

Hierarchical Combination

Serdar Erbatur⁵, Deepak Kapur^{1,*}, Andrew M. Marshall^{2,**},
Paliath Narendran^{3,***}, and Christophe Ringeissen⁴

¹ University of New Mexico (USA)

² Naval Research Laboratory (USA)

³ University at Albany, SUNY (USA)

⁴ LORIA – INRIA Nancy-Grand Est (France)

⁵ Dipartimento di Informatica Università degli Studi di Verona (Italy)

Abstract. A novel approach is described for the combination of unification algorithms for two equational theories E_1 and E_2 which share function symbols. We are able to identify a set of restrictions and a combination method such that if the restrictions are satisfied the method produces a unification algorithm for the union of non-disjoint equational theories. Furthermore, we identify a class of theories satisfying the restrictions. The critical characteristics of the class is the hierarchical organization and the shared symbols being restricted to “inner constructors”.

1 Introduction

Unification (or equation solving) is a fundamental problem in science and has been studied in different forms for thousands of years. In computer science, the concept of unification was popularized by Robinson with his resolution rule of inference for proving valid formulas in first-order predicate calculus. Since then, unification has been studied very extensively in theoretical computer science and artificial intelligence. Unification is extensively used in automated reasoning systems and other tools, such as cryptographic protocol analysis tools.

Even though the unification problem over a general first-order theory is undecidable, the problem is decidable for many first-order theories and numerous algorithms have been proposed. A critical question in the search for such methods is how to obtain a unification algorithm for the combination of non-disjoint equational theories when there exists unification algorithms for the constituent theories. The problem is known to be difficult and can easily be seen to be undecidable in the general case. Therefore, previous work has focused on identifying specific conditions and methods in which the problem is decidable. We continue the investigation in this paper, building on previous combination results, [5, 11, 18] and [13]. We are able to develop a novel approach to the non-disjoint combination problem. The approach is based on a new set of restrictions and combination method such that if the restrictions are satisfied the

* Partially supported by the NSF grant CNS-0905222.

** Supported by an ASEE postdoctoral fellowship under contract to the NRL.

*** Partially supported by the NSF grant CNS-0905286.

method produces an algorithm for the unification problem in the union of non-disjoint equational theories. Furthermore, we identify a set of properties on the constituent theories, E_1 and E_2 , such that theories characterized by these properties satisfy the restrictions and thus the combination algorithm is valid for these theories. The main properties of this class are: a hierarchical organization of E_1 and E_2 , R_1 is a left-linear, convergent rewrite system corresponding to E_1 , and the shared symbols are “inner constructors” of R_1 which are not equated in E_2 .

Due to space constraints we do not attempt to give a complete review of the *related work* in this active research field but rather point the reader to a non-exhaustive list of related and important works. Combination methods for theories with disjoint signatures have been investigated in [22, 21], solved in general in [19] and extended in [5, 9]. Combination methods for theories with non-disjoint signatures is an active area. Some excellent work in this area includes [11, 18, 8, 7, 3].

2 Preliminaries

We use the standard notation of equational unification [6] and term rewriting systems [4]. The set of Σ -terms, denoted by $T(\Sigma, \mathcal{X})$, is built over the signature Σ and the (countably infinite) set of variables $\mathcal{X}$. The terms $t|_p$ and $t[u]_p$ denote respectively the subterm of t at the position p , and the term t having u as subterm at position p . The symbol of t occurring at the position p (resp. the top symbol of t) is written $t(p)$ (resp. $t(\epsilon)$). A Σ -rooted term is a term whose top symbol is in Σ . The set of variables of a term t is denoted by $Var(t)$. A term is *ground* if it contains no variables. A term t is *linear* if each variable of t occurs only once in t . A Σ -substitution σ is an endomorphism of $T(\Sigma, \mathcal{X})$ denoted by $\{x_1 \mapsto t_1, \dots, x_n \mapsto t_n\}$ if there are only finitely many variables $x_1, \dots, x_n$ not mapped to themselves. We call domain of σ the set of variables $\{x_1, \dots, x_n\}$ and range of σ the set of terms $\{t_1, \dots, t_n\}$. Application of a substitution σ to a term t (resp. a substitution ϕ) may be written $t\sigma$ (resp. $\phi\sigma$).

An alien subterm of a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted term t (resp. Σ_2 -rooted term) is a Σ_2 -rooted subterm (resp. $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted subterm) of t such that all its superterms are $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted (resp. Σ_2 -rooted).

Given a first-order signature Σ , and a set E of Σ -axioms (i.e., pairs of Σ -terms, denoted by $l = r$), the *equational theory* $=_E$ is the congruence closure of E under the law of substitutivity. By a slight abuse of terminology, E will be often called an equational theory. An axiom $l = r$ is *variable-preserving* if $Var(l) = Var(r)$. An axiom $l = r$ is *linear* (resp. *collapse-free*) if l and r are linear (resp. non-variable terms). An equational theory is *variable-preserving* (resp. linear/collapse-free) if all its axioms are variable-preserving (resp. linear/collapse-free). An equational theory E is *finite* if for each term t , there are finitely many terms s such that $t =_E s$.

A Σ -equation is a pair of Σ -terms denoted by $s =^? t$. An E -unification problem is a set of Σ -equations, $\mathcal{S} = \{s_1 =^? t_1, \dots, s_m =^? t_m\}$. The set of variables

of $\mathcal{S}$ is denoted by $\text{Var}(\mathcal{S})$. Given a signature $\Sigma' \subseteq \Sigma$, $\mathcal{S}|_{\Sigma'}$ denotes the set of Σ' -equations occurring in $\mathcal{S}$.

A solution to $\mathcal{S}$, called an *E-unifier*, is a substitution σ such that $s_i\sigma =_E t_i\sigma$ for all $1 \leq i \leq m$. A substitution σ is *more general modulo E* than θ on a set of variables V , denoted as $\sigma \leq_E^V \theta$, if there is a substitution τ such that $x\sigma\tau =_E x\theta$ for all $x \in V$. Two substitutions θ_1 and θ_2 are *equivalent modulo E* on a set of variables V , denoted as $\theta_1 \equiv_E^V \theta_2$, if and only if $x\theta_1 =_E x\theta_2$ for all $x \in V$. A *Complete Set of E-Unifiers* of $\mathcal{S}$ is a set of substitutions denoted by $\text{CSU}_E(\mathcal{S})$ such that each $\sigma \in \text{CSU}_E(\mathcal{S})$ is an *E-unifier* of $\mathcal{S}$, and for each *E-unifier* θ of $\mathcal{S}$, there exists $\sigma \in \text{CSU}_E(\mathcal{S})$ such that $\sigma \leq_E^{\text{Var}(\mathcal{S})} \theta$.

A set of equations $\mathcal{S}$ is said to be in *standard form* over a signature Σ if and only if every equation in $\mathcal{S}$ is of the form $x =^? t$, where x is a variable and t is one of the following: (a) a variable different from x , (b) a constant, or (c) a term of depth 1 that contains no constants. It is not generally difficult to decompose equations of a given problem into simpler standard forms.

Definition 1. A set of equations is said to be in *dag-solved form* (or *d-solved form*) if and only if they can be arranged as a list $x_1 =^? t_1, \dots, x_n =^? t_n$ where (a) each left-hand side x_i is a distinct variable, and (b) $\forall 1 \leq i \leq j \leq n: x_i$ does not occur in t_j ([16]). Each x_i in this case is called a *solved variable*. A set of equations $\mathcal{S}$ are said to be in Σ -*solved form* if and only if it is in *standard form* and $\mathcal{S}|_{\Sigma}$ is in *dag-solved form*.

Definition 2. A theory E is *subterm collapse-free* if and only if for all terms t it is not the case that $t =_E u$ where u is a strict subterm of t .

Definition 3. We say there exists a cycle in a unification problem if it contains a set of equations $\bigcup_{i=1}^{n-1} \{x_i =^? t_i[x_{i+1}]\} \cup \{x_n =^? t_n[x_1]\}$ where $t_1, \dots, t_n$ are non-variable terms and $x_1, \dots, x_n$ are distinct variables.

The following is a well know property of subterm collapse-free theories (see [10]).

Proposition 1. If E is subterm collapse-free, then any *E-unification problem* containing a cycle has no solution.

Definition 4. For a convergent rewrite system R we define a constructor of R to be a function symbol f which does not appear at the root on the left-hand side of any rewrite rule of R . We define an inner constructor to be a constructor f the satisfies the following additional restrictions:

- (i) f does not appear on the left-hand side on any rule in R .
- (ii) f does not appear as the root symbol on the right-hand side of any rule in R .
- (iii) there are no function symbols below f on the right-hand side of any rule in R .

We consider two equational theories E_1 and E_2 built over the signatures Σ_1 and Σ_2 . Let $\Sigma_{(1,2)} = \Sigma_1 \cap \Sigma_2$. Here we will use some notations and methods already developed in the combination literature (See [5]).

The elements of Σ_i are called *i*-symbols. A term t is called an *i*-term if its root symbol is an *i*-symbol or t is a variable. Notice, that the shared symbols are both 1 and 2 symbols. We say that a Σ_i -term and a Σ_i -equation are *i*-pure.

We also use the notion of an *alien subterm*. An alien subterm of a 1-term (2-term), t , is a non-variable subterm s that is a 2-term (1-term). A decomposition algorithm can be defined that uses *variable abstraction* ([5]) to replace any alien subterm u in a term t by a fresh variable x and adds the equation $x =^? u$. By first converting a unification problem into standard form we have also applied variable abstraction on the entire unification problem. Decomposition problems usually employ a splitting procedure to split non-pure equations, $s =^? t$ into two pure equations $x =^? t$ and $x =^? s$ where x is a new variable. An equation between two variables is always 1 and 2-pure.

We call a term (e.g., a variable) *fresh* if it is created by applying an inference rule (or a unification algorithm) and did not previously exist.

3 Combination Procedure

We want to investigate conditions to build an $E_1 \cup E_2$ -unification algorithm by using two algorithms A_1 and A_2 solving two different kinds of $E_1 \cup E_2$ -unification problems. Consider an $E_1 \cup E_2$ -unification problem $\mathcal{P}$ in standard form. The approach taken is as follows:

1. Run A_1 on the $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations of $\mathcal{P}$. Let $\mathcal{P}'$ denote the resulting set of equations.
2. Run A_2 on the Σ_2 -equations of $\mathcal{P}'$.
3. Collect the resulting problems that are in dag-solved forms.

As in disjoint combination [5], we perform a variable identification step before applying A_1 and A_2 in order to guess a priori all possible identifications of variables occurring in $\mathcal{P}$. Hence, we apply A_1 and A_2 not only on $\mathcal{P}$ but on all possible unification problems obtained from $\mathcal{P}$ by identifying some variables. In Section 3.2, we show why the classical variable identification introduced for disjoint combination is sufficient in our non-disjoint setting.

For this approach to work we need to place some restrictions on the theories and algorithms. These restrictions are introduced next, in Section 3.1. The combination algorithm, denoted by $\mathfrak{C}$, is formally presented in Figure 2. The correctness of $\mathfrak{C}$ is proved in Section 3.3. After introducing the combination algorithm and proving its correctness based on the following restrictions we show, in Section 4, how these restrictions can be satisfied by a class of equational theories.

3.1 Restrictions

We present a set of “restrictions” or assumptions on the theories and corresponding algorithms. Let Σ_1 and Σ_2 be finite signatures and let $\mathcal{X}$ be a countably infinite set of variables. Consider two *subterm collapse-free* equational theories

E_1 over the set of terms $T(\Sigma_1, \mathcal{X})$ and likewise E_2 over $T(\Sigma_2, \mathcal{X})$ such that $\Sigma_1 \cap \Sigma_2 = \Sigma_{(1,2)} \neq \emptyset$.

Restriction 1. (*Algorithm A_1*) Let $\mathcal{P}$ be a set of $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations. Algorithm A_1 computes a set of problems $\{\mathcal{Q}_k\}_{k \in K}$ such that

$\bigcup_{k \in K} CSU_{E_1 \cup E_2}(\mathcal{Q}_k)$ is a $CSU_{E_1 \cup E_2}(\mathcal{P})$ and for each $k \in K$:

- (i) $\mathcal{Q}_k$ consists of $(\Sigma_1 \setminus \Sigma_{(1,2)})$ -equations and $\Sigma_{(1,2)}$ -equations.
- (ii) $\mathcal{Q}_k$ is in standard form and $(\Sigma_1 \setminus \Sigma_{(1,2)})$ -solved form.
- (iii) No fresh variable occurring in a nonvariable $\Sigma_{(1,2)}$ -term in $\mathcal{Q}_k$ can appear as solved in $\mathcal{Q}_k$.

Note that A_1 is a special type of algorithm that returns a “partial” solution to an $E_1 \cup E_2$ -unification problem. A_1 is needed to solve some portion of the problem, namely the $\Sigma_1 \setminus \Sigma_{(1,2)}$ -pure, but a standard E_1 -unification algorithm is not sufficient, even for $\Sigma_1 \setminus \Sigma_{(1,2)}$ -pure $E_1 \cup E_2$ problems. A standard E_1 -unification algorithm may return an error even though a solution exists.

Example 1. Let $E_1 := \{h(a, x, y) = g(x * y), h(b, x, y) = g(y * x)\}$ and let E_2 be the commutative theory for $*$. Then, $h(a, a, z) =^? h(b, a, b)$ is not solvable in E_1 but is in $E_1 \cup E_2$.

Restriction 1 ensures that completeness is not lost by ensuring that a CSU for all the partial solutions is a CSU for the original problem. To explain Restriction 1(iii), let us note that A_1 may generate fresh $\Sigma_{(1,2)}$ -equations, e.g. $z =^? f(x, y)$ where $f \in \Sigma_{(1,2)}$, together with some $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations, e.g. $x =^? s, y =^? t$. If A_2 later generates $x =^? y$, then this may lead to the reapplication of A_1 , to solve $x =^? s, x =^? t$. If x and y are from the initial set of variables, this problem can be discarded without loss of generality, since the variable identification performed initially generates another unification problem where x and y are already identified. The problem remains if x or y are fresh variables. In order to avoid it, we introduce Restriction 1(iii), where only the occurrences of fresh variables are restricted (see also Example 4).

Restriction 2. (*Algorithm A_2*)

Algorithm A_2 computes a finite complete set of 2-pure unifiers of 2-pure $E_1 \cup E_2$ -unification problems.

Example 2. Consider $E_1 = \{r(f(x)) = r(g(x))\}$ and $E_2 = \{r(g(x)) = r(h(x))\}$ and the unification problem $r(f(x)) =^? r(h(x))$. The problem in standard form consists of the equations $\{x_1 =^? r(x_2), x_1 =^? r(x_3), x_2 =^? f(x), x_3 =^? h(x)\}$, which is already in $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form. Thus the problem to be solved by A_2 is the 2-pure problem $\{x_1 =^? r(x_2), x_1 =^? r(x_3), x_3 =^? h(x)\}$, which admits the $E_1 \cup E_2$ -unifiers $\sigma_1 = \{x_1 \mapsto r(f(x)), x_2 \mapsto f(x), x_3 \mapsto h(x)\}$ and $\sigma_2 = \{x_1 \mapsto r(h(x)), x_2 \mapsto h(x), x_3 \mapsto h(x)\}$. Notice that the solution σ_1 fails Restriction 2 because it is not 2-pure. Since we require that A_2 be able to solve

2-pure $E_1 \cup E_2$ -unification problems with *only* 2-pure solutions, this theory is beyond the scope of our algorithm $\mathfrak{C}$. In Section 4 we give a family of theories that satisfy this restriction.

We restrict A_2 to compute only 2-pure substitutions. This is used to avoid possible reapplications of A_1 after A_2 .

Restriction 3. (*Errors*)

- (i) A $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted term cannot be $E_1 \cup E_2$ -equal to a Σ_2 -rooted term.
- (ii) $E_1 \cup E_2$ is subterm collapse-free. Therefore, an $E_1 \cup E_2$ -unification problem including a cycle has no solution.

Example 3. Consider $E_1 = \{f(x) = 0\}$ and $E_2 = \{g(x) = 0\}$. Restriction 3 is not satisfied since $f(x) =_{E_1 \cup E_2} 0$ where $f \in \Sigma_1 \setminus \Sigma_{(1,2)}$ and $0 \in \Sigma_2$. Hence, checking the unifiability of $f(x) =_{E_1 \cup E_2}^? g(x)$ is beyond the scope of our combination algorithm $\mathfrak{C}$.

The failure rules associated with Restriction 3 are given in Figure 1.

An Example Theory. We present a theory, called $\mathcal{E}_{AC}$, for which we applied our approach [13] and show that $\mathcal{E}_{AC}$ satisfies the restrictions 1, 2 and 3. The axioms of $\mathcal{E}_{AC}$ are as follows:

$$\begin{aligned} \exp(\exp(x, y), z) &= \exp(x, y \circledast z) \quad (1) & (x \circledast y) \circledast z &= x \circledast (y \circledast z) \quad (3) \\ \exp(x * y, z) &= \exp(x, z) * \exp(y, z) \quad (2) & x \circledast y &= y \circledast x \quad (4) \end{aligned}$$

Here, $\mathcal{E}_{AC} = E_1 \cup E_2$ where E_1 consists of axioms (1) and (2) and E_2 axioms (3) and (4). Also, $\Sigma_1 = \{\exp, *, \circledast\}$. $\Sigma_2 = \{\circledast\}$. In other words, $E_1 \cup E_2$ involves AC operator $\circledast$. The theory $\mathcal{E}_{AC}$ has the following AC -convergent system [13]:

$$\begin{aligned} \exp(\exp(x, y), z) &\rightarrow \exp(x, y \circledast z) \\ \exp(x * y, z) &\rightarrow \exp(x, z) * \exp(y, z) \end{aligned}$$

Note that $\Sigma_1 \setminus \Sigma_{(1,2)} = \{\exp, *\}$. In [13], we develop a procedure A_1 (called $\mathfrak{R}_1$ in [13]) that returns a $\{\exp, *\}$ -solved form. Furthermore, it is shown in [13] that A_1 terminates and that it is sound and complete. This implies that A_1 satisfies Restriction 1(i). The second part of the restriction, is satisfied by the checking of A_1 which does not have any rule as defined. $\mathcal{E}_{AC}$ satisfies also Restriction 2 and 3, where A_2 is the standard AC -unification algorithm. This is a consequence of results showing the fact that $\circledast$ is an inner constructor with respect to the AC -convergent rewrite system associated with axioms (1) and (2) above. Therefore, we get the following result.

Proposition 2. $\mathcal{E}_{AC}$ satisfies Restriction 1, Restriction 2 and Restriction 3.

We will use this theory to illustrate more results in the paper.

Conflict: If $s(\epsilon) \in \Sigma_1 \setminus \Sigma_{(1,2)}$ and $t(\epsilon) \in \Sigma_2$.

$$\{v =^? s, v =^? t\} \cup P \longrightarrow \text{Fail}$$

Cycle: If P contains a cycle

$$P \longrightarrow \text{Fail}$$

Fig. 1. Failure rules

Some of the steps below are non-deterministic, hence lead to computation paths. Throughout each path, the failure rules of Figure 1 are applied eagerly after each step. Let $\mathcal{V} = \text{Var}(\mathcal{P})$.

Step 0: Variable Identification

Guess a partition on $\mathcal{V}$ and set variables in each subset equal to each other. This requires adding fresh equations of type $u =^? w$, where $u, w \in \mathcal{V}$ and u and w belong to the same subset in the partition. Let us denote the enumeration of the partitions of $\mathcal{V}$ as $\pi_1, \pi_2, \dots, \pi_m$. Once a partition is selected, the unification problem $\mathcal{P}$ is modified by adding the fresh equations between variables to the unification problem. We denote the modified problems by $\mathcal{P}^{\pi_i}$, $1 \leq i \leq m$.

Step 1: Run A_1

We apply A_1 to the $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations of $\mathcal{P}^{\pi_i}$. If A_1 fails for all π_i , report failure and stop. Otherwise, we now have a modified, by A_1 , set of unification problems from $\mathcal{P}^{\pi_i}$, $\mathcal{P}_1^{\pi_i}, \dots, \mathcal{P}_n^{\pi_i}$ for $n \geq 1$, such that all the $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations in each $\mathcal{P}_j^{\pi_i}$, are in $(\Sigma_1 \setminus \Sigma_{(1,2)})$ -solved form.

Step 2: Run A_2

Run Algorithm A_2 on the 2-pure equations of each $\mathcal{P}_j^{\pi_i}$. If A_2 equates variables in $\mathcal{V}$ not equated by $\pi_{i,j}$ discard that particular $\mathcal{P}_j^{\pi_i}$. If none of the remaining $\mathcal{P}_1^{\pi_i}, \dots, \mathcal{P}_n^{\pi_i}$ exit with success, return failure. Otherwise, return success.

Fig. 2. $\mathcal{E}$: Unification Algorithm for the combined theory $E_1 \cup E_2$

3.2 Variable Identification

We are concerned with the fact that some variables from $\mathcal{P}$ could be equated by A_2 and then cause reapplication of the inference rules of A_1 . The way to avoid ping-ponging between the two unification algorithms is to guess, right at the beginning, a successful partition of the set of variables of $\mathcal{P}$ in a way that variables contained in each subset of each partition are set equal to each other. That is, we perform *variable identification* on $\text{Var}(\mathcal{P})$. For variable identification, we initially consider the set of all variables $\text{Var}(\mathcal{P})$ in the problem. However, some improvements are possible depending on syntactic properties of the equational theory considered, as is done in [13].

Let us assume that A_1 creates fresh variables that could cause the need for reapplication of the rules of A_1 during the application of algorithm A_2 . We call

such variables “Ping-Pong Variables” as it implies back-and-forth mechanism between the algorithms.

Example 4. Consider again the theory $\mathcal{E}_{AC}$ and suppose that z is a fresh variable created by A_1 such that $z =^? \exp(x_1, y_1)$, and $x_2 =^? y_2 \otimes z$. Now if this could happen the AC -unification algorithm (A_2) could equate z , by using the equation $x_2 =^? y_2 \otimes z$, to another variable, say x_3 , such that $x_3 =^? x_4 * x_5$, resulting in the equation $\exp(x_1, y_1) =^? x_4 * x_5$. Since this fresh equation would cause failure in the AC algorithm we would need to “ping-pong” back to A_1 to handle the fresh equation. The process could continue to repeat. Therefore, we ensure these variables don’t occur in the following results.

Definition 5. (*Ping-Pong Variable*) *A variable x is a ping-pong variable if*

1. *x is a fresh variable created by A_1 .*
2. *A_1 did not create an equation $x =^? y$, where y is a variable from the initial standard form problem.*
3. *x occurs in $x =^? t$ where t is a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted term.*
4. *x occurs in a $\Sigma_{(1,2)}$ -rooted term generated by A_1 .*

Lemma 1. *Let $\mathcal{P}$ be a modified problem produced by $\mathfrak{C}$ in step (2) (i.e., an output of A_1) and assume that z is fresh variable created by A_1 . If A_2 equates z to another variable y and that equality converts a subset of $\mathcal{P}$ into non-($\Sigma_1 \setminus \Sigma_{(1,2)}$)-solved form, then either z is a ping-pong variable or $\mathcal{P}$ is not unifiable.*

Proof. Let z be the fresh variable created by A_1 and assume that A_2 creates the equality $z = y$, where due to the equality a subset of equations, $\rho \in \mathcal{P}$, is not in dag-solved form. Now, if $\rho = \{x_1 =^? t_1, \dots, x_m =^? t_m\}$ is not in solved form then one of the following situations must be present; (1) the left hand sides are not all distinct, (2) there exists a cycle in the equations. To create situation (1) the equality $z = y$ needs to create duplicate left hand sides. This implies that $x_i = z =^? t_i$ and $x_j = y =^? t_j$. If z is equated to another variable by A_2 it must occur in the 2-pure equations. Since z cannot be both equal to a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -term and a $\Sigma_{(1,2)}$ -term, an error by Restriction 3, z must be contained in a $\Sigma_{(1,2)}$ -rooted term. Thus, z is a ping-pong variable. Situation (2) implies there is a cycle between $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations, a failure condition. $\square$

Based on Restriction 1 of A_1 we get the following result.

Lemma 2. *Let $\mathcal{Q}$ be a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form, computed by A_1 . There are no ping-pong variables in $\mathcal{Q}$.*

Example 5. In [13] it is shown that $\mathfrak{R}_1$ does not produce equations of the form $z =^? \exp(x, y)$ or $z =^? x * y$, where z is a fresh variable occurring in a $\otimes$ -rooted equation.

Algorithm A_2 could still create different subsets of equated variables from $\mathcal{V}$. Because $\mathcal{V}$ is finite we easily get the following result.

Lemma 3. *For each possible set of equalities $\mathcal{I}$ of the variables in $\mathcal{V}$ by the algorithm A_2 there exists a partition π_i on $\mathcal{V}$ that corresponds to $\mathcal{I}$ when the variables in each subset of π_i are equated.*

Therefore, we can “guess” a (or generate all the) partition(s) of $\mathcal{V}$. For each partition we can modify the original unification problem by pre-equating the variables in the subsets of the partition, thus guessing the action of A_2 . Because there exists a partition for each action of A_2 on $\mathcal{V}$, including not equating any of the variables, we only have to apply A_1 once. This is because if for any particular partition, say π_i , algorithm A_2 equates two variables in $\mathcal{V}$ not equated by π_i we can discard that particular modified unification problem because we know there exists a partition that does correctly correspond to the action of A_2 .

3.3 Termination, Soundness and Completeness

Given two theories E_1 and E_2 , with corresponding algorithms A_1 and A_2 , satisfying Restrictions 1 through 3, we show $\mathfrak{C}$ is terminating, sound and complete.

Theorem 1. $\mathfrak{C}$ terminates.

Proof. We assume that the algorithms A_1 and A_2 terminate. There is a finite number of partitions of $\mathcal{V}$, thus only a finite number of $\mathcal{P}^{\pi_i}$. Since A_1 terminates and is finitary, there are only a finite number of $\mathcal{P}_j^{\pi_i}$ problems produced. $\square$

Lemma 4. *Let $\mathcal{P}$ be an $E_1 \cup E_2$ -unification problem and σ an $E_1 \cup E_2$ -unifier of $\mathcal{P}$. Then, there exists a partition π_i of $\mathcal{V}$, and an index j , such that σ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}_j^{\pi_i}$.*

Proof. σ naturally induces a partition of the set of variables from $\mathcal{P}$, i.e., if $x, y \in \mathcal{V}$ and $x\sigma =_{E_1 \cup E_2} y\sigma$ then x and y are in the same subset of the partition. Denote this partition as π_i . Then we can modify $\mathcal{P}$ to $\mathcal{P}^{\pi_i}$ and σ remains a unifier of $\mathcal{P}^{\pi_i}$. $\square$

We want to prove that if a problem is unifiable there exists at least one partial solved form produced from A_1 such that if $\mathfrak{C}$ returns success after running A_2 that solved form is still intact.

Lemma 5. *Let θ be an $E_1 \cup E_2$ -unifier of an $E_1 \cup E_2$ -unification problem $\mathcal{P}$. There exist a partition π_i , a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form $\mathcal{P}'$ obtained by applying A_1 on $\mathcal{P}^{\pi_i}$, a Σ_2 -solved form $\mathcal{P}''$ obtained by applying A_2 on $\mathcal{P}'_{|\Sigma_2}$ such that $\mathcal{P}'_{|\Sigma_1 \setminus \Sigma_{(1,2)}} \cup \mathcal{P}''$ is in a dag-solved form for which the related substitution σ satisfies $\sigma \leq_{E_1 \cup E_2}^{Var(\mathcal{P})} \theta$.*

Proof. By assumption, A_1 and A_2 transform an $E_1 \cup E_2$ -unification problem into an equivalent set of $E_1 \cup E_2$ -unification problems. Moreover, the failure rules are also sound and complete. It remains to show that the final problems (different from *Fail*) are indeed dag-solved forms. The algorithm A_2 will apply to $\mathcal{P}'_{|\Sigma_2}$

and may possibly add to $\mathcal{P}'_{|\Sigma_1 \setminus \Sigma_{(1,2)}}$ fresh equations between variables. But these additional equations cannot convert $\mathcal{P}'_{|\Sigma_1 \setminus \Sigma_{(1,2)}}$ into a non-solved form according to Lemma 1 and Lemma 2. Therefore, all final problems $\mathcal{P}'''$ different from *Fail* are such that $\mathcal{P}'''_{|\Sigma_1 \setminus \Sigma_{(1,2)}}$ is in solved form and $\mathcal{P}'''_{|\Sigma_2}$ is in solved form. Such a final problem $\mathcal{P}'''$ is necessarily in dag-solved form, for otherwise a failure rule would apply. $\square$

Theorem 2. *Let A_1 and A_2 be algorithms satisfying Restrictions 1 through 3. The algorithm $\mathfrak{C}$ is sound and complete, which means that $E_1 \cup E_2$ -unification is finitary.*

Proof. The soundness directly follows from the fact that each time an $E_1 \cup E_2$ -unification problem $\mathcal{P}$ is transformed into $\mathcal{P}'$ by $\mathfrak{C}$, then any $E_1 \cup E_2$ -unifier of $\mathcal{P}'$ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}$. The completeness is proved by Lemma 5. $\square$

4 A Class of Hierarchically Combinable Theories

In this section we consider a class of hierarchically defined theories. We investigate the problem of satisfying Restrictions 1 through Restriction 3. The class of theories is defined by the following set of properties. The two key properties being the hierarchical organization and the shared symbols being inner constructors. We assume that E_1 and E_2 are *subterm collapse-free*. The class is then defined by the following properties.

1. Properties of E_1 :
 R_1 is a left-linear, convergent term rewrite system corresponding to E_1 .
2. Properties of E_2 :
 E_2 is a linear, finite equational theory.
3. Properties of the shared symbols:
 If $f \in \Sigma_{(1,2)}$, then f is an *inner constructor* of R_1 .
 If f and g are inner constructors of R_1 , then f -rooted terms cannot be equated to g -rooted terms in E_2 .

The $\mathcal{E}_{AC}$ theory, is an example of a theory that is contained in the above defined class. The convergent rewrite theory associated with the first two axioms is left-linear and $\otimes$ is an inner constructor. The lower theory is the *AC*-theory for $\otimes$.

By assuming these properties, we can show that an E_2 -unification algorithm is sufficient to satisfy Restriction 2. Moreover, these properties are sufficient to satisfy Restriction 3. With respect to Restriction 1, it is possible to build a complete “syntactic” method to generate the partial solved forms of A_1 .

4.1 Satisfying Restriction 1

The first and critical step to satisfying Restriction 1 is the construction of an inference system $\mathcal{G}$ such that the standard forms of leaves in the search tree

generated by $\mathcal{G}$ are the $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved forms corresponding to a complete set of $E_1 \cup E_2$ -unifiers of the input problem, thus satisfying the majority of Restriction 1. Another way to view the inference system $\mathcal{G}$ is as a method for reducing $E_1 \cup E_2$ -unification problems to a set of E_2 -unification problems.

We still have to identify examples of theories for which $\mathcal{G}$ can be turned into a terminating algorithm. Moreover, the computed $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved forms must satisfy Restriction 1 (iii) to avoid ping-pong variables when running $\mathfrak{C}$. Before presenting the procedure we need some results on the relation between R_1 and E_2 . Immediate from the properties we have the following.

Lemma 6. *Let s_1, s_2 and t be terms such that $s_1 \longleftrightarrow_{E_2} s_2 \longrightarrow_{R_1} t$. Then there exists a term t' such that $s_1 \longrightarrow_{R_1} t' \longleftrightarrow_{E_2}^* t$.*

Lemma 7. *$\longrightarrow_{R_1} \circ \longleftrightarrow_{E_2}^*$ is terminating if and only if $\longrightarrow_{R_1}$ is terminating.*

Proof. The “only if” part is trivial. The “if” part is a consequence of the fact that $(\longrightarrow_{R_1} \circ \longleftrightarrow_{E_2}^*)^* \subseteq \longrightarrow_{R_1}^* \circ \longleftrightarrow_{E_2}^*$ according to Lemma 6. $\square$

Lemma 8. *R_1 is convergent modulo E_2 . That is, for any s, t*

$$s =_{R_1 \cup E_2} t \text{ if and only if } s \downarrow_{R_1} =_{E_2} t \downarrow_{R_1}$$

Proof. The result is a consequence of [15], Theorem 3.5. There a set of restrictions is given on a rewrite theory and equational theory. The key restrictions of [15] are:

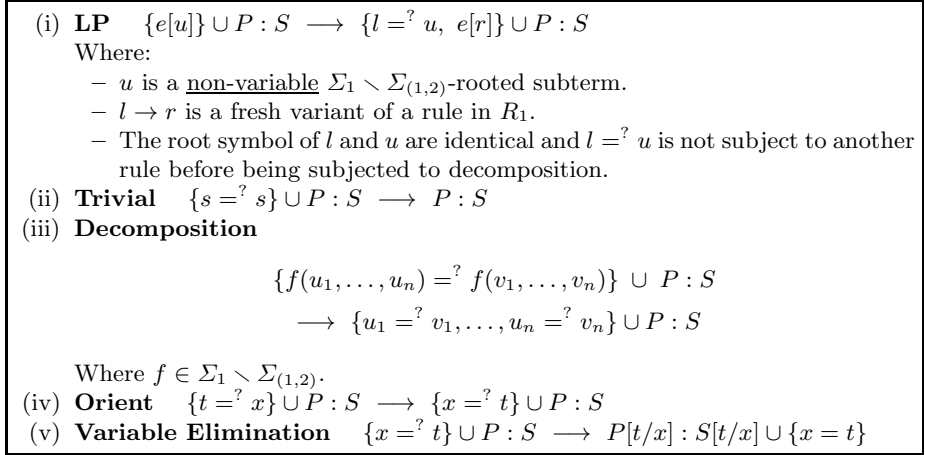
(1) R_1 be left-linear. (2) Critical pairs of R_1 are joinable modulo E_2 . (3) Critical pairs of R_1 and E_2 are joinable modulo E_2 . (4) $\longrightarrow_{R_1} \circ \longleftrightarrow_{E_2}^*$ is Noetherian.

(1) is satisfied by assumption. (2) is a consequence of R_1 being a convergent system. (3) is a consequence of the fact that the only shared symbols are inner-constructors of R_1 . Since inner-constructors don’t appear on the left-hand sides of the rules in R_1 , there is no overlap between the left hand sides of R_1 with either side of the identities in E_2 . Finally, (4) follows from Lemma 7. $\square$

We modify the general syntactic E -unification procedure $\mathcal{G}$, presented in [6], which consists of a set of non-deterministic inference rules. This procedure is based on the methods developed in [14, 20]. The modifications are to the rules (i) and (ii). The resulting system is presented in Figure 3. Note that in constructing this procedure one could also start with another general E -unification method such as [14, 17, 12]. The algorithm could also be constructed from scratch as in [13].

Let $e[u]$ denote an equation with subterm u , let P and S denote sets of equations. Initially P is a set of $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations and S is empty. A *computation path* in the modified $\mathcal{G}$ (from now just called $\mathcal{G}$) is a sequence of rule starting from an initial system $P : \emptyset$ and ending in a system $P : S$ such that no additional rules from $\mathcal{G}$ can be applied and S is in solved form.

We start by showing that the R_1 portion of any $E_1 \cup E_2$ proof corresponds to a computation path in $\mathcal{G}$. Note, in our procedure $\mathcal{G}$, we introduce fresh variables by considering the fresh variants of rules in R_1 . These variables can be seen

Fig. 3. Unification procedure $\mathcal{G}$

as existentially quantified variables [16]. Hence, an equation $x =^? t$ can be removed from a solved form when x is existentially quantified, and so existentially quantified variables do not appear in the domain of unifiers.

Lemma 9. *Let $s =^? t$ be a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equation. For each $E_1 \cup E_2$ -unifier θ of $s =^? t$ such that θ is R_1 -normalized and $s\theta, t\theta$ are ground terms, there exists a sequence of LP applications in $\mathcal{G}$ that correspond to the R_1 steps in $s\theta \rightarrow_{R_1}^* (s\theta) \downarrow_{R_1=E_2} (t\theta) \downarrow_{R_1}^* \leftarrow_{R_1}^* t\theta$ such that θ is an $E_1 \cup E_2$ -unifier of any unification problem in this sequence.*

Proof. For an R_1 -normalized θ , a ground term $s\theta$ and any R_1 -rewrite sequence $s\theta \rightarrow_{R_1}^* s'$ reducing $s\theta$ to the R_1 -normal form s' , there exists a rewrite proof $s\theta \rightarrow^* s'$ such that no rewrite step can take place at a term introduced by any substitution. This is known as a “basic” rewrite proof (see [6], Section 4). According to the assumption and Lemma 8, we have $s\theta \rightarrow_{R_1}^* s'$, $t\theta \rightarrow_{R_1}^* t'$, and $s' =_{E_2} t'$ where s' and t' are respectively the R_1 -normal forms of $s\theta$ and $t\theta$. We examine $s\theta \rightarrow_{R_1}^* s'$, as $t\theta \rightarrow_{R_1}^* t'$ follows analogously. If $s\theta \rightarrow_{R_1}^* s'$ is a basic rewrite proof then the result follows from a simple induction argument on the length of the derivation. If there are no rewrite steps, $s\theta = s'$, then no LP step is required as it is already in normal form and E_2 -unifiable. Otherwise, we get a set of rewrite steps of the form

$$s\theta = s_1 \xrightarrow[l_1 \rightarrow r_1]{p_1, \theta} s_1[r_1\theta]_{p_1} = s_2 \rightarrow \dots \rightarrow s_n \xrightarrow[l_n \rightarrow r_n]{p_n, \theta} s_n[r_n\theta]_{p_n} = s'$$

Since at each reduction the redex is not contained in θ , each reduction corresponds to an application of LP at the same position. That is, for any $(k)^{th}$ step of the rewrite proof $s_k \xrightarrow[l_k \rightarrow r_k]{p_k, \theta} s_k[r_k\theta]_{p_k} = s_{k+1}$ there exists a LP application such that $\{s''[u]_{p_k} =^? t\} \cup P; S \implies_{LP} \{l_k =^? u, s''[r_k]_{p_k} =^? t\} \cup P; S$ where

$s''\theta = s_k$. Furthermore since $u\theta = s_k|_{p_k} = l_k\theta$ and $(s''[u]_{p_k})\theta \rightarrow_{R_1} (s''[r_k]_{p_k})\theta$, θ is a $E_1 \cup E_2$ -unifier of $\{l_k =^? u, s''[r_k]_{p_k} =^? t\}$. $\square$

Let us denote a system $P; S$ resulting from an exhaustive application of the rules in $\mathcal{G}$ as a $\mathcal{G}$ -normal form.

Lemma 10. *Let $\mathcal{P}$ be a set of $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations. If θ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}$, then there exists a sequence, $\mathcal{P}; \emptyset \Longrightarrow^* \mathcal{P}'; S$, in $\mathcal{G}$ such that $\mathcal{P}'; S$ is in normal form w.r.t $\mathcal{G}$ and θ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}' \cup S$.*

Proof. We can modify the same proof strategy used in [6] (Lemma 4.9) but modified due to Lemma 9 and the fact that our procedure stops before considering E_2 . Let θ be a R_1 -normal solution to $\mathcal{P}$. For all $u =^? v \in P$, If $u\theta =_{R_1 \cup E_2} v\theta$ then by Lemma 8 $u\theta \downarrow_{R_1=E_2} v\theta \downarrow_{R_1}$. By Lemma 9, there exists a $\mathcal{P}'; S$ having the same solution, obtained by a sequence of LP applications, $\mathcal{P}; S \Longrightarrow_{LP} \mathcal{P}'; S$, such that $\mathcal{P}'; S$ is a R_1 -normal form. In addition, since the LP rule doesn't add equations to S , $S = \emptyset$. Thus, S is in solved form.

Now if $u\theta = v\theta$ is in R_1 -normal form. Then, only rules (ii)-(v) can be applied. In this case the problem reduces to syntactic unification. In [6] a complexity measure is defined and it is shown that the syntactic unification rules will transform the system into a system lower in the complexity measure and having the same solution. In addition, it is also shown that any equation introduced into S maintains a solved form. $\square$

We can characterize the normal forms produced in $\mathcal{G}$ by the following result.

Lemma 11. *The standard form of any $\mathcal{G}$ -normal form is a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form.*

Proof. Consider the equations $s =^? t$ such that $s(\epsilon), t(\epsilon) \in \mathcal{X} \cup (\Sigma_1 \setminus \Sigma_{(1,2)})$. When these equations occur in a $\mathcal{G}$ -normal form, $P; S$, they occur necessarily in S . Otherwise, one of the rules of $\mathcal{G}$ could be applied contradicting the assumption that $P; S$ is a $\mathcal{G}$ -normal form. A $\mathcal{G}$ -normal form, $P; S$, can be converted to a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form by converting the system into standard form. The equations in S will clearly be in $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form and the equations in P will be converted to $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form due to variable abstraction. In addition the conversion maintains unifiability. $\square$

Lemma 12. *Let $\mathcal{P}$ be a set of $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations. Then, the normal forms computed by $\mathcal{G}$ are equivalent to $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved forms $\{\mathcal{Q}_k\}_{k \in K}$ such that*

$$\bigcup_{k \in K} CSU_{E_1 \cup E_2}(\mathcal{Q}_k) \text{ is a } CSU_{E_1 \cup E_2}(\mathcal{P})$$

In addition, $\mathcal{Q}_k$ consists of $(\Sigma_1 \setminus \Sigma_{(1,2)})$ -equations and $\Sigma_{(1,2)}$ -equations.

Proof. If $\mathcal{P}$ is $E_1 \cup E_2$ -unifiable the result follows from Lemma 10 and Lemma 11. If the problem is not $E_1 \cup E_2$ -unifiable then there are two possibilities. First, no $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form exist, in which case the result follows. Second, there exist one or more $\mathcal{G}$ -normal forms. But since all the conversions maintain unifiability, the resulting solved forms will not be E_2 -unifiable. Lastly, since R_1 can only introduce Σ_1 symbols the last result follows. $\square$

4.2 Satisfying Restriction 2

We prove that an E_2 -unification algorithm is sufficient to solve 2-pure problems modulo $E_1 \cup E_2$, the restriction on A_2 . To state this result, let us consider some technical concepts introduced initially for disjoint combination [5]. We define the following bijection between the set of $(E_1 \cup E_2)$ -equivalence classes of terms and a set of variables, so-called abstraction variables. This is helpful to obtain the proof of next lemma which states that variable abstractions are not affected by rewrite steps at any position in a term.

Let $\mathcal{X}$ and $\mathcal{Y}$ be disjoint sets of variables that are countably infinite. Let $T(\Sigma_1 \cup \Sigma_2, \mathcal{X})$ be the set of $\Sigma_1 \cup \Sigma_2$ -terms over $\mathcal{X}$. We define a bijection $\phi : T(\Sigma_1 \cup \Sigma_2, \mathcal{X})_{/=E_1 \cup E_2} \rightarrow \mathcal{Y}$ and denote t^ϕ the ϕ -abstraction of t defined as follows:

If $t \in \mathcal{X}$, then $t^\phi = t$. If $t = f(t_1, \dots, t_n)$ and $f \in \Sigma_2$, then $t^\phi = f(t_1^\phi, \dots, t_n^\phi)$. Otherwise, $t^\phi = \phi([t]_{E_1 \cup E_2})$, where $[t]_{E_1 \cup E_2}$ denotes the equivalence class of t modulo $E_1 \cup E_2$.

Note that third case applies to $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted terms and ϕ -abstractions of 2-terms are 2-pure terms.

Lemma 13. *For any terms s and t , if $s \longleftrightarrow_{E_1 \cup E_2} t$, then $s^\phi =_{E_2} t^\phi$.*

Proof. Note that if s is $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted (resp Σ_2 -rooted), then $s \longleftrightarrow_{E_1 \cup E_2} t$ implies that t is $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted (resp Σ_2 -rooted), due to the form of rules in R_1 and equalities in E_2 , plus the subterm collapse-freeness of E_1 and E_2 .

Let $s \longleftrightarrow_{E_1 \cup E_2} t$ and assume that the $\longleftrightarrow_{E_1 \cup E_2}$ -step is applied at a position p in s . Let us first consider that s is Σ_2 -rooted. There are two cases to consider: 1. p is not in an alien subterm. We can only apply $\longleftrightarrow_{E_2}$ at p according to our assumptions on E_1 and E_2 . Then there exists an equation $l = r$ in E_2 and a substitution σ such that $s[l\sigma]_p \longleftrightarrow s[r\sigma]_p = t$. Consider the ϕ -abstraction s^ϕ of s . Note that p is still a non-variable position in s^ϕ and we can find substitution σ' such that $s^\phi[l\sigma']_p \longleftrightarrow_{E_2} s^\phi[r\sigma']_p = t^\phi$. Thus we get $s^\phi \longleftrightarrow_{E_2} t^\phi$.

2. p is in an alien subterm occurring at position q . A $\longleftrightarrow_{E_1 \cup E_2}$ -step is applied at p , which implies that $s|_p =_{E_1 \cup E_2} t|_p$, and so $s|_q =_{E_1 \cup E_2} t|_q$. Hence the alien subterms $s|_q$ and $t|_q$ are abstracted to the same variable by ϕ , that is $(s|_q)^\phi = (t|_q)^\phi$. Following the definition of ϕ -abstraction, we get $s^\phi = t^\phi$.

If s is $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted, then t is $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted. Hence, $s^\phi = t^\phi$, and so $s^\phi =_{E_2} t^\phi$.

Finally, if s is a variable, then there is no t such that $s \longleftrightarrow_{E_1 \cup E_2} t$. □

Lemma 14. *E_2 -unification is sound and complete for solving 2-pure $E_1 \cup E_2$ -unification problems.*

Proof. Consider an $E_1 \cup E_2$ -unifier σ of $s =^? t$ where s and t are 2-pure. By induction and Lemma 13, $s\sigma =_{E_1 \cup E_2} t\sigma$ implies $s\sigma^\phi =_{E_2} t\sigma^\phi$ where in fact σ is an instance of σ^ϕ . □

Hence, an E_2 -unification algorithm provides an algorithm A_2 satisfying Restriction 2 since it computes a complete set of 2-pure unifiers.

4.3 Satisfying Restriction 3

We first show that the assumptions imply the first part of Restriction 3.

Lemma 15. *A $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted term cannot be $E_1 \cup E_2$ -equal to a Σ_2 -rooted term.*

Proof. If a $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted term is $E_1 \cup E_2$ -equal to a Σ_2 -rooted term, then Lemma 13 contradicts the fact that E_2 is subterm collapse-free. $\square$

The following lemma is useful to show that $E_1 \cup E_2$ is subterm collapse-free.

Lemma 16. *For any term t and any strict subterm u of t , if $t \rightarrow_{R_1}^+ t'$ then $t' \neq_{E_2} u$.*

Proof Sketch. If $t \rightarrow_{R_1}^+ t'$, then $t' \neq u$ for any strict subterm u of t , otherwise it would contradict the fact that E_1 is subterm collapse-free. To prove that $t' \neq_{E_2} u$, let us first introduce the notion of layer defined via the bijection ϕ used to define ϕ -abstraction in Section 4.2. A *1-layer* (resp. *2-layer*) of a term s is a 1-pure (resp. 2-pure) term which is obtained from any $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted (resp. Σ_2 -rooted) subterm s' of s having no direct $\Sigma_1 \setminus \Sigma_{(1,2)}$ -rooted (resp. Σ_2 -rooted) superterm, by replacing each alien subterm s'' of s' by the variable $\phi([s'']_{E_1 \cup E_2})$, where $[s'']_{E_1 \cup E_2}$ denotes the equivalence class of s'' modulo $E_1 \cup E_2$. We are now ready to state the following facts:

(1) According to the form of rules in R_1 , we can show that t' and t have necessarily different 1-layers. Then, this can be extended to any subterm of t , since E_1 is subterm collapse-free: t' and any subterm of t have different 1-layers. Hence, t' and u have necessarily different 1-layers.

(2) Since E_2 is subterm collapse-free, E_2 is a set of equalities between non-variable 2-pure terms having the same set of variables. Therefore, $t' =_{E_2} u$ implies that t' and u have the same 1-layers.

Consequently, it is not possible to have $t' =_{E_2} u$. $\square$

Theorem 3. *$E_1 \cup E_2$ is subterm collapse-free.*

Proof. Assume that $t =_{E_1 \cup E_2} u$ for a strict subterm u of t . We can assume that u is in R_1 -normal form. By Lemma 8, we have that $t \downarrow_{R_1} =_{E_2} u$. If $t \downarrow_{R_1} = t$, then we get $t =_{E_2} u$, which contradicts the fact that E_2 is subterm collapse-free. If $t \downarrow_{R_1} \neq t$, then we get a contradiction by Lemma 16. $\square$

4.4 Unifiability of Partial Solved Forms

Theorem 4. *$E_1 \cup E_2$ -unifiability of $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved forms is decidable if general E_2 -unification is decidable.*

Proof. We use the notion of unification with (linear) constant restriction [5] and the fact that E_2 -unification with linear constant restriction corresponds to general E_2 -unification. A $\Sigma_1 \setminus \Sigma_{(1,2)}$ -solved form can be decomposed into $\mathcal{P}_1 \cup \mathcal{P}_2$

where $\mathcal{P}_1$ denotes the solved form containing only $\Sigma_1 \setminus \Sigma_{(1,2)}$ -equations between a variable and a non-variable. Let σ_1 be the idempotent substitution corresponding to the solved form $\mathcal{P}_1$. This substitution defines a constant restriction: If x is in the domain of σ_1 , then x must be considered as a (free) constant in E_2 . If x is in the domain of σ_1 and y occurs in $x\sigma_1$, then the constant x must not occur in the solution of y in E_2 . Let $<$ be a linear constant restriction satisfying the above constant restriction. We are now ready to state that $\mathcal{P}_1 \cup \mathcal{P}_2$ is $E_1 \cup E_2$ -unifiable iff $\mathcal{P}_2$ is E_2 -unifiable with the linear constant restriction $<$: ($\Leftarrow$) We can show that for any E_2 -unifier σ_2 of $\mathcal{P}_2$ satisfying the linear constant restriction $<$, $\sigma_1 \circ \sigma_2$ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}_1 \cup \mathcal{P}_2$. ($\Rightarrow$) If σ is an $E_1 \cup E_2$ -unifier of $\mathcal{P}_1 \cup \mathcal{P}_2$, then σ^ϕ is an E_2 -unifier of $\mathcal{P}_2$. Moreover, σ^ϕ satisfies a linear constant restriction extending the restriction given by σ_1 . Otherwise, it would be possible to contradict Restriction 3. $\square$

As a consequence, we can consider in our assumptions a general E_2 -unifiability algorithm (that is, E_2 -unifiability with free symbols) instead of an E_2 -unification algorithm to get an $E_1 \cup E_2$ -unifiability algorithm.

4.5 Inner Constructor Definitions and Undecidable Results

Suppose we weaken Definition 4, such that inner constructors can appear on the left hand side and have constants below them. The following convergent system, R_1^u , satisfying the new definition, where $+$ is the inner constructor:

$$\begin{aligned} B(x_1 * x_2, x_3 * x_4) &\rightarrow f(b + a, g(x_1, x_2, x_3, x_4)) \\ f(a + b, g(x_1, x_2, x_3, x_4)) &\rightarrow B(x_1, x_3) * B(x_2, x_4) \end{aligned}$$

Now let our second theory, E_2^u , be the commutative theory for $+$. Then $R_1^u \cup E_2^u$ -unification is undecidable due to the undecidability result for the theory of Synchronous Distributivity, $B(u * v, x * y) \longrightarrow B(u, x) * B(v, y)$, presented in [2].

5 Conclusion and Perspective

Building on previous combination results [5, 11, 18], and our own earlier work [13], we are able to identify a set of restrictions and a combination method such that if the restrictions are satisfied the method produces a unification algorithm in the union of non-disjoint equational theories. Furthermore, we identify a class of theories that satisfy the restrictions and thus the combination problem for these theories is decidable. The theory $\mathcal{E}_{AC}$, which is of interest in the area of cryptographic protocol analysis, is also presented as a theory for which the method is applicable.

Another candidate theory where the approach may be applicable is a partial theory of Cipher Block Chaining, also of interest in cryptographic protocol analysis and studied in [1]. The theory has the axiom: $bc(cons(x, Y), z) = cons(h(x, z), bc(Y, h(x, z)))$ which can be oriented from left to right. Notice that the symbol h in a left to right orientation would be an inner constructor.

The symbol h could then be given some equational properties, like AC , which would be a natural extension. The AC theory for h would form the lower theory E_2 . The combination method would then be applicable for obtaining an unification algorithm for this AC -cipher block chaining theory. Interestingly, the full theory studied in [1], includes the additional axiom $bc(nil, z) = nil$. Although this would violate the restriction on subterm collapse-free theories, it may still be possible to extend the combination result to this and similar cases by first “removing” the collapsing conditions by a closure method.

With respect to efficiency, it should be possible to improve the algorithm by cutting down on the number of partitions. The first example of this can be seen in [13] where only a particular subset of the variables are needed to form the partitions and thus the number of partitions is reduced. This type of improvement may be possible in many cases. More efficient methods for enumerating these types of partitions could be developed, where the fact that these partitions will correspond to equalities between variables is taken into account. Some efficiency can be obtained if the enumeration is done by enumerating the more “granular” (less variables equated) partitions first and then proceeding down the lattice of partitions to partitions where more variables are equated. If a unifier is found for one particular partition, all partitions that are both comparable and lower in the lattice are unifiable but with less general unifiers. Therefore, paths in the lattice of partitions could be pruned. Finally, it may also be possible to apply a deterministic approach. Rather than guessing a priori all possible variable identifications, allow the algorithms to ping-pong but provide a measure that is reduced at each application of an algorithm.

Acknowledgments. We are grateful to Maria Paola Bonacina and the anonymous referees for their fruitful comments. A. M. Marshall gratefully acknowledges the support of an ASEE postdoctoral fellowship. S. Erbatur was supported by the Department of Computer Science of the University at Albany and then INRIA Nancy-Grand Est during a portion of this work.

References

- [1] Anantharaman, S., Bouchard, C., Narendran, P., Rusinowitch, M.: Unification modulo chaining. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 70–82. Springer, Heidelberg (2012)
- [2] Anantharaman, S., Erbatur, S., Lynch, C., Narendran, P., Rusinowitch, M.: Unification modulo synchronous distributivity. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 14–29. Springer, Heidelberg (2012)
- [3] Baader, F., Ghilardi, S., Tinelli, C.: A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 183–197. Springer, Heidelberg (2004)
- [4] Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press, New York (1998)

- [5] Baader, F., Schulz, K.U.: Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computation* 21(2), 211–243 (1996)
- [6] Baader, F., Snyder, W.: Unification theory. In: Robinson, J.A., Voronkov, A. (eds.) *Handbook of Automated Reasoning*, pp. 445–532. Elsevier and MIT Press (2001)
- [7] Baader, F., Tinelli, C.: Combining equational theories sharing non-collapse-free constructors. In: Kirchner, H. (ed.) *FroCos 2000. LNCS (LNAI)*, vol. 1794, pp. 260–274. Springer, Heidelberg (2000)
- [8] Baader, F., Tinelli, C.: Combining decision procedures for positive theories sharing constructors. In: Tison, S. (ed.) *RTA 2002. LNCS*, vol. 2378, pp. 352–366. Springer, Heidelberg (2002)
- [9] Boudet, A.: Combining unification algorithms. *Journal of Symbolic Computation* 16(6), 597–626 (1993)
- [10] Bürckert, H.-J., Herold, A., Schmidt-Schauß, M.: On equational theories, unification, and (un)decidability. *Journal of Symbolic Computation* 8(1-2), 3–49 (1989)
- [11] Domenjoud, E., Klay, F., Ringeissen, C.: Combination techniques for non-disjoint equational theories. In: Bundy, A. (ed.) *CADE 1994. LNCS*, vol. 814, pp. 267–281. Springer, Heidelberg (1994)
- [12] Dougherty, D.J., Johann, P.: An improved general E-unification method. *Journal of Symbolic Computation* 14(4), 303–320 (1992)
- [13] Erbatur, S., Marshall, A.M., Kapur, D., Narendran, P.: Unification over distributive exponentiation (sub)theories. *Journal of Automata, Languages and Combinatorics (JALC)* 16(2-4), 109–140 (2011)
- [14] Gallier, J.H., Snyder, W.: Complete sets of transformations for general E-unification. *Theoretical Computer Science* 67(2-3), 203–260 (1989)
- [15] Huet, G.P.: Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM (JACM)* 27(4), 797–821 (1980)
- [16] Jouannaud, J.-P., Kirchner, C.: Solving equations in abstract algebras: A rule-based survey of unification. In: *Computational Logic - Essays in Honor of Alan Robinson*, pp. 257–321 (1991)
- [17] Morawska, B.: General E-unification with eager variable elimination and a nice cycle rule. *Journal of Automated Reasoning* 39, 77–106 (2007)
- [18] Ringeissen, C.: Unification in a combination of equational theories with shared constants and its application to primal algebras. In: Voronkov, A. (ed.) *LPAR 1992. LNCS*, vol. 624, pp. 261–272. Springer, Heidelberg (1992)
- [19] Schmidt-Schauß, M.: Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation* 8, 51–99 (1989)
- [20] Snyder, W.: *A Proof Theory for General Unification*, Birkhauser. *Progress in Computer Science and Applied Logic*, vol. 11 (1991)
- [21] Tidén, E.: Unification in combinations of collapse-free theories with disjoint sets of function symbols. In: Siekmann, J.H. (ed.) *CADE 1986. LNCS*, vol. 230, pp. 431–449. Springer, Heidelberg (1986)
- [22] Yelick, K.A.: Unification in combinations of collapse-free regular theories. *Journal of Symbolic Computation* 3(1-2), 153–181 (1987)