as in [GNPS90] in order to keep the set of equations general.
[7] This can be strengthened by replacing $u$ by $IP(u, s)(u)$ and $v$ by $IP(v, s)(v)$.

1. a new sort $EQ$,

2. a new function symbol $T : s \mapsto EQ$,

3. a new function symbol $F : LS(u) \cdot LS(v) \mapsto EQ$, and

4. a new function symbol $eq : LS(u) \cdot LS(v) \mapsto EQ$.

Given $E$, a set of equations over $\mathcal{T}_\Sigma(X)$ , let $z \in X_s$ be a variable not occurring in $E$. We consider finite sets of equations of the form

$$E_{u,v} = E \ \cup \ \{eq(u,v) \doteq F(u,v), eq(z,z) \doteq T(z)\}$$

where $u, v \in \mathcal{T}_\Sigma(X)$. Notice that $eq(u,v) \doteq F(u,v)$ and $eq(z,z) \doteq T(z)$ are general. tcomment because (for $eq(u,v) \doteq F(u,v)$),

1. $LS(eq(u,v)) = EQ = LS(F(u,v))$,

2. $Var(eq(u,v)) = Var(F(u,v))$, and

3. for any variable renaming $\rho$, $LS(\rho(eq(u,v))) = EQ = LS(\rho(F(u,v)))$.

Similarly, $eq(z,z) \doteq T(z)$ is general. Hence, if $E$ is general, so is $E_{u,v}$. Notice that the choice of $\Sigma^s$ is nondeterministic because $s$ is not uniquely specified. As long as every member of $LBd(LS(u), LS(v))$ can be picked in polynomial time, our algorithm will remain in NP. For $\Sigma$ finite this is, of course, the case.

The next lemma shows that one can use the system $E_{u,v}$ to find rigid $(\Sigma, E)$-unifier of $u$ and $v$ provided no extraneous terms are introduced in the process.

**Lemma 9.2** A $\Sigma$-substitution $\theta$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$ iff there is some sort $s$ and some $\Sigma^s$-substitution $\theta'$ such that

1. $\theta'$ is over $\mathcal{T}_\Sigma(X)$, i.e. none of the new symbols are used in $\theta'$,

2. $\theta = \theta'|_{D(\theta')-\{z\}}$ and

3. $\theta'$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $T(z)$ and $F(u,v)$.

The proof is given in appendix A.6.

We are now ready to present the method. It is based upon a single transformation which is similar to the one presented in [GNPS90] but does without the order assignment and uses a different reduction procedure.

**Definition 9.3** We define a nondeterministic transformation on systems of the form $\langle S, E\rangle$, where $S$ is a term system and $E$ is a set of equations as above:

$$\langle S,\ E\rangle\ \Rightarrow_{l_1 \doteq r_1, l_2 \doteq r_2, \beta, \sigma_T} \langle S',\ E'\rangle,$$

where $l_1 \doteq r_1,\ l_2 \doteq r_2 \in E \cup E^{-1}$, either $l_1/\beta$ is *not* a variable or $l_2 \doteq r_2$ is degenerate, $l_1/\beta \neq l_2$, $TU(l_1/\beta, l_2)$ represents a member of $CSU_{\Sigma^s}(l_1/\beta, l_2)$, which is a $\Sigma^s$-substitution over $\mathcal{T}_\Sigma(X)$, in special triangular form, [8] $\sigma_T = [t_1/x_1, \ldots, t_p/x_p]$ where $TU(l_1/\beta, l_2) =$ $= \{\langle x_1, t_1\rangle, \ldots, \langle x_p, t_p\rangle\}$,

$$E'' = \sigma_T((E - \{l_1 \doteq r_1\})^\dagger \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\}),$$

$S' = S \cup TU(l_1/\beta, l_2)$, and $E'' \Rightarrow_{\mathcal{R}} E'$.

The triangular form $TU(l_1/\beta, l_2)$ is obtained by running the non-deterministic quasi-linear algorithm $CTU$ described in section 4.3 which returns either a triangular form or fails. If it fails, the transformation fails. Notice that, due to the nature of the equations, one can restrict $CSU_{\Sigma^s}(l_1/\beta, l_2)$ to a set of substitutions over $\mathcal{T}_\Sigma(X)$ instead of $\mathcal{T}_{\Sigma^s}(X)$, and obtain a set which is complete for all $\Sigma^s$-unifiers over $\mathcal{T}_\Sigma(X)$. Therefore, $\sigma_T$ satisfies condition 1 of lemma 9.2.

Also note that the rigid reduced system $E'$ is obtained nondeterministically from $E''$. The non-determinism is introduced by the $CTU$ procedure as explained above and by the non-deterministic nature of reduction procedure $\mathcal{R}$. The idea is that some $E'$ will be compatible with the orientation imposed by $\theta$. In essence, this is a guess of the orientation $\preceq_\theta$ imposed by $\theta$ on $E$.

Notice that we do not apply a unifier $\sigma$ in the transformation, but its associated $\Sigma$-substitution $\sigma - T$. This guarantees that the size of the system being transformed does not grow too much. As a matter of fact, since $\sigma_T$ only uses terms already appearing in the system, it can

[8] Note that we are requiring that $l_1/\beta$ and $l_2$ have a *nontrivial* $\Sigma$-unifier. The triangular form of $\Sigma$-unifiers is important for the NP-completeness of this method.

be implemented by moving pointers in a DAG, hence the system which results from applying $\sigma_T$ is at worst as large as the original one. This plays a significant role in placing our method in $NP$.

Although $\sigma_T(l_1[\beta \leftarrow r_2] \doteq r_1)$ looks like a critical pair of equations in $E \cup E^{-1}$, it is not. This is because a critical pair is formed by applying the order-sorted unifier of $l_1/\beta$ and $l_2$ to $l_1[\beta \leftarrow r_2] \doteq r_1$, but $[t_1/x_1, \ldots, t_p/x_p]$ is usually not a unifier of $l_1/\beta$ and $l_2$. It is the composition $[t_1/x_1]\,; \ldots\,; [t_p/x_p]$ that is a unifier of $l_1/\beta$ and $l_2$. In addition note that in general, a / $\tau$ associated with the triangular form of a unifier of $l_1/\beta$ and $l_2$ does not have to preserve sorts, i.e. $LS(\tau(l_1/\beta)$ and $LS(\tau(l_2)$ do not necessarily have to agree. The reason for using *special* triangular forms is to take care of this problem.

**Lemma 9.4** Let $E$ be a system of general $\Sigma$-equations and $S$ a set of pairs of the form $< x, t >$ with $t \in \mathcal{T}_\Sigma(X)$ and $LS(t) \leq LS(x)$.

Suppose that $\langle S, E\rangle \Rightarrow \langle S', E'\rangle$, then

1. all pairs in $S'$ are of the form $< x, t >$ with $x$ a variable, $t \in \mathcal{T}_\Sigma(X)$ and $LS(t) \leq LS(x)$.
2. $E'$ is a set of general equations, in particular its terms are well sorted, and
3. for any $\Sigma$-unifier $\varphi$ of $S'$, $\varphi(E)$ and $\varphi(E')$ are rigid equivalent.

See the proof in appendix A.7.

By iterating lemma 9.4 we can prove by induction the following.

**Lemma 9.5** Suppose that $\langle \emptyset, E\rangle \Rightarrow^+ \langle S', E'\rangle$, then,

1. $S'$ consists of pairs of the form $< x, t >$ with $x$ a variable, $t \in \mathcal{T}_\Sigma(X)$ and $LS(t) \leq LS(x)$ (in particular $S'$ consists of $\Sigma$-terms).
2. $E'$ is a set of general equations, in particular of order-sorted equations, and
3. for any $\Sigma$-unifier $\varphi$ of $S'$, $\varphi(E)$ and $\varphi(E')$ are rigid equivalent.

For the previous lemma to hold it is fundamental that the evolving equation system remains general, because that guarantees that all terms are order-sorted hence the substitution being built in $S$ is a $\Sigma$-substitution.

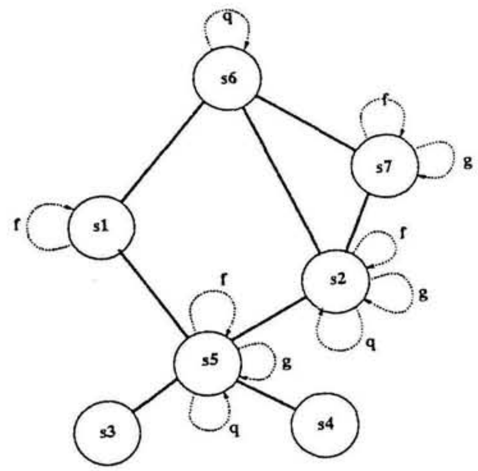


Figure 7: The signature $\Sigma$.

Given a finite coherent order-sorted signature $\Sigma$, a set $E$ of general $\Sigma$-equations and two $\Sigma$-terms $u$ and $v$, the method to find rigid $(\Sigma, E)$-unifier for $u$ and $v$ is the following.

## Method

If $Lbd(LS(u), LS(v))$ is empty announce **failure**. Otherwise non-deterministically pick $s \in Lbd(LS(u), LS(v))$. Construct the signature $\Sigma^s$ and the set $E_{u,v}$ of general $\Sigma^s$-equations. Find a reduced set $E_0$ of general rewrite rules equivalent to $E_{u,v}$ by running the nondeterministic procedure $\mathcal{R}$, i.e. $E \Rightarrow_{\mathcal{R}} E_0$. Let $m$ the total number of variables in $E_0$, and $V = Var(E) \cup Var(u, v)$. For any sequence $\langle \emptyset, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle$ consisting of at most $m$ transformation steps, where $k \leq m$, if the non-deterministic algorithm for $CSU_{\Sigma^s}(S_k)$ (over $\mathcal{T}_\Sigma(X)$) produces a $\Sigma$-unifier $\theta_{S_k}$, and $k$ is the first integer in the sequence such that $F(w, w) \doteq T(w') \in E_k$ for some $w, w' \in \mathcal{T}_\Sigma(X)$ of sort $s$, return the $\Sigma$-substitution $\theta_{S_k}|_V$.

We shall prove that the finite set of all $\Sigma$-substitutions returned by our method forms a complete set of rigid $(\Sigma, E)$-unifiers $u$ and $v$. In particular, the method provides a decision procedure that is in NP. But first let us show how the method works via an example.

Consider the coherent signature $\Sigma$ of figure 7.

In order to facilitate the notation we will denote the variables by the letter $z$ with a subscript to indicate its sort. For example $z_3$ is a variable of sort $s_3$.

Let $E = \{g(f(z_7)) \doteq f(z_7), g(f(z_2)) \doteq q(z_2)\}$. Consider the question of finding a rigid $(\Sigma, E)$-unifier of the $\Sigma$-terms $u = q(z_6)$ and $v = f(z_1)$.

First we guess a sort below the least sorts of $u$ and $v$. Let $s_3$ be our guess. We construct the set of general equations $E_{u,v}$ over $\Sigma^{s_3}$ as follows:

$$E_{u,v} = E \cup \{eq(z_3, z_3) \doteq T(z_3), eq(q(z_6), f(z_1)) \doteq F(q(z_6), f(z_1))\}.$$

**1)** The reduction procedure does not change the set, it just orients it as the equations are written above. We obtain $E_0$:

$$\begin{aligned} E_0 \ = \{ \ & g(f(z_7)) \rightarrow f(z_7), \\ & g(f(z_2)) \rightarrow q(z_2), \\ & eq(z_3, z_3) \rightarrow T(z_3), \\ & eq(q(z_6), f(z_1)) \rightarrow F(q(z_6), f(z_1))\} \end{aligned}$$

**2)** There is an overlap between the first two rules at the root. Let $\sigma_1 = [z_7/z_2]$, then $TU(g(f(z_7)), g(f(z_2))) = [< z_7, z_2 >]$ and $\sigma_{T,1} = \sigma_1$. By applying $\sigma_{T,1}$ to the system resulting from the overlap we obtain:

$$\begin{aligned} E_1' \ = \{ \ & q(z_2) \doteq f(z_2), \\ & g(f(z_2) \doteq q(z_2), \\ & eq(z_3, z_3) \rightarrow T(z_3), \\ & eq(q(z_6), f(z_1)) \rightarrow F(q(z_6), f(z_1))\}. \end{aligned}$$

**3)** We reduce the second equation to obtain

$$\begin{aligned} E_1 \ = \{ \ & f(z_2) \rightarrow q(z_2), \\ & g(q(z_2) \rightarrow q(z_2), \\ & eq(z_3, z_3) \rightarrow T(z_3), \\ & eq(q(z_6), f(z_1)) \rightarrow F(q(z_6), f(z_1))\} \end{aligned}$$

and $S_1 = \{< z_7, z_2 >\}$.

**4)** There is an overlap between the fourth and the first rules. A unifier of $f(z_1)$ and $f(z_2)$ is chosen: $\sigma_1 = [z_1/z_5, z_2/z_5]$. The resulting set of equations is already reduced:

$$\begin{aligned} E_2 \ = \{ \ & f(z_5) \to q(z_5), \\ & g(q(z_5) \to q(z_5), \\ & eq(z_3, z_3) \to T(z_3), \\ & eq(q(z_6), q(z_5)) \to F(q(z_6), q(z_5))\} \end{aligned}$$

and $S_2 = \{< z_2, z_5 >, < z_1, z_5 >, < z_7, z_2 >\}$.

**5)** We overlap the last two rewrite rules using the unifier $\sigma_2 = [z_3/q(z_3'), z_5/z_3', z_6/z_3']$. We need to compute a triangular form $TU(eq(z_3, z_3), eq(q(z_6), q(z_5)))$. One such triangular form is given by $\{< z_3, q(z_3') >, < z_5, z_3' >, < z_6, z_3' >\}$ where $z_3'$ is a *new* variable of sort $s3$. We obtain

$$\begin{aligned} E_3' \ = \{ \ & f(z_3') \doteq q(z_3') \\ & g(q(z_3') \doteq q(z_3') \\ & F(q(z_3'), q(z_3')) \doteq T(q(z_3')), \\ & eq(q(z_3'), q(z_3')) \doteq F(q(z_3'), q(z_3'))\}. \end{aligned}$$

This system is already reduced, thus we have

$$\begin{aligned} E_3 \ = \{ \ & f(z_3') \to q(z_3') \\ & g(q(z_3') \to q(z_3') \\ & F(q(z_3'), q(z_3')) \to T(q(z_3')), \\ & eq(q(z_3'), q(z_3')) \to F(q(z_3'), q(z_3'))\}. \end{aligned}$$

We have $S_3 = \{< z_3, q(z_3') >, < z_5, z_3' >, < z_6, z_3' >, < z_2, z_5 >, < z_1, z_5 >, < z_7, z_2 >\}$.

Now, we managed to obtain an equation of the form $T(w') \doteq F(w, w)$, thus the method stops. We can find a $\Sigma$-unifier $\theta_1$ of $S_3$, $\theta_1 = [z_1/z_3', z_2/z_3', z_3/q(z_3'), z_5/z_3', z_6/z_3', z_7/z_3']$. Restricted to the variables in $E_{u,v}$ we obtain:

$$\theta' = [z_1/z_3', z_2/z_3', z_6/z_3', z_7/z_3'].$$

And indeed:

$$\theta'(u) = \theta'(q(z_6)) = q(z_3')\ {}_{[\theta'(g(f(z_2))\doteq q(z_2))]}\!\leftarrow g(f(z_3')) \rightarrow_{[\theta'(g(f(z_7))\doteq f(z_7))]} f(z_3') = \theta'(v)$$

shows that $\theta'$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$.

If instead of choosing $s = s_3$ at the very first step, when constructing $E_{u,v}$, had we chosen $s = s_5$, we would have obtained a different rigid $(\Sigma, E)$-unifier, for example:

$$\theta'' = [z_1/z_5', z_2/z_5', z_6/z_5', z_7/z_5'].$$

There is also choice in the selection of $\sigma_1$ and $\sigma_2$, all of which lead to different rigid unifiers.

We now show the soundness of the method.

**Theorem 9.6** [Soundness] Let $E_0$ be a reduced form of $E_{u,v}$, i.e. $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$; $S_0 = \emptyset$; $m$ the total number of variables in $E_0$; and $V = Var(E) \cup Var(u,v)$. If

$$\langle S_0, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle,$$

if $\theta_{S_k}$ is a $\Sigma^s$-unifier in $CSU_{\Sigma^s}(S_k)$ over $\mathcal{T}_\Sigma(X)$, $F(w,w) \doteq T(w') \in E_k$, for $w, w' \in \mathcal{T}_\Sigma(X)$ of sort $s$ and $F(t,t') \doteq T(t'') \notin E_i$ for all $i$, $0 \le i < k \le m$, then $\theta_{S_k}|_V$ is a rigid $E$-unifier of $u$ and $v$.

**Proof:** We shall prove the following claim by induction on $k$.

*Claim.* Given any set of the form $E_{u,v} = E \cup \{eq(u,v) \doteq F(u,v),\ eq(z,z) \doteq T(z)\}$, with $E$ a set of general $\Sigma(X)$-equations and $u, v \in \mathcal{T}_\Sigma(X)$, for any pair $\langle S_0, E_0 \rangle$ where $S_0$ is any set of pairs of the form $< x, t >$ with $t \in \mathcal{T}_\Sigma(X)$ and $LS(t) \le LS(x)$, and $E_0$ is rigid reduced and rigid equivalent to $E_{u,v}$, if

$$\langle S_0, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle,$$

if $\theta_{S_k}$ is a $\Sigma^s$-unifier in $CSU_{\Sigma^s}(S_k)$ over $\mathcal{T}_\Sigma(X)$, $F(w,w) \doteq T(w') \in E_k$ for some $t \in \mathcal{T}_\Sigma(X)$, and $F(t,t') \doteq T(t'') \notin E_i$ for any $\Sigma^s$-terms $t, t', t''$, for all $i$, $0 \le i < k \le m$, then $\theta_{S_k}$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $T(z)$ and $F(u,v)$, where $\theta_{S_k} \in CSU_{\Sigma^s}(S_k)$ and $\theta_{S_k}$ is over $\mathcal{T}_\Sigma(X)$.

*Proof of claim.*

In the base case, we must have $k = 1$ because $F(w,w) \doteq T(w') \notin E_0 \cup E_0^{-1}$. In order that $F(w,w) \doteq T(w')$ be in $E_1$, the transformation step must be

$$\langle S_0,\ E_0\rangle \Rightarrow \langle S_0 \cup TU(eq(z,z), eq(u,v)), E_1\rangle,$$

where $E_1' = \sigma((E_0 - \{eq(z,z) \doteq T(z)\}) \cup \{F(u,v) \doteq T(z)\})$, $E_1' \Rightarrow_{\mathcal{R}} E_1$, $TU(eq(z,z), eq(u,v))$ is a triangular form of a $\Sigma^s$-unifier of $eq(z,z)$ and $eq(u,v)$ (over $\mathcal{T}_\Sigma(X)$), $\sigma$ is the $\Sigma^s$-substitution (over $\mathcal{T}_\Sigma(X)$), associated with $TU(eq(z,z), eq(u,v))$ and $\theta' = \theta_{S_1}$ is in $CSU_{\Sigma^s}(S_1)$ over $\mathcal{T}_\Sigma(X)$.

By lemma 4.12 $\sigma;\theta' = \theta'$, hence $\theta'(F(u,v)) \doteq \theta'(T(z)) \in \theta'(E_1)$. Since by lemma 9.4, $\theta'(E_0)$ and $\theta'(E_1)$ are rigid equivalent, $\theta'(F(u,v)) \overset{*}{\cong}_{\theta'(E_0)} \theta'(T(z))$. This shows that $\theta'$ is a rigid $(\Sigma^s, E_0)$-unifier of $F(u,v)$ and $T(z)$. The soundness of the reduction procedure $\mathcal{R}$ (see theorem 8.7) implies that $\theta'$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $F(u,v)$ and $T(z)$.

For the induction step, assume that

$$\langle S_0,\ E_0\rangle \Rightarrow \langle S_1,\ E_1\rangle \Rightarrow^+ \langle S_k,\ E_k\rangle,$$

where $S_1 = S_0 \cup TU(l_1/\beta, l_2)$, $E_1' \Rightarrow_{\mathcal{R}} E_1$ with

$$E_1' = \sigma((E_0 - \{l_1 \doteq r_1\})^\dagger \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\}),$$

if $\theta' = \theta_{S_k}$ is a $\Sigma^s$-unifier in $CSU_{\Sigma^s}(S_k)$ over $\mathcal{T}_\Sigma(X)$, $F(w,w) \doteq T(w') \in E_k$, $F(t,t') \doteq T(t'') \notin E_i$ for all $i$, $0 \le i < k \le m$, $TU(l_1/\beta, l_2)$ represents a $\Sigma^s$-unifier over $\mathcal{T}_\Sigma(X)$ of $l_1/\beta$ and $l_2$ in triangular form, $\sigma = [t_1/x_1, \ldots, t_p/x_p]$ where $TU(l_1/\beta, l_2) = \{\langle x_1, t_1\rangle, \ldots, \langle x_p, t_p\rangle\}$. Thus the induction hypothesis applies to $\langle S_1, E_1\rangle$, and the $\Sigma^s$-unifier $\theta'$ of $S_k$ (over $\mathcal{T}_\Sigma(X)$) is a rigid $(\Sigma^s, E_1)$-unifier of $T(z)$ and $F(u,v)$ (over $\mathcal{T}_\Sigma(X)$). Since $S_1 \subseteq S_k$ and $\theta'$ unifies $S_k$, by lemma 9.5, $\theta'(E_0)$ and $\theta'(E_1)$ are rigid equivalent. Hence $\theta'$ is a rigid $(\Sigma^s, E_0)$-unifier of $T(z)$ and $F(u,v)$ (over $\mathcal{T}_\Sigma(X)$). This concludes the induction step and the proof of the claim. □

Applying the claim to $S_0 = \emptyset$ and an $E_0$ such that $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$, we have that $\theta'$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $T(z)$ and $F(u,v)$ over $\mathcal{T}_\Sigma(X)$, where $\theta' = \theta_{S_k}$ is in $CSU_{\Sigma^s}(S_k)$ and is over $\mathcal{T}_\Sigma(X)$, and by lemma 9.2, $\theta_{S_k}|_V$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$. □

The main technique in the proof of the completeness part is the removal of peaks by the use of critical pairs (Bachmair [Bac89], Bachmair, Dershowitz, and Plaisted [BDP87], Bachmair, Dershowitz, and Hsiang [BDH86]).

**Theorem 9.7** [Completeness] Let $E$ be a set of general $\Sigma$-equations over $\mathcal{T}_\Sigma(X)$ and $u$, $v$ two terms in $\mathcal{T}_\Sigma(X)$. Let $\theta$ be a rigid $(\Sigma, E)$-unifier of $u$ and $v$ and let $s = LS(\theta(u)) = LS(\theta(v))$. Consider the order-sorted signature $\Sigma^s$ and the set of general axiom $E_{u,v}$ as described above. Then, there is a reduced set $E_0$ of general axioms rigid equivalent to $E_{u,v}$ and letting $S_0 = \emptyset$, $m$ the number of variables in $E_0$, and $V = Var(E) \cup Var(u,v)$, there is a sequence of transformations

$$\langle S_0, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle,$$

and there exists $\theta_{S_k} \in CSU(S_k)$ over $\mathcal{T}_\Sigma(X)$, where $k \leq m$, $F(w,w) \doteq T(w') \in E_k$, $F(t,t') \doteq T(t'') \notin E_i$ for all $i$, $0 \leq i < k$. Furthermore, $\theta_{S_k}|_V$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$.

**Proof:** First, since it is clear that the method is *pure*, thus it can be assumed that $\theta$ is a ground substitution and that $V \subseteq D(\theta)$. By lemma 9.2, $\theta$ can be extended to a $\Sigma^s$-substitution $\theta'$ over $\mathcal{T}_\Sigma$(X) such that $\theta = \theta'|_{D(\theta')-\{z\}}$ and $\theta'$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $T(z)$ and $F(u,v)$ over $\mathcal{T}_\Sigma$(X). By lemma 7.6, there is a minimal element $\theta_1 \in S_{E_{u,v},T,F,\theta'}$ that is a ground $\Sigma^s$-substitution satisfying

1. $\theta_1 \sqsubseteq_{E_{u,v}} \theta'$,

2. $\theta_1$ is a rigid $(\Sigma^s, E_{u,v})$-unifier of $T(z)$ and $F(u,v)$,

3. $\theta_1$ is reduced w.r.t. $\theta_1(E_{u,v})$,

4. since $D(\theta) = D(\theta_1)$ and $V \subseteq D(\theta)$, we also have $V \subseteq D(\theta_1)$ and

5. since $\theta$ is over $\mathcal{T}_\Sigma$(X), so is $\theta_1$.

Let $\preceq_{\theta_1}$ be the total simplification ordering on $\mathcal{T}_\Sigma(X)$ induced by $\theta_1$ and $\preceq$ as in definition 8.10. By theorem 8.9 there exists $E_0$ reduced with respect to $\preceq_{\theta_1}$ such that $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$. Since $E_0$ and $E_{u,v}$ are rigid equivalent, by lemma 7.8 $\theta_1$ must be reduced w.r.t. $\theta_1(E_0)$. We shall prove the following claim.

*Claim.* Given a ground $\Sigma^s$-substitution $\theta_1$ such that $V \subseteq D(\theta_1)$, letting $E_0$ be a set of general axioms such that $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$ and $E_0$ is reduced with respect to $\preceq_{\theta_1}$, if $\theta_1$ is reduced with respect to $\theta_1(E_0)$, is a $\Sigma^s$-unifier over $\mathcal{T}_\Sigma(\mathrm{X})$ of $S_0$ and is a rigid $(\Sigma^s, E_0)$-unifier of $T(z)$ and $F(u, v)$, then there is a sequence of transformations

$$\langle S_0, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle$$

where $k \leq m$, $S_k$ is unifiable, $F(t, t') \doteq T(t'') \in E_k$, $F(t, t') \doteq T(t'') \notin E_i$ for all $i$, $0 \leq i < k$, and $\theta_1$ unifies $S_k$ (over $\mathcal{T}_\Sigma(X)$).

*Proof of claim.* Let

$$T(w') = u_0 \longleftrightarrow_{\theta_1(E_0)} u_1 \longleftrightarrow_{\theta_1(E_0)} \cdots \longleftrightarrow_{\theta_1(E_0)} u_{n-1} \longleftrightarrow_{\theta_1(E_0)} u_n = F(w, w)$$

be a proof that $\theta_1(T(z)) \stackrel{*}{\cong}_{\theta_1(E_0)} \theta_1(F(u, v))$. We proceed by induction on the pair $\langle m, \{u_0, \ldots, u_n\}\rangle$, where $m$ is the number of variables in $E_0$ and $\{u_0, \ldots, u_n\}$ is the multiset of terms occurring in the proof. We use the well-founded ordering on pairs where the ordering on the first component is the ordering on the natural numbers, and the ordering on the second component is the multiset ordering $\prec_m$ extending $\prec$. First, observe that since $T \prec F \prec r \prec eq(s, t)$ for all $r, s, t \in \mathcal{T}_\Sigma$, the above proof must have some peak because oriented instances of the equations $eq(u, v) \doteq F(u, v)$ and $eq(z, z) \doteq T(z)$ are of the form $eq(s, t) \to F(s, t)$ and $eq(s, s) \to T(s)$. Thus, in the base case, we have $m = 1$, $n = 2$, and $u_1 = \theta_1(eq(u, v)) = \theta_1(eq(z, z))$. Hence, $\theta_1$ is a unifier of $eq(z, z)$ and $eq(u, v)$. Let $\sigma$ be an idempotent and variable decreasing $\Sigma^s$-unifier over $\mathcal{T}_\Sigma(X)$ such that $\sigma \preceq \theta_1$ (guaranteed to exist by completeness of the $CSU$ procedure), and let $TU(eq(z, z), eq(u, v))$ be a triangular form of $\sigma$. By lemma 4.16, since $\theta_1$ unifies $eq(z, z)$ and $eq(u, v)$, it unifies $TU(eq(z, z), eq(u, v))$. Let $E_1' = \sigma_T((E_0 - \{eq(z, z) \doteq T(z)\}) \cup \{F(u, v) \doteq T(z)\})$ where $\sigma_T$ is associated with $TU(eq(z, z), eq(u, v))$. We have

$$\langle S_0, E_0 \rangle \Rightarrow \langle S_1, E_1 \rangle,$$

with $S_1 = S_0 \cup TU(eq(z, z), eq(u, v))$ and $E_1' \Rightarrow_{\mathcal{R}} E_1$.

Since $\theta_1$ unifies $S_0$ and $TU(eq(z, z), eq(u, v))$, it unifies $S_1$ and the claim holds.

For the induction step, consider a peak $u_{i-1} \longleftarrow_{\theta_1(E_0)} u_i \longrightarrow_{\theta_1(E_0)} u_{i+1}$. Note that $u_i \succ u_{i-1}$ and $u_i \succ u_{i+1}$. Assume that

$$u_i \longrightarrow_{\beta_1, \theta_1(l_1 \doteq r_1)} u_{i-1}$$

and

$$u_i \xrightarrow{\cdot}_{\beta_2,\theta_1(l_2 \doteq r_2)} u_{i+1},$$

where $l_1 \doteq r_1$, $l_2 \doteq r_2 \in E_0 \cup E_0^{-1}$ and $\beta_1$ and $\beta_2$ are addresses in $u_i$. By lemma 7.9, we have that $u_j = \theta_1(u'_j)$ for $j = i-1, i, i+1$ and $\beta_1, \beta_2 \in Dom(u'_i)$. We need to examine overlaps carefully. There are two cases.

*Case* 1. $\beta_1$ and $\beta_2$ are independent. Then, letting $v = u_i[\beta_1 \leftarrow \theta_1(r_1), \beta_2 \leftarrow \theta_1(r_2)]$, we have $u_{i-1} \longrightarrow_{\theta_1(E_0)} v \longleftarrow_{\theta_1(E_0)} u_{i+1}$, and $u_i \succ v$. We obtain a proof with associated sequence $\langle u_0, \ldots, u_{i-1}, v, u_{i+1}, \ldots, u_n \rangle$. Since $u_i \succ v$,

$$\{u_0, \ldots, u_n\} \succ_m \{u_0, \ldots, u_{i-1}, v, u_{i+1}, \ldots, u_n\},$$

and we conclude by applying the induction hypothesis.

*Case* 2. $\beta_1$ is an ancestor of $\beta_2$ (the case where $\beta_2$ is an ancestor of $\beta_1$ is similar), and letting $\beta_2 = \beta_1\beta$, we see that

$$\theta_1(l_1)/\beta = (\theta_1(u'_i)/\beta_1)/\beta = \theta_1(u'_i)/\beta_2 = \theta_1(l_2) \tag{1}$$

Hence $\theta_1(l_1) \rightarrow_{[\beta,\theta_1(l_2 \doteq r_2)]} \theta_1(l_1)[\beta \leftarrow \theta_1(r_2)]$. Since $\theta_1$ is reduced with respect to $E_0$ we have again by lemma 7.9 that $\beta \in Dom(l_1)$ hence by (1) $\theta_1(l_1/\beta) = \theta(l_2)$. Therefore, $l_1/\beta$ and $l_2$ are unifiable. Since $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ are in $E_0$ with that orientation because $E_0$ is reduced with respect to $\preceq_{\theta_1}$, it must be the case that $l_1/\beta \neq l_2$. Not only is it important that $E_0$ is interreduced, but that the orientation of the rules is as in $\theta(E_{u,v})$.

Let $\sigma \preceq \theta_1$ be an idempotent and variable decreasing $\Sigma^s$-unifier (over $\mathcal{T}_\Sigma(\mathrm{X})$) in $CSU_{\Sigma^s}(l_1/\beta, l_2)$, let $TU(l_1/\beta, l_2)$ be a triangular form of $\sigma$ and let $\sigma_T$ be the associated $\Sigma^s$-substitution. Notice that $\sigma_T$ is over $\mathcal{T}_\Sigma(\mathrm{X})$. Thus we have

$$\langle S_0, E_0 \rangle \Rightarrow < S_1, E_1 > .$$

Since $\theta_1$ is ground, it is idempotent, and since it unifies $l_1/\beta$ and $l_2$, by lemma 4.16, $\theta_1$ unifies $TU(l_1/\beta, l_2)$ as well. Hence $\theta_1$ unifies $S_1$. Since $\theta_1(E_0)$ and $\theta_1(E_1)$ are rigid equivalent, $\theta_1$ is also a rigid $(\Sigma^s, E_1)$-unifier of $T(z)$ and $F(u,v)$. Since $\theta_1$ is is minimal in $S_{E_{u,v},T,F,\theta'}$,

$\theta_1(E_{u,v})$, $\theta_1(E_0)$, and $\theta_1(E_1)$ are rigid equivalent, and $\theta_1 \sqsubseteq_{E_{u,v}} \theta'$, as argued previously, $\theta_1$ is also reduced w.r.t. $\theta_1(E_1)$. Also note that since $\sigma$ is variable decreasing, so is $\sigma_T$, hence at least one variable in the set $\{x_1, \ldots, x_p\}$ does not occur in $I(\sigma_T)$. Thus, this variable does not occur in $E_1$, and $m' < m$ where $m'$ is the number of variables in $E_1$. Therefore, we can apply the induction hypothesis to $\theta_1$, $S_1$ and $E_1$, and obtain a sequence

$$\langle S_1, E_1 \rangle \Rightarrow^+ \langle S_k, E_k \rangle,$$

where $k \leq m'$, $S_k$ is unifiable, $F(w,w) \doteq T(w') \in E_k$, $F(t,t') \doteq T(t'') \notin E_i$ for all $i$, $0 \leq i < k$, and $\theta_1$ is a $\Sigma^s$-unifier of $S_k$ over $\mathcal{T}_\Sigma(X)$. This concludes the induction step and the proof of the claim. □

Let us apply this claim to prove the theorem. Let $S_0 = \emptyset$ and $E_0$ be a rigid reduced set with respect to $\preceq_{\theta_1}$ such that $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$. By the claim, there is a sequence of at most $m$ transformations as stated in the theorem, and $\theta_1$ is a $\Sigma^s$-unifier of $S_k$ over $\mathcal{T}_\Sigma(X)$. Since the set $CSU_{\Sigma^s}(S_k)$ restricted to substitutions over $\mathcal{T}_\Sigma(X)$ is a complete set of $\Sigma^s$-unifier over $\mathcal{T}_\Sigma(X)$, there exists some $\theta_{S_k} \in CSU_{\Sigma^s}(S_k)$ such that $\theta_{S_k} \leq \theta_1[V]$. We know that $\theta_1 \sqsubseteq_{E_{u,v}} \theta'$, so we have $\theta_{S_k} \leq_{E_{u,v}} \theta'[V]$. Therefore, $\theta_{S_k}|_V \leq_E \theta[V]$. Finally, by theorem 9.6, we see that $\theta_{S_k}|_V$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$. □

Theorem 9.7 also shows that order-sorted rigid-unification is decidable for general axioms.

**Corollary 9.8** For $\Sigma$ a finite coherent order-sorted signature, $E$ a set of general axioms, Rigid $(\Sigma, E)$-unification is decidable.

**Proof:** By theorem 9.7, a (ground) rigid $(\Sigma, E)$-unifier $\theta$ of $u$ and $v$ exists iff there is some sort $s \in \Sigma$, a set $E_{u,v}$ of general over $\Sigma^s$ obtained as described above, a rigid reduced form $E_0$ of $E$, i.e. $E \Rightarrow_{\mathcal{R}} E_0$ and some sequence of transformations

$$\langle \emptyset, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle$$

of at most $k \leq m$ steps where $m$ is the number of variables in $E_0$, and such that $S_k$ is $\Sigma^s$ unifiable (over $\mathcal{T}_\Sigma(X)$), $F(w,w) \doteq T(w') \in E_k$ for some $w \in \mathcal{T}_\Sigma(X)_s$ and $F(t,t') \doteq T(t'') \notin E_i$ for all $i$, $0 \leq i < k$, all $t, t', t'' \in \mathcal{T}_\Sigma(X)$. Clearly, all these conditions are finitary and can be tested. Thus, order-sorted rigid $E$-unification is decidable. □

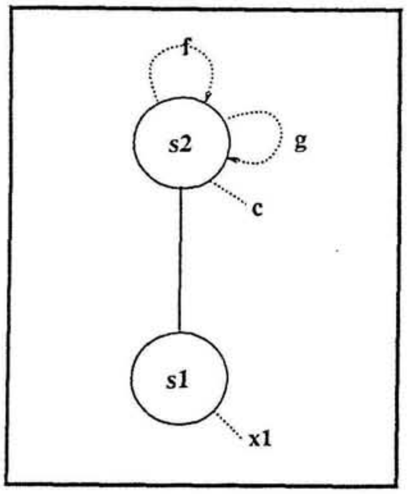


Figure 8: $f(c) \doteq g(c)$

Combining the results of theorem 9.6 and 9.7 we also obtain the fact that for any set $E$ of general axioms, any $\Sigma$-terms $u$, $v$, there is always a finite complete set of rigid $(\Sigma, E)$-unifiers.

**Theorem 9.9** Let $E$ be a set of general equations over $\mathcal{T}_\Sigma(X)$, $u$, $v$ two terms in $\mathcal{T}_\Sigma(X)$, $m$ the number of variables in $E \cup \{u, v\}$, and $V = Var(E) \cup Var(u, v)$. There is a finite complete set of rigid $(\Sigma, E)$-unifiers for $u$ and $v$ given by the set

$$\{\theta_{S_k}|_V \mid \theta_{S_k} \in CSU_{\Sigma^s}(S_k) \text{ is over } \mathcal{T}_\Sigma(X), \text{ and } \langle \emptyset, E_0 \rangle \Rightarrow^+ \langle S_k, E_k \rangle,\ k \leq m\},$$

with $E_{u,v} \Rightarrow_{\mathcal{R}} E_0$, and where $S_k$ is unifiable, $F(w, w) \doteq T(w') \in E_k$, $F(t, t') \doteq T(t'') \notin E_i$ for all $i$, $0 \leq i < k$.

Let us now illustrate via two examples how the method takes advantage of sort information.

**Example 9.10** Consider the problem presented at the end of section 4.2. The signature is shown in figure 9.10. Consider the equation system $E = \{f(c) \doteq g(c)\}$, and let us try to find a rigid $(\Sigma, E)$-unifier for $u = f(x_1)$ and $v = g(x_1)$. In this case $LS(u) = LS(v) = s_2$. Let us pick $s \in LBd(\{s_2\})$. The choices are $s_1$ and $s_2$. Clearly, no solution can have sort $s_1$ because for any $\Sigma$-substitution $\theta, LS(\theta(u)) = s_2$. Let us therefore pick $s = s_2$. We construct the system $E_{u,v}$ as follows:

$$f(c) \doteq g(c)$$

$$eq(f(x_1), g(x_1)) \doteq F(f(x_1), g(x_1))$$
$$eq(z, z) \doteq T(z)$$

By interreducing we obtain the system $E_0$:

$$f(c) \rightarrow g(c)$$
$$eq(f(x_1), g(x_1)) \rightarrow F(f(x_1), g(x_1))$$
$$eq(z, z) \rightarrow T(z)$$

There is no overlap possible between the last two equations because $f(x_1)$ and $g(x_1)$ are not unifiable. An overlap between the first and the last equations leads to a dead end. Therefore, the only possibility involves overlapping the first and second equations. This entails finding $TU(f(x_1), f(c))$. However, $[c/x_1]$ which is not well sorted! Therefore the algorithm returns **failure**. Hence $u = f(x_1)$ and $v = g(x_1)$ are not rigid $(\Sigma, E)$-unifiable.
This is indeed correct. Notice that an unsorted algorithm would return the substitution $[c/x_1]$ as a solution. A further attempt to obtain a $\Sigma$-substitution from it would fail. Thus, the order-sorted is more efficient because it detects failure at an earlier stage.

**Example 9.11 AC** (Adapted from [MGS89].)
Let the set of sorts be $S = \{Elt, Mult\}$with $Elt \leq Mult$, and let $\Sigma$ consist of a binary operator $\cdot : MultMult \mapsto Mult$ with the syntax of juxtaposition. The equations are associativity and commutativity:

$$z_1 \cdot z_2 \doteq z_2 \cdot z_1$$
$$w_1 \cdot (w_2 \cdot w_3) \doteq (w_1 \cdot w_2) \cdot w_3$$

Consider the terms $u = x \cdot s$ and $v = y \cdot t$, with $x, s, y$ and $t$ variables of sort $Elt$.
The system has the following covering of unsorted $E$-unifiers:

1. $[t/x, y/t]$
2. $[y/x, t/s]$
3. $[(y \cdot q)/s, (x \cdot q)/t]$

4. $[(y \cdot q)/x, (s \cdot q)/t]$

5. $[(t \cdot q)/s, (x \cdot q)/y]$

6. $[(t \cdot q)/x, (s \cdot q)/y]$

7. $[(q \cdot p)/x, (q' \cdot p')/s, (p \cdot p')/y, (q \cdot q')/t]$

However only the first two are well sorted. Also, the first two are rigid unsorted $E$-unifiers. The third one is not, because its proof requires two instances of associativity. However, by expanding the system $E$ to a system $E'$ which includes an additional instance of associativity, the third substitution represents an unsorted rigid $E'$-unifier.

Let us see how our method computes the first rigid $(\Sigma, E)$-unifier. The system of equations $E_{u,v}$ is:

$$z_1 \cdot z_2 \doteq z_2 \cdot z_1$$
$$w_1 \cdot (w_2 \cdot w_3) \doteq (w_1 \cdot w_2) \cdot w_3$$
$$eq(x \cdot s, y \cdot t) \doteq F(x \cdot s, y \cdot t)$$
$$eq(z, z) \doteq T(z)$$

After reducing, we obtain $E_0$:

$$z_1 \cdot z_2 \rightarrow z_2 \cdot z_1$$
$$w_1 \cdot (w_2 \cdot w_3) \rightarrow (w_1 \cdot w_2) \cdot w_3$$
$$eq(x \cdot s, y \cdot t) \rightarrow F(x \cdot s, y \cdot t)$$
$$eq(z, z) \rightarrow T(z)$$

There is an overlap between the first and third rewrite rules, with $\sigma_T = TU(x \cdot s, z_1 \cdot z_2) = [z_1/x, s/z_2]$. After rewriting and applying $\sigma_T$ we obtain $E_1'$:

$$z_1 \cdot s \doteq s \cdot z_1$$
$$w_1 \cdot (w_2 \cdot w_3) \doteq (w_1 \cdot w_2) \cdot w_3$$
$$eq(s \cdot z_1, y \cdot t) \doteq F(z_1 \cdot s, y \cdot t)$$
$$eq(z, z) \doteq T(z)$$

After reducing we obtain $E_1$:

$$\begin{aligned}
&z_1 \cdot s \rightarrow s \cdot z_1 \\
&w_1 \cdot (w_2 \cdot w_3) \rightarrow (w_1 \cdot w_2) \cdot w_3 \\
&eq(s \cdot z_1, y \cdot t) \rightarrow F(s \cdot z_1, y \cdot t) \\
&eq(z, z) \rightarrow T(z)
\end{aligned}$$

Next, the last two rules are overlapped. One can then obtain $TU(eq(s \cdot z_1, y \cdot t), eq(z, z)) = [s \cdot z_1/z, s/y, t/z_1]$. The system $E_2$ is obtained by applying the transformation and interreducing:

$$\begin{aligned}
&t \cdot s \rightarrow s \cdot t \\
&w_1 \cdot (w_2 \cdot w_3) \rightarrow (w_1 \cdot w_2) \cdot w_3 \\
&T(s \cdot t) \rightarrow F(s \cdot t, s \cdot t) \\
&eq(s \cdot t, s \cdot t) \rightarrow T(s \cdot t)
\end{aligned}$$

Thus the method terminates and produces the rigid $(\Sigma, E)$-unifier $[t/x, s/y, t/z_1, s/z_2]$. It is interesting to see how the sort information can be used to discard a substitution at an early stage. For example, the substitution $[(y \cdot q)/s, (x \cdot q)/t]$ is not well sorted because $s$ and $t$ are of sort *Elt* while the co-arity of a term containing $\cdot$ has to be *Mult*. Let us see how this is witnessed by our method. First, the system $E_{u,v}$ now contains an extra instance of the associativity equation:

$$\begin{aligned}
&z_1 \cdot z_2 \doteq z_2 \cdot z_1 \\
&w_1 \cdot (w_2 \cdot w_3) \doteq (w_1 \cdot w_2) \cdot w_3 \\
&w_1' \cdot (w_2' \cdot w_3') \doteq (w_1' \cdot w_2') \cdot w_3' \\
&eq(x \cdot s, y \cdot t) \doteq F(x \cdot s, y \cdot t) \\
&eq(z, z) \doteq T(z)
\end{aligned}$$

On attempting to overlap the second and fourth rule (as a matter of fact any of the two associativity rules with the fourth one), we have to compute $TU(x \cdot s, z_1 \cdot (z_2 \cdot z_3))$. There is no such $\Sigma$-unifier since $s$ and $(z_2 \cdot z_3)$ do not unify (by virtue of $s$ being a variable of type *Elt*.) As a matter of fact, due to this reason, none of the other $E$-unifiers is well sorted.

Again, out method stops before computing an ill-typed unsorted unifier. This explains the sense in which the order-sorted method is more efficient than the unsorted one.

# 10 NP-Completeness of Rigid $(\Sigma, E)$-unification

First, recall that rigid $E$-unification is NP-hard. This holds even for sets of ground unsorted equations, as shown by Kozen [Koz76, Koz77].

**Theorem 10.1** Rigid $(\Sigma, E)$-unification is NP-complete.

**Proof:** By corollary 9.8, the problem is decidable. It remains to show that it is in NP. From corollary 9.8, $u$ and $v$ have some rigid $E$-unifier iff there is some sequence of transformations $\langle \mathcal{S}_0, \mathcal{E}_0 \rangle \Rightarrow^+ \langle \mathcal{S}_k, \mathcal{E}_k \rangle$ of at most $k \leq m$ steps where $m$ is the number of variables in $\mathcal{E}_0$, and a there is a $\Sigma$-unifier $\theta_{\mathcal{S}_k}$ of $\mathcal{S}_k$ such that $F(w,w) \doteq T(w') \in \mathcal{E}_k$ and $F(t,t') \doteq T(t'') \notin \mathcal{E}_i$ for all $i$, $0 \leq i < k$. We need to verify that it is possible to check these conditions in polynomial time.

We first show that each $\Rightarrow$ step takes time polynomial on its input. Let $n_i = size(< \mathcal{S}_i, \mathcal{E}_i >) = |\mathcal{S}_i| + |\mathcal{E}_i|$ where $|\mathcal{S}_i|$ is the size of the DAG representing all terms in $\mathcal{S}_i$ and $|\mathcal{E}_i|$ is defined similarly. The first part of $\Rightarrow$ consists of picking the equations $l_1 \doteq r_1$ and $l_2 \doteq r_2$; choosing an address $\beta$ in $l_1$; checking that either $l_1/\beta$ is not a variable or $l_2 \doteq r_2$ is degenerate; and finally making sure that $l_1/\beta \neq l_2$. These steps can all be done in time linear on $n_i$. Next, $TU(l_1/\beta, l_2)$ is obtained by running the $CTU$ algorithm which is quasi-linear on its input. The next two steps involve a) adding $TU(l_1/\beta, l_2)$ to $\mathcal{S}_i$ which takes at most time $O(n_i)$ and then finding a reduced set via the reduction procedure $\Rightarrow_{\mathcal{R}}$ which runs in time $O(|E_i|log(|E_i|) \leq O(n_i log(n_i)))$. Thus it takes at most time $O(n_i log(n_i))$ to do the transformation $\langle \mathcal{S}_i, \mathcal{E}_i \rangle \Rightarrow \langle \mathcal{S}_{i+1}, \mathcal{E}_{i+1} \rangle$. After applying the transformation we run the non-deterministic unification algorithm to compute elements of $CSU(\mathcal{S}_i)$. This procedure runs in quasi-linear time. Provided we obtain $\theta_{S_k} \in CSU(\mathcal{S}_i)$, we still have to check whether $F(w,w) \doteq T(w') \in \mathcal{E}_i$. This is linear on the size of $\mathcal{E}_i$. Therefore, the transformation together with the guessing of a $\Sigma$-unifier for $\mathcal{S}_i$ and checking for the halting condition still takes $O(n_i log(n_i))$ time.

Since the transformations are applied in sequence, in order to guarantee polynomial time for $k$ transformation steps, we should make sure that the size of the system does not grow too much. Since $TU(l_1/\beta, l_2)$ is constructed using elements of $\mathcal{E}_i$ exclusively, its size is at most $|\mathcal{E}_i|$, and since $\mathcal{S}_{i+1}$ is obtained by adding $TU(l_1/\beta, l_2)$ to $\mathcal{S}_i$, it follows that $|\mathcal{S}_i| \leq |\mathcal{S}_i| + |\mathcal{E}_i|$. Since $\mathcal{S}_0 = 0$, we see that $|\mathcal{S}_i| \leq i \times |\mathcal{E}_0| = i \times n_0$. The equational part of the system, $\mathcal{E}_i$ is obtained in three steps. First rewriting an equation, which does not increase the size of $\mathcal{E}$ since it involves changing pointers in a DAG. Then, $\sigma_T$ is applied, which again can be implemented by rearranging pointers. Finally the $\Rightarrow_{\mathcal{R}}$ is applied which as explained in section 8 does not increase the size of $\mathcal{E}_i$. Thus for $0 \leq i, |\mathcal{E}_i| = |\mathcal{E}_0|$ and $n_i \leq (i+1) \times n_0$.

Therefore, the total time for $k$ transformation steps is bounded by

$O(\Sigma_{i=0}^{i=k} n_i \times log(n_i)) \leq$

$O(\Sigma_{i=0}^{i=k} (i \times n_0) \times log(i \times n_0)) =$

$O(n_0 \times \Sigma_{i=0}^{i=k} i \times log(i) + n_0 \times log(n_0) \times \Sigma_{i=0}^{i=k} i) \leq$

$O(n_0 \times k^4 + n_0 \times log(n_0) \times k^2) = O(n_0 \times k^4)$.

Since $k \leq n_0$ we have that the total time for $k$ transformations along with the checks for the halting condition is at most $O(n_0^5)$, hence polynomial on the size of $E$. Thus we have an NP-algorithm.

□

# 11 Conclusion and Further Research

The contribution of this paper is the presentation of an Order-Sorted method for Rigid $(\Sigma, E)$-unification. We show that the problem is decidable, furthermore that it is in NP. The method is intrinsically order-sorted and uses the triangular forms produced by a non-deterministic order-sorted unification algorithm presented in [Isa89]. The fact that order-sorted rigid unification remains in NP is quite impressive given the intricacy of the procedures involved. Not only do we present an order-sorted method, but we propose improvements to the original unsorted algorithm [GNPS90] which substantially simplify it. A significant improvement of our method over the unsorted rigid E-unification one is that we do not use order assignments to guess the right orientation of the rewrite rules. We have managed to include the guessing into the reduction procedure.

It is important to note that the order-sorted method is more efficient than the unsorted one because it is able to *weed out* unfit substitutions as these are built, as oposed to doing this after the fact, when the substitution has already been generated.

The method presented only works for general axioms. In the future, we plan to extend our results to larger classes of axioms. Let us point out that the main difficulty lies in the use of congruence closure to build up $\Sigma$-unifiers. If the equations are not general, ill-typed terms might be formed thereby *infecting* the method. An alternative is to refine the reduction procedure of section 8, so as to keep the systems order-sorted.

## Acknowledgements

The authors would like to thank Joseph Goguen, José Meseguer, Val Brezau-Tannen, Carl Gunter and Wayne Snyder for their valuable comments.

## References


[And81] Peter Andrews. Theorem Proving via General Matings. *Journal of the ACM*, 28(2):193–214, 1981.

[Bac89] Leo Bachmair. *Canonical Equational Proofs*. Wiley and Sons, 1989.

[BDH86] Leo Bachmair, Nachum Dershowitz, and J. Hsiang. Orderings for equational proofs. In *LICS'86, Cambridge, Massachusetts*, pages 346–357, 1986.

[BDP87] Leo Bachmair, Nachum Dershowitz, and David Plaisted. Completion without Failure. In *Proceedings of CREAS, Lakeway, Texas. Also submitted for publication.*, 1987.

[Der87] Nachum Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[DST80] Peter J. Downey, Ravi Sethi, and Endre R. Tarjan. Variations on the Common Subexpressions Problem. *J.ACM*, 27(4):758–771, 1980.

[GM84] Joseph Goguen and José Meseguer. Eqlog: Equality, Types and Generic Modules for Logic Programming. In Douglas DeGroot and Gary Lindstrom, editors, *Functional and Logic Programming*, pages 295–363. Prentice-Hall, 1984.

[GM87a] Joseph Goguen and José Meseguer. Models and Equality for Logic Programming. In *Proceedings of the International Joint Conference on Theory and Practice of Software Development, Lecture Notes in Computer Science 250*, pages 1–22. Springer-Verlag, 1987.

[GM87b] Joseph Goguen and José Meseguer. Order-Sorted Algebra I: Partial and Overloaded Operations, Errors, Inheritance. Technical report, SRI Computer Science Laboratory, 1987.

[GNP+92] Jean H. Gallier, P. Narendran, David Plaisted, Stan Raatz, and Wayne Snyder. An Algorithm for Finding Canonical Sets of Ground Rewrite Rules in Polynomial Time. *Journal of the ACM*, ?, 1992.

[GNPS90] Jean H. Gallier, P. Narendran, David Plaisted, and Wayne Snyder. Rigid *E*-Unification: NP-Completeness and Applications To Equational Matings. *Information and Computation*, 87(1/2):129–195, July-August 1990.

[Gog78] Joseph Goguen. Order-Sorted Algebra. Semantics and Theory of Computation Report 14, UCLA Computer Science Department, 1978.

[GRS87] Jean H. Gallier, Stan Raatz, and Wayne Snyder. Theorem Proving using Rigid *E*-Unification: Equational Matings. In *Proceedings of LICS'87, Ithaca, New York*, pages 338–346, 1987.

[Hue76] Gérard Huet. *Résolution d'Equations dans le Languages d'Ordre* $1, 2, \ldots, \omega$. PhD thesis, Université de Paris VII, 1976.

[Isa89] Tomás Isakowitz. *Theorem Proving Methods for Order-Sorted Logic.* PhD thesis, University of Pennsylvania, Philadelphia, PA 19104, December 1989.

[Kir88] Claude Kirchner. Order-sorted equational unification. In Robert A. Kowalski and Keneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium in Logic Programming, Seattle 88*, 1988.

[Koz76] Dexter Kozen. Complexity of Finitely Presented Algebras. Technical Report 76-294, Department of Computer Science, Cornell University, Ithaca, NY, 1976.

[Koz77] Dexter Kozen. Complexity of Finitely Presented Algebras. In *9th STOC Symposium, Boulder, Colorado*, pages 164–177, May 1977.

[MGS89] José Meseguer, Joseph Goguen, and Gert Smolka. Order-Sorted Unification. *Journal of Symbolic Computation*, 8:383–413, 1989.

[MM82] Alberto Martelli and Ugo Montanari. An efficient Unification Algorithm. *ACM Transactions on Programming Languages and Systems*, 4:158–282, 1982.

[NO80] Greg Nelson and Derek C. Oppen. Fast Decision Procedures Based on Congruence Closure. *Journal of the ACM*, 27(2):356–364, 1980.

[PW78] M. S. Paterson and M. N. Wegman. Linear Unification. *Journal of Computer and System Sciences*, 16(1):158–167, 1978.

[Rob65] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Smo86] Gert Smolka. Order-Sorted Horn Logic. SEIKI Report SR-86-17, FB Informatik, Universitat Kaiserslauten, West Germany, 1986.

[Sny88] Wayne Snyder. *Complete Set of transformation for General Unification.* PhD thesis, University of Pennsylvania, Philadelphia, PA 19104, 1988.

[Sny89] Wayne Snyder. A fast algoritm for generating Reduced sets of Ground Rewrite Rules equivalent to a set of Ground Equations $E$. In *Proceedings of RTA-89*, may 1989.

[SS87] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations.* PhD thesis, Fachbereich Informatik, Universitat Kaiserslauten, 1987.

[Wal84] Christoph Walther. Unification in many-sorted theories. In *Proceedings of the* $6^{th}$ *European Conference on Artificial Intelligence (ECAI-84)*, 1984.

[Wal87] Christoph Walther. *A Many-Sorted Calculus based on Resolution and Paramodulation.* Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers, Los Altos, California, 1987.

# A Appendix of Proofs

## A.1 Proof of lemma 4.12

Since $\theta$ is a $\Sigma$-unifier of $T$, we have $\theta(x_i) = \theta(t_i) = \theta(\sigma_T(x_i))$ for every $i$, $1 \leq i \leq k$. Since $\sigma_T(y) = y$ for all $y \notin \{x_1, \ldots, x_k\}$, $\theta = \sigma_T\,;\theta$ holds.

## A.2 Proof of lemma 4.13

By the definition of triangular forms we have that $\sigma = [x_1/t_1]; \ldots ; [x_n/t_n]$. The proof relies upon the following claim:

For $1 \leq i \leq n$,

$$\sigma_T^{(n+1-i)}(x_i) = \sigma(x_i). \tag{2}$$

Suppose the claim has been proven then

$$\sigma_T^{(n)}(x_i) = \sigma_T^{(i-1)}(\sigma_T^{(n+1-i)}(x_i)) = \sigma_T^{(i-1)}(\sigma(x_i)). \tag{3}$$

Since $\sigma$ is idempotent, the variables $x_1, \ldots, x_n$ do not appear in $\sigma(x_i)$, therefore $\sigma_T(\sigma(x_i))$=$\sigma(x_i)$, hence $\sigma_T^{(i-1)}(\sigma(x_i)) = \sigma(x_i)$. Therefore from 3 we obtain for $1 \leq i \leq n$:

$$\sigma_T^{(n)}(x_i) = \sigma(x_i). \tag{4}$$

Since $x_1, \ldots, x_n$ are all the variables in $D(\sigma)$ and $D(\sigma_T)$, we have $\sigma_T^{(n)} = \sigma$ as wanted.

The proof of the claim proceeds by descending induction. First, it is clear that $\sigma_T(x_n) = t_n = \sigma(x_n)$. Suppose the claim is true for $i + 1$ then

$$\begin{aligned}
\sigma_T^{(n+1-i)}(x_i) &= \sigma_T^{(n-i)}(\sigma_T(x_i)) \\
&= \sigma_T^{(n+1-(i+1))}(\sigma_T(x_i)) \\
&= \sigma_T^{(n+1-(i+1))}(t_i).
\end{aligned}$$

By the definition of a triangular form, the only variables in $t_i$ that can be affected by $\sigma_T$ are $x_{i+1}, \ldots, x_n$. By the inductive hypothesis, we have that for $i + 1 \leq k \leq n$,
$\sigma_T^{(n+1-(i+1))}(x_k) = \sigma(x_k)$. Therefore

$$\sigma_T^{(n+1-(i+1))}(t_i) = \sigma(t_i). \tag{5}$$

This completes the proof of the claim and of the lemma.

### A.3 Proof of lemma 4.16

By lemma 4.13, $\sigma = \sigma_T^{(n)}$. Since $\sigma$ is idempotent, none of the variables in the domain $\sigma_T$ appear in $\sigma_T^{(n)}(x_i)$. Therefore $\sigma_T^{(n)}(x_i) = \sigma_T^{(n+1)}(x_i)$. Thus,

$$\sigma(x_i) = \sigma_T^{(n)}(x_i) = \sigma_T^{(n-1)}(t_i) = \sigma_T^{(n)}(t_i) = \sigma(t_i).$$

### A.4 Proof of lemma 7.9

By hypothesis $\theta(t)/\beta = \theta(l)$, and $t'' = \theta(t)[\beta \leftarrow \theta(r)]$.

Suppose that $\beta$ is an address not in $Dom(t)$, since $\beta \in Dom(\theta(t))$, it has to be below the address $\beta_1$ of a variable $x$ in $t$. That is, $\beta = \beta_1\beta'$ with $t/\beta_1 = x$. We therefore have

$$\theta(x)/\beta' = \theta(t)/\beta = \theta(l).$$

This means that $\theta(x) \rightarrow_{\beta', \theta(l \doteq r)} \theta(x)[\beta' \leftarrow \theta(r)]$ which contradicts the assumption that $\theta$ is reduced with respect to $< \theta(E), \preceq >$. Therefore it must be the case that $\beta \in Dom(t)$. This proves part 1. As a consequence we have that $\theta(t)/\beta = \theta(t/\beta)$ hence

$$t'' = \theta(t)[\beta \leftarrow \theta(r)] = \theta(t[\beta \leftarrow r]) = \theta(t').$$

That $t' \in \mathcal{T}_\Sigma$ follows from the fact that $E$ is general, hence $LS(l) = LS(r)$, thus $LS(t) = LS(t')$. This proves part 2.

### A.5 Proof that $\preceq_\theta$ is a total ordering

We claim that $\preceq_\theta$ is a total ordering on $\mathcal{T}_\Sigma(X)$ that is monotonic and has the subterm property. The only problem is in showing that $\preceq_\theta$ is total, as the other conditions are then easily verified. The proof is similar to one given in [GNPS90].

Notice that $\theta$ defines an equivalence relation $\equiv_\theta$ on $\mathcal{T}_\Sigma(X)$ as follows:

$$u \equiv_\theta v \text{ if and only if } \theta(u) = \theta(v).$$

Due to clause **(1)** of the definition of $\prec'_\theta$, it is enough to show that for any two distinct elements $u, v$ in some nontrivial class $C$ modulo $\equiv_\theta$, either $u \preceq_\theta v$ or $v \preceq_\theta u$, but not both. Note that the set of classes modulo $\equiv_\theta$ is totally ordered: $C \ll C'$ iff $\theta(C) \prec \theta(C')$, where $\theta(C)$ denotes the common value of all terms $\theta(t)$ where $t \in C$. We proceed by induction on this well-ordering of the classes. Consider the least class $C$. It cannot contain a composite term $t = f(u_1, \dots, u_n)$ because by the subterm property of $\prec$, $\theta(u_i) \prec \theta(t)$ hence $[u_i] \ll [t] = C$ contradicts the minimality of $C$. Therefore $C$ contains some variable and at most one constant. But then, it is already totally ordered by $\prec'$. Given any other nontrivial class $C$, if $u$ and $v$ are both variables, we already know by **(2a)** that either $u \prec' v$ or $v \prec' u$, but not both. If $u$ is a variable and $v$ is not, by **(2a)** we can only have $u \prec' v$. If both $u$ and $v$ are not variables, then they must be of the form $u = f(u_1, \dots, u_n)$ and $v = f(v_1, \dots, v_n)$, since $C$ is unified by $\theta$. Since $u \neq v$, there is a least $i$ such that $u_i \neq v_i$, and since $\theta$ unifies $u$ and $v$, $\theta$ unifies $u_i$ and $v_i$. But then, because $\prec$ has the subterm property, $u_i, v_i$ belong to some class $C_i$ such that $C_i \ll C$. Therefore, either $u_i \preceq_\theta v_i$ or $v_i \preceq_\theta u_i$, but not both, and thus by **(2b)**, either $u \preceq_\theta v$ or $v \preceq_\theta u$, but not both.

Denote by $\prec_\theta$ the irreflexive portion of $\preceq_\theta$, i.e. $\prec_\theta = \preceq_\theta \setminus \{(t,t) | t \in \mathcal{T}_\Sigma(X)\}$. Clearly, $\prec_\theta$ is a simplification ordering on $\mathcal{T}_\Sigma(X)$. We will be somewhat ambiguous in not differentiating between $\prec_\theta$ and $\preceq_\theta$, and we will say that $\prec_\theta$ is a total simplification ordering on $\mathcal{T}_\Sigma(X)$. (The nuance is that a simplification ordering is strict, hence irreflexive, hence it cannot be total.)

### A.6 Proof of lemma 9.2

If a $\Sigma$-substitution $\theta$ is a rigid $E$-unifier of $u$ and $v$ then $\theta(u) \overset{*}{\cong}_{\theta(E)} \theta(v)$, let $s = LS(\theta(u))$[9], construct $\Sigma^s$ and $E_{u,v}$ as described above, with $z : s$. Extend $\theta'$ such that $\theta'(z) = \theta(u)$. Since $LS(\theta'(z)) = LS(\theta(u)) = s = LS(z)$, $\theta'$ is order-sorted. Since $\theta(eq(u, v)) \overset{*}{\cong}_{\theta(E)} eq(\theta(u), \theta(u))$, clearly

$$\begin{aligned} \theta'(F(u,v)) &\overset{*}{\cong}_{\theta'(E_{u,v})} \theta'(eq(u,v)) \\ &\overset{*}{\cong}_{\theta'(E_{u,v})} \theta'(eq(z,z)) \\ &\overset{*}{\cong}_{\theta'(E_{u,v})} \theta'(T(z)). \end{aligned}$$

[9] Since $E$ is general $LS(\theta(v)) = s$ as well.

Conversely, if there is some $\Sigma^s$-substitution $\theta'$ over $\mathcal{T}_\Sigma(X)$ such that $\theta'(T(z)) \overset{*}{\cong}_{\theta'(E_{u,v})} \theta'(F(u,v))$, because $eq$, $T$, $F$ are not in $\Sigma$, from the way congruence closure works, it must be the case that $\theta'(eq(z,z)) \overset{*}{\cong}_{\theta'(E_{u,v})} \theta'(eq(u,v))$. Letting $\theta = \theta'|_{D(\theta')-\{z\}}$, since the terms in the range of $\theta'$ are in $\mathcal{T}_\Sigma(X)$ and $eq$, $T$, $F$ are not in $\Sigma$, we must also have $\theta'(z) \overset{*}{\cong}_{\theta(E)} \theta(u)$ and $\theta'(z) \overset{*}{\cong}_{\theta(E)} \theta(v)$. Thus $\theta(u) \overset{*}{\cong}_{\theta(E)} \theta(v)$, showing that $\theta$ is a rigid $(\Sigma, E)$-unifier of $u$ and $v$.

### A.7 Proof of lemma 9.4

Let $l_1 \doteq r_1$ and $l_2 \doteq r_2$ be the equations in $E$ involved in the transformation, $\beta$ the address in $Dom(l_1)$ such that $l_1/\beta$ and $l_2$ are $\Sigma$-unifiable via a $\Sigma$-unifier $\sigma$. Let $TU(l_1/\beta, l_2)$ be the triangular form used in the transformation with associated $\Sigma$-substitution $\sigma_T$.

Point 1. This follows from the fact that $TU(l_1/\beta, l_2)$ and $S$ are of the desired form; and $S' = S \cup TU(l_1/\beta, l_2)$.

Point 2. Recall that $E'$ is obtained as follows:

- $E'' = \sigma_T((E - \{l_1 \doteq r_1\})^\dagger \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\})$, and
- $E'' \Rightarrow_{\mathcal{R}} E'$.

By lemma 3.3, $\sigma_T((E - \{l_1 \doteq r_1\})^\dagger$ is general. To show that $\sigma_T(l_1[\beta \leftarrow r_2] \doteq r_1)$ is a general equation we first realize that, by the way $\sigma_T$ was chosen (a special triangular form), $\sigma_T(l_1[\beta \leftarrow r_2]$ is a $\Sigma$-term. Indeed, $LS(\sigma_T(l_1/\beta)) = LS(\sigma_T(l_2))$, and since $l_2 \doteq r_2$ is general, $LS(\sigma_T(l_2)) = LS(\sigma_T(r_2))$. Therefore the result of replacing $\sigma_T(l_2)$ by $\sigma_T(r_2)$ does not violate sort constraints hence $\sigma_T(l_1[\beta \leftarrow r_2])$ is a $\Sigma$-term[10]. Clearly $Var(\sigma_T(l_1[\beta \leftarrow r_2])) = Var(\sigma_T(r_1))$. Similarly, $LS(\rho(\sigma_T(l_1[\beta \leftarrow r_2]))) = LS(\rho(\sigma_T(r_1)))$ for any variable renaming. Hence $\sigma_T(l_1[\beta \leftarrow r_2] \doteq r_1)$ is a general equation. Therefore, $E''$ is general. Since the reduction procedure preserves general axioms (see lemma 8.4), $E'$ is also general.

[10]Actually the reason why we *push* the terms $\sigma_T(l_1/\beta)$ and $\sigma_T(l_2)$ to be of the same sort is precisely to guarrantee that the term resulting from rewriting one by the other be well typed.

3) We only use the fact that $\varphi$ unifies $TU(l_1/\beta, l_2)$, which is true because $TU(l_1/\beta, l_2) \subseteq S'$. First, notice that since $\varphi$ unifies $TU(l_1/\beta, l_2)$ and $\sigma_T$ is the $\Sigma$-substitution associated with $TU(l_1/\beta, l_2)$, by lemma 4.12, $\sigma_T; \varphi = \varphi$, hence

$$\begin{aligned} \varphi(E'') &= \varphi(\sigma_T((E - \{l_1 \doteq r_1\})^\dagger \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\})) \\ &= \varphi((E - \{l_1 \doteq r_1\})^\dagger \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\}). \end{aligned}$$

We now show that $\varphi(E)$ and $\varphi(E'')$ are rigid equivalent. By the above, it is enough to show that

**(a)** $\varphi(l_1 \doteq r_1)$ can be deduced from $\varphi(l_1[\beta \leftarrow r_2] \doteq r_1)$ and $\varphi(l_2 \doteq r_2)$; and vice versa, that

**(b)** $\varphi(l_1[\beta \leftarrow r_2] \doteq r_1)$ can be deduced from $\varphi(l_1 \doteq r_1)$ and $\varphi(l_2 \doteq r_2)$.

By lemma 4.15, since $\varphi$ unifies $TU(l_1/\beta, l_2)$, it unifies $l_1/\beta$ and $l_2$. To show **(a)**, notice that

$$\varphi(l_1) = \varphi(l_1)[\beta \leftarrow \varphi(l_2)] = \varphi(l_1[\beta \leftarrow l_2]) \rightarrow \varphi(l_1[\beta \leftarrow r_2]) \rightarrow \varphi(r_1).$$

To see that **(b)** holds, notice that

$$\varphi(l_1[\beta \leftarrow r_2]) \ {}_{\varphi(l_2 \rightarrow r_2)}\!\leftarrow \varphi(l_1[\beta \leftarrow l_2]) = \varphi(l_1) \rightarrow_{\varphi(l_1 \doteq r_1)} \varphi(r_1).$$

By the soundness of the reduction procedure (theorem 8.7) $E''$ and $E'$ are also rigid equivalent, hence for any $\Sigma$-substitution $\varphi$, $\varphi(E'')$ and $\varphi(E')$ are rigid equivalent. Since we just showed that $\varphi(E)$ and $\varphi(E'')$ are rigid equivalent, we have that $\varphi(E)$ and $\varphi(E')$ are rigid equivalent.

# Contents