

An Extension of Baaz's Algorithm to E-semi-unification with One Inequality

Preliminary Report

Alberto Oliart

Laboratorio Nacional de Informática Avanzada A.C.
aoliart@xalapa.lania.mx*

Abstract

In this paper we present a preliminary research report on the problem of finding procedures to solve some cases of the Uniform E -Semi-unification Problem, where E is a set of equations defining an equational theory. The problem in general is undecidable, but for certain theories, as is the case with E -Unification, there may be theories for which it is decidable.

1 Introduction

The Semi-Unification problem has applications in areas of computer science like programming languages, computational linguistics, term rewriting, and automated deduction. Despite this, the body of knowledge on Semi-unification is not comparable to that of Unification, which has been for a long time an ongoing branch of study in computer science.

The general problem of semi-unification (see below for definitions) has been shown to be undecidable, but there are a number of restrictions and/or generalizations that can be shown to be decidable.

In this paper we present an introduction and a preliminary research report on the problem of uniform semi-unification but where equality is now modulo an equational theory E . We call this Uniform E -Semi-Unification. The analogous E -Unification is a problem with a large list of results, while very little is known about E -Semi-Unification.

The paper is divided as follows: Next section gives some basic definitions. Section 3 describes an algorithm for solving the Uniform Semi-Unification problem. An extension of this algorithm is used later on to describe a solution to E -Semi-Unification for a particular E . Section 4 introduces the Uniform E -Semi-Unification problem, giving a very short introduction to E -Unification. The following section describes a general method for finding a solution if one exists, but that may fail to stop if none exists, and finally we describe an algorithm to solve

the C -Semi-unification problem, where C is a particular theory.

2 Basic Definitions

Let $\Sigma = \{f_1, f_2, \dots, f_k\}$, $k \geq 1$ be a first order signature. Let X a countably infinite set of variables, and let $\mathcal{T}_{\Sigma, X}$ be the set of all first order terms over Σ and X .

Definition 1 A substitution is a function $\sigma : X \rightarrow \mathcal{T}_{\Sigma, X}$ such that $\sigma(x) \in X$ for almost all $x \in X$. The set of variables x such that $\sigma(x) \notin X$ is called $Dom(\sigma)$. A substitution can be extended naturally to a function $\sigma : \mathcal{T}_{\Sigma, X} \rightarrow \mathcal{T}_{\Sigma, X}$. An injective substitution σ is called a variable renaming if and only if for all $x \in X$, $\sigma(x) \in X$.

Definition 2 If σ is a substitution and $V \subseteq X$ then $\sigma|_V$, the restriction of σ to V is defined as follows :

$$\sigma|_V(x) = \begin{cases} \sigma(x), & \text{if } x \in V \\ x, & \text{otherwise.} \end{cases}$$

Definition 3 If σ_1 and σ_2 are substitution then we say that $\sigma_1 \leq \sigma_2$ if and only if there is a substitution ρ such that $\sigma_2 = \rho \circ \sigma_1$, where \circ denotes functional composition.

Definition 4 If $t, u \in \mathcal{T}_{\Sigma, X}$ then we say that $t \leq u$ if and only if there is a substitution σ such that $\sigma(t) = u$.

Definition 5 A unification instance is a set of pairs of terms $\Gamma = \{t_1 \stackrel{?}{=} u_1, \dots, t_n \stackrel{?}{=} u_n\}$. We say that a substitution σ is a solution (unifier) of Γ if and only if $\sigma(t_1) = \sigma(u_1), \dots, \sigma(t_n) = \sigma(u_n)$.

Definition 6 A semi-unification instance is a set of pairs of terms $\Gamma = \{t_1 \leq^? u_1, \dots, t_n \leq^? u_n\}$. We say that a semi-unification instance Γ has a solution (semi-unifier) σ if and only if σ is a substitution such that $\sigma(t_1) \leq \sigma(u_1), \dots, \sigma(t_n) \leq \sigma(u_n)$.

The substitutions $\rho_1, \rho_2, \dots, \rho_n$ such that $\rho_1(\sigma(t_1)) = \sigma(u_1), \dots, \rho_n(\sigma(t_n)) = \sigma(u_n)$ are called matching substitutions. These substitutions are determined uniquely by the semi-unifier.

The *semi-unification problem* (SUP for short) is: Given an arbitrary semi-unification instance Γ decide

whether Γ has a semi-unifier. If Γ contains a single pair, then we call it the *uniform semi-unification problem* (USUP). The name “uniform” comes from the fact that it is equivalent to restricting the matching substitutions to be the same substitution.

It has been shown in [Kfoury *et al.*, 1993] that SUP is undecidable. USUP, on the other hand, is decidable [Kapur *et al.*, 1991; Henglein, 1993; Pudlack, 1988], with a time complexity of $O(n^2\alpha(n)^2)$, where $\alpha(n)$ is the inverse of Ackerman’s function [Oliart and Snyder, 1998].

Definition 7 *A substitution σ is the most general unifier of a unification instance if and only if, for any other solution σ' , $\sigma \leq \sigma'$.*

A substitution σ is the principal solution of a semi-unification instance if and only if for any other solution σ' , $\sigma \leq \sigma'$.

As is the case with unification, it is the case that if a semi-unifier exists, then there is a principal one. It is not the case that all solutions can be calculated from the principal one, as is the case with unification.

Definition 8 *Given a set of terms $\{t_1, \dots, t_n\}$, the set of all variables occurring in t_1, \dots, t_n is denoted by $V(t_1, \dots, t_n)$.*

Definition 9 *If $V \subseteq X$ is a set of variables then:*

1. *If σ_1, σ_2 are substitutions then $\sigma_1 \leq_V \sigma_2$ if and only if $\sigma_1|_V \leq \sigma_2|_V$.*
2. *If σ_1, σ_2 are substitutions then $\sigma_1 =_V \sigma_2$ if and only if $\sigma_1|_V = \sigma_2|_V$.*

3 Baaz’s Procedure for Uniform Semi-Unification.

In [Baaz, 1993], Mathias Baaz introduces a partial procedure for SUP. Since SUP is not decidable, the procedure may never stop if there is no solution to the given instance. The procedure presented here is a modification of the one introduced by Baaz, and is restricted to the uniform case. In our case the procedure is complete, always terminates, and produces a principal solution for the given instance if one exists. Baaz’s procedure, as presented in [Baaz, 1993] will not always terminate, even for the Uniform case.

We need to extend the set X of variables to a set of variables \overline{X} whose members are of the form x_w , where $x \in X$ and $w \in \mathbb{N}$. Let $\mathcal{T}_{\Sigma, \overline{X}}$ be the set of all finite terms over \overline{X} and Σ . The variable $x_0 \in \overline{X}$ is identified with the variable $x \in X$. This way we can think of X as a subset of \overline{X} , and also of \mathcal{T} as a subset of $\overline{\mathcal{T}}$. Let $[\] : \overline{X} \rightarrow \overline{X}$ be the function defined by :

$$[x_v] = x_{v+1}$$

for every $x_v \in \overline{X}$. This function can be naturally extended to a function $[\]_w : \overline{\mathcal{T}} \rightarrow \overline{\mathcal{T}}$.

The procedure starts with an instance $\Gamma = \{t \leq^? u\}$ with variables in X . The procedure does the following:

1. Let $\sigma_0 = id$ and let $i = 0$.
2. Let $\Gamma_{i+1} = \{[\sigma_i(t)] \stackrel{?}{=} \sigma_i(u)\}$ be an instance of unification. If Γ_{i+1} has no solution, then STOP, Γ has no semi-unifiers. Else, continue.
3. Let γ_{i+1} be the m.g.u. of Γ_{i+1} . If $\gamma_{i+1}|_{V(\sigma_i(u))}$ is a renaming, then STOP. Γ has a solution, namely σ_i . Else, continue.
4. Let $\sigma_{i+1} = \gamma_{i+1} \circ \sigma_i$. If there is an $x_n \in \overline{X}$ such that $\sigma_{i+1}(x_n) = f(\dots, x_{n+j}, \dots)$ where $j \in \mathbb{N}$, $j \geq 0$ then STOP. Γ has no semi-unifier (this is, in a sense, equivalent to the occurs check in unification). Else, let $i = i + 1$, go to 2.

Example 1 $\Gamma = \{f(f(x, y), u) \leq^? f(u, v)\}$

Consider instance $\Gamma = \{f(f(x, y), u) \leq^? f(u, v)\}$, then we have :

$\Gamma_1 = \{f(f(x_1, y_1), u_1) \stackrel{?}{=} f(u, v)\}$ and $\sigma_1 = \{u := f(x_1, y_1)\}$

$\Gamma_2 = \{f(f(x_1, y_1), f(x_2, y_2)) \stackrel{?}{=} f(f(x_1, y_1), v)\}$ and $\sigma_2 = \{u := f(x_1, y_1), v := f(x_2, y_2)\}$.

The last instance produced is:

$$\{f(f(x_1, y_1), f(x_2, y_2)) \stackrel{?}{=} f(f(x_1, y_1), f(x_2, y_2))\}$$

which has as most general unifier the identity, which is a renaming on the variables of the term on the left. This ends the procedure and the principal semi-unifier is σ_2 .

Example 2 $\Gamma = \{f(x, y) \leq^? x\}$

In this case, we have that the first instance of unification is

$$\Gamma_1 = \{f(x_1, y_1) \stackrel{?}{=} x\},$$

which has as unifier the substitution

$$\gamma_1 = \{x := f(x_1, y_1)\}.$$

In this case the procedure stops because we have the cycle condition.

The proof of correctness of the algorithm is based in the following lemmas. Due to lack of space, we leave the proofs of the lemmas for the full paper.

Lemma 1 *Let t and u be first order terms. If γ is the most general unifier of $\{[\sigma_i(t)] \stackrel{?}{=} \sigma_i(u)\}$ where σ_i is the i -th is the substitution generated by the i -th step of the algorithm for some $i \in \mathbb{N}$, then γ is the most general substitution over $V(t, u)$ such that $t \leq \gamma(u)$.*

Lemma 2 (Completeness) *Let $\Gamma = \{t \stackrel{?}{\leq} u\}$ be an instance of semi-unification. If S is a semi-unifier of Γ then $\forall m \exists S_m$ such that $S =_{V(t, u)} S_m \circ \sigma_m$, where σ_m is the substitution generated by the m -th step of the algorithm, and there is a m.g.u. of $\{[\gamma_m(t)] \stackrel{?}{=} \gamma_m(u)\}$.*

Lemma 3 *If S is a semi-unifier of $\Gamma = \{t \stackrel{?}{\leq} u\}$, then there is an m such that $\sigma_m = \sigma_{m+1}$.*

Lemma 4 *If m is the least number such that $\sigma_m = \sigma_{m+1}$, and γ_{m+1} exists, then σ_m is a semi-unifier.*

Lemma 5 *The procedure presented above always halts.*

4 Uniform E-Semi-unification

In this section we define the E-Semi-unification problem, as well as its uniform restriction. The basic difference between Semi-unification (which we will call *syntactic semi-unification*) and E-semi-unification is that equality is now modulo an equational theory E . We also give a very brief introduction to the E-Unification with a survey of relevant results. For a more complete survey see [Baader and Snyder, 1999]. Some of the material presented here has been taken from this survey.

4.1 E-Unification

Definition 10 *Let E be a set identities (pairs of terms). If $(t, u) \in E$ we write $t \approx u$. An equational theory defined by E^1 , is the least congruence relation on $\mathcal{T}_{\Sigma, X}$ that contains E and is closed under substitutions. We denote this relation by $=_E$.*

As an example, let $C = \{f(x, y) \approx f(y, x)\}$, where f is a binary symbol in the signature. We obviously have that $f(x, y) =_C f(y, x)$. We also have the following: $f(f(a, b), c) =_C f(c, f(a, b))$, where a, b, c are functional symbols of arity zero (constants).

Definition 11 *Let E be a set of identities. An E-unification instance is a set of pairs of terms $\Gamma = \{t_1 =_E^? u_1, \dots, t_n =_E^? u_n\}$. A E-unification instance has a solution (E-unifier) if and only if there is a substitution σ such that $\sigma(t_1) =_E \sigma(u_1), \dots, \sigma(t_n) =_E \sigma(u_n)$.*

The E-unification problem (EUP) is: Given an E-unification instance Γ , whether Γ has a solution.

We denote the set of all E-unifiers of a given Γ as $\mathcal{U}_E(\Gamma)$

Definition 12 *Let E be an equational theory, and X a set of variables. A substitution σ_1 is more general modulo E on X than σ_2 if and only if there is a substitution ρ such that $\rho(\sigma_1(x)) = \sigma_2(x)$ for all $x \in X$.*

Syntactic unification is clearly a special case of E-unification, with $E = \emptyset$. Obviously, an instance Γ of E-unification has a solution if and only if $\mathcal{U}_E(\Gamma) \neq \emptyset$. Unlike the unification problem, EUP is in general undecidable. This is due to the undecidability of the word problem. There are a number of other properties of syntactic unification that are not satisfied in general. It is the case that, for a given set of equations E , given an instance of E-Unification Γ , if Γ has a syntactic solution σ , this is also E-unifier, but the sets $\mathcal{U}_E(\Gamma)$ and $\mathcal{U}_\emptyset(\Gamma)$ are not necessarily the same. As a matter of fact, we have that $\mathcal{U}_\emptyset(\Gamma) \subseteq \mathcal{U}_E(\Gamma)$.

¹We identify the set of equations E with its equational theory.

As an example, consider again the equational theory $C = \{f(x, y) = f(y, x)\}$, and consider the instance $\Gamma = \{f(x, y) =_C^? f(a, b)\}$, where a, b are constants. This has as a solution the substitution σ_1 defined by:

$$\sigma_1(z) = \begin{cases} a & \text{if } z = x \\ b & \text{if } z = y \\ z & \text{otherwise} \end{cases}$$

This substitution is clearly a syntactic unifier, but the substitution σ_2 defined by:

$$\sigma_2(z) = \begin{cases} b & \text{if } z = x \\ a & \text{if } z = y \\ z & \text{otherwise} \end{cases}$$

Clearly, σ_2 is not a syntactic unifier. We also have that $\sigma_1 \not\leq_C \sigma_2$ and $\sigma_2 \not\leq_C \sigma_1$, and therefore there is no most general C-unifier for Γ .

Definition 13 *Let Γ be an E-unification instance, and let X be the set of variables occurring in Γ . A complete set of E-unifiers for Γ is a set CU such that $CU \subseteq \mathcal{U}_E(\Gamma)$ and for each $\sigma_1 \in \mathcal{U}_E(\Gamma)$ there is $\sigma_2 \in CU$ such that $\sigma_2 \leq_E \sigma_1$ over X .*

We say that CU is a minimal complete set of E-unifiers if and only if for any two elements of CU σ_1, σ_2 we have that σ_1 and σ_2 are not comparable with respect to leq_E .

Given a theory E , an instance Γ of E-unification has a most general E-unifier if and only if any minimal complete set of E-unifiers has only one element. As we have already seen in the case of C , for some theories not all instances have most general E-unifiers.

Definition 14 *Let E be an equational theory. Let Γ be an instance of E-unification. Γ is of type unitary (finitary, infinitary) if and only if it has a minimal complete set of E-unifiers of cardinality 1 (finite cardinality, infinite cardinality). It is of type 0 (zero) if a minimal complete set does not exist.*

The unification type of E with respect to Σ is the maximal type of an E-unification problem over Σ .

The unification type of $\Gamma = \{f(x, y) =_C^? f(a, b)\}$ is finitary, since the minimal complete set of C-unifiers is the one shown above. As a matter of fact, C is of type finitary over $\Sigma = \{f\}$.

Consider now the theory $A = \{g(x, g(y, z)) = g(g(x, y), z)\}$, and consider the instance $\Gamma = \{g(a, x) =_A^? g(x, a)\}$. Γ is an example of an infinitary instance of A-Unification, so therefore A is an infinitary equational theory.

There is another issue, which refers to the signatures of the equational theory and the terms in the instances of unification. The unification properties of a particular equational theory may depend on whether the signature of the terms in the instances is a subset or a superset of the signature of the equational theory.

Definition 15 Let E be an equational theory over the set of symbols \mathcal{F} , and let Γ be an E -Unification instance over Σ . We say the Γ is elementary if and only if $\Sigma \subseteq \mathcal{F}$. We say that Γ is an instance with constants if and only if $\Sigma \setminus \mathcal{F}$ is a set of constants. We say the Γ is general if and only if $\mathcal{F} \subseteq \Sigma$.

It turns out (see [Baader and Snyder, 1999]) that the actual type of a E -Unification instance, given an equational theory E depends in part on whether it is elementary, with constants or general. As an example, while C is finitary for the three kinds of instances, the combination of commutativity, associativity, and $U = \{f(x, e) = x\}$ which we call ACU is unitary for elementary instances, but finitary for all other kinds.

One more difference between syntactic unification and E -Unification is that only for the general case it is true that an instance with $n \in \mathbb{N}$ pairs is equivalent, in general, to an instance with only one pair. This will also be the case with E -Semi-Unification.

4.2 Uniform E -Semi-Unification

We now define the Uniform E -Semi-unification problem. We call it *uniform* because, as in the syntactic case, we only consider instances that contain exactly one pair.

Definition 16 Let E be a set of identities. We say that $t \leq_E u$, where t and u are terms, if there is a substitution ρ such that $\rho(t) =_E u$.

Definition 17 Let E be a set of identities. An E -semi-unification instance is a set of pairs of terms $\Gamma = \{t_1 =_E^? u_1, \dots, t_n =_E^? u_n\}$. A E -semi-unification instance has a solution if and only if there is a substitution σ such that $\sigma(t_1) \leq_E \sigma(u_1), \dots, \sigma(t_n) \leq_E \sigma(u_n)$.

We denote the set of all E -Semi-unifiers of a given Γ as $\mathcal{S}_E(\Gamma)$

Of course, the same issues discussed in section 4.1 apply in the case of Semi-unification. We may no longer have the principal solution property, and we also have to take into account the signatures of the equational theory and the instances to be considered. We now give analogous definitions to the ones given for E -Unification.

Definition 18 Let Γ be a uniform E -Semi-unification instance, and let X be the set of variables occurring in Γ . A complete set of E -Semi-unifiers for Γ is a set CSU such that $CSU \subseteq \mathcal{S}_E(\Gamma)$ and for each $\sigma_1 \in \mathcal{S}_E(\Gamma)$ there is $\sigma_2 \in CSU$ such that $\sigma_2 \leq_E \sigma_1$ over X .

We say that CU is a minimal complete set of E -Semi-unifiers if and only if for any two elements of CU σ_1, σ_2 we have that σ_1 and σ_2 are not comparable with respect to leq_E .

Definition 19 Let E be an equational theory. Let Γ be an instance of uniform E -Semi-unification. Γ is of type unitary (finitary, infinitary) if and only if it has a minimal complete set of E -Semi-unifiers of cardinality 1 (finite, infinite). It is of type 0 (zero) if a minimal complete set does not exist.

The Semi-unification type of E with respect to Σ is the maximal type of an E -Semi-unification problem over Σ .

Definition 20 Let E be an equational theory over the set of symbols \mathcal{F} , and let Γ be an E -Semi-unification instance over Σ . We say the Γ is elementary if and only if $\Sigma \subseteq \mathcal{F}$. We say that Γ is an instance with constants if and only if $\Sigma \setminus \mathcal{F}$ is a set of constants. We say the Γ is general if and only if $\mathcal{F} \subseteq \Sigma$.

We consider now some examples of C -Semi-unification instances. The first one is $\Gamma_1 = \{f(x, y) \leq_C^? f(a, b)\}$, where a, b are constants. This is an instance with constants, and it has type 1, since the identity is a C -semi-unifier. Let $\Gamma_2 = \{f(a, b) \leq_C^? f(x, y)\}$. Γ_2 is of type finitary, since the C -Semi-unifiers are:

$$\sigma_1(z) = \begin{cases} a & \text{if } z = x \\ b & \text{if } z = y \\ z & \text{otherwise} \end{cases}$$

and

$$\sigma_2(z) = \begin{cases} b & \text{if } z = x \\ a & \text{if } z = y \\ z & \text{otherwise} \end{cases}$$

and as seen before, neither is more general than the other. It turns out that C is of Semi-unification type finitary. This is shown in section 6, where we present an algorithm to solve it which is an extension of the procedure presented in next section.

5 An Extension to Baaz's Procedure

The idea of applying Baaz's procedure to E -Semi-Unification is due to Rittri in [Rittri, 1994; 1995]. In his paper Rittri applies blindly the procedure using two different equational theories. The two papers apply the procedure to the general case of E -Semi-Unification, and in our case, we want to apply the same procedure to the Uniform case. Another difference is that Rittri considers only theories for which the Unification problem is of class 1 (they have the most general unifier property), and in our case we will try to use it in the case of finitary E -Unification problems. It is obvious that in the case of infinitary problems the procedure will not work.

The basic idea is to use the algorithm that solves a finitary E -Unification problem where the procedure uses the Unification algorithm. This doesn't mean that with this we will have algorithms for the given Uniform E -Semi-Unification problem, probably some kind of checks will have to be added to the procedure, as is the case with the algorithm shown for the syntactic case of Uniform Semi-Unification. We have applied this procedure successfully with commutativity, as will be explained in the next section. First we explain the general form of the procedure, and in the next section we will explain the modifications made for commutativity. We note that the general form will always find a solution if one exists, but it may

not terminate when one does not exist, see [Rittri, 1994; 1995].

We now describe the general algorithm. As in the case of syntactic Semi-unification, the algorithm receives as input an instance of E-Semi-unification $\Gamma = \{t \leq^? u\}$. The unification algorithm used here is an E-Unification algorithm for a finitary EUP.

We need to change the definition of \overline{X} , the set that extends the set of variables, and \square so that we can generalize the procedure presented in section 3. Instead of using natural numbers as indices for the variables appearing in the terms, we will use strings of natural numbers. Now the elements of \overline{X} are of the form x_w , where $w \in \mathbb{N}^*$. Let $\overline{\mathcal{T}}_{\Sigma, \overline{X}}$ be the set of all finite terms over \overline{X} and Σ . The variable $x_\varepsilon \in \overline{X}$, where ε is the empty string, is identified with the variable $x \in X$. This way we can think of X as a subset of \overline{X} , and also of \mathcal{T} as a subset of $\overline{\mathcal{T}}$. For $w \in \mathbb{N}^*$ let $[\]_w : \overline{X} \rightarrow \overline{X}$ be the function defined by:

$$[x_v]_w = x_{vw}$$

for every $x_v \in \overline{X}$.

We now show a generalized version of the Baaz procedure. The procedure presented next is an extension to Baaz's procedure as presented in [Baaz, 1993], but in our case with the restriction that an instance can only have one inequality.

1. Let $U_0 = \{\Sigma_0 = id\}$ and $i = 0$.
2. Let $U_{i+1} = \emptyset$. For each $\sigma_i^j \in U_i$ do
 - (a) Let $\Gamma_i^j = \{[\sigma_i^j(t)]_i =_C^? \sigma_i^j(u)\}$ be an instance of C -Unification.
 - (b) Let $\mathcal{U}_C(\Gamma_i^j)$ be the minimal complete set of C -Unifiers. For each $\gamma \in \mathcal{U}_C(\Gamma_i^j)$ do:
 - i. If γ is a renaming on $\sigma_i^j(u)$, then σ_i^j is a solution, add it to $S\mathcal{U}_C(\Gamma)$, if there is no $\sigma \in S\mathcal{U}_C(\Gamma)$ such that $\sigma \leq_C \sigma_i^j$.
 - ii. Else, $U_{i+1} = U_{i+1} \cup \{\gamma \circ \sigma_i^j\}$.
3. If $U_{i+1} = \emptyset$ then STOP. If $S\mathcal{U}_C(\Gamma)$ is empty, then Γ has no solutions, else return $S\mathcal{U}_C(\Gamma)$.
4. Goto 2.

This algorithm builds a set of E -Semi-unifiers. It may never finish, even if there are some solutions to the algorithm, given that it searches for approximations in a breath first search fashion, but if there is a E -Semi-unifier it will find it sooner or later. On the other hand, it may run forever in some cases when there are no solutions. The proof of this is very similar to the one for the general case given by [Baaz, 1993].

If the corresponding unification type of E is 1, this is, it has the most general unifier property, then this algorithm becomes very similar to the one presented in section 3. The only real difference is that in this case we

have no way to "catch" cycles. For example, if we run the above algorithm for the syntactic case ($E = \emptyset$), the algorithm will not stop if we give it as input the second example in section 3. This is because there is no way of "catching" the cycle. This shows that, in order to have a total procedure when this general procedure does not give us one, we need to have some "external" check, or modify the procedure in some way. In the syntactic case, we used the trick of using natural numbers as the indexes, and instead of using the concatenation we incremented the indexes. This actually counts the number of times the function \square is applied to the terms, allowing the cycle stop if a cycle condition was found.

Example 3 $\Gamma = \{f(a, b) \leq_C^? f(x, y)\}$

We consider now the case of commutativity. In the example above, the application of the renamings have no effect, so $\Gamma_0 = \{f(a, b) =_C^? f(x, y)\}$. The C -Unification algorithm produces the minimal set of C -unifiers: $\mathcal{U}_C(\Gamma_1) = \{\gamma_1^1 = \{x := a, y := b\}, \gamma_2 = \{x := b, y := a\}\}$. With this we get the set $U_1 = \{\sigma_1^1 = \gamma_1, \sigma_2^1 = \gamma_2\}$. For the next cycle, we have the instances of C -Unification $\Gamma_1^1 = \{f(a, b) =_C^? f(a, b)\}$ and $\Gamma_2^1 = \{f(b, a) =_C^? f(a, b)\}$ both having the identity as unique solution, and with this the algorithm ends.

Example 4 $\Gamma = \{f(f(x, y), z) \leq_C^? f(x, z)\}$

In this case, we get as the first instance of C -Unification is

$$\Gamma_1 = \{f(f(x_1, y_1), z_1) =_C^? f(x, z)\}$$

which has the minimal set of solutions $\mathcal{U}_C(\Gamma_1) = \{\gamma_1^1 = \{x := f(x_1, y_1), z_1 := z\}, \gamma_1^2 = \{z := f(x_1, y_1), z_1 := x\}\}$.

It is easy to see that $\sigma_1^1 = \gamma_1^2$ actually leads to a solution of the original problem, while the other will produce a cycle. In this case, the algorithm never stops, but it actually finds a solution.

This last example shows that if we use the same trick we used in the algorithm for syntactic uniform semi-unification, then we would obtain a total procedure that finds a minimal set of C -semi-unifiers. In the next section we describe the total algorithm.

6 An Example: Commutativity

We now apply the procedure described above to the instances of C -Semi-unification. As we have seen before, C -Unification is finitary for all kinds of instances, which means that we can apply the above procedure. The only problem is that none of the algorithms known to solve the C -Unification problem directly produces a minimal complete set of unifiers. This can be easily solved because the algorithms actually return a finite set of unifiers, and a minimal one can be obtained by brute force. Another advantage of the C case is its close similarity to the syntactic case.

If we apply the algorithm described in section 5 to the C -Semi-unification problem as is, it would not produce

a total procedure, given that the same example shown in section 3 will run indefinitely, without an answer. We need to use the same trick used in the syntactic case. The procedure shown below is actually an extension to the procedure in section 3, which in turn can be seen as an extension to the procedure shown in section 5. The output is a set of solutions, which we denote $SU_C(\Gamma)$.

1. Let $U_0 = \{\Sigma_0 = id\}$ and $i = 0$.
2. Let $U_{i+1} = \emptyset$. For each $\sigma_i^j \in U_i$ do
 - (a) Let $\Gamma_i^j = \{[\sigma_i^j(t)]_i =_C^? \sigma_i^j(u)\}$ be an instance of C -Unification.
 - (b) Let $\mathcal{U}_C(\Gamma_i^j)$ be the minimal complete set of C -Unifiers. For each $\gamma \in \mathcal{U}_C(\Gamma_i^j)$ do:
 - i. If γ is a renaming on $\sigma_i^j(u)$, then σ_i^j is a solution, add it to $SU_C(\Gamma)$, if there is no $\sigma \in SU_C(\Gamma)$ such that $\sigma \leq_C \sigma_i^j$.
 - ii. Else, if there is no x_k such that $\gamma(\sigma_i^j(x_k)) = f(\dots, x_{k+m}, \dots)$ for $m \geq 0$, then $U_{i+1} = U_{i+1} \cup \{\gamma \circ \sigma_i^j\}$.
3. If $U_{i+1} = \emptyset$ then STOP. If $SU_C(\Gamma)$ is empty, then Γ has no solutions, else return $SU_C(\Gamma)$.
4. Goto 2.

It should be easy for the reader to check that for examples 2 and 4 the algorithm presented here ends, returning no solution for the instance of example 2, and a minimal complete set of solutions for the instance in example 4.

7 Conclusions and Future Work

We presented here two modifications of an algorithm introduced by Mathias Baaz to show that the general syntactic semi-unification problem has the principal solution property. The first modification gives us a total algorithm to solve the Uniform Semi-Unification Problem. The idea for the second modification is due to Rittri, who used it in a more primitive way trying to solve E -Semi-unification problems that had to do with typing systems for programming languages. We also showed how to apply the technique to one particular example, namely commutativity. We are now using this technique trying to find a more general way of finding procedures to solve E -Semi-unification problems for some equational theories.

We have worked with some theories that are of type 1, as is the case of Boolean Rings, with only partial success. We are trying to embed into the Baaz algorithm some of the techniques that have given good results in E -Unification, as is the case of narrowing.

References

- [Baader and Snyder, 1999] F. Baader and W. Snyder. Unification theory. In *Handbook on Automated Deduction*. Springer, 1999.
- [Baaz, 1993] Matthias Baaz. Note on the existence of most general unifiers. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory and Logical Complexity*, pages 20–29. Oxford University Press, 1993.
- [Henglein, 1993] F. Henglein. Type inference with polymorphic recursion. *ACM Transactions on Programming Languages and Systems*, 15(2), April 1993.
- [Kapur et al., 1991] D. Kapur, D. Musser, P. Narendran, and Stillman J. Semi-unification. *Theoretical Computer Science*, 81(2):169–188, April 1991.
- [Kfoury et al., 1993] A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. The undecidability of the semi-unification problem. *Information and Computation*, 102(1):83–101, January 1993.
- [Oliart and Snyder, 1998] A. Oliart and W. Snyder. A fast algorithm for semi-unification. In *CADE-15*, LNCS. Springer-Verlag, 1998.
- [Pudlack, 1988] P. Pudlack. On a unification problem related to kreisel’s conjecture. *Commentationes Mathematicae Universitatis Carolinae*, 1988. Prague Czechoslovakia.
- [Rittri, 1994] M. Rittri. Semi-unification of two terms in abelian groups. *Info. Processing Letters*, (52):61–68, 1994.
- [Rittri, 1995] M. Rittri. Dimensions inference under polymorphic recursion. In *7th ACM Conference on Functional Programming Languages and Computer Architecture*, pages 147–159. ACM, ACM Press, 1995.