Unification in Non-Disjoint Combinations with Forward-Closed Theories

Ajay K. Eeralla, Serdar Erbatur, Andrew M. Marshall, Christophe Ringeissen

Unification in Non-Disjoint Combinations with Forward-Closed Theories

Ajay K. Eeralla, Serdar Erbatur, Andrew M. Marshall, Christophe Ringeissen

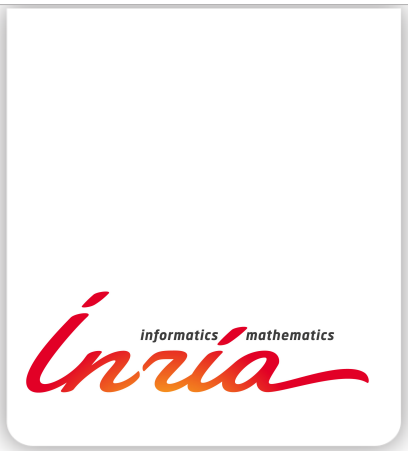
RESEARCH REPORT

N° 9252

February 2019

Project-Team Pesto

ISSN 0249-6399 ISRN INRIA/RR--9252--FR+ENG



Unification in Non-Disjoint Combinations with Forward-Closed Theories

Ajay K. Eeralla*, Serdar Erbatur†, Andrew M. Marshall‡,

Christophe Ringeissen§ ¶

Project-Team Pesto

Research Report n° 9252 — February 2019 — 23 pages

Abstract: We investigate the unification problem in theories defined by rewrite systems which are both convergent and forward-closed. These theories are also known in the context of protocol analysis as theories with the finite variant property and admit a variant-based unification algorithm. In this paper, we present a new rule-based unification algorithm which can be seen as an alternative to the variant-based approach. In addition, we define forward-closed combination to capture the union of a forward-closed convergent rewrite system with another theory, such as the Associativity-Commutativity, whose function symbols may occur in right-hand sides of the rewrite system. Finally, we present a combination algorithm for this particular class of non-disjoint unions of theories.

Key-words: term rewriting, unification, combination, forward-closure

* University of Missouri, Columbia (USA)
† Ludwig-Maximilians-Universität München (Germany)
‡ University of Mary Washington (USA)
§ Université de Lorraine, CNRS, Inria, LORIA (France)
¶ This work has received funding from the European Research Council (ERC) under the H2020 research and innovation program (grant agreement No 645865-SPOOC).

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Unification dans des mélanges non-disjoints avec des théories fermées en avant

Résumé : On étudie le problème d'unification dans les théories définies par des systèmes de réécriture qui sont à la fois convergents et fermés en avant. Ces théories sont connues dans le contexte de l'analyse de protocoles de sécurité comme les théories ayant la propriété des variants finis et admettant de ce fait un algorithme d'unification à base de variants. Dans ce papier, on présente un nouvel algorithme d'unification à base de règles qui peut être vu comme une alternative à l'approche basée sur le calcul de variants. On étudie l'union d'un système de réécriture convergent et fermé en avant avec une autre théorie dont les symboles de fonction peuvent apparaître dans les membres droits du système de réécriture. Finalement, on présente un algorithme de combinaison pour cette classe particulière d'unions non-disjointes de théories.

Mots-clés : réécriture, unification, combinaison, fermeture en avant

1 Introduction

Unification plays a central role in logic systems based on the resolution principle, to perform the computation in declarative programming, and to deduce new facts in automated reasoning. Syntactic unification is particularly well-known for its use in logic programming. Being decidable and unitary are remarkable properties of syntactic unification. More generally, we may consider equational unification, where the problem is defined modulo an equational theory E, like for instance the Associativity-Commutativity. Equational unification, say E-unification, is undecidable in general. However, specialized techniques have been developed to solve the problem for particular classes of equational theories, many of high practical interest. It is not uncommon to have such equational theories include Associativity-Commutativity, which is useful to represent arithmetic operators. Nowadays, security protocols are successfully analyzed using dedicated reasoning tools [6, 5, 13, 19] in which protocols are usually represented by clauses in first-order logic with equality. In these protocol analyzers, equational theories are used to specify the capabilities of an intruder [1]. To support the reasoning in these equational theories E, one needs to use E-unification procedures. When the equational theory E has the Finite Variant Property (FVP) [9], there exists a reduction from E-unification to syntactic unification via the computation of finitely many variants of the unification problem. When this reduction is used, we talk about variant-based unification. The class of equational theories with the FVP has attracted a considerable interest since it contains theories that are crucial in protocol analysis [14, 8, 7, 10, 20]. The concept of narrowing is another possible unification technique when E is given by a convergent term rewrite system (TRS). Narrowing is a generalization of rewriting which is widely used in declarative programming. It is complete for E-unification, but it terminates only in some very particular cases. A particular narrowing strategy, called folding variant narrowing, has been shown complete and terminating for any equational theory with the FVP [14]. When E has the property of being *syntactic* [17, 22], it is possible to apply a rule-based unification procedure in the same vein as the one known for syntactic unification [16], which is called a mutation-based unification procedure. Unfortunately, being syntactic is not a sufficient condition to insure the termination of this unification procedure. Finally, another important scenario is given by an equational theory E defined as a union of component theories. To solve this case, it is quite natural to proceed in a modular way by reusing the unification algorithms available in the component theories. There are terminating and complete combination procedures for signature-disjoint unions of theories [23, 3], but the non-disjoint case remains a challenging problem [12].

In this paper, we investigate the impact of considering an equational theory with the FVP in order to get a terminating mutation-based unification procedure and a terminating combination procedure for some non-disjoint unions of theories. Instead of directly talking about the FVP, we study the equivalent class of theories defined by forward-closed convergent TRSs [7]. Actually, a forward-closed convergent TRS is a syntactic theory admitting a terminating mutation-based unification procedure. Here, we consider the unification problem in the class of *forward-closed combinations* defined as unions of a forward-closed convergent TRS plus an equational theory over function symbols that may only occur in the right-hand sides of the TRS. To solve this problem we need a mutation procedure for the forward-closed component of the combination. Rather than reusing the mutation procedure given in [18] we develop a new mutation procedure which is more conducive to combination. By adding some standard combination rules, we show how to extend this new mutation procedure in order to solve the unification problem in forward-closed combinations.

The rest of the paper is organized as follows. Section 2 recalls the standard notions and Section 3 introduces the class of forward-closed theories. In Section 4, we present a terminating

mutation-based unification procedure for forward-closed theories. In Section 5, we introduce forward-closed combinations. The related combination method is given in Section 6, by proving its termination and correctness. For the sake of completeness, Section 7 discusses two brute force reduction methods: the first one relies on the computation of variants, while the second one is mutation-based and variant-free. Finally, Section 9 discusses some limitations and possible extensions of this work.

2 Preliminaries

We use the standard notation of equational unification [4] and term rewriting systems [2]. 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 $Var(t) = \emptyset$. A term is *linear* if all its variables occur only once. For any position p in a term t (including the root position ϵ), $t(p)$ is the symbol at position p, $t|_p$ is the subterm of t at position p, and $t[u]_p$ is the term t in which $t|_p$ is replaced by u. A substitution is an endomorphism of $T(\Sigma, V)$ with only finitely many variables not mapped to themselves. A substitution is denoted by $\sigma = \{x_1 \mapsto t_1, \ldots, x_m \mapsto t_m\}$, where the domain of σ is $Dom(\sigma) = \{x_1, \ldots, x_m\}$. Application of a substitution σ to t is written $t\sigma$. 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 (by a slight abuse of terminology, E is often called an equational theory). Equivalently, $=_E$ can be defined as the reflexive transitive closure $\leftrightarrow_E^*$ of an equational step $\leftrightarrow_E$ defined as follows: $s \leftrightarrow_E t$ if there exist a position p of s, $l = r$ (or $r = l$) in E, and substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. An axiom $l = r$ is *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 *regular* (resp., linear/collapse-free) if all its axioms are regular (resp., linear/collapse-free). A theory E is *syntactic* if it has finite *resolvent presentation* S, defined as a finite set of axioms S such that each equality $t =_E u$ has an equational proof $t \leftrightarrow_S^* u$ with at most one equational step $\leftrightarrow_S$ applied at the root position. One can easily check that $C = \{x * y = y * x\}$ (Commutativity) and $AC = \{x * (y * z) = (x * y) * z,\ x * y = y * x\}$ (Associativity-Commutativity) are regular, collapse-free, linear, and finite. Moreover, C and AC are syntactic [17]. A Σ-equation is a pair of Σ-terms denoted by $s =^? t$ or simply $s = t$ when it is clear from the context that we do not refer to an axiom. An E-unification problem is a set of Σ-equations, $G = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$, or equivalently a conjunction of Σ-equations. The set of variables in G is denoted by $Var(G)$. A solution to G, called an *E-unifier*, is a substitution σ such that $s_i\sigma =_E t_i\sigma$ for all $1 \leq i \leq n$, written $E \models G\sigma$. 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 $x \in V$. An E-unification algorithm computes a (finite) *Complete Set of E-Unifiers* of G, denoted by $CSU_E(G)$, which is a set of substitutions such that each $\sigma \in CSU_E(G)$ is an E-unifier of G, and for each E-unifier θ of G, there exists $\sigma \in CSU_E(G)$ such that $\sigma \leq_E^{Var(G)} \theta$. Given a unifiable equation $s =^? t$, a syntactic unification algorithm computes a unique most general unifier denoted by $mgu(s, t)$. An inference rule $G \vdash G'$ for E-unification is *sound* if each E-unifier of G' is an E-unifier of G; and *complete* if for each E-unifier σ of G, there exists an E-unifier σ' of G' such that $\sigma' \leq_E^{Var(G)} \sigma$. An inference system for E-unification is *sound* (resp. *complete*) if all its inference rules are sound (resp. complete). A set of equations $G = \{x_1 =^? t_1,\ \ldots,\ x_n =^? t_n\}$ is said to be in *tree solved form* if each x_i is a variable occurring once in G. Given an idempotent substitution $\sigma = \{x_1 \mapsto t_1,\ \ldots,\ x_n \mapsto t_n\}$ (such that $\sigma\sigma = \sigma$), $\hat{\sigma}$ denotes the corresponding tree solved form. A set of equations is said to be in *dag 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. A set of equations $\{x_1 =^? t_1,\ \ldots,\ x_n =^? t_n\}$ is a *cycle* if for any $i \in [1, n-1], x_{i+1} \in Var(t_i)$, $x_1 \in Var(t_n)$, and there exists $j \in [1, n]$ such that t_j is not a variable. Given two variables x and y, $x = y$ is said to be *solved* in a set of equations G if x does not occur in $G \backslash \{x = y\}$. Then, x is said to be *solved* in G. Given two disjoint signatures Σ_1 and Σ_2 and any $i = 1, 2$, Σ_i-terms (including the variables) and Σ_i-equations (including the equations between variables) are called *i-pure*. For any $\Sigma_1 \cup \Sigma_2$-theory E, an E-unification problem is in *separate form* if it is a conjunction $G_1 \wedge G_2$, where G_i is a conjunction of Σ_i-equations for $i = 1, 2$. A term t is called a *Σ_i-rooted* term if its root symbol is in Σ_i. 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. We define *general E-unification* as the unification problem in the equational theory obtained by extending E with arbitrary free function symbols. A *term rewrite system* (TRS) is a pair (Σ, R), where Σ is a 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 $Var(r) \subseteq Var(l)$. A term s *rewrites* to a term t w.r.t R, 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 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 R, we have the existence and the uniqueness of R-normal forms, denoted by $t{\downarrow_R}$ for any term t. A substitution σ is *normalized* if, for every variable x in the domain of σ, $x\sigma$ is a normal form. A convergent TRS R is said to be *subterm convergent* if for any $l \to r \in R$, r is either a strict subterm of l or a constant. To simplify the notation, we often use tuples of terms, say $\vec{u} = (u_1, \ldots, u_n)$, $\vec{v} = (v_1, \ldots, v_n)$. Applying a substitution σ to $\vec{u}$ is the tuple $\vec{u}\sigma = (u_1\sigma, \ldots, u_n\sigma)$. The tuples $\vec{u}$ and $\vec{v}$ are said E-equal, denoted by $\vec{u} =_E \vec{v}$, if $u_1 =_E v_1, \ldots, u_n =_E v_n$. Similarly, $\vec{u} \to_R^* \vec{v}$ if $u_1 \to_R^* v_1, \ldots, u_n \to_R^* v_n$, $\vec{u}$ is R-normalized if $u_1, \ldots, u_n$ are R-normalized, and $\vec{u} =^? \vec{v}$ is $u_1 =^? v_1 \wedge \cdots \wedge u_n =^? v_n$.

3 Forward Closure

In this section, we define the central notion of finite forward closure. To define the forward closure as in [7], let us first introduce the notion of redundancy. For a given convergent TRS R, assume a reduction ordering $<$ such that $r < l$ for any $l \to r \in R$ and $<$ is total on ground terms. Since (rewrite) rules are multisets of two terms, the multiset extension of $<$ leads to an ordering on rules, also denoted by $<$, which is total on ground instances of rules. A rule ρ is *strictly redundant in R* if any ground instance $\rho\sigma$ of ρ is a logical consequence of ground instances of R that are strictly smaller w.r.t $<$ than $\rho\sigma$. A rule ρ is *redundant in R* if ρ is strictly redundant in R or ρ is an instance of some rule in R. Given two rules $\rho_1 = (g \to d)$, $\rho_2 = (l \to r)$ and a non-variable position p of d such that $d|_p$ and l are unifiable, $Fwd(\rho_1, \rho_2, p)$ denotes the rule $(g \to d[r]_p)\sigma$ where $\sigma = mgu(d|_p, l)$. Forward closure steps are inductively defined as follows:

- $FC_0(R) = NR_0(R) = R$,
- $FC_{k+1}(R) = FC_k(R) \cup NR_{k+1}(R)$ where $NR_{k+1}(R)$ is the set of rules $\rho_3 = Fwd(\rho_1, \rho_2, p)$ such that $\rho_1 \in NR_k(R)$, $\rho_2 \in R$, p is a non-variable position of the right-hand side of ρ_1, and ρ_3 is not redundant in $FC_k(R)$.

The *forward closure* of R is $FC(R) = \bigcup_{k \geq 0} FC_k(R)$. A TRS R is *forward-closed* if $FC(R) = R$. A TRS is *forward-closed convergent* if it is both forward-closed and convergent.

Example 1 *Any subterm convergent TRS has a finite forward closure. Subterm convergent TRSs are often used in the verification of security protocols [1], e.g.,* $\{dec(enc(x, y), y) \to x\}$ *and* $\{fst(pair(x, y)) \to x,\ snd(pair(x, y)) \to y\}$.

Example 2 *The following TRSs are forward-closed convergent:*

- $\{f(x) + f(y) \to f(x * y)\}$,
- $\{f(x) + y \to f(x * y)\}$,
- $\{exp(exp(a, x), y) \to exp(a, x * y)\}$,
- $\{g(h(x, y), z) \to h(x, y * z)\}$,
- $\{d(e(x, a), a) \to x * a\}$,
- $\{pdt(pair(x, y)) \to x * y\}$,
- $\{g(x, 0) \to x,\ g(0, y) \to y,\ g(s(x), s(y)) \to x * y\}$.

In all these TRSs, the function symbol $*$ occurs only in the right-hand sides of rules. In this paper, we will study the unification problem in a combination of any of these TRSs with an equational theory over $*$, such as $C = \{x * y = y * x\}$ (Commutativity) or $AC = \{x * (y * z) = (x * y) * z, x * y = y * x\}$ (Associativity-Commutativity). In such a combination, the function symbol $*$ is shared by both theories.

It has been shown in [7] that for any convergent TRS R, R has a finite forward closure if and only if R is bounded. A convergent TRS R is *bounded* if for any term t there exists some natural number n such that for any R-normalized substitution σ, $t\sigma \xrightarrow[R]{\leq n} (t\sigma)\downarrow_R$. In other words, for any normalized σ, the normal form of $t\sigma$ can be reached in less than n steps, independent of σ. We can see that the rewrite systems in Example 2 are all bounded since the normal form of $t\sigma$ can be reached using an innermost strategy in a number of steps that does not depend on σ. This is due to the fact that replacing the variables in any of the left-hand sides of the TRSs will not add new redexes that are not fully contained in the substitution itself.

It has been shown in [9] that for any convergent TRS R, R is bounded if and only if R has the finite variant property.

Definition 1 (R-variant) *Let R be a convergent TRS. An* R-variant *of a term t is a pair (u, θ) such that $u = (t\theta)\downarrow_R$, the domain of θ is included in $Var(t)$ and θ is R-normalized. Given two variants (u, θ) and (v, γ) of a term t, (u, θ) is* more general *than (v, γ), denoted by $(u, \theta) \leq (v, \gamma)$ if there exists a substitution τ such that $u\tau = v$ and $\theta\tau = \gamma$. A set $V_R(t) = \bigcup_{i\in I}\{(u_i, \theta_i)\}$ is a complete set of R-variants of t if for any R-variant (v, γ) of t, there is some $i \in I$ such that $(u_i, \theta_i) \leq (v, \gamma)$. R is said to have the* Finite Variant Property (FVP, for short) *if any term admits a finite complete set of R-variants.*

The notion of R-variant can be lifted to any R-unification problem G by considering it as a term, where $=^?$ and $\wedge$ are viewed as additional function symbols [20], and $V_R(G)$ denotes a finite complete set of R-variants of G. When R has the FVP, any R-unification problem G reduces to syntactic unification problems in $V_R(G)$. In many cases, computing the complete set of variants $V_R(G)$ can be prohibitive even with an efficient implementation of folding variant narrowing [14, 10]. For these cases, it is interesting to have an alternative to this brute force method, possibly via a rule-based R-unification procedure that does not impose a full reduction to syntactic unification.

4 Rule-Based Unification in Forward-Closed Theories

To design a rule-based unification procedure for forward-closed theories, we basically reuse the *BSM* unification procedure initially developed for the class of theories saturated by paramodulation [18], where *BSM* stands for *Basic Syntactic Mutation*. The *BSM* procedure extends syntactic unification with some additional mutation rules applied in a *don't know* non-deterministic way. These mutation rules are parameterized by a finite set of axioms corresponding to a resolvent presentation (cf. Section 2). The resulting *BSM* unification procedure is similar to the mutation-based unification procedures designed for syntactic theories [17, 22] but with the additional property of being terminating. To get termination, it makes use of boxed terms. Variables can be considered as implicitly boxed, and terms are boxed according to the following rules:

- Subterms of boxed terms are also boxed.
- Terms boxed in the premises of an inference rule remain boxed in the conclusion.
- When the "box" status of a term is not explicitly given in an inference rule, it can be either boxed or unboxed. For instance, each occurrence of f in the premise of **Imit** rule (cf. Figure 1) can be either boxed or unboxed.

Boxed terms allow us to focus on particular R-normalized solutions of a unification problem. Hence, we are interested in R-normalized solutions σ such that $t\sigma$ is R-normalized for each boxed term t occurring in the unification problem.

Definition 2 *Let G be a unification problem and σ be a substitution. We say that (G, σ) is* R-normalized *if σ is R-normalized, and for any term t in G, $t\sigma$ is R-normalized whenever t is boxed.*

Assuming a forward-closed convergent TRS R is sufficient to replay the correctness proofs of *BSM* originally stated for theories saturated by paramodulation. Thus, *BSM* can be rephrased by using directly a forward-closed convergent TRS R as input. In this setting, the equational theory of any forward-closed convergent TRS R is syntactic and a resolvent presentation is used as the parameter of *BSM* mutation rules. This leads to a *BSM* procedure providing a unification algorithm for forward-closed theories, detailed in Appendix A.

In this paper, we are also interested in solving the unification problem in the union of a forward-closed theory R_1 and a non-disjoint theory E_2. For this more general problem we develop a new and simplified mutation-based unification algorithm called BSM'. The new algorithm simplifies conflicts and therefore we need only a single mutation rule and thus a simpler mutation algorithm overall. These changes in turn allow for simpler correctness proofs as there are fewer cases to check. The single mutation rule, called **MutConflict** in Figure 1, aims at applying rewrite rules in R instead of equalities in the resolvent presentation. This restriction to R is sufficient if there is no equation between two non-variable terms. This form of equations can be easily avoided by splitting such equation $s = t$ into two equations $x = s$ and $x = t$ involving a common fresh variable x. Thanks to this additional transformation called **Split** in Figure 1, the classical decomposition rule of syntactic unification is superfluous.

All the BSM' rules are given in Figure 1. Let $\mathbf{B}'$ be the subset of BSM' that consists of rules with boxed terms, i.e., **Imit**, **MutConflict** and **ImitCycle**. BSM' rules are applied according to the following order of priority (from higher to lower): **Coalesce**, **Split** and $\mathbf{B}'$, where all $\mathbf{B}'$ rules are applied in a non-deterministic way (using a "don't know" non-determinism). The BSM' unification procedure consists in applying repeatedly the BSM' rules until reaching normal forms. The procedure then only returns those sets of equations which are in dag solved form.

Coalesce $\{x = y\} \cup G \vdash \{x = y\} \cup (G\{x \mapsto y\})$
where x and y are distinct variables occurring both in G.

Split $\{f(\vec{v}) = t\} \cup G \vdash \{x = f(\vec{v}), x = t\} \cup G$
where t is a non-variable term and x is a fresh variable.

Imit $\bigcup_i \{x = f(\vec{v}_i)\} \cup G \vdash \{x = \boxed{f(\vec{y})}\} \cup \bigcup_i \{\vec{y} = \vec{v}_i\} \cup G$
where $i > 1$, $\vec{y}$ are fresh variables and there are no more equations $x = f(\dots)$ in G.

MutConflict $\{x = f(\vec{v})\} \cup G \vdash \{x = \boxed{t}, \boxed{\vec{s}} = \vec{v}\} \cup G$
where a fresh instance $f(\vec{s}) \to t \in R$, $f(\vec{v})$ is unboxed, and (there is another equation $x = u$ in G with a non-variable term u or $x = f(\vec{v})$ occurs in a cycle).

ImitCycle $\{x = f(\vec{v})\} \cup G \vdash \{x = \boxed{f(\vec{y})}, \vec{y} = \vec{v}\} \cup G$
where $f(\vec{v})$ is unboxed, $\vec{y}$ are fresh variables and $x = f(\vec{v})$ occurs in a cycle.

Figure 1: BSM' rules

The BSM' unification can be used as an equivalent alternative to BSM. Compared to BSM, the BSM' alternative has the advantage of being easily combinable as shown in Section 6.

Theorem 1 *If R is a forward-closed convergent TRS, then the BSM' unification procedure provides an R-unification algorithm.*

Theorem 1 is subsumed by Theorem 2 that will be presented in Section 6.

5 Forward-Closed Combination

Along the lines of hierarchical combination [12], we study a form of non-disjoint combination defined as a convergent TRS R_1 combined with a base theory E_2. The TRS R_1 must satisfy some properties to ensure that $E = R_1 \cup E_2$ is a conservative extension of E_2. We focus here on cases where it is possible to reduce the E-equality between two terms into the E_2-equality of their R_1-normal forms. In addition, we assume that R_1 is forward-closed. The following definition clearly introduces the forward-closed combinations studied in the rest of the paper.

Definition 3 *Let Σ_1 and Σ_2 be two disjoint signatures. A* forward-closed combination *(FC-combination, for short) is a pair (E_1, E_2) such that*

- *E_1 is an equational $\Sigma_1 \cup \Sigma_2$-theory given by a forward-closed convergent TRS R_1 whose left-hand sides are Σ_1-terms;*
- *E_2 is a regular and collapse-free equational Σ_2-theory;*
- *for any terms s, t, we have (i) $s =_{E_1 \cup E_2} t$ iff $s\downarrow_{R_1} =_{E_2} t\downarrow_{R_1}$, and (ii) if $s =_{E_2} t$ then s is R_1-reducible iff t is R_1-reducible.*

Let us discuss the ingredients of the above definition. First of all, it is important to note that Σ_1 and Σ_2 are disjoint signatures. Thus, the TRS is a standard rewrite system defined on the signature $\Sigma_1 \cup \Sigma_2$ where Σ_2-symbols can occur only in right-hand sides. For this TRS, we do not have to rely on the notions of E_2-confluence and E_2-coherence introduced for class rewrite systems [15].

Proposition 1 *Assume Σ_1, Σ_2 and E_2 are given as in Definition 3. If E_1 is an equational $\Sigma_1 \cup \Sigma_2$-theory given by a forward-closed convergent TRS whose left-hand sides are linear Σ_1-terms, then (E_1, E_2) is an FC-combination.*

Proof. Let us first prove that $s \downarrow_{R_1} =_{E_2} t \downarrow_{R_1}$ if $s \leftrightarrow_{E_2} t$. We proceed just like in a critical pair lemma, by using the fact that the left-hand sides of R_1 are Σ_1-terms and E_2 is a Σ_2-theory with $\Sigma_1 \cap \Sigma_2 = \emptyset$. This fact is called the disjointness assumption in the rest of this proof.

Consider $s' \leftarrow^q_{R_1} s \leftrightarrow^p_{E_2} t$. By the disjointness assumption, p and q are necessarily distinct positions.

- If p and q are incomparable positions, then there exists $t \rightarrow^q_{R_1} t'$, such that $s' =_{E_2} t'$.
- If $q < p$, then there exists $t \rightarrow^q_{R_1} t'$ by the linearity of left-hand sides of R_1 and the disjointness assumption, such that $s' =_{E_2} t'$.
- If $p < q$, then there exists $t \rightarrow^q_{R_1} t'$ by the disjointness assumption and the fact that E_2 is regular and collapse-free. Note that s' and t' are not necessarily E_2-equal since E_2 can be non-linear. To retrieve an E_2-equality, we have to R_1-normalize below the position p. Let $s|_p = g\mu$ and $t = s[d\mu]_p$ for $g = d \in E_2$. Then, we have

$$s[(s|_p) \downarrow_{R_1}]_p = s[g(\mu \downarrow_{R_1})]_p \leftrightarrow_{E_2} s[d(\mu \downarrow_{R_1})]_p = t[(t|_p) \downarrow_{R_1}]_p$$

From previous cases, it follows that $s \downarrow_{R_1} =_{E_2} t \downarrow_{R_1}$ if $s \leftrightarrow_{E_2} t$. Since $s \downarrow_{R_1} = t \downarrow_{R_1}$ if $s \leftrightarrow_{E_1} t$, we get the properties defining an FC-combination. □

Example 3 *Consider R_1 as any TRS mentioned in Example 2 and $\Sigma_2 = \{*\}$. An FC-combination is defined by R_1 together with any regular and collapse-free Σ_2-theory E_2, such as C or AC.*

From now on, we assume $E = E_1 \cup E_2$ and (E_1, E_2) is an FC-combination given by a forward-closed convergent TRS R_1.

As shown below, a syntacticness property also holds for a restricted form of E-equalities.

Lemma 1 *For each equality $u =_E v$ such that u is Σ_1-rooted and v is R_1-normalized, one of the following is true:*

1. *$u = f(\vec{u})$, $v = f(\vec{v})$ and $\vec{u} =_E \vec{v}$.*
2. *$u = f(\vec{u})$, there exist $f(\vec{s}) \rightarrow t \in R_1$ and a R_1-normalized substitution σ such that $\vec{u} =_E \vec{s}\sigma$, $v =_{E_2} t\sigma$ and $\vec{s}\sigma, t\sigma$ are R_1-normalized.*

Proof. Let us analyze the possible rewrite proofs $\rightarrow^*_{R_1} \circ =_{E_2}$ of $u =_E v$.

- There is no step at the root position. Then we get $u = f(\vec{u}) \rightarrow^*_{R_1} f(\vec{u'}) =_{E_2} v$ where $\vec{u} \rightarrow^*_{R_1} \vec{u'}$ and $\vec{u'}$ are R_1-normalized. Since f is a free symbol for E_2, we have that $v = f(\vec{v})$ and $\vec{u'} =_{E_2} \vec{v}$. Hence, $\vec{u} =_E \vec{v}$ since $\vec{u} =_E \vec{u'}$.
- There is one step at the root position. Then, we have:

$$u = f(\vec{u}) \rightarrow^*_{R_1} f(\vec{u'}) = f(\vec{s})\sigma \rightarrow_{R_1,\epsilon} t\sigma =_{E_2} v$$

  where $f(\vec{s}) \rightarrow t \in R_1$, $\vec{u} \rightarrow^*_{R_1} \vec{u'}$, $\vec{u'}$ are R_1-normalized, $\vec{u'} = \vec{s}\sigma$, and so $\sigma, \vec{s}\sigma$ are R_1-normalized. Since $t\sigma =_{E_2} v$, and v is R_1-normalized, we have $t\sigma$ is R_1-normalized. □

6 Unification in Forward-Closed Combinations

We now study how the BSM' unification procedure can be combined with an E_2-unification algorithm to solve the unification problem in $E = E_1 \cup E_2$.

VA $\{s = t[u]\} \cup G \vdash \{s = t[x], x = u\} \cup G$
where u is an alien subterm of t, x is a fresh variable, and u is boxed iff $t[u]$ is boxed.

Solve $G_1 \wedge G_2 \vdash \bigvee_{\sigma_2 \in CSU_{E_2}(G_2)} G_1 \wedge \hat{\sigma}_2$
if $G_1 \wedge G_2$ is a separate form, G_2 is E_2-unifiable and not in tree solved form, where w.l.o.g $\forall \sigma_2 \in CSU_{E_2}(G_2)\ \forall x \in Dom(\sigma_2)$, ($x\sigma_2$ is a variable) $\Rightarrow x\sigma_2 \in Var(G_2)$.

Figure 2: Additional Rules for the combination with E_2

Consider the inference system for Basic Syntactic Combination, say BSC, given by **Coalesce**, **Split** and $\mathbf{B}'$ rules defining BSM' in Section 4, where f is now supposed to be a function symbol in Σ_1; plus the two additional rules given in Figure 2, namely **VA** and **Solve**. The rule **VA** applies the classical Variable Abstraction transformation [23, 3, 11] to purify terms and so to get a separate form, while **Solve** calls an E_2-unification algorithm to solve the set of Σ_2-equations in a separate form. The repeated application of rules in $\{\mathbf{Coalesce}, \mathbf{Split}, \mathbf{VA}, \mathbf{Solve}\}$ computes particular separate forms defined as follows.

Definition 4 *A separate form $G_1 \wedge G_2$ is* mutable *if **Coalesce** does not apply on $G_1 \wedge G_2$, G_1 is a set of Σ_1-equations $x = t$ (where x is a variable), and G_2 is a set of Σ_2-equations in solved form. A* compound solved form *is a mutable separate form in dag solved form.*

Example 4 *From Example 2, let $R_1 = \{exp(exp(a, x), y) \to exp(a, x * y)\}$ and let E_2 be the AC theory for $*$. Now consider the unification problem $G = \{exp(x_1, x_2) = exp(a, x_2 * x_3)\}$. After applying the rules **VA** and **Split** we obtain the* mutable separate form, *$G = G_1 \wedge G_2$, where $G_1 = \{z_2 = exp(x_1, x_2),\ z_2 = exp(a, z_1)\}$ and $G_2 = \{z_1 = x_2 * x_3\}$. However, this is not a* compound solved form *since it is not in dag solved form. Notice that rules such as **Imit** and **MutConflict** can still be applied.*

BSC rules are applied according to the following order of priority (from higher to lower): **Coalesce**, **Split**, **VA**, **Solve**, and $\mathbf{B}'$ where **Solve** computes each solution of the subproblem G_2 in a separate form $G_1 \wedge G_2$ and $\mathbf{B}'$ rules are applied on a separate form $G_1 \wedge G_2$ in a non-deterministic way as in Section 4. Due to the order of priority, **Solve** applies only if **Coalesce**, **Split**, **VA** are not applicable and $\mathbf{B}'$ rules apply only if $G_1 \wedge G_2$ is a mutable separate form. Notice that any compound solved form is in normal form w.r.t BSC.

Lemma 2 *Given an E-unification problem G as input, the repeated application of BSC rules always terminates.*

Proof. The E_2-unification algorithm used in **Solve** may generate as usual some fresh variables, but when it produces an equation between variables, these variables are not fresh. Hence, **Coalesce** only applies on variables that are unsolved and not generated by E_2-unification; and variables generated by E_2-unification will remain in G_2. By denoting p the number of variables that are unsolved and not generated by E_2-unification, we have that p strictly decreases by **Coalesce** and p is not increased by **Solve**.

Given a goal G, we use a complexity measure similar to the one introduced in [18] for BSM, defined by the following tuple of natural numbers:

- m is the number of (occurrences of) unboxed Σ_1-symbols in G,
- n is the number of (occurrences of) Σ_1-symbols in G,
- i_1 is the sum of sizes of impure terms in G,
- i_2 is the number of equations $f(\vec{v}) = t$ in G such that $f \in \Sigma_1$ and t is a non-variable term,
- p is the number of variables in G that are unsolved and not generated by E_2-unification,
- $q \in \{0, 1\}$ such that $q = 0$ iff G is a separate form $G_1 \wedge G_2$ and G_2 is in solved form.

By using the corresponding lexicographic ordering, this complexity measure decreases with the application of each rule.

Rule	m	n	i_1	i_2	p	q
Imit	$\geq$	$>$				
MutConflict	$>$					
ImitCycle	$>$					
Coalesce	$\geq$	$\geq$	$\geq$	$\geq$	$>$	
VA	$\geq$	$\geq$	$>$			
Split	$\geq$	$\geq$	$\geq$	$>$		
Solve	$\geq$	$\geq$	$\geq$	$\geq$	$\geq$	$>$

□

Given an E-unification problem G, $\texttt{BSC}(G)$ denotes the normal forms of G w.r.t BSC, which correspond to the compound solved forms of G. Following Lemma 2, the BSC unification procedure works as follows: apply the BSC rules on a given E-unification problem G until reaching normal forms, and return all the dag solved forms in $\texttt{BSC}(G)$. The completeness of the BSC unification procedure relies on the following lemmas.

First, we state that an E_2-unification algorithm can be reused without loss of completeness to E-unify any conjunction of Σ_2-equations. This is a classical result, already used in hierarchical combination [12], which can be easily lifted to FC-combination:

Lemma 3 *Assume **Solve** is applied on an E-unification problem G, leading to a disjunction $Sol(G)$ of E-unification problems. For any substitution σ, if $E \models G\sigma$ then there exist some G' in $Sol(G)$ and a substitution σ' such that $E \models G'\sigma'$ and $\sigma'_{|Var(G)} = \sigma$.*

Below, we show that BSC rules are applied without loss of completeness. We denote by $G \xrightarrow[BSC]{} G'$ an application of a BSC rule to a unification problem G producing a modified problem G'.

Lemma 4 *If (G, σ) is R_1-normalized, $E \models G\sigma$ and G is not a compound solved form, then there exist some G' and a substitution σ' such that $G \xrightarrow[BSC]{} G'$, (G', σ') is R_1-normalized, $E \models G'\sigma'$ and $\sigma' \leq_E^{Var(G)} \sigma$.*

Proof. Let us consider all the cases where G is not a compound solved form.

(A) G is not a mutable separate form. Then, some rule among **Coalesce**, **VA**, **Split**, **Solve** applies, and the E-unifiers of G' restricted to $Var(G)$ are the E-unifiers of G. For **Coalesce**, we define σ' such that $\sigma' = \sigma$. For **VA**, $\sigma'_{|Var(G)} = \sigma$ and $x\sigma' = (u\sigma) \downarrow_{R_1}$ where x is the fresh variable. For **Split**, $\sigma'_{|Var(G)} = \sigma$ and $x\sigma' = (t\sigma) \downarrow_{R_1}$ where x is the fresh variable.

Assume **Solve** applies. By Lemma 3, if $E \models G\sigma$ then there exist some G' generated by **Solve** and a substitution σ' such that $E \models G'\sigma'$ and $\sigma'_{|Var(G)} = \sigma$. The substitution σ' can be considered as R_1-normalized. Since **Solve** does not introduce new boxed terms, (G', σ') is R_1-normalized.

(B) G is a mutable separate form which is not in dag solved form. Several subcases must be considered.

(i) Assume G contains a subset $G_x = \{x = v \in G \mid v \text{ is non-variable }\}$ such that $|G_x| > 1$. According to Lemma 1, there are two possibilities.

- For any equation $x = v \in G_x$, $x\sigma =_E v\sigma$, $x\sigma = f(\vec{w})$, $v = f(\vec{v})$ and $\vec{w} =_E \vec{v}\sigma$. Then **Imit** applies to get G'. The substitution σ' is defined to be equal to σ on $Var(G)$, and $\vec{y}\sigma' = \vec{w}$. Since the term $f(\vec{y})$ is boxed, we have that (G', σ') is R_1-normalized.
- there are some equations $x = f(\vec{u}), x = v \in G_x$, and a rule $f(\vec{s}) \to t \in R_1$, where x is a variable, such that $x\sigma =_E f(\vec{u})\sigma =_E v\sigma$, $\vec{u}\sigma =_E \vec{s}\sigma'$, and $t\sigma' =_E x\sigma$ for a R_1-normalized substitution σ'. Thus **MutConflict** applies to get G'. The substitution σ' can be defined equal to σ on $Var(G)$. Hence, σ' is a R_1-normalized solution of G', and (G', σ') is R_1-normalized since $\vec{s}\sigma'$ and $t\sigma'$ are R_1-normalized by Lemma 1.

(ii) Assume G contains a cycle. Since σ is an E-unifier of G, there is necessarily some Σ_1-equation $x = f(\vec{v})$, such that $f(\vec{v})\sigma$ is R_1-reducible. According to Lemma 1, there are two possibilities:

- $x\sigma = f(\vec{w}) =_E f(\vec{v})\sigma$ and $\vec{w} =_E \vec{v}\sigma$. Then **ImitCycle** applies to get G'. The substitution σ' is defined to be equal to σ on $Var(G)$, and $\vec{y}\sigma' = \vec{w}$. Since the term $f(\vec{y})$ is boxed, we have that (G', σ') is R_1-normalized.
- There exist a rule $f(\vec{s}) \to t \in R_1$ and a R_1-normalized substitution σ' such that $f(\vec{v})\sigma =_E x\sigma =_E t\sigma'$ and $\vec{s}\sigma' =_E \vec{v}\sigma$. Then **MutConflict** applies to get G'. The substitution σ' can be defined equal to σ on $Var(G)$. In that case, σ' is an R_1-normalized solution of G'. By Lemma 1, $\vec{s}\sigma'$ and $t\sigma'$ are R_1-normalized, and so (G', σ') is R_1-normalized. □

Hence *BSC* leads to a terminating and complete E-unification procedure.

Theorem 2 *Given any FC-combination (E_1, E_2) and an E_2-unification algorithm, BSC provides an $E_1 \cup E_2$-unification algorithm.*

According to Definition 3, an FC-combination can be obtained by considering an arbitrary forward-closed convergent TRS R_1 and the empty theory E_2 over the empty signature Σ_2. In that particular case, *BSC* reduces to *BSM′* and so the fact that *BSC* is both terminating and correct provides a proof for Theorem 1.

Example 5 *Assume f, g, a are in Σ_1 and E_2 is the C theory for $*$. Consider the separate form $G = \{x = f(y), x = z * y\}$ and the following possible cases:*

- *If $R_1 = \{f(v) \to a\}$, then **MutConflict** can be applied on G and we get $\{x = \boxed{a}, x = z*y\}$, which is in normal form w.r.t BSC but not in dag solved form. So, it has no solution.*

- *If* $R_1 = \{f(v) \to v * a\}$, *then* ***MutConflict*** *can be applied on* G *and we obtain* $\{x = \boxed{y * a}, x = z * y\}$. *Then* ***Solve*** *leads to the solved form* $\{x = y * a, z = a\}$.
- *If* $R_1 = \{g(v) \to v * a\}$, *then* G *is in normal form w.r.t BSC but not in dag solved form. So, it has no solution.*

Example 6 *(Example 4 continued).*
Consider the problem $G = \{exp(x_1, x_2) = exp(a, x_2 * x_3)\}$ *and a run of BSC on* G. *Applying* ***VA*** *and* ***Split*** *leads to the mutable separate form* $\{z_2 = exp(x_1, x_2),\ z_2 = exp(a, z_1),\ z_1 = x_2 * x_3\}$. *At this point one possibility is to apply* ***MutConflict*** *(introducing* z_3, z_4*) followed by* ***Coalesce*** *(replacing* z_4 *by* x_2*), leading to* $\{z_2 = \boxed{exp(a, z_3 * x_2)},\ x_1 = \boxed{exp(a, z_3)},\ x_2 = z_4,\ z_2 = exp(a, z_1),\ z_1 = x_2 * x_3\}$ *Then* ***VA*** *applies, leading to* $\{z_2 = \boxed{exp(a, z_5)},\ x_1 = \boxed{exp(a, z_3)},\ x_2 = z_4,\ z_2 = exp(a, z_1),\ z_1 = x_2 * x_3,\ z_5 = \boxed{z_3 * x_2}\}$, *By applying* ***Imit*** *(introducing* z_6, z_7*) followed by* ***Coalesce*** *(replacing* z_5, z_7 *by* z_1*), we obtain* $\{z_2 = \boxed{exp(z_6, z_1)},\ z_6 = a,\ z_7 = z_5,\ z_7 = z_1,\ x_1 = \boxed{exp(a, z_3)},\ x_2 = z_4,\ z_1 = x_2 * x_3,\ z_1 = \boxed{z_3 * x_2}\}$ *At this point an AC-unification algorithm can be used to solve* $\{z_1 = x_2 * x_3,\ z_1 = z_3 * x_2\}$. *The AC-unifier* $\{z_3 \mapsto x_3\}$ *leads to an expected solution of* G, *which is* $\{x_1 \mapsto exp(a, x_3)\}$. *A detailed view of this example can be found in Appendix B.*

BSC is also useful to the general E-unification problem. Actually, the E-unification algorithm given by *BSC* can be lifted to a general E-unification algorithm by considering that E_2 may contain the free function symbols. In that case, a general E_2-unification algorithm can be obtained from an E_2-unification algorithm by applying the disjoint combination algorithm known for regular and collapse-free theories [25].

Theorem 3 *Given any FC-combination* (E_1, E_2) *and an* E_2*-unification algorithm, it is possible to construct a general* $E_1 \cup E_2$*-unification algorithm.*

7 Reduction Methods

We demonstrate two alternative solutions to the unification problem in forward-closed combinations. Both methods are brute force in nature with the first requiring the computation of a complete set of variants, and the second relying completely on brute force non-determinism. We conclude the section by comparing the approaches to *BSC*. As previously, we assume an FC-combination (E_1, E_2) given by a forward-closed TRS R_1 and $E = E_1 \cup E_2$.

7.1 Variant-Based Reduction

The first method is a variant based approach that reduces any E-unification problem G into some general E_2-unification problems corresponding to the finite set $V_{R_1}(G)$ of R_1-variants of G (cf. Definition 1).

Proposition 2 *Let* $E = E_1 \cup E_2$ *such that* (E_1, E_2) *is an FC-combination given by a forward-closed convergent TRS* R_1. *For any* E*-unification problem* G, *the set of substitutions* $\{\sigma'\mu \mid (G', \sigma') \in V_{R_1}(G), \mu \in CSU_{E_2}(G')\}$ *is a* $CSU_E(G)$.

Proof. Soundness being easy to prove, we focus on completeness. Assume σ is any R_1-normalized E-unifier of G. By Definition 3, $E \models G\sigma$ iff $E_2 \models (G\sigma)\downarrow_{R_1}$. Since R_1 has the FVP, we have $E_2 \models$

$G'\tau$ where $G' = (G\sigma') \downarrow_{R_1}$, $(G', \sigma') \in V_{R_1}(G)$, and $\sigma = \sigma'\tau$. Consider $Dom(\sigma') = \{x \mid x\sigma' \neq x\}$ and $VRan(\sigma') = \bigcup_{x \in Dom(\sigma')} Var(x\sigma')$. By definition, $Dom(\sigma') \subseteq Var(G)$.

Since τ is an E_2-unifier of G', there exists some $\mu \in CSU_{E_2}(G')$ such that $\mu \leq_E^{Var(G')} \tau$. The substitution μ is such that $x\mu = x$ for any variable x in $Var(G\sigma')\backslash Var(G')$. Therefore, there exists a substitution ρ such that $x\tau =_E x\mu\rho$ for any $x \in Var(G\sigma')$. By definition of σ', $Var(G\sigma')$ is equal to $VRan(\sigma') \cup (Var(G)\backslash Dom(\sigma'))$. Then, we have for any $x \in Dom(\sigma')$, $x\sigma = x\sigma'\tau =_E x\sigma'\mu\rho$ and for any $x \in Var(G)\backslash Dom(\sigma')$, $x\sigma = x\sigma'\tau = x\tau =_E x\mu\rho = x\sigma'\mu\rho$. In other words, $\sigma'\mu \leq_E^{Var(G)} \sigma$. □

7.2 Mutation-Based Reduction

Another possible reduction method is based on the application of a non-deterministic mutation rule similar to the ones used in BSM'. However, this can be considered as another brute force method where the mutation rule is applied in a completely non-deterministic way and not just applied in the case of conflict as in BSM'.

A *flat* E-unification problem is a set of equalities $\bigcup_{k\in K}\{x_k =^? t_k\}$ such that, for any $k \in K$, x_k is a variable and t_k is either flat or boxed. A flat E-unification problem $\bigcup_{k\in K}\{x_k =^? t_k\}$ is *fully boxed* if, for each $k \in K$, t_k is either a boxed term or a variable.

Lemma 5 *Let G be any fully boxed flat E-unification problem. Then, any $CSU_{E_2}(G)$ is a $CSU_E(G)$.*

Proof. Assume without loss of generality that σ is an R_1-normalized E-unifier of $G = \bigcup_{k\in K}\{x_k =^? t_k\}$. For each $k \in K$, $x_k\sigma =_E t_k\sigma$ iff $x_k\sigma =_{E_2} t_k\sigma$ by Definition3 and the fact that $x_k\sigma$ and $t_k\sigma$ are R_1-normalized due to the assumption on G. Thus, $E \models G\sigma$ iff $E_2 \models G\sigma$. □

Consider the inference system MI defined in Figure 3 by two rules: a mutation rule **M** and an imitation rule **I**. Both rules are assumed to be applied in a non-deterministic way. The inference system MI is terminating since the number of unboxed terms is strictly decreasing by each rule application.

M $\{x = f(\vec{y})\} \cup G \vdash \{x = \boxed{t}, \vec{y} = \boxed{\vec{s}}\} \cup G$
where $f(\vec{y})$ is unboxed, $f(\vec{s}) \to t \in R_1$.

I $\{x = f(\vec{y})\} \cup G \vdash \{x = \boxed{f(\vec{y})}\} \cup G$
where $f(\vec{y})$ is unboxed.

Figure 3: MI rules

Lemma 6 *If (G, σ) is R_1-normalized, $E \models G\sigma$, G is flat and not fully boxed, then there exist some G' and a substitution σ' such that $G \underset{MI}{\longrightarrow} G'$, G' is flat, (G', σ') is R_1-normalized, $E \models G'\sigma'$ and $\sigma' \leq_E^{Var(G)} \sigma$.*

Proof. If G is not fully boxed, then there exists some $x = f(\vec{y}) \in G$ such that $f(\vec{y})$ is unboxed, $x\sigma =_E f(\vec{y})\sigma$, and $x\sigma, \vec{y}\sigma$ are R_1-normalized.

Then two cases can appear:

- if $f(\vec{y})\sigma$ is R_1-normalized, then the rule **I** applies.

- Otherwise, $f(\vec{y})\sigma$ is R_1-reducible, and so there exists a rule $f(\vec{s}) \to t \in R_1$ such that $\vec{y}\sigma = \vec{s}\delta$, for some substitution δ. The terms $\vec{s}\delta$ are R_1-normalized since $\vec{y}\sigma$ are R_1-normalized. Moreover, $t\delta$ is R_1-normalized because R_1 is forward-closed. Thus, the rule **M** applies to get G'. If we extend σ to σ' by the new variables in $\vec{s}$ according to δ, then $\sigma' \leq_E^{Var(G)} \sigma$ and σ' is a unifier of G'. □

Given any input flat E-unification problem G, $\mathtt{MI(G)}$ denotes the normal forms of G w.r.t MI. All the E-unification problems in $\mathtt{MI(G)}$ are flat and fully boxed.

Proposition 3 *Let $E = E_1 \cup E_2$ such that (E_1, E_2) is an FC-combination given by a forward-closed TRS R_1. For any flat E-unification problem G, the set of substitutions $\bigcup_{G' \in \mathtt{MI}(G)} CSU_{E_2}(G')$ is a $CSU_E(G)$.*

The inference system MI is very similar to the unification algorithm described in [21] for optimally reducing TRSs. This is not surprising due to the connection between optimally reducing TRSs and the FVP.

7.3 Reduction versus Combination

Proposition 2 leads to a non-deterministic reduction method based on the computation of a complete set of variants. As shown by Proposition 3, another possible non-deterministic reduction method can be designed using a mutation-based approach, and so without relying on the notion of variant. From our point of view, it is interesting to have a simple alternative to these brute force reduction methods, possibly via a rule-based E-unification procedure that does not impose a full reduction to general E_2-unification. Actually, the BSC procedure described in Section 6 provides this rule-based alternative. It includes two sources of non-determinism, one for R_1 via the mutation/imitation rules and another one for E_2 via the **Solve** rule. These sources of non-determinism cannot be avoided without loss of completeness. Compared to the variant-based reduction method, one advantage of BSC is its simplicity: mutation/imitation rules are easy to understand and to implement. On the other hand, the variants and the folding variant narrowing can be considered as a complicated machinery.

Proposition 2 and Proposition 3 correspond to reduction methods that aim at computing general E_2-unification problems, where symbols in Σ_1 are considered as free symbols. These general E_2-unification problems can be naturally solved by applying the disjoint combination algorithm known for regular and collapse-free theories [25]. The BSC procedure can be viewed as the result of the integration of a mutation mechanism into this simple disjoint combination algorithm.

8 Implementation

When choosing to implement the algorithms developed in this paper, we have selected the Maude programming language[1]. Maude provides a nice environment for a number of reasons. First, it provides a more natural environment for expressing the rules of algorithms such as BSM'. Second, it has both variant generation and several unification algorithms, such as AC, built-in. Indeed, having both the variant-based unification and the rule-based unification developed here implemented in Maude is the best way to compare them in practice. In addition, having both approaches implemented offers alternatives for selecting the most suitable method for an

[1] `http://maude.cs.illinois.edu/w/index.php/The_Maude_System`

application (for example, in cases when the number of variants is high [24]). One can now easily switch between the most appropriate approach for their situation.

Implementation of the above procedures is ongoing[2]. Currently, the focus of the implementation is on the BSM' algorithm which in itself provides a new alternative method for solving the unification problem in forward closed theories. Significantly, once the forward closure of a system is computed, the implementation of BSM' provides a unification procedure for any problem in the theory. In other words, the computation of a forward closure can be reused for any unification problem for that theory. The implementation also takes advantage of the flexibility of Maude, allowing the rules of the BSM' procedure to be instantiated by a theory input to the algorithm via a Maude-module. This will also make the program easier to incorporate into a larger tools.

After the BSM' implementation the focus will be on the combination and experimentation. Due to the importance of AC in practical applications, we plan to focus on the case of forward-closed combinations with AC-symbols, for which it is possible to reuse the AC-unification algorithm implemented in Maude in a way similar to [11].

9 Conclusion

In this paper we develop a rule-based unification algorithm which can be easily combined, even for some non-disjoint unions of theories, and does not require the computation of variants. By applying this rule-based unification algorithm, we present, in addition, a new non-disjoint, terminating combination procedure for a base theory extended with a non-disjoint forward-closed TRS. The new combination allows for the addition of such often used theories as AC and C.

Until now, we assume that the TRS is defined in a simple way, by using syntactic matching for the rule application. A possible extension would be to consider an equational TRS defined modulo the base theory, where equational matching is required for the rule application. Considering an equational TRS instead of a classical one, two natural problems arise. First, the possible equivalence between the finite forward closure and the FVP is an open problem when the TRS is equational. Second, another problem is to highlight a combination algorithm for solving unification problems modulo an equational TRS having the FVP.

References

[1] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006.

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

[3] 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.

[4] 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.

[5] D. A. Basin, S. Mödersheim, and L. Viganò. An on-the-fly model-checker for security protocol analysis. In E. Snekkenes and D. Gollmann, editors, *Computer Security - ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October*

[2] `https://github.com/ajayeeralla/BSM`

13-15, 2003, Proceedings, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[6] B. Blanchet. Modeling and verifying security protocols with the Applied Pi calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016.

[7] C. Bouchard, K. A. Gero, C. Lynch, and P. Narendran. On forward closure and the finite variant property. In P. Fontaine, C. Ringeissen, and R. A. Schmidt, editors, *Frontiers of Combining Systems - 9th International Symposium, FroCoS 2013, Nancy, France, September 18-20, 2013. Proceedings*, volume 8152 of *Lecture Notes in Computer Science*, pages 327–342. Springer, 2013.

[8] S. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. *J. Autom. Reasoning*, 48(2):219–262, 2012.

[9] H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. In J. Giesl, editor, *Rewriting Techniques and Applications*, volume 3467 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

[10] F. Durán, S. Eker, S. Escobar, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. Built-in variant generation and unification, and their applications in Maude 2.7. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, volume 9706 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2016.

[11] A. K. Eeralla and C. Lynch. Bounded ACh Unification. *CoRR*, abs/1811.05602, 2018.

[12] S. Erbatur, D. Kapur, A. M. Marshall, P. Narendran, and C. Ringeissen. Hierarchical combination. In M. P. Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, volume 7898 of *Lecture Notes in Computer Science*, pages 249–266. Springer, 2013.

[13] S. Escobar, C. A. Meadows, and J. Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In A. Aldini, G. Barthe, and R. Gorrieri, editors, *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2007.

[14] S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. *J. Log. Algebr. Program.*, 81(7-8):898–928, 2012.

[15] J. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986.

[16] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In *Computational Logic - Essays in Honor of Alan Robinson*, pages 257–321, 1991.

[17] 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.

[18] 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.

[19] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

[20] J. Meseguer. Variant-based satisfiability in initial algebras. *Sci. Comput. Program.*, 154:3–41, 2018.

[21] P. Narendran, F. Pfenning, and R. Statman. On the unification problem for cartesian closed categories. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 57–63. IEEE Computer Society Press, 1993.

[22] 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.

[23] M. Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8:51–99, July 1989.

[24] F. Yang, S. Escobar, C. A. Meadows, J. Meseguer, and P. Narendran. Theories of homomorphic encryption, unification, and the finite variant property. In O. Chitil, A. King, and O. Danvy, editors, *Proceedings of the 16th International Symposium on Principles and Practice of Declarative Programming, Kent, Canterbury, United Kingdom, September 8-10, 2014*, pages 123–133. ACM, 2014.

[25] K. A. Yelick. Unification in combinations of collapse-free regular theories. *Journal of Symbolic Computation*, 3(1-2):153–181, 1987.

A Basic Syntactic Mutation

The material contained in this appendix is not required for understanding or constructing the procedures in the main body of the paper. However, for completeness we include it here.

We present a *BSM* unification procedure adapted from [18]. This mutation-based unification procedure is terminating, sound and complete for the class of forward-closed theories. The correctness proofs detailed below follow the ones developed in [18].

Dec $\{f(\vec{u}) = f(\vec{v})\} \cup G \ \vdash \ \{\vec{u} = \vec{v}\} \cup G$

Mut $\{f(\vec{u}) = g(\vec{v})\} \cup G \ \vdash \ \{\vec{u} = \boxed{\vec{s}}, \boxed{\vec{t}} = \vec{v}\} \cup G$
where $f(\vec{u})$ is unboxed and $f(\vec{s}) = g(\vec{t}) \in S$.

Imit $\bigcup_i \{x = f(\vec{v}_i)\} \cup G \ \vdash \ \{x = \boxed{f(\vec{y})}\} \cup \bigcup_i \{\vec{y} = \vec{v}_i\} \cup G$
where $i > 1$ and there are no more equations $x = f(\dots)$ in G.

MutImit
$\{x = f(\vec{u}), x = g(\vec{v})\} \cup G \ \vdash$
$\{x = f(\vec{y}), \vec{y} = \boxed{\vec{s}}, \boxed{\vec{s}} = \vec{u}, \boxed{\vec{t}} = \vec{v}\} \cup G$
where $f(\vec{s}) = g(\vec{t}) \in S$, and

1. if $f(\vec{u})$ is boxed, $g(\vec{v})$ is unboxed, then $f(\vec{y})$ is boxed;
2. if $f(\vec{u})$ and $g(\vec{v})$ are unboxed, then $f(\vec{y})$ is unboxed.

Coalesce $\{x = y\} \cup G \ \vdash \ \{x = y\} \cup (G\{x \mapsto y\})$
where x and y are distinct variables occurring both in G.

VarMut $\{f(\vec{u}) = v\} \cup G \ \vdash \ \{\vec{u} = \boxed{\vec{s}}, y = v\} \cup G$
where $f(\vec{u})$ is unboxed, $f(\vec{s}) = y \in S$, y is a variable, and if v is a variable, then there is another equation $v = t \in G$ with a non-variable term t, or $v = f(\vec{u})$ occurs in a cycle.

ImitCycle $\{x = f(\vec{v})\} \cup G \ \vdash \ \{x = \boxed{f(\vec{y})}, \vec{y} = \vec{v}\} \cup G$
where $f(\vec{v})$ is unboxed and $x = f(\vec{v})$ occurs in a cycle.

MutImitCycle $\{x = f(\vec{v})\} \cup G \ \vdash \ \{x = \boxed{g(\vec{t})}, \boxed{\vec{s}} = \vec{v}\} \cup G$
where $f(\vec{v})$ is unboxed, $f(\vec{s}) = g(\vec{t}) \in S$, and $x = f(\vec{v})$ occurs in a cycle.

Figure 4: *BSM* rules

Assume R is a forward-closed convergent TRS. The resolvent presentation of R is given by the finite set $S = RHS(R)$ of right-hand side critical pairs [18] defined as follows: $RHS(R) = \{l = r \mid l \to r \in R\} \cup \{l\sigma = g\sigma \mid l \to r \in R, g \to d \in R, \sigma \in mgu(r, d), l\sigma \neq g\sigma\}$. All the *BSM* rules are given in Figure 4. Let $\mathbf{B}$ be the subset of *BSM* rules with boxed terms: $\mathbf{B} = \{\mathbf{Mut}, \mathbf{Imit}, \mathbf{MutImit}, \mathbf{VarMut}, \mathbf{ImitCycle}, \mathbf{MutImitCycle}\}$. Given a unification problem G, the *BSM* unification procedure consists in applying repeatedly the *BSM* rules until reaching normal forms. Rules are applied according to the following order of priority (from higher to lower): **Coalesce**, **Dec** and $\mathbf{B}$, where $\mathbf{B}$ rules are applied in a non-deterministic way (using a "don't know" non-determinism). The procedure then only returns those sets of equations which are in dag solved form.

Lemma 7 *Let $S = RHS(R)$ where R is a convergent forward-closed rewrite system. Assume that $u =_S v$. Then, one of the following is true:*

1. *$u = f(\vec{u})$, $v = f(\vec{v})$ and $\vec{u} =_S \vec{v}$.*
2. *$u = f(\vec{u})$, $v = g(\vec{v})$, there exists $f(\vec{s}) = t \in S$ and a R-normalized substitution σ such that $\vec{u} =_S \vec{s}\sigma$; $g(\vec{v}) =_S t\sigma$; if $t = g(\vec{t})$ then $\vec{v} =_S \vec{t}\sigma$; and $\vec{t}\sigma$, $\vec{s}\sigma$ are R-normalized.*

Proof. It follows the same approach to the proof of Theorem 2 in [18] but modified to account for the new use of a forward-closed rewrite system.

Without loss of generality we can consider an innermost R-rewrite proof of $u =_S v$. Since R is forward-closed, for any term there exists an innermost rewrite proof to its normal form with at most one root reduction, which additionally is the final rewrite step. We consider the cases the root reduction can take:

1. No reduction at the root of either side:

   $$u = f(\vec{u}) \to_R^* f(\vec{u}') \leftarrow_R^* f(\vec{v}) = v$$

   Then $\vec{u} \to_R^* \vec{u}' \leftarrow_R^* \vec{v}$ and so $\vec{u} =_S \vec{v}$.

2. One root reduction on one side. Assume, without loss of generality, that the root reduction occurs on the left side, $u = f(\vec{u}) \to_R^* u'$. Then, $u' = g(\vec{v}')$ and there exists a rule in R such that $f(\vec{u}') \to g(\vec{v}')$ s.t $\vec{u}'$ is R-normalized. Two cases are possible:

   (a) There is an equality $f(\vec{s}) = g(\vec{t})$ in S. Consider the matching substitution σ from the rewrite step. We can assume σ is R-normalized. Then: $\vec{s}\sigma = \vec{u}'$; $\vec{t}\sigma = \vec{v}'$; $\vec{u} =_S \vec{s}\sigma$ from the left side of the proof; $g(\vec{v}) =_S g(\vec{t})\sigma$ from the right side of the proof; and $\vec{v} =_S \vec{t}\sigma$ from the final decomposition step.

   (b) There is an equality $f(\vec{s}) = x \in S$ for a variable x such that $\vec{s}\sigma = \vec{u}'$ and $x\sigma = g(\vec{v}')$. Then again, if $\vec{u} \to_R^* \vec{u}'$ then $\vec{u} =_S \vec{u}'$. In addition, if on the right hand side of the proof $g(\vec{v}) \to_R g(\vec{v}')$, then $g(\vec{v}) =_S x\sigma$.

3. There is a root reduction on both sides:

   $$u = f(\vec{u}) \to_R^* w \leftarrow_R^* g(\vec{v}) = v$$

   We can assume that the terms on the left and right, before the root rewrites, are not equal otherwise we reduce to case (1).

   Now consider the two root rewrite steps: $f(\vec{u}') \to_R w$ and $g(\vec{v}') \to_R w$. Therefore there must be equalities in S of the form $f(\vec{s}) = t$ and $g(\vec{t}) = t'$ and an R-normalized substitution, σ, such that $\vec{s}\sigma = \vec{u}'$ and $\vec{t}\sigma = \vec{v}'$. Furthermore, $t\sigma = w$ and $t'\sigma = w$. This implies there are two rules in R, $l \to t$ and $g \to t'$, such that $t\sigma = t'\sigma$, i.e., they satisfy the definition of $RHS(R)$. Therefore, there exists a substitution $\theta \in mgu(t, t')$ s.t. $\theta \leq \sigma$ and $f(\vec{s})\theta = g(\vec{t})\theta$ is in S. Finally, since $\theta \in mgu(t, t')$, there exists a substitution γ such that $\vec{s}\theta\gamma = \vec{s}\sigma$ and $\vec{t}\theta\gamma = \vec{t}\sigma$. □

When R is collapse-free, the second case of Lemma 7 can be simplified since the equality in S is necessarily of the form $f(\vec{s}) = g(\vec{t})$. This simplification will be used in Lemma 1 (cf. Section 6).

In the following we denote by $G \xrightarrow[BSM]{} G'$ an application of a *BSM* rule to a unification problem G producing a modified problem G'.

Lemma 8 *Let $S = RHS(R)$. If (G, σ) is R-normalized, $S \models G\sigma$ and G is not in dag solved form, then there exist some G' and a substitution σ' such that $G \xrightarrow[BSM]{} G'$, (G', σ') is R-normalized, $S \models G'\sigma'$ and $\sigma' \leq_S^{Var(G)} \sigma$.*

Proof. It follows the same approach to the proof of Lemma 2 in [18]. Indeed, after Lemma 7 has been established the same strategy used in [18] can be used here with small modifications due to the use of a forward-closed rewrite system.

If G is not in solved form then a *BSM* rule can be applied. Each rule applies to one or more equations from G. We consider below the different forms that these equations can take. For each case we show three properties:

1. If G is not in dag solved form, then there is a *BSM* rule that applies such that $G \xrightarrow[BSM]{} G'$.
2. For each G' there is an unifier σ' s.t. $\sigma' \leq_S^{Var(G)} \sigma$.
3. When terms are boxed in G', those terms instantiated by σ' are R-normalized.

Cases:

1. $x = y \in G$, where x and y are variables. Then, $G \xrightarrow[BSM]{} G'$ by the rule **Coalesce** and σ is a unifier of G'.
2. $u = v \in G$, s.t. u and v are not variables. According to Lemma 7, there are several sub-cases based on the form of $u\sigma =_S v\sigma$.
   (a) There is no root reduction. Then, $u = f(\vec{u})$, $v = f(\vec{v})$, and $G \xrightarrow[BSM]{} G'$ by the rule **Dec** and $\vec{u}\sigma =_S \vec{v}\sigma$.
   (b) There is a root reduction. Then, $u = f(\vec{u})$, $v = g(\vec{v})$ and there exists an equality $f(\vec{s}) = t \in S$ s.t. $\vec{u}\sigma' =_S \vec{s}\sigma'$ and $g(\vec{v})\sigma' =_S t\sigma'$, where σ' is σ extended to including the variables introduced from S. Depending on t there are several rules from *BSM* that could apply:
      i. If $t = g(\vec{t})$, then $\vec{v}\sigma' =_S \vec{t}\sigma'$. In this case $G \xrightarrow[BSM]{} G'$ by the rule **Mut**. Note, $\sigma'|_{Var(G)} = \sigma$ and by Lemma 7 terms $\vec{s}\sigma'$ and $\vec{t}\sigma'$ introduced by the rule are R-normalized.
      ii. If $t = x$, then $f(\vec{s}) = x \in S$ and $x\sigma' =_S g(\vec{v})\sigma'$. Then $G \xrightarrow[BSM]{} G'$ by the rule **VarMut** and σ' is a unifier of G'. In addition, by Lemma 7, $\vec{s}\sigma'$ is R-normalized.
3. $x = v \in G$ where x is a variable, $v = f(\vec{v})$, there is no other $x = v' \in G$, and $x = v$ is part of a cycle. By assumption $x\sigma =_S f(\vec{v})\sigma$. Then, by Lemma 7, there are two cases:
   (a) The first case is no root reduction in the proof of $x\sigma =_R f(\vec{v})\sigma$. In this case, from Lemma 7, we have that $x\sigma = f(\vec{u})\sigma$ and $\vec{u}\sigma =_S \vec{v}\sigma$. If $x = f(\vec{v})$ is part of a cycle then by the rule **ImitCycle** $G \xrightarrow[BSM]{} G'$ and σ is a unifier of G'. The term $x\sigma$ is R-normalized, thus if $x\sigma = f(\vec{w})$ for some R-normalized $\vec{w}$ and σ is extended to a R-normalized σ' that includes new variables $\vec{y}$ s.t. $\vec{y}\sigma' = \vec{w}$, then $f(\vec{y})\sigma'$ is R-normalized.
   (b) Now assume there is a reduction at the root. In this case, from Lemma 7, $x\sigma = g(\vec{u})$ and there is an equation $g(\vec{t}) = t \in S$ such that $\vec{u} =_S \vec{t}\sigma'$ where σ' is an extension of σ with the new variables from S. In addition, $\vec{t}\sigma'$ is R-normalized, $x\sigma' =_S g(\vec{t})\sigma'$,

and $f(\vec{v})\sigma' =_S t\sigma'$. If $t = f(\vec{s})$, then $\vec{v}\sigma' =_S \vec{s}\sigma'$ and $\vec{s}\sigma'$ is R-normalized. Then $G \xrightarrow[BSM]{} G'$ by rule **MutImitCycle**. If t is a variable the case reduces to a **VarMut** application as in 2(b)ii.

4. $x = v_1$ and $x = v_2 \in G$ such that x is a variable and v_1 and v_2 are non-variable terms. In this case $x\sigma =_S v_1\sigma$ and $x\sigma =_S v_2\sigma$. Then there are four cases:
   (a) Lemma 7 part 1 holds for both $x\sigma =_R v_1\sigma$ and for $x\sigma =_R v_2\sigma$. That is, in both R-rewrite proofs there is no reduction at the root. Assume $x\sigma = f(\vec{w})$, $v_1 = f(\vec{u})$, and $v_2 = f(\vec{v})$. Then it's the case that $\vec{w} =_S \vec{u}\sigma$ and $\vec{w} =_S \vec{v}\sigma$. Hence $\vec{u}\sigma =_S \vec{v}\sigma$. Then, $G \xrightarrow[BSM]{} G'$ by the rule **Imit**. If $x\sigma = f(\vec{w})$ and σ is R-normalized, then $\vec{w}$ is R-normalized. Now consider a term $f(\vec{y})\sigma'$, where σ' is an extension of σ to include the new variables, $\vec{y}$, defined as $\vec{y}\sigma' = \vec{w}$. Then, if $\vec{w}$ is R-normalized, $\vec{y}\sigma'$ is R-normalized. Furthermore, $f(\vec{u})\sigma' =_S f(\vec{y})\sigma' =_S f(\vec{v})\sigma'$.
   (b) There is no root reduction in the R-rewrite proof of $x\sigma =_R v_1\sigma$ and there is a root reduction in the proof of $x\sigma =_R v_2\sigma$. Then $x\sigma(\epsilon) = v_1\sigma(\epsilon)$. Assume $x\sigma = f(\vec{w})$. Then, $v_1 = f(\vec{u})$ and $v_2 = g(\vec{v})$. In addition, $\vec{w} =_S \vec{u}\sigma$. Then, there must be an equality in S of the form $g(\vec{t}) = t$ such that $\vec{v}\sigma =_S \vec{t}\sigma'$. Now we have two cases based on the form of t.
      i. If $t = f(\vec{s})$, then $\vec{w} =_S \vec{s}\sigma'$ where σ' is an extension of σ to include the new variables from S. By Lemma 7 $\vec{s}\sigma'$ is R-normalized. Then, $G \xrightarrow[BSM]{} G'$ via rule **MutImit** and σ' is a R-normalized unifier of G' s.t. $\sigma = \sigma'|_{Var(G)}$. If $f(\vec{u})\sigma$ is R-normalized, then $f(\vec{u})\sigma = f(\vec{w})$. Now consider a term $f(\vec{y})$ and extend σ to include the new variables $\vec{y}$ s.t. $\vec{y}\sigma' = \vec{w}$. Then $f(\vec{y})\sigma' = f(\vec{u})\sigma$, $\vec{y}\sigma' = \vec{u}\sigma$, and $f(\vec{y})\sigma'$ is R-normalized.
      ii. If $t = z$, where z is a variable, then by Lemma 7 $z\sigma =_S f(\vec{u})\sigma$. Then, $G \xrightarrow[BSM]{} G'$ via the rule **VarMut** as in 2(b)ii.
   (c) There are root reductions in the R-rewrite proof of $x\sigma =_R v_1\sigma$ and there is no root reduction in the proof of $x\sigma =_R v_2\sigma$. This case is symmetric to the previous case.
   (d) There are root reductions in the R-rewrite proof of $x\sigma =_R v_1\sigma$ and root reductions in the proof of $x\sigma =_R v_2\sigma$. Let $x\sigma = f(\vec{w})$, $v_1 = g(\vec{v})$ and $v_2 = h(\vec{u})$. In addition: $h(\vec{s}) = t \in S$ s.t. $\vec{u}\sigma =_S \vec{s}\sigma$; $x\sigma =_S t\sigma$; $g(\vec{t}) = t' \in S$ s.t. $\vec{v}\sigma =_S \vec{t}\sigma$; $x\sigma =_S t'\sigma$. Therefore, we have two reduction steps at the root position. If $v_1\sigma =_R v_2\sigma$ or $g(\vec{v})\sigma =_R h(\vec{u})\sigma$ then by Lemma 7 there exists a $h(\vec{s}) = t \in S$ s.t. $\vec{u}\sigma =_S \vec{s}\sigma'$, where σ' is just σ extended, and R-normalized, by the new variables from S. In addition, $t\sigma' =_S g(\vec{t})$.
      - If $t = g(\vec{t})$, then $\vec{t}\sigma' =_S \vec{v}\sigma$ and $G \xrightarrow[BSM]{} G'$ by the rule **Mut** as in 2(b)i.
      - If t is a variable then $G \xrightarrow[BSM]{} G'$ by the rule **VarMut** as in 2(b)ii. □

Termination of the *BSM* unification procedure follows from [18] where termination is already proven (cf. Lemma 2 for a similar proof). Completeness follows from Lemma 8.

Corollary 1 *Assume $S = RHS(R)$. Let G be an R-unification problem. Consider the set of normal forms of G w.r.t BSM rules (Figure 4) denoted by $\mathtt{BSM}_S(G)$. The set of dag solved forms in $\mathtt{BSM}_S(G)$ provides a $CSU_R(G)$.*

Theorem 4 *If R is a forward-closed convergent TRS, then the BSM unification procedure provides an R-unification algorithm.*

B Example run of BSC

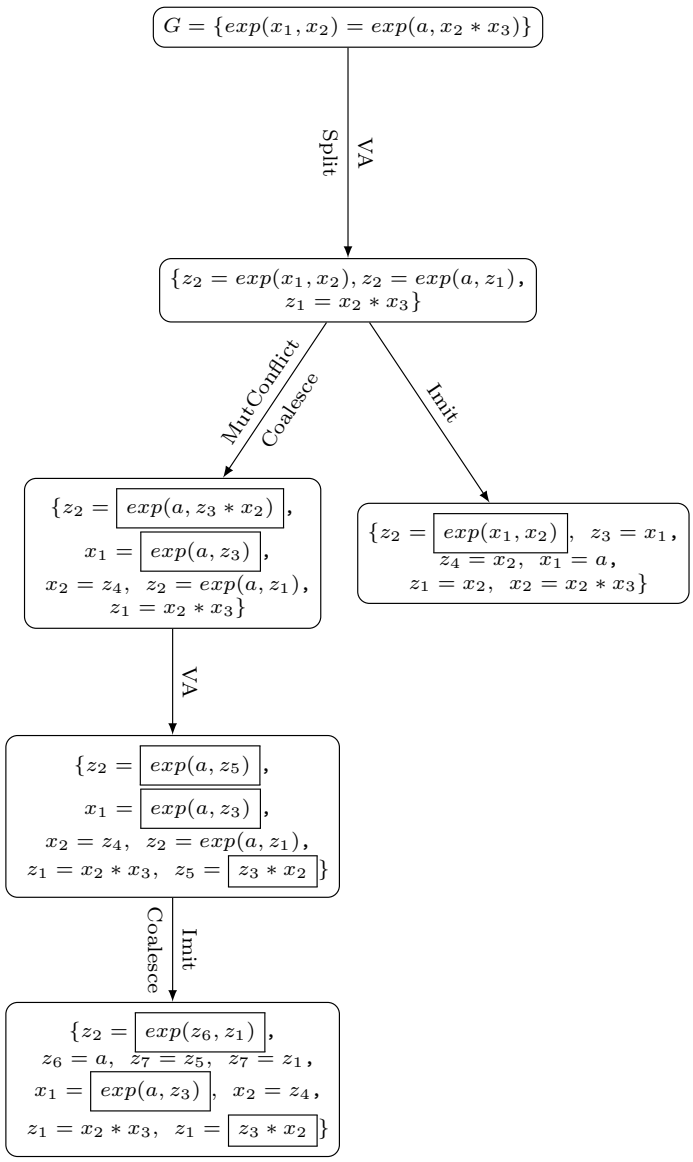
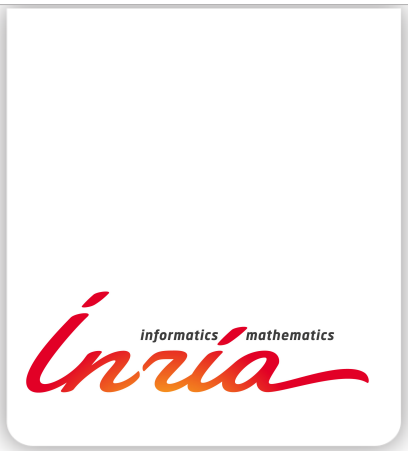


Figure 5: Example run of BSC



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399