Unification and Matching in Hierarchical Combinations of Syntactic Theories

Serdar Erbatur, Deepak Kapur, Andrew M. Marshall, Paliath Narendran, Christophe Ringeissen

Unification and Matching in Hierarchical Combinations of Syntactic Theories

Serdar Erbatur[1], Deepak Kapur[2], Andrew M. Marshall[3], Paliath Narendran[4], and Christophe Ringeissen[5]

[1] Ludwig-Maximilians-Universität, München (Germany)
[2] University of New Mexico (USA)
[3] University of Mary Washington (USA)
[4] University at Albany, SUNY (USA)
[5] LORIA – INRIA Nancy-Grand Est (France)

Abstract. We investigate a hierarchical combination approach to the unification problem in non-disjoint unions of equational theories. In this approach, the idea is to extend a base theory with some additional axioms given by rewrite rules in such way that the unification algorithm known for the base theory can be reused without loss of completeness. Additional techniques are required to solve a combined problem by reducing it to a problem in the base theory. In this paper we show that the hierarchical combination approach applies successfully to some classes of syntactic theories, such as shallow theories since the required unification algorithms needed for the combination algorithm can always be obtained. We also consider the matching problem in syntactic extensions of a base theory. Due to the more restricted nature of the matching problem, we obtain several improvements over the unification problem.

1 Introduction

A critical question in matching and unification is how to obtain an algorithm for the combination of non-disjoint equational theories when there exist algorithms for the constituent theories. In recent work [7], a new "hierarchical" approach to the unification problem in the combination of non-disjoint theories is developed and classes of theories are identified for which the method can be applied. The main property of these classes is a hierarchical organization of the two equational theories E_1 and E_2, where E_1 is a set of axioms extending a base theory E_2. The method is successful in providing a unification method to the combination of theories for which no previous combination method was applicable. However, the main difficulty in applying the new combination method is that a new type of unification algorithm, denoted by A_1, is required to incorporate the axioms of E_1. The A_1 algorithm is not actually a black box unification algorithm. Rather the algorithm constructs a specific type of solved-form which has the property of reducing an $E_1 \cup E_2$ unification problem to one or more E_2-unification problems. A general A_1 method is developed in [7], based on the general E-unification methods studied in [10, 19]. However, as with the general unification methods,

there is no general termination proof for an arbitrary theory. Termination must be proven for A_1 to satisfy the restrictions of [7].

Here we are able to overcome this limitation by showing that for some classes of theories, such as shallow equational theories, the unification *algorithms* for those theories can be used for the A_1 algorithm. This replaces the need to construct a dedicated unification algorithm via the method presented in [7] and allows one to use an "off the shelf" available algorithm.

In this paper, we also consider the matching problem in this new hierarchical framework and obtain a new non-disjoint combination method for the matching problem. Due to the more restricted nature of the matching problem, the combination method introduced in [7] for unification can be simplified. For the matching problem, we are not restricted to shallow theories but syntactic theories [5, 11, 12] are allowed. By assuming a resolvent presentation of a syntactic theory E_1, we are able to construct in a modular way a matching algorithm for hierarchical combination. The matching algorithm we present can be seen as an extension of the one known for disjoint unions of syntactic (finite) theories [15] and a variation of the one existing for disjoint unions of regular theories [16, 17]. A preliminary version of this algorithm has been presented in [8]. Compared to [8], we now rely on the standard definition of syntactic theory.

Let us give a brief preview of the remaining portions of the paper. Section 2 presents the preliminary background material. Section 3 presents an overview of the hierarchical combination. The overview given here is actually an improvement via simplification to the original hierarchical presentation in [7]. Section 4 applies hierarchical combination to shallow theories. Section 5 extends the hierarchical approach from the unification problem to the matching problem by considering a syntactic theory E_1.

2 Preliminaries

We use the standard notation of equational unification [3] and term rewriting systems [1]. Given a first-order signature Σ and a (countable) set of variables V, the set of Σ-terms over variables V is denoted by $T(\Sigma, V)$. The set of variables in a term t is denoted by $Var(t)$. A term t is *ground* if $Var(t) = \emptyset$. For any position p in a term t, $t|_p$ denotes the subterm of t at position p, and $t[u]_p$ denotes the term t in which $t|_p$ is replaced by u. Given 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. For any Σ-term t, the equivalence class of t with respect to $=_E$ is denoted by $[t]_E$. The E-free algebra over V is denoted by $T(\Sigma, V)/=_E$. By a slight abuse of terminology, E will be often called an equational theory. An axiom $l = r$ is *variable-preserving* (also called *regular*) 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., regular/linear/collapse-free) if all its axioms are variable-preserving (resp., regular/linear/collapse-free). An equational theory E is *finite* if for each term t, there are only finitely many terms s such that $t =_E s$.

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. A theory E is *syntactic* if it has a finite *resolvent presentation* S, that is a presentation S such that each equality $s =_E t$ has an equational proof $s \leftrightarrow_S^* t$ with at most one step $\leftrightarrow_S$ applied at the root position. A theory E is *shallow* if variables can only occur at a position at most 1 in axioms of E. When E is both subterm collapse-free and shallow, variables can only occur at position 1 in axioms of E. Shallow theories are known to be syntactic theories [6]. Let us recall some results connecting syntactic theories and unification. First, shallow theories admit a mutation-based unification algorithm [6]. Second, finite syntactic theories admit a mutation-based matching algorithm [16]. We will reuse both of these algorithms in this paper. In addition, any collapse-free and finitary unifying theory is syntactic [12]. For instance the Associative-Commutative (AC) theory is syntactic, as well as the $\mathcal{E}_{AC}$ theory introduced in [9] as a finitary unifying theory of distributive exponentiation.

A Σ-equation is a pair of Σ-terms denoted by $s =^? t$. An E-unification problem is a set of Σ-equations, $\mathcal{S} = \{s_1 =^? t_1, \ldots, s_m =^? t_m\}$. The set of variables of $\mathcal{S}$ is denoted by $Var(\mathcal{S})$. When $t_1, \ldots, t_n$ are ground, $\mathcal{S}$ is called a matching problem, also denoted by $\{s_1 \leq^? t_1, \ldots, s_m \leq^? t_m\}$, where $s_i \leq^? t_i$ denotes a match-equation. A solution to an E-unification problem $\mathcal{S}$, 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 $CSU_E(\mathcal{S})$ such that each $\sigma \in CSU_E(\mathcal{S})$ is an E-unifier of $\mathcal{S}$, and for each E-unifier θ of $\mathcal{S}$, there exists $\sigma \in CSU_E(\mathcal{S})$ such that $\sigma \leq_E^{Var(\mathcal{S})} \theta$. An inference rule $\mathcal{S} \vdash \mathcal{S}'$ is sound (resp. complete) for E-unification if $CSU_E(\mathcal{S}) \subseteq CSU_E(\mathcal{S}')$ (resp. $CSU_E(\mathcal{S}) \supseteq CSU_E(\mathcal{S}')$). An inference system is sound (resp. complete) for E-unification if all its inference rules are sound (resp. complete). A set of equations is said to be in *dag-solved form* (or *d-solved form*) if they can be arranged as a list $x_1 =^? t_1, \ \ldots, \ x_n =^? t_n$ 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. Each x_i in this case is called a *solved variable*. 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.

Consider the union $\Sigma_1 \cup \Sigma_2$ of two disjoint signatures Σ_1 and Σ_2. A term t is called a Σ_i-rooted term if its root symbol is in Σ_i. Variables, Σ_i-terms and Σ_i-equations are *i-pure*. We also use the notion of an *alien subterm*. An alien subterm of a Σ_i-rooted term t is a Σ_j-rooted subterm s ($i \neq j$) such that all superterms of s are Σ_i-rooted. A *purification* procedure can be defined that uses *variable abstraction* [2] to replace any alien subterm u in a term t by a fresh variable x and adds the equation $x =^? u$. Then, equations $s =^? t$ between a 1-pure term s and a 2-pure term t are split into two pure equations $x =^? s$ and $x =^? t$ where x is a new variable. An equation between two variables is always both 1-pure and 2-pure. Given a set of pure equations P, we denote by $P|_{\Sigma_i}$ the

set of i-pure equations in P (for $i = 1, 2$), and we say that P is in Σ_i-*solved form* (*partial* solved form) if $P|_{\Sigma_i}$ is in dag-solved form.

A *term rewriting system* (TRS) is a pair (Σ, R), where Σ is the signature and R is a finite set of rewrite rules of the form $l \to r$ such that l, r are Σ-terms, l is not a variable and $Var(r) \subseteq Var(l)$. Given a TRS $R = (\Sigma, R)$, the signature Σ is often partitioned into two disjoint sets $\Sigma := C \uplus D$, where $D := \{f | f(t_1, \ldots, t_n) \to r \in R\}$ and $C := \Sigma \setminus D$. Symbols in C are called *constructors*, and symbols in D are called *defined functions*. A term s *rewrites* to a term t, denoted by $s \to_R t$ (or simply $s \to t$), if there exist a position p of s, $l \to r \in R$, and substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. A term s is a *normal form with respect to the relation* $\to_R$ (or simply a normal form), if there is no term t such that $s \to_R t$. This notion is lifted to substitutions as follows: a substitution σ is *normalized* if, for every variable x in the domain of σ, $x\sigma$ is a normal form. A TRS R is *terminating* if there are no infinite reduction sequences with respect to $\to_R$. A TRS R is *confluent* if, whenever $t \to_R s_1$ and $t \to_R s_2$, there exists a term w such that $s_1 \to_R^* w$ and $s_2 \to_R^* w$. A confluent and terminating TRS is called *convergent*. In a convergent TRS, we have the existence and the unicity of R-normal forms, denoted by $t{\downarrow_R}$ for any term t. We define an *inner constructor* to be a constructor f that satisfies the following additional restrictions: (1) f does not appear on the left-hand side on any rule in R; (2) f does not appear as the root symbol on the right-hand side of any rule in R; (3) there are no non-constant function symbols below f on the right-hand side of any rule in R. We consider a $\Sigma_1 \cup \Sigma_2$-theory E_1 and a Σ_2-theory E_2. The $\Sigma_1 \cup \Sigma_2$-theory E_1 is given by a Σ_1-*rooted* TRS R_1 such that for each $l \to r \in R_1$, l and r are Σ_1-rooted terms. Moreover, Σ_2-symbols do not occur in left-hand sides of R_1. We use the notion of *convergence modulo an equational theory*. When R_1 is convergent modulo E_2, we have that for any terms s and t, $s =_{R_1 \cup E_2} t$ if and only if $s{\downarrow_{R_1}} =_{E_2} t{\downarrow_{R_1}}$.

3 Hierarchical Combination

Our hierarchical combination has been introduced in [7]. We present a simplified version in Definition 1. The motivation is to simplify the relationships between the theories and allows for an easier notation. However, the new definition follows the one presented in [7]. For completeness, we also repeat several results proven in [7] on hierarchical combination. These results will prove useful in showing the applicability of hierarchical combination to shallow theories, and then more generally to syntactic theories.

Definition 1. *A* hierarchical combination *is a pair* (E_1, E_2) *such that*

- $\Sigma_1 \cap \Sigma_2 = \emptyset$*;*
- E_1 *is a subterm collapse-free equational* $\Sigma_1 \cup \Sigma_2$*-theory given by a* TRS R_1 *which is* Σ_1*-rooted and convergent modulo* E_2*;*
- E_2 *is a finite equational* Σ_2*-theory;*
- Σ_2*-symbols are* inner constructors *in* R_1*.*

A hierarchical combination (E_1, E_2) is finite (resp., shallow/syntactic) if E_1 is finite (resp., shallow/syntactic). The equational theory of (E_1, E_2) is $E_1 \cup E_2$.

Since a finite theory is subterm collapse-free, we have that both E_1 and E_2 are subterm collapse-free.

Proposition 1. *[7] If (E_1, E_2) is a hierarchical combination, then $E_1 \cup E_2$ is subterm collapse-free.*

Let us now introduce a key notion of great interest to relate combined equational proofs with pure ones. The notion of variable abstraction is widely used in the context of combined equational theories [2,4,18]. In the following, we use a slight adaptation to abstract ground terms by constants ($\mathcal{C}$) and non-ground terms by variables ($\mathcal{Y}$). According to our assumptions, theories are regular and so a ground term cannot be equal to a non-ground one.

Definition 2 (Variable Abstraction). *Let $\mathcal{Y}$ be a countably infinite set of variables, $\mathcal{C}$ be a countably infinite set of constants such that V, $\mathcal{Y}$ and $\mathcal{C}$ are pairwise disjoint. We consider a bijection*

$$\pi : (T(\Sigma_1 \cup \Sigma_2 \cup V)_{\downarrow_{R_1}} \setminus V)/ =_{E_2} \longrightarrow \mathcal{Y} \cup \mathcal{C}$$

*such that $\pi([t\downarrow_{R_1}]_{E_2}) \in \mathcal{C}$ if and only if t is ground. For $i = 1, 2$, the π_i-*abstraction *of t is denoted by t^{π_i} and defined as follows:*

- *If $t \in V$, then $t^{\pi_i} = t$.*
- *If $t = f(t_1, \ldots, t_n)$ and $f \in \Sigma_i$, then $t^{\pi_i} = f(t_1^{\pi_i}, \ldots, t_n^{\pi_i})$.*
- *Otherwise, $t^{\pi_i} = \pi([t\downarrow_{R_1}]_{E_2})$.*

An inverse mapping of π *is a mapping $\pi^{-1} : \mathcal{Y} \cup \mathcal{C} \longrightarrow (T(\Sigma_1 \cup \Sigma_2 \cup V) \setminus V)$ such that $\pi([\pi^{-1}(z)\downarrow_{R_1}]_{E_2}) = z$ for any $z \in \mathcal{Y} \cup \mathcal{C}$.*

In a hierarchical combination, a key feature is the ability to reuse an E_2-unification algorithm, say A_2, to solve 2-pure unification problems without loss of completeness.

Lemma 1. *[7] Let (E_1, E_2) be any hierarchical combination.*

- *For any terms s and t, if $s \longleftrightarrow_{E_1 \cup E_2} t$, then $s^{\pi_2} =_{E_2} t^{\pi_2}$.*
- *E_2-unification is a sound and complete method to solve 2-pure $E_1 \cup E_2$-unification problems.*

Let us now present the combination procedure.

3.1 Combination Procedure for Hierarchical Combination

We describe the combination procedure in an abstract way. By applying variable abstraction, we split the input set of equations P into two sets of 1-pure and 2-pure equations, denoted respectively by P_1 and P_2. Two algorithms A_1 and A_2 aim at solving P_1 and P_2:

- A_1 returns a Σ_1-solved form of 1-pure $E_1 \cup E_2$-unification problems. Thus, A_1 is a special type of algorithm that returns a "partial" solved form. We address the problem of building such a procedure in Section 4.
- A_2 is an E_2-unification algorithm.

Lemma 1 shows that an E_2-unification algorithm is the right tool for solving 2-pure problems. For unification problems that are not 2-pure, the idea is to use a rule-based unification procedure A_1. The role of A_1 is to reduce the problem to a form for which A_2 can be applied. The approach taken is as follows:

1. Run A_1 on P_1 to obtain P_1'.
2. Run A_2 on P_2 to obtain P_2'.
3. Combine P_1' and P_2' into a set of equations P'.
4. If P' is not solved, then iterate the procedure with P' as input.

In comparison with the combination algorithm presented in [7], the above description represents the core procedure of hierarchical combination. The algorithm of [7] can be seen as a single iteration of the loop in the above procedure. To ensure a single iteration is sufficient there are some additional technical restrictions present in [7], such as restricting the occurrence of "ping-pong variables". Additionally, some machinery, such as variable identification, is also required.

4 Unification in Shallow Hierarchical Combination

In this section we show that shallow theories are another class of theories for which a hierarchical combination approach can be applied. Let E_1 be a shallow equational theory [6] and E_2 an equational theory, both satisfying Definition 1. Thus, R_1 is a rewrite system where variables occur in rules at depth 1 (a variable cannot occur at depth 0 since R_1 is collapse-free). In this particular case, an inner constructor can only occur in a right hand side of a rule and as the root symbol of a ground flat term. By applying constant abstraction on right-hand sides of rules, we can build an "equivalent" shallow rewrite system which is disjoint from E_2. As shown below, we can rely on a unification algorithm for shallow equational theories [6, 13, 14] to build an A_1 procedure.

Let $R_1^\pi = \{l \to r^{\pi_1} \mid l \to r \in R_1\}$. Obviously, R_1^π is also a shallow rewrite system.

Lemma 2. *Let s be a Σ_1-rooted term such that its alien subterms are R_1-normalized. If $s \to_{R_1} t$, then*

- *t is a Σ_1-rooted term such that its alien subterms are R_1-normalized,*
- *$s^{\pi_1} \to_{R_1^\pi} t^{\pi_1}$.*

Proof. Let $s \to_{R_1} t$.

1. E_1 is subterm collapse free and Σ_2 symbols in R_1 appear as inner constructors. Thus, if s is Σ_1-rooted then t is Σ_1-rooted.

2. If $s = s[l\sigma]_p \to_R s[r\sigma]_p = t$ for some R-normalized σ and position p, then p must not occur in an alien subterm of s since we assume s has R_1-normalized alien subterms. This implies, $s^{\pi_1} = s^{\pi_1}[(l\sigma)^{\pi_1}]_p$ and $t^{\pi_1} = s^{\pi_1}[(r\sigma)^{\pi_1}]_p$. Let $(\sigma)^{\pi_1} := \{x \mapsto (x\sigma)^{\pi_1} \mid x \in Var(l)\}$. Note, l contains only Σ_1 symbols and any Σ_2 symbols in r must be roots of ground terms. Thus, $(l\sigma)^{\pi_1} = l(\sigma)^{\pi_1}$ and $(r\sigma)^{\pi_1} = r^{\pi_1}(\sigma)^{\pi_1}$. This implies

$$s^{\pi_1} = s^{\pi_1}[(l\sigma)^{\pi_1}]_p = s^{\pi_1}[l(\sigma)^{\pi_1}]_p \to_{R_1^\pi} s^{\pi_1}[r^{\pi_1}(\sigma)^{\pi_1}]_p = s^{\pi_1}[(r\sigma)^{\pi_1}]_p = t^{\pi_1}$$

□

Lemma 3. *Let s and t be Σ_1-rooted terms. If $s =_{E_2} t$ then $s^{\pi_1} = t^{\pi_1}$.*

Proof. Let $s =_{E_2} t$. For each pair of alien subterms $s' \in s$ and $t' \in t$ such that $s' \leftrightarrow^*_{E_2} t'$, $\pi(s') = \pi(t')$. In addition, s and t have the same root symbol. Therefore, $s^{\pi_1} = t^{\pi_1}$. □

Proposition 2. *Let $s =^? t$ be a Σ_1-equation, and σ be a R_1-normalized substitution. If $s\sigma =_{E_1 \cup E_2} t\sigma$, then $s\sigma^{\pi_1} =_{R_1^\pi} t\sigma^{\pi_1}$.*

Proof. Since $s\sigma =_{E_1 \cup E_2} t\sigma$, we have $s\sigma \to^*_{R_1} (s\sigma)\downarrow_{R_1} =_{E_2} (t\sigma)\downarrow_{R_1} \leftarrow^*_{R_1} t\sigma$. By Lemma 2, we have $s\sigma^{\pi_1} \to^*_{R_1^\pi} ((s\sigma)\downarrow_{R_1})^{\pi_1}$ and $t\sigma^{\pi_1} \to^*_{R_1^\pi} ((t\sigma)\downarrow_{R_1})^{\pi_1}$. By Lemma 3, $((s\sigma)\downarrow_{R_1})^{\pi_1} = ((t\sigma)\downarrow_{R_1})^{\pi_1}$, and so $s\sigma^{\pi_1} =_{R_1^\pi} t\sigma^{\pi_1}$. □

Proposition 2 shows that the problem of solving 1-pure equations can be reduced to a R_1^π-unification problem. Let us now show that the hierarchical combination approach initiated in [7] can be applied to the theories under consideration. It is shown in [7] that if three restrictions are satisfied by the combined theory, there exists a combined unification algorithm. We present those restrictions below, rephrased as to conform to the current presentation.

Restriction 1 *(Algorithm A_1) Let P_1 be a set of 1-pure equations. Algorithm A_1 applied to P_1 computes a set of problems $\{\mathcal{Q}_k\}_{k \in K}$ such that*

$$\bigcup_{k \in K} CSU_{E_1 \cup E_2}(\mathcal{Q}_k) \text{ is a } CSU_{E_1 \cup E_2}(P_1) \text{ and for each } k \in K\text{:}$$

(i) $\mathcal{Q}_k$ is in Σ_1-solved form.
(ii) No fresh variable occur under Σ_2-rooted terms.

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

Restriction 3 *(Errors)*

(i) A Σ_2-rooted term cannot be $E_1 \cup E_2$-equal to a Σ_1-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.

Lemma 4. *Shallow hierarchical combination (E_1, E_2) satisfies Restrictions (1) through (3).*

Proof. Let us consider the different restrictions.

- Restriction 1: Since R_1^π is shallow, R_1^π-unification is finitary and a unification algorithm for shallow theories is known. Given a set of Σ_1-equations P_1, it computes a finite complete set of R_1^π-unifiers of P_1, say $\{\sigma_k\}_{k\in K}$. By Proposition 2, $\{\sigma_k\pi^{-1}\}_{k\in K}$ is a $CSU_{E_1\cup E_2}(P_1)$. By purification, each σ_k can be equivalently written as a unification problem $\mathcal{Q}_k$ such that $\mathcal{Q}_k$ is in Σ_1-solved form, and $\mathcal{Q}_k$ contains only ground Σ_2-rooted terms.
- Restriction 2: This restriction follows directly from Lemma 1.
- Restriction 3: $E_1 \cup E_2$ is subterm collapse-free due to Proposition 1. Since E_1 and E_2 are both subterm collapse-free, Σ_2-rooted terms cannot be $E_1 \cup E_2$-equal to Σ_1-rooted terms without contradicting Lemma 1. □

Directly from Lemma 4 and [7] we have the following.

Theorem 1. *For any shallow hierarchical combination (E_1, E_2), there exists a* combined *$E_1 \cup E_2$-unification algorithm, provided that an E_2-unification algorithm is known.*

Proof. In [6], unification is shown decidable and finitary in shallow theories. In addition, they provide a method for constructing a unification algorithm for an arbitrary shallow theory. Thus, we can construct a finitary unification algorithm, A_1, from the shallow equational theory of R_1^π. By Lemma 4, the result follows. □

Example 1. Consider the (ground) hierarchical combination (E_1, E_2) where the TRS R_1 associated with E_1 is $\{f(a,b) \to g(a+b), f(b,a) \to g(b+a)\}$ and $E_2 = \{x+y = y+x\}$. The TRS R_1 is convergent modulo E_2. The TRS R_1^π is $\{f(a,b) \to g(c), f(b,a) \to g(c)\}$ where $c = \pi([a+b]_{E_2})$. A R_1^π-unification algorithm can be obtained by adding to a syntactic unification algorithm some mutation rules [6]. Then, most general $E_1 \cup E_2$-unifiers of a 1-pure unification problem can be derived from most general R_1^π-unifiers by replacing the constant c with $a + b$.

Shallow theories are particular syntactic theories admitting a terminating mutation-based unification procedure. In [13], Lynch and Morawska have investigated a larger class of syntactic theories admitting a terminating mutation-based unification procedure. In Section 5, we consider a mutation-based approach for the matching problem. This particular unification problem is particularly interesting with respect to our combination method since there will be no termination issue due to the fact that solutions are necessarily ground.

5 Matching in Finite Syntactic Hierarchical Combination

In this section we consider the matching problem in any finite syntactic hierarchical combination. Due to the more restricted nature of the matching problem

we obtain several improvements over the unification problem. One of the improvements is that we are able to relax several restrictions we assumed for the unification problem. In the unification setting it was necessary to restrict variables which could cause reapplication of the first unification algorithm, denoted as "ping pong" variables in [7]. This restriction can be easily fulfilled if most general solutions can be expressed without any new variable. Considering matching problems in regular theories, there are only ground solutions, and so no ping pong variables. Since subterm collapse-free theories are regular, the theories we are interested in seem well-suited for a hierarchical approach to the matching problem.

The combination algorithm we propose is similar to the one existing for matching in disjoint unions of regular theories [16, 17]. In that context, matching algorithms A_1 and A_2 are applied repeatedly until reaching normal forms that correspond to most general solutions. The key principle of the combination algorithm for matching is to purify only the left-hand sides of matching problems. Thus, this purification introduces a "pending" equation $s =^? X$ that will lead to a match-equation: since X occurs in a match-equation solved by A_1 or A_2, X will be instantiated by a ground term, say u, transforming eventually $s =^? X$ into a match-equation $s \leq^? u$.

Definition 3. *Let P be a conjunction of match-equations and equations. The set of* ground-solved *variables in P is the smallest set of variables in P such that*

1. *variables occurring in left-hand sides of match-equations are ground-solved.*
2. *if $t =^? t'$ is an equation in P such that variables in t' are ground-solved, then variables in t are also ground-solved.*

When P contains only ground-solved variables, it is called an extended matching problem. *A matching problem is an extended matching problem containing no equations. For any match-equation in an extended matching problem, we assume that the right-hand side is R_1-normalized. Hence, there will be an implicit normalization of right-hand sides when new match-equations are generated by some inference rules.*

In the following, we present a rule-based combination method to solve $E_1 \cup E_2$-matching problems. Let us briefly introduce the main steps of that combination method. On the one hand, solving match-equations with 2-pure left-hand sides generate solved match-equations. On the other hand, solving match-equations whose left-hand sides are Σ_1-rooted generate new equalities while preserving the syntactic form of an extended matching problem. Then, there are some combination rules to remove successively the equations introduced during the purification and the solving phases. Eventually, we obtain a matching problem in solved form. As shown below, this method is sound and complete when (E_1, E_2) is a finite syntactic hierarchical combination, which means that E_1, E_2 are finite and E_1 admits a resolvent presentation denoted by S_1. If E_1 and E_2 are finite, then $E_1 \cup E_2$ is finite according to the result below.

Proposition 3. *(E_1, E_2) is a finite hierarchical combination if and only if $E_1 \cup E_2$ is finite.*

Proof. (If direction) Straightforward.
(Only if direction) Assume that this is not the case, that is, there exists a term $t \in T(\Sigma_1 \cup \Sigma_2, \mathcal{X})$ with an infinite set of terms S such that $t =_{E_1 \cup E_2} s$ for each $s \in S$. Since there exists a convergent rewrite system R_1 which is equivalent to E_1, we can consider the unique R_1-normal form of each term $s \in S$. Let $S' = \{s{\downarrow_{R_1}} \mid s \in S\}$. The set of terms S' is also infinite, otherwise it would contradict that E_1 is finite. Since R_1 is convergent modulo E_2, $t{\downarrow_{R_1}} =_{E_2} s'$ for each $s' \in S'$. This implies the existence of an infinite E_2 equivalence class $[t{\downarrow_{R_1}}]_{E_2}$, which contradicts our assumption that E_2 is finite. □

Since matching is finitary in a finite theory, we could take a brute for approach to construct an $E_1 \cup E_2$-matching algorithm [15]. Actually, a match-equation $s \leq^?_{E_1 \cup E_2} t$ has the same set of (ground) $E_1 \cup E_2$-solutions as the syntactic matching problem

$$\bigvee_{\{t' \mid t' =_{E_1 \cup E_2} t\}} s \leq^?_{\emptyset} t'$$

where $\emptyset$ denotes the empty equational theory. However, similarly to [15], we can also proceed in a modular way, by using some "pure" matching algorithms, say A_1 and A_2 dedicated respectively to E_1 and E_2. In the context of matching, A_1 aims at handling match-equations whose left-hand sides are Σ_1-rooted, whilst A_2 denotes an E_2-matching algorithm to solve match-equations whose left-hand sides are 2-pure.

Restriction 4 *(E_1, E_2) is a finite syntactic hierarchical combination with the following constituent algorithms:*

1. *A_1 is given by the mutation-based algorithm $\mathcal{M}$ depicted in Fig. 1, where S_1 denotes a resolvent presentation of the finite syntactic theory E_1.*
2. *A_2 is an E_2-matching algorithm.*

Lemma 5. *E_2-matching is a sound and complete method to solve $E_1 \cup E_2$-matching problems whose left-hand sides are 2-pure, provided that the right-hand sides are replaced by their π_2-abstractions.*

Proof. Let $s \leq^? t$ be a match-equation such that s is 2-pure term, and consider the corresponding 2-pure match-equation $s \leq^? t^{\pi_2}$, where t^{π_2} can be effectively computed (up to a renaming). The soundness is obvious: any E_2-solution σ of $s \leq^? t^{\pi_2}$ leads to an $E_1 \cup E_2$-solution $\sigma\pi^{-1}$ of $s \leq^? t$. For the completeness, consider an $E_1 \cup E_2$-solution σ of $s \leq^? t$ where s is 2-pure. By Lemma 1 (and induction on the length of derivation), $s\sigma =_{E_1 \cup E_2} t$ implies $s\sigma^{\pi_2} =_{E_2} t^{\pi_2}$ where $\sigma =_{E_1 \cup E_2} \sigma^{\pi_2}\pi^{-1}$. □

5.1 Mutation-based Procedure

We now consider the question of applying a mutation-based inference system to simplify match-equations whose left-hand sides are Σ_1-rooted. The mutation inference is known to be complete for syntactic theories [5,11,12], and so for E_1. The next lemma shows that this property is preserved when considering $E_1 \cup E_2$.

Lemma 6. *Let S_1 be a resolvent presentation of E_1, and let s and t be Σ_1-rooted terms such that t is R_1-normalized. If $s =_{E_1 \cup E_2} t$, then there exists an equational proof of $s =_{E_1 \cup E_2} t$ with at most one S_1-equational step applied at the root position.*

Lemma 6 justifies the eager R_1-normalization of the right-hand sides of match-equations, and it implies the completeness of the inference system $\mathcal{M}$ (Fig. 1).

Mutate

$$\{f(s_1, \ldots, s_m) \leq^? g(t_1, \ldots, t_n)\} \cup P$$
$$\longrightarrow \{r_1 \leq^? t_1, \ldots, r_n \leq^? t_n,\ s_1 =^? l_1, \ldots, s_m =^? l_m\} \cup P$$

If $f(l_1, \ldots, l_m) = g(r_1, \ldots, r_n)$ is a fresh variant of an axiom in S_1.

Matching Decomposition

$$\{f(s_1, \ldots, s_m) \leq^? f(t_1, \ldots, t_m)\} \cup P$$
$$\longrightarrow \{s_1 \leq^? t_1, \ldots, s_m \leq^? t_m\} \cup P$$

Where $f \in \Sigma_1$.

Matching Clash

$$\{f(s_1, \ldots, s_m) \leq^? g(t_1, \ldots, t_n)\} \cup P$$
$$\longrightarrow \mathit{Fail}$$

Where $f \in \Sigma_1, f \neq g$ and **Mutate** does not apply.

Fig. 1. Mutation-based inference system $\mathcal{M}$ for Matching

Lemma 7. *$\mathcal{M}$ (Fig. 1) is sound, complete and terminating.*

Proof. The soundness is straightforward. The completeness is a consequence of Lemma 6. Let us now prove the termination. The multiset consisting of the sizes of the right-hand sides of match-equations can be used as a complexity measure. The three rules of $\mathcal{M}$ decrease this complexity measure, and so $\mathcal{M}$ terminates. □

Corollary 1. *Let P be a set of match-equations whose left-hand sides are Σ_1-rooted. There exists a finite derivation $P \vdash \cdots \vdash P'$ in $\mathcal{M}$ such that P and P' have the same set of solutions, and P' is in normal form w.r.t. $\mathcal{M}$.*

Lemma 8. *If P is an extended matching problem in normal form w.r.t. $\mathcal{M}$, then P does not contain match-equations whose left-hand sides are Σ_1-rooted.*

Proof. By contradiction. Let P_1 be the set of match-equations in P whose left-hand sides are Σ_1-rooted. If P_1 is non-empty, then some rule in $\mathcal{M}$ must apply, which means that P is not a normal form w.r.t. $\mathcal{M}$. □

To conclude this section we introduce a complexity measure, which will be useful to prove the termination of the combined matching procedure (Fig. 2). Since $E_1 \cup E_2$ is finite, any extended matching problem P has only finitely many ground solutions, and each non-ground right-hand side of an equation in P, say t, can only be instantiated among a (possibly empty) finite set of ground terms $Gnd(t)$. Then, the set of match-equations *encoded by* P is inductively defined as follows:

$$ms(\{s =^? t\} \cup P) = \bigcup_{t' \in Gnd(t)} \{s \leq^? t'\} \cup ms(P) \text{ if } t \text{ is non-ground}$$
$$ms(\{s =^? t\} \cup P) = \{s \leq^? t\} \cup ms(P) \text{ if } t \text{ ground}$$
$$ms(\{s \leq^? t\} \cup P) = \{s \leq^? t\} \cup ms(P)$$

To compare match-equations we use the ordering: $(s \leq^? t) \prec (s' \leq^? t')$ if

- s is a strict subterm of s',
- or $Var(s) \cap Var(s') = \emptyset$ and t is a strict subterm of t',
- or $t = t'$ and s strictly subsumes s'.

The ordering $\prec$ is well-founded and so its multiset extension $\prec_m$ is well-founded.

Lemma 9. *Any rule in $\mathcal{M}$ strictly decreases ms with respect to $\prec_m$.*

The above lemma is another way to prove the termination of $\mathcal{M}$.

5.2 Combination Procedure

Based on the restrictions related to A_1 and A_2, we can give a new matching procedure for the hierarchical combination.

Consider $\mathfrak{C}_{\mathcal{M}}$ the inference system depicted in Fig. 2, with the following inferences rules $\{\mathbf{Solve}_1, \mathbf{Solve}_2, \mathbf{VA}, \mathbf{RemEq}, \mathbf{Rep}, \mathbf{Merge}, \mathbf{Clash}\}$. We can easily verify that each rule in $\mathfrak{C}_{\mathcal{M}}$ preserves the set of $E_1 \cup E_2$-solutions. This is clear for the rules in $\{\mathbf{VA}, \mathbf{RemEq}, \mathbf{Rep}, \mathbf{Merge}, \mathbf{Clash}\}$. Moreover, this is true by Lemma 7 for $\mathbf{Solve}_1$, and by Lemma 5 for $\mathbf{Solve}_2$. The inference system $\mathfrak{C}_{\mathcal{M}}$ aims at computing matching problems in solved form.

Lemma 10. *Normal forms with respect to $\mathfrak{C}_{\mathcal{M}}$ are matching problems in solved form.*

Proof. Assume P is an extended matching problem which is not a matching problem in solved form. Then we can always show that a rule in $\mathfrak{C}_{\mathcal{M}}$ can be applied:

Solve$_1$:
$$\frac{P_1 \cup P}{A_1(P_1) \cup P} \qquad P_1 \text{ is a set of match-equations with } \Sigma_1\text{-rooted left-hand sides}$$
Solve$_2$:
$$\frac{P_2 \cup P}{A_2(P_2) \cup P} \qquad P_2 \text{ is a set of match-equations with 2-pure left-hand sides}$$
VA:
$$\frac{\{s[u] \leq^? t\} \cup P}{\{s[X] \leq^? t, u =^? X\} \cup P} \qquad \text{if } s \text{ is } \Sigma_2\text{−rooted, } u \text{ is an alien subterm}$$
RemEq:
$$\frac{\{t =^? t'\} \cup P}{\{t \leq^? t'\} \cup P} \qquad \text{if } t' \text{ is ground}$$
Rep:
$$\frac{\{Y \leq^? u,\ t =^? t'[Y]\ \} \cup P}{\{Y \leq^? u,\ t =^? t'[u]\} \cup P}$$
Merge:
$$\frac{\{X \leq^? t,\ X \leq^? s\} \cup P}{\{X \leq^? t\} \cup P} \qquad \text{if } s =_{E_1 \cup E_2} t$$
Clash:
$$\frac{\{X \leq^? t,\ X \leq^? s\} \cup P}{\mathit{Fail}} \qquad \text{if } s \neq_{E_1 \cup E_2} t$$

Fig. 2. $\mathfrak{C}_{\mathcal{M}}$: inference system for the combination of matching

1. If there is some equation $t =^? t'$ in P, we have two possible cases. First, if t' is ground, then **RemEq** applies. Second, if t' contains some ground-solved variable, then a match-equation containing some fresh variable must also occur in P. If this match-equation is solved, then **Rep** applies. Otherwise, just like in the third case below, either **Solve**$_1$ or **Solve**$_2$ or **VA** applies on this unsolved match-equation.
2. If there are $X \leq^? t$ and $X \leq^? s$ in P, then either **Merge** or **Clash** can be applied.
3. If there is some match-equation $s \leq^? t$ in P where s is not a variable, then either **Solve**$_1$ or **Solve**$_2$ or **VA** can be applied.

□

To show the soundness and completenes of $\mathfrak{C}_{\mathcal{M}}$, it remains to show that $\mathfrak{C}_{\mathcal{M}}$ terminates for any input.

Lemma 11. *Let P be any input matching problem. Any repeated application of rules in $\mathfrak{C}_{\mathcal{M}}$ on P terminates.*

Proof. Consider the complexity measure ms introduced for Lemma 9 plus the following ones:

- m_1: number of equations (denoted by $=^?$)
- m_2: number of Σ_2-rooted match-equations
- m_3: number of match-equations
- m_4: number of variables occurring in equations (denoted by $=^?$)

Then, the termination of $\mathfrak{C}_{\mathcal{M}}$ can be obtained by considering a lexicographic combination of these complexity measures, more precisely the one given by the tuple (ms, m_1, m_2, m_3, m_4), as shown in the table below.

	ms	m_1	m_2	m_3	m_4
Solve$_1$	$\downarrow$				
VA	$\downarrow$				
RemEq	$\downarrow=$	$\downarrow$			
Solve$_2$	$\downarrow=$	$\downarrow=$	$\downarrow$		
Merge/Clash	$\downarrow=$	$\downarrow=$	$\downarrow=$	$\downarrow$	
Rep	$\downarrow=$	$\downarrow=$	$\downarrow=$	$\downarrow=$	$\downarrow$

□

Theorem 2. *For any finite syntactic hierarchical combination (E_1, E_2), there exists a* combined *$E_1 \cup E_2$-matching algorithm (Fig. 1 and Fig. 2), provided that an E_2-matching algorithm is known.*

Proof. Direct consequence of Lemmas 10 and 11. □

5.3 Example: Matching in a Theory of Distributive Exponentiation

As an example of the above combination method for matching algorithms we consider in this section a theory of distributive exponentiation, namely $\mathcal{E}_{AC}$ for which a rule-based unification algorithm is presented in [9]. We recall that $\mathcal{E}_{AC} = (E_1, E_2)$, where $E_2 = AC(\circledast) = \{(x \circledast y) \circledast z = x \circledast (y \circledast z), x \circledast y = y \circledast x\}$ and E_1 is given by the following convergent rewrite system modulo E_2:

$$R_1 = \begin{cases} exp(exp(x, y), z) \to exp(x, y \circledast z) \\ exp(x * y, z) \to exp(x, z) * exp(y, z) \end{cases}$$

Lemma 12. *$\mathcal{E}_{AC}$ is a finite syntactic hierarchical combination $(E_1, AC(\circledast))$ where E_1 admits the following resolvent presentation:*

$$S_1 = \begin{cases} exp(exp(x, y), z) = exp(x, y \circledast z) \\ exp(x * y, z) = exp(x, z) * exp(y, z) \end{cases}$$

Note that Restriction 4(2) is addressed trivially since there are well-known AC-matching algorithms. Therefore, the main task is to instantiate the inference system $\mathcal{M}$ (Fig. 1). This leads to the mutation rules for $\mathcal{E}_{AC}$-matching shown in Fig. 3.

Example 2. Consider the equational theory $\mathcal{E}_{AC}$ and the matching problem

$$exp(X, V \circledast c_1) \leq^? exp(b, c_1 \circledast c_2 \circledast c_3)$$

The combination algorithm $\mathfrak{C}_{\mathcal{M}}$ works as follows with this input. First, **Matching Decomposition** is applied and leads to $\{X \leq^? b, V \circledast c_1 \leq^? c_1 \circledast c_2 \circledast c_3\}$. Then **Solve**$_2$ applies and provides a first solved form $\{X \leq^? b, V \leq^? c_2 \circledast c_3\}$.

Another possibility is to apply the first **Mutate** rule from Fig. 3, yielding $\{Y \circledast V \circledast c_1 \leq^? c_1 \circledast c_2 \circledast c_3, X =^? exp(b, Y)\}$. By **Solve**$_2$, the above 2-pure match-equation has two solutions. The first solution is $\{Y \leq^? c_2, V \leq^? c_3\}$. After **Rep** and **RemEq**, we obtain a new solved form: $\{V \leq^? c_3, X \leq^? exp(b, c_2)\}$. Similarly, for the second solution $\{Y \leq^? c_3, V \leq^? c_2\}$, we get the solved form $\{V \leq^? c_2, X \leq^? exp(b, c_3)\}$. The other **Mutate** rules lead to a failure thanks the application of **Matching Clash**.

Mutate

$$\{exp(s_1, s_2) \leq^? exp(t_1, t_2)\} \cup P \longrightarrow \{Y \circledast s_2 \leq^? t_2, s_1 =^? exp(t_1, Y)\} \cup P$$

$$\{exp(s_1, s_2) \leq^? exp(t_1, t_2)\} \cup P \longrightarrow \{exp(s_1, Y) \leq^? t_1, s_2 =^? Y \circledast t_2\} \cup P$$

$$\{exp(s_1, s_2) \leq^? t_1 * t_2\} \cup P \longrightarrow \{exp(X, s_2) \leq^? t_1, exp(Y, s_2) \leq^? t_2, s_1 =^? X * Y\} \cup P$$

$$\{s_1 * s_2 \leq^? exp(t_1, t_2)\} \cup P \longrightarrow \{X * Y \leq^? t_1, s_1 =^? exp(X, t_2), s_2 =^? exp(Y, t_2)\} \cup P$$

Fig. 3. Mutation rules for $\mathcal{E}_{AC}$-matching

6 Conclusion

We have presented a collection of new results about our hierarchical combination approach for solving unification problems. First we defined a simpler reformulation of the combination method, which is sufficient for the problems we focus on in this paper. Our application to shallow extensions complement and improve our earlier work on hierarchical combination presented in [7,9]. Hierarchical combination requires a solver A_1, taking in account the axioms of E_1, to produce partial solved forms. Although a general sound and complete method is available to construct A_1, the problem of termination still remains. We solve this problem for shallow theories by showing how to exploit unification algorithms known for them. Second, we have shown another combination method for the matching problem in finite syntactic extensions. Future work includes applying the general method developed here to partly ground unification problems and finding conditions which allow us to combine unification algorithms of larger classes of equational theories.

References

1. F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, New York, NY, USA, 1998.

2. F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computation*, 21(2):211 – 243, 1996.
3. F. Baader and W. Snyder. Unification theory. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.
4. A. Boudet. Combining unification algorithms. *Journal of Symbolic Computation*, 16(6):597–626, 1993.
5. A. Boudet and E. Contejean. On n-syntactic equational theories. In H. Kirchner and G. Levi, editors, *Algebraic and Logic Programming*, volume 632 of *Lecture Notes in Computer Science*, pages 446–457. Springer Berlin Heidelberg, 1992.
6. H. Comon, M. Haberstrau, and J. Jouannaud. Syntacticness, cycle-syntacticness, and shallow theories. *Inf. Comput.*, 111(1):154–191, 1994.
7. S. Erbatur, D. Kapur, A. M. Marshall, P. Narendran, and C. Ringeissen. Hierarchical combination. In M. P. Bonacina, editor, *Automated Deduction (CADE-24)*, volume 7898 of *Lecture Notes in Computer Science*, pages 249–266. Springer Berlin Heidelberg, 2013.
8. S. Erbatur, D. Kapur, A. M. Marshall, P. Narendran, and C. Ringeissen. Hierarchical combination of matching algorithms. In *Twentyeighth International Workshop on Unification (UNIF-2014), Vienna, Austria*, 2014.
9. S. Erbatur, A. M. Marshall, D. Kapur, and P. Narendran. Unification over distributive exponentiation (sub)theories. *Journal of Automata, Languages and Combinatorics (JALC)*, 16(2–4):109–140, 2011.
10. J. H. Gallier and W. Snyder. Complete sets of transformations for general E-unification. *Theoretical Computer Science*, 67(2–3):203 – 260, 1989.
11. J.-P. Jouannaud. Syntactic theories. In B. Rovan, editor, *Mathematical Foundations of Computer Science 1990*, volume 452 of *Lecture Notes in Computer Science*, pages 15–25. Springer Berlin Heidelberg, 1990.
12. C. Kirchner and F. Klay. Syntactic theories and unification. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 270–277, Jun 1990.
13. C. Lynch and B. Morawska. Basic syntactic mutation. In A. Voronkov, editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, volume 2392 of *Lecture Notes in Computer Science*, pages 471–485. Springer, 2002.
14. R. Nieuwenhuis. Decidability and complexity analysis by basic paramodulation. *Inf. Comput.*, 147(1):1–21, 1998.
15. T. Nipkow. Proof transformations for equational theories. In *Logic in Computer Science, 1990. LICS '90, Proceedings., Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 278–288, Jun 1990.
16. T. Nipkow. Combining matching algorithms: The regular case. *J. Symb. Comput.*, 12(6):633–654, 1991.
17. C. Ringeissen. Combining decision algorithms for matching in the union of disjoint equational theories. *Inf. Comput.*, 126(2):144–160, 1996.
18. M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8:51–99, July 1989.
19. W. Snyder. *A Proof Theory for General Unification*, volume 11 of *Progress in Computer Science and Applied Logic*. Birkhauser, 1991.