Combining Generalization Algorithms in Regular Collapse-Free Theories

Mauricio Ayala-Rincón
Exact Sciences Institute, Universidade de Brasília, Brazil

David M. Cerna
Institute of Computer Science, Czech Academy of Sciences, Prague, Czechia

Temur Kutsia
Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria

Christophe Ringeissen
Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

We look at the generalization problem modulo some equational theories. This problem is dual to the unification problem: given two input terms, we want to find a common term whose respective two instances are equivalent to the original terms modulo the theory.

There exist algorithms for finding generalizations over various equational theories. We focus on modular construction of equational generalization algorithms for the union of signature-disjoint theories. Specifically, we consider the class of regular and collapse-free theories, showing how to combine existing generalization algorithms to produce specific solutions in these cases.

Additionally, we identify a class of theories that admit a generalization algorithm based on the application of axioms to resolve the problem. To define this class, we rely on the notion of syntactic theories, a concept originally introduced to develop unification procedures similar to the one known for syntactic unification. We demonstrate that syntactic theories are also helpful in developing generalization procedures similar to those used for syntactic generalization.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Computing methodologies → Symbolic and algebraic manipulation

Keywords and phrases Generalization, Anti-unification, Equational theories, Combination

Digital Object Identifier 10.4230/LIPIcs.FSCD.2025.7

Funding *Mauricio Ayala-Rincón*: Brazilian National Council for Scientific and Technological Development (CNPq), Grant Universal 409003/21-2 and RG 313290/21-0.
David M. Cerna: Czech Science Foundation Grant 22-06414L; Cost Action CA20111 EuroProofNet.
Temur Kutsia: Austrian Science Fund (FWF), Project P 35530; Cost Action CA20111 EuroProofNet.

1 Introduction

The problem of generalization of two terms s and t asks to find a term r such that s and t are substitution instances of r. The generalizations of interest are those that maximally retain similarities between the given terms while abstracting their differences uniformly. In other words, they should be the least general among generalizations of the given terms. The generalization problem is also called the problem of anti-unification since it can be seen as a dual to unification: while unification focuses on making two expressions more specific by finding their common instance via unifiers, anti-unification abstracts them to a more general form.

The first generalization algorithms were developed in the 1970s [38, 41], motivated by the usefulness of this technique for inductive reasoning. Nowadays, application areas of anti-unification are quite diverse, including, e.g., automated program repair [8, 19, 44], software code or specification synthesis [21, 40, 49], code similarity detection and change analysis [48, 9, 33], learning-related tasks [18, 36, 39], natural language processing [3, 24, 45], indexing/compression [12, 37, 25], just to name a few. Generalization computation techniques have been investigated for various structures (see, e.g., the recent survey [16] for an overview), among them, for first-order equational theories, which is also the subject of this paper.

In equational anti-unification, syntactic equality is replaced by equality modulo the given equational theory. As a consequence, the given terms do not necessarily have a single least general generalization. Instead, problems are characterized by their minimal complete set of generalizations, which might be a singleton, a finite set, an infinite set, or might not exist at all due to a clash between the minimality and completeness requirements. Several authors considered (first-order) equational anti-unification, studying the problem in associative, commutative, associative-commutative, unital, idempotent, absorptive theories, semirings, theories that lead to regular congruence classes, etc. [1, 2, 4, 5, 11, 13, 14, 15, 31, 47].

A more general question is related to anti-unification in a combined equational theory: Is it possible to derive a generalization algorithm for a union of equational theories from the existing generalization algorithms for the component theories? This is called the *combination problem for generalization algorithms* and is very important for applications. To the best of our knowledge, our work is the first attempt to address the modular construction of generalization algorithms.

On the other hand, the combination problem has been studied quite intensively for unification; see, e.g., [7, 20, 22, 26, 28, 30, 42, 43, 46]. These modularity results impose certain restrictions on the component theories to guarantee the existence and good properties of the combined algorithm, the main restriction being that the theories are signature-disjoint. Considering regular collapse-free theories is also a classical restriction. In the class of regular collapse-free theories, the left-hand side and the right-hand side of any equational axiom are non-variable terms with exactly the same variables. A combination method is known for unification algorithms in any union of disjoint regular collapse-free theories [46], and another one exists for matching algorithms in any union of disjoint regular theories [35] where regular theories may have collapse axioms between a non-variable left-hand side and a variable right-hand side. It is important to notice that when we relax the restriction on equational theories, we may need to impose a stronger restriction on the kind of algorithms available in the component theories. Hence, the combination method introduced in [7] for the unification problem works in any union of disjoint theories. Still, as a counterpart, it assumes the existence of general unification procedures capable of dealing with free function symbols.

Our contributions

We consider the combination problem for generalization algorithms in the union of equational theories E_1 and E_2 over the signatures Σ_1 and Σ_2 and a set of free constants C, such that:

- $\Sigma_1 \cap \Sigma_2 = \emptyset$, i.e., the theories are signature-disjoint (and $E_i \cap C = \emptyset$ holds for $i = 1, 2$, since C consists of free constants);
- both E_1 and E_2 are regular collapse-free;
- for each E_i, $i = 1, 2$, there exists a complete generalization algorithm $\mathfrak{G}_i$ which
  - handles ground terms built over $\Sigma_i \cup C$, and
  - for any given ground terms t and u returns a triple (r, ϕ, ψ), where r is a generalization of (t, u) with respective *solved* substitutions (ϕ, ψ), meaning that application of (ϕ, ψ) to r does not lead to further (more specific) generalizations.

For such equational theories E_1 and E_2, a combination algorithm for generalization problems in $E_1 \cup E_2$ is designed, which relies on E_i-generalization algorithms $\mathfrak{G}_i$ for $i = 1, 2$, and its completeness is shown. This is our main result, which we further refine in two ways, as explained below.

1. The class of finite theories (i.e., theories where every equivalence class is finite) satisfies the above-mentioned conditions, and we show that combined generalization problems in this class can be solved in a modular way. Furthermore, we provide a rule-based generalization algorithm for the subclass of finite syntactic theories. The notion of syntactic theories was originally introduced to develop unification procedures similar to the one known for syntactic unification (cf. [27, 29, 34]). We demonstrate that syntactic theories are also helpful in developing a rule-based generalization algorithm that extends syntactic generalization. This extension is obtained by introducing a new mutation rule where the resolvent axioms of the theory are applied to simplify the generalization problems.
2. For theories that admit rule-based generalization algorithms working on configurations of a special form, we design a combination algorithm that uses the rules of the component algorithms (instead of using them as black-boxes), thus obtaining a combined rule-based algorithm with rules of the same form as the component algorithms. We call it the white-box combination method. This leads to a modular result: a combined algorithm obtained in such a way can be used as a component algorithm for further white-box combinations. This white-box combination method applies to any instance of the rule-based generalization algorithm introduced for finite syntactic theories.

Organization

After this introduction, Section 2 presents the required background. Then, Section 3 presents some results connecting equational generalization to free generalization, showing that equational generalization in finite theories is reducible to free generalization, and so in finite theories equational generalization is finitary. In Section 4, we discuss the combination of generalization for regular collapse-free theories. Then, Section 5 explores the class of finite syntactic theories by showing a rule-based generalization algorithm. Section 6 considers the problem of combining rule-based generalization algorithms. Finally, Section 7 concludes and briefly discusses possible future work.

2 Preliminaries

2.1 Terms and Substitutions

We consider a first-order alphabet consisting of a signature Σ (set of fixed arity function symbols) and a set of variables $\mathcal{V}$. The set of terms $\mathcal{T}(\Sigma, \mathcal{V})$ over Σ and $\mathcal{V}$ is defined in the standard way: $t \in \mathcal{T}(\Sigma, \mathcal{V})$ iff t is defined by the grammar $t := x \mid f(t_1, \ldots, t_n)$, where $x \in \mathcal{V}$ and $f \in \Sigma$ is an n-ary symbol with $n \geq 0$.

We denote arbitrary function symbols by f, g, h, constants by a, b, c, variables by x, y, z, v, and terms by s, t, r, u. The notions of *term depth*, *term size*, and a *position in a term* are defined in the standard way, see, e.g., [6]. By $t|_p$, we denote the subterm of t at position p, and by $t[s]_p$, a term obtained from t by replacing the subterm at position p with the term s. For any position p in a term t (including the root position ϵ), $t(p)$ is the symbol at position p. The set of all variables in t is denoted by $\mathit{Var}(t)$. A term is called *linear* if no variable occurs in it more than once, and *ground* if $\mathit{Var}(t) = \emptyset$.

A sequence of terms $t_1, \ldots, t_n$ may be written $\bar{t}$ when n is clear from the context.

Given any signature Σ and any finite set of constants C such that $\Sigma \cap C = \emptyset$, the signature $\Sigma \cup C$ is denoted by Σ^C.

A *substitution* is a mapping from $\mathcal{V}$ to $\mathcal{T}(\Sigma, \mathcal{V})$, which is the identity almost everywhere. We use the Greek letters $\sigma, \vartheta, \varphi$ to denote substitutions, except for the identity substitution, which is written as *Id*. We represent substitutions using the usual set notation, e.g., $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where $\{x_1, \ldots, x_n\}$ is the domain of σ, denoted by $Dom(\sigma)$. *Application* of a substitution σ to a term t, denoted by $t\sigma$, is defined as $x\sigma = \sigma(x)$ and $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$. Substitution *composition* is defined as a composition of mappings. It is associative but not commutative, with *Id* playing the role of the unit element. We write $\sigma\vartheta$ for the composition of σ with ϑ.

2.2 Equational Theories

Let E be a set of term pairs, i.e., elements of $\mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$. An *equational* Σ*-theory generated by* E, denoted by $=_E$, is the least congruence relation on $\mathcal{T}(\Sigma, \mathcal{V})$ that is closed under substitution application and contains E. We call E *a presentation* of $=_E$, and the elements of E are called the *axioms* of $=_E$, written as $l = r$. If $t =_E u$, we say that u is equal modulo E to t. When $E = \emptyset$, every term is equal only to itself, and the theory is called the *empty* or *free* theory.

It is common to slightly abuse the terminology in the literature by calling both E and $=_E$ an equational theory. For a given E, we denote by $\mathsf{sig}(E)$ the set of all function symbols occurring in E.

A sequence of E-equalities $t_1 =_E u_1, \ldots, t_n =_E u_n$ may be written $\bar{t} =_E \bar{u}$ when n is clear from the context.

An axiom $l = r$ is *regular* if $Var(l) = Var(r)$. An axiom $l = r$ is *collapse-free* if l and r are non-variable terms. An axiom $l = r$ is *shallow* if variables can only occur at a position at depth at most 1 in both l and r. An equational theory is *regular* (resp., *collapse-free/shallow*) if all its axioms are regular (resp., collapse-free/shallow). An equational theory E is *finite* if, for each term t, there are only finitely many terms u such that $t =_E u$. An equational theory E is *subterm collapse-free* if there are no terms t, u such that $t =_E u$ and u is a strict subterm of t. A subterm collapse-free theory is necessarily collapse-free, and a finite theory is necessarily regular and subterm collapse-free.

A theory E is *syntactic* if it has a finite *resolvent presentation* RP, defined as a finite set of axioms RP such that each equality $t =_E u$ has an equational proof $t \leftrightarrow^*_{RP} u$ with at most one equational step $\leftrightarrow_{RP}$ applied at the root position. Note that any axiom in RP can be applied in both directions: from the left to the right and from the right to the left. Any shallow theory is syntactic [17], and any collapse-free theory with finitary unification is syntactic [29].

One can easily check that $A = \{x * (y * z) = (x * y) * z\}$ (Associativity), $C = \{x * y = y * x\}$ (Commutativity), and $AC = A \cup C$ (Associativity-Commutativity) are regular and collapse-free. Moreover, A, C and AC are finite and syntactic [34, 29], but only C is shallow. Since A, C, and AC are finite theories, they are also subterm collapse-free.

2.3 Combination of Equational Theories

The rest of the paper assumes that E_i is a regular collapse-free Σ_i-theory for $i = 1, 2$, where Σ_1 and Σ_2 are disjoint signatures. For the sake of brevity, the combined signature $\Sigma_1 \cup \Sigma_2$ is abbreviated to $\Sigma_{1,2}$, and the combined theory $E_1 \cup E_2$ is also denoted by $E_{1,2}$.

A Σ-rooted term t is a term whose root symbol is in Σ. Given any $\Sigma_{1,2}^C$-term t where t is Σ_i-rooted, an *alien* subterm of t is any subterm s of t which is not Σ_i-rooted and such that any proper superterm of s in t is Σ_i-rooted. The set of alien subterms of t is denoted by $Alien(t)$. Given any term t, its *height of theory* is formally defined as follows: $ht(t) = 1 + \max\{ht(t') \mid t' \in Alien(t)\}$. Given a finite set T of ground $\Sigma_{1,2}^C$-terms, we consider $Aliens = \bigcup_{t\in T} Alien(t)$ and a mapping $\pi : Aliens \to FC$ such that FC is a set of fresh free constants ($FC \cap C = \emptyset$) and for any $a, a' \in Aliens$, $\pi(a) = \pi(a')$ iff $a =_{E_1\cup E_2} a'$. Conversely, we define $\pi^{-1} : FC \to Aliens$ as a mapping such that for any $a \in Aliens$, $\pi^{-1}(\pi(a)) =_{E_1\cup E_2} a$. The i-abstraction of any term $t \in T$ is denoted by t^{π_i} and is inductively defined as follows:

- $c^{\pi_i} = c$ if $c \in C$,
- $(f(t_1, \dots, t_n))^{\pi_i} = f(t_1^{\pi_i}, \dots, t_n^{\pi_i})$ if $f \in \Sigma_i$,
- $(f(t_1, \dots, t_n))^{\pi_i} = \pi(f(t_1, \dots, t_n))$ if $f \in \Sigma_j$ for $j \neq i$.

▶ **Lemma 1** (Correspondence for Equality [46]). *Let E_1 and E_2 be two signature-disjoint regular collapse-free theories. For $i = 1, 2$ and any Σ_i^C-rooted terms t and u, $t =_{E_1\cup E_2} u$ iff $t^{\pi_i} =_{E_i} u^{\pi_i}$.*

Lemma 1 holds when the component theories E_1 and E_2 are regular collapse-free. In the general case where the component theories are arbitrary, we need to assume that t and u are in layer-reduced form [23] to have the same correspondence with their i-abstractions.

2.4 Generalization Problems

Given an equational theory E, a term t is *more general than u modulo E*, or is an *E-generalization of u*, if there exists a substitution σ such that $t\sigma =_E u$. In such a case, we write $t \preceq_E u$ and also say the term u is *less general than t modulo E* or that u is an *E-instance* of t. The relation $\preceq_E$ is a quasi-order and generates the equivalence relation, denoted by $\simeq_E$. The strict part of $\preceq_E$ is denoted by $\prec_E$.

An equational generalization problem is formulated as follows:
Given: an equational theory E and two terms t and u;
Find: a term r such that $r \preceq_E t$ and $r \preceq_E u$ (r is said to be an *E-generalization* of t and u).

Given an equational theory E and two terms t and u, a set of terms $\mathcal{G}$ is called a *complete set of E-generalizations* of t and u if it satisfies the following properties:

1. Soundness: every element of $\mathcal{G}$ is an E-generalization of t and u,
2. Completeness: for every E-generalization q of t and u there exists $r \in \mathcal{G}$ such that $q \preceq_E r$.

The set $\mathcal{G}$ is called a *minimal complete set of E-generalizations* of t and u, denoted $mcsg_E(t, u)$ if it, in addition, satisfies the following:

3. Minimality: if there exist $r, q \in \mathcal{G}$ such that $r \preceq_E q$, then $r = q$.

The existence and cardinality of $mcsg_E$ define what is called the *generalization type of an equational theory* E as follows: E has the generalization type 0 (or is *nullary*) if there are two terms t and u such that $mcsg_E(t, u)$ does not exist. Otherwise, the generalization type of E is

- unitary, if $mcsg_E(t, u)$ is a singleton for all t and u (in this case, the sole element of $mcsg_E(t, u)$ is called a least general E-generalization of t and u and is denoted by $lgg_E(t, u)$),
- finitary, if $mcsg_E(t, u)$ is finite for all t and u, and there exists at least one pair of terms for which this set is not a singleton,
- infinitary, if there exist t and u for which $mcsg_E(t, u)$ is infinite.

In the rest of the paper, we consider generalization algorithms defined as terminating inference systems transforming anti-unification triples (AUTs, for short) written $x : t \triangleq u$ where x is a variable, and t and u are the two ground terms to generalize. More precisely, we are focusing on rule-based systems working on configurations $\mathcal{C}$ of the form $A; S; \sigma$, where A and S are sets of AUTs, called the *active set* and *store* of $\mathcal{C}$, respectively, and σ is a substitution, called the *generalization component* of $\mathcal{C}$. From now on, $\mathcal{C}$ is said to be a *PSS-configuration*; it consists of a problem together with a store and a substitution. Given a set of AUTs $P = \bigcup_{i=1}^{n}\{x_i : t_i \triangleq u_i\}$, we associate two substitutions: $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is the *left* substitution of P and $\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$ is the *right* substitution of P. A set of AUTs may be written using the disjoint union symbol $\uplus$ to isolate a specific AUT from that set.

3 Equational Generalization vs Free Generalization

The following two theorems characterize equational generalizations with the help of free least general generalizations.

▶ **Theorem 2** (Free *lgg*'s versus E-Generalizations)**.** *Given an equational theory E and two terms t and u, if a term r is an E-generalization of t and u, then there exist terms t', u', and r' such that $t' =_E t$, $u' =_E u$, r' is a free least general generalization of t' and u', and $r \preceq_\emptyset r'$.*

Proof. From r being an E-generalization of t and u we get the existence of two substitutions σ and ϑ such that $r\sigma =_E t$ and $r\vartheta =_E u$. Take $t' = r\sigma$ and $u' = r\vartheta$. Then we have $t' =_E t$ and $u' =_E u$. Moreover, r is a free generalization of t' and u'. Then there exists a free least general generalization r' of t' and u', and we get $r \preceq_\emptyset r'$. ◀

▶ **Theorem 3** (Completeness and Finiteness through Free *lgg*'s)**.** *Let E be an equational theory, t and u be two terms, and $\mathcal{G}$ be the set of terms $\mathcal{G}_E(t, u) = \{lgg_\emptyset(t', u') \mid t' \in [t]_E, u' \in [u]_E\}$. Then*
1. *$\mathcal{G}_E(t, u)$ is a complete set of E-generalizations of t and u.*
2. *$\mathcal{G}_E(t, u)$ is not necessarily minimal.*
3. *If E is a finite theory, then $\mathcal{G}_E(t, u)$ is finite modulo variable renaming. Consequently, every finite theory E has the E-generalization type at most finitary.*
4. *For any E, if $[t]_E$ is finite, then $\mathcal{G}_E(t, u)$ (as well as $\mathcal{G}_E(u, t)$) is also finite modulo variable renaming.*

Proof.

1. Every element $r' \in \mathcal{G}$ is an E-generalization of t and u, because there exist substitutions σ and ϑ such that $r'\sigma = t' =_E t$ and $r'\vartheta = u' =_E u$. Thus, together with Theorem 2, we may conclude that for every E-generalization r of t and u there exists an E-generalization r' of t and u such that $r' \in \mathcal{G}$ and $r \preceq_\emptyset r'$, which implies the completeness of $\mathcal{G}$.
2. Let E be the equational theory that asserts the commutativity of f, and take $t = f(a, b)$ and $u = f(a, b)$. Then $\mathcal{G}_E(t, u) = \{f(a, b), f(x, y)\}$, because $f(a, b) = lgg_\emptyset(f(a, b), f(a, b))$ and $f(x, y) = lgg_\emptyset(f(a, b), f(b, a))$. Obviously, $f(x, y)$ is more general than $f(a, b)$ and, hence, $\mathcal{G}_E(t, u)$ is not minimal.
3. If E is a finite theory, then $[t]_E$ and $[u]_E$ are finite sets. Free least general generalizations are unique (modulo variable renaming). Hence, $\mathcal{G}_E(t, u)$ is finite modulo renaming. Since E-matching is finitary in any finite theory E, $\mathcal{G}_E(t, u)$ can always be minimized, and we get that E-generalization type in finite theories is at most finitary.

4. First, note that the depth of a free generalization of two terms never exceeds the minimum of their depths, and such a generalization does not contain any function symbol that does not appear in both of them. Therefore, for a fixed $t' \in [t]_E$, the depth of $lgg_\emptyset(t', u')$ is bounded by the depth of t', and the set of function symbols that potentially may appear in $lgg_\emptyset(t', u')$ is restricted to the set of function symbols of t'. Hence, only finitely many terms (over a fixed-arity alphabet like ours) can satisfy these conditions. Hence, whatever u' we take, the candidates for $lgg(t', u')$ may come only from a fixed finite (modulo variable renaming) set that depends on t' only: the elements of this set are terms whose free instance is t'. Therefore, for the given t', the set of all terms $\{lgg_\emptyset(t', u') \mid u' \in \mathcal{T}(\mathcal{F}, \mathcal{V})\}$ is finite. Since by assumption $[t]_E$ is finite, we get that $\mathcal{G}_E(t, u)$ is a finite union of finite sets, which implies that it is finite. ◀

▶ **Example 4.** This example illustrates an infinite $\mathcal{G}_E$ for E that is non-finite subterm-collapse free (the simplest class of non-finite theories according to [10]). Let $E = \{f(a, g(x)) = f(a, x), f(b, g(x)) = f(b, x)\}$, $t = f(a, c)$, and $u = f(b, d)$. Then

$$\mathcal{G}_E(t, u) = \{f(x, y), f(x, g(y)), f(x, g(g(y)), \ldots\}.$$

In fact, this set contains an infinite chain

$$f(x, y) \prec_E f(x, g(y)) \prec_E f(x, g(g(y)) \prec_E \cdots .$$

4 Combined Generalization in Regular Collapse-Free Theories

In this section, we consider regular collapse-free theories E. When E is regular, the E-equivalence class of any ground term only contains ground terms. Thus, any solution of an E-matching problem necessarily maps a variable occurring in the problem to a ground term. As a consequence, the following property holds when E is assumed to be regular: given any generalization r of a pair of ground terms (t, u), any substitutions ϕ, ψ such that $r\phi =_E t$ and $r\psi =_E u$ satisfy that $x\phi$ and $x\psi$ are ground for any variable $x \in Var(r)$. This assumption is useful in the following definition, where the considered substitutions are necessarily ground.

▶ **Definition 5** (Generalizations with Solved Instantiations). *Let E be a regular Σ-theory. Given any pair of ground Σ^C-terms (t, u), an E-*generalization of (t, u) with respective instantiations (ϕ, ψ) *is a Σ^C-term r together with a pair of substitutions (ϕ, ψ) such that $Var(r) = Dom(\phi) = Dom(\psi)$, $r\phi =_E t$ and $r\psi =_E u$. Given any pair of ground Σ^C-terms (t, u), a triple representing an E-*generalization of (t, u) with solved instantiations (an E-GSI triple of (t, u), for short) *is any triple (r, ϕ, ψ) such that*

- *r is an E-generalization of (t, u) with respective instantiations (ϕ, ψ),*
- *for any $x \in Var(r)$, there is no non-variable E-generalization of $(x\phi, x\psi)$,*
- *for any $x, y \in Var(r)$, $x\phi =_E y\phi$ and $x\psi =_E y\psi$ implies $x = y$[1].*

A set ST of E-GSI triples of (t, u) is said to be complete *if $\bigcup_{(r,\phi,\psi)\in ST}\{r\}$ is a complete set of E-generalizations of (t, u). We say that E admits* an algorithm computing a complete set of E-generalizations with solved instantiations (GSI algorithm, for short) *if there exists a computable function returning, for each input pair of ground Σ^C-terms (t, u), a complete set of E-GSI triples of (t, u) denoted by $GSI_E(t, u)$.*

[1] This third condition may be dropped when considering *linear* generalizations.

A GSI algorithm can be constructed via the repeated application of a generalization algorithm together with a matching algorithm. In the case of a finite theory, the generalization problem is finitary, and the matching problem as well. Consequently, this iteration can be implemented, and it is necessarily terminating in the case of a finite theory since the size of any generalization is necessarily bounded. Thus, we have that:

▶ **Lemma 6.** *Any finite theory admits a GSI algorithm.*

▶ Remark 7. In addition to the sketch of the proof given in the paragraph preceding Lemma 6, notice that another proof can be obtained using Theorem 3 and the existence of a GSI algorithm for the empty theory.

▶ **Example 8.** In Definition 5, the notion of GSI triple is introduced for any regular theory E. In particular, E can be the non-finite regular theory from Example 4. Continuing with that example where $t = f(a, c)$, $u = f(b, d)$, and $\mathcal{G}_E(t, u) = \{f(x, y), f(x, g(y)), f(x, g(g(y)), \ldots\}$, one can check that

$$GSI_E(t, u) = \{(r, \phi, \psi) \mid r \in \mathcal{G}_E(t, u), \phi = \{x \mapsto a, y \mapsto c\}, \psi = \{x \mapsto b, y \mapsto d\}\}.$$

Section 5 introduces another class of theories admitting a GSI algorithm. This class includes non-finite theories.

In general, for the theories we want to combine, we could imagine deriving GSI algorithms from existing rule-based generalization algorithms transforming PSS-configurations (see Section 6). Then, the resulting GSI algorithms can be combined directly as black-boxes, provided the theories are regular collapse-free.

▶ **Theorem 9** (Modular Property for GSI Algorithm). *The class of regular collapse-free theories admitting a GSI algorithm and with decidable equality is closed by disjoint union.*

Proof. The rule-based procedure given in Figure 1 can be shown terminating thanks to a complexity measure defined as follows: to each configuration $(A; S; \sigma)$, we associate a pair $(mht(A), |S|)$ where $mht(A)$ is the multiset of heights of theory $\bigcup_{(x':t' \triangleq u') \in A}\{ht(t'), ht(u')\}$, and $|S|$ is the cardinality of S. These pairs are ordered lexicographically, the first component being ordered by the multiset extension of the (Noetherian) ordering on positive integers, while the second component is ordered by the (Noetherian) ordering on positive integers. One can easily verify that $\boldsymbol{E_i}$**-Gen** and $\boldsymbol{E_{1,2}}$**-Sol** strictly decrease the first component of the pair, and $E_{1,2}$**-Mer** does not increase the first component but strictly decreases the second one.

The normal forms with respect to the rule-based procedure are the configurations of the form $(\emptyset; S; \sigma)$ upon which $\boldsymbol{E_{1,2}}$**-Mer** does not apply. Indeed, any configuration $(A; S; \sigma)$ with $A \neq \emptyset$ is necessarily reducible by either $\boldsymbol{E_i}$**-Gen** or $\boldsymbol{E_{1,2}}$**-Sol**.

For any derivation starting with the input configuration $(\{x : t \triangleq u\}, \emptyset, Id)$ and leading to a final configuration $(\emptyset; S; \sigma)$, we can associate a tuple $(x\sigma, \phi_S, \psi_S)$ where

- $\phi_S = \{y \mapsto t' \mid y : t' \triangleq u' \in S, y \in Var(x\sigma)\}$,
- $\psi_S = \{y \mapsto u' \mid y : t' \triangleq u' \in S, y \in Var(x\sigma)\}$.

Then, the set of all such tuples defines a $GSI_{E_1 \cup E_2}(t, u)$ as stated in Definition 5. This is a consequence of the following facts:

- $\boldsymbol{E_i}$**-Gen** is sound and complete by Lemma 10 and Corollary 11 (see below),
- $\boldsymbol{E_{1,2}}$**-Sol** is sound and complete since E_1 and E_2 are collapse-free. ◀

E_i-Gen: Generalization in the E_i-theory

$$\{x : t \triangleq u\} \uplus A; S; \sigma \Longrightarrow (P \setminus S_i) \cup A; S_i \cup S; \sigma\{x \mapsto (r\pi^{-1})\},$$

where

- $t(\epsilon) \in \Sigma_i^C, u(\epsilon) \in \Sigma_i^C$,
- $(r, \phi, \psi) \in GSI_{E_i}(t^{\pi_i}, u^{\pi_i})$,
- $P = \{y : (y\phi)\pi^{-1} \triangleq (y\psi)\pi^{-1} \mid y \in Var(r)\}$,
- $S_i = \{y : t' \triangleq u' \mid (y : t' \triangleq u') \in P, t'(\epsilon) \in \Sigma_i^C, u'(\epsilon) \in \Sigma_i^C\}$.

$E_{1,2}$-Sol: Solving in the combined theory

$$\{x : t \triangleq u\} \uplus A; S; \sigma \Longrightarrow A; S \cup \{x : t \triangleq u\}; \sigma, \qquad \text{if } t(\epsilon) \in \Sigma_i \text{ and } u(\epsilon) \in \Sigma_{3-i}.$$

$E_{1,2}$-Mer: Merging in the combined theory

$$\emptyset; \{x : t \triangleq u, x' : t' \triangleq u'\} \uplus S; \sigma \Longrightarrow \emptyset; S \cup \{x : t \triangleq u\}; \sigma\{x' \mapsto x\},$$

if $t =_{E_1 \cup E_2} t'$ and $u =_{E_1 \cup E_2} u'$.

Figure 1 Combined Generalization Procedure for Regular Collapse-Free Theories.

We now show the completeness of the algorithm presented in Figure 1 by considering any generalization of the given terms over the combined theory $E_1 \cup E_2$ and showing that we can construct from it an E_i-generalization, where i depends on the root symbol, that uses the same variable instantiations as the combined generalization. In essence, our combination algorithm builds on the completeness of the algorithms for the individual theories and is thus complete as well.

The following technical lemma and corollary are used in the proof of Theorem 9.

▶ **Lemma 10** (Correspondence for Generalization). *Let E_1 and E_2 be two signature-disjoint regular collapse-free theories over respectively the signatures Σ_1 and Σ_2, t and u any Σ_i^C-rooted terms for some $i = 1, 2$, and r any $E_1 \cup E_2$-generalization r of (t, u) with respective instantiations (ϕ, ψ). Consider a bijective mapping $\Pi_V : Alien(r) \to V$ where V is a set of fresh variables and the following terms and substitutions:*

- *r_i is the i-pure term obtained from r by replacing each $r' \in Alien(r)$ by $\Pi_V(r')$.*
- *ϕ_i and ψ_i are two substitutions such that $Dom(\phi_i) = Dom(\psi_i) = Var(r_i)$ and defined as follows:*
  - *for any $x \in Var(r_i) \setminus Var(r)$, $x\phi_i = ((\Pi_V^{-1}(x))\phi)^{\pi_i}$ and $x\psi_i = ((\Pi_V^{-1}(x))\psi)^{\pi_i}$,*
  - *for any $x \in Var(r)$, $x\phi_i = (x\phi)^{\pi_i}$.*

Then, r_i is an E_i-generalization of (t^{π_i}, u^{π_i}) with respective instantiations (ϕ_i, ψ_i), and for any $x \in Var(r_i) \setminus Var(r)$, $\Pi_V^{-1}(x)$ is an $E_1 \cup E_2$-generalization of $(x\phi_i)\pi^{-1}, (x\psi_i)\pi^{-1})$ with respective instantiations (ϕ, ψ).

Proof. By assumption, $r\phi =_{E_1 \cup E_2} t$ and $r\psi =_{E_1 \cup E_2} u$. By Lemma 1, $(r\phi)^{\pi_i} =_{E_1 \cup E_2} t^{\pi_i}$ and $(r\psi)^{\pi_i} =_{E_1 \cup E_2} u^{\pi_i}$. Then, by definition of r_i, ϕ_i and ψ_i, we have $r_i\phi_i = (r\phi)^{\pi_i}$ and $r_i\psi_i = (r\psi)^{\pi_i}$. Consequently, $r_i\phi_i =_{E_1 \cup E_2} t^{\pi_i}$ and $r_i\psi_i =_{E_1 \cup E_2} u^{\pi_i}$.

For each $x \in Var(r_i) \setminus Var(r)$, by definition of ϕ_i and ψ_i, we have that $(\Pi_V^{-1}(x)\phi)^{\pi_i} = x\phi_i$ and $(\Pi_V^{-1}(x)\psi)^{\pi_i} = x\psi_i$. Applying π^{-1} on these identities, we get $(\Pi_V^{-1}(x)\phi)^{\pi_i}\pi^{-1} = (x\phi_i)\pi^{-1}$ and $(\Pi_V^{-1}(x)\psi)^{\pi_i}\pi^{-1} = (x\psi_i)\pi^{-1}$. By definition of π^{-1}, $s =_{E_1 \cup E_2} s^{\pi_i}\pi^{-1}$ for any term s. Consequently, $(\Pi_V^{-1}(x))\phi =_{E_1 \cup E_2} (x\phi_i)\pi^{-1}$ and $(\Pi_V^{-1}(x))\psi =_{E_1 \cup E_2} (x\psi_i)\pi^{-1}$. ◀

▶ **Corollary 11.** *Let E_1 and E_2 be two signature-disjoint regular collapse-free theories over respectively the signatures Σ_1 and Σ_2, and any Σ_i^C-rooted terms t, u where $i = 1, 2$. Then, there is no non-variable $E_1 \cup E_2$-generalization of (t, u) iff there is no non-variable E_i-generalization of (t^{π_i}, u^{π_i}).*

5 Rule-Based Generalization in Finite Syntactic Theories

In this section, we explore a class of equational theories admitting a rule-based generalization algorithm similar to the one known for the empty theory. For the class of finite syntactic theories, we construct a rule-based algorithm fully parametrized by the axioms of a resolvent presentation of a finite syntactic theory. A similar rule-based algorithm can be found in [27].

▶ **Example 12.** The class of finite syntactic theories includes A, C, AC, finite shallow theories [17] such as $f(x) = g(x)$, and permutative theories closed by paramodulation [32]. Notice that $\{f(a) = a\}$ and $\{f(f(x)) = f(x)\}$ are syntactic but not finite theories.

▶ **Lemma 13.** *For any finite syntactic theory E with a resolvent presentation RP, the inference system given in Figure 4 provides an E-generalization algorithm, using the axioms of RP to*

- *check E-equality (Figure 2),*
- *solve E-matching problems (Figure 3),*
- *control the construction of E-generalizations.*

Proof. The rule-based procedure in Figure 4 terminates since E is assumed to be finite. Indeed, the size of the generalizations associated with configurations is necessarily bounded, and so there are only finitely many applications of **Mut**. The rule **Sol** is sound and complete: when an AUT cannot be transformed by **Mut** or **Dec**, we cannot have a generalization rooted by a function symbol; a variable is the only possible way to get a generalization for this AUT. The soundness and completeness of **Mut** and **Dec** follow because RP is a resolvent presentation of E.

For any input PSS-configuration $(\{x : t \triangleq u\}; \emptyset; Id)$ where t, u are ground Σ^C-terms, the procedure terminates by computing finitely many PSS-configurations of the form $(\emptyset; S; \sigma)$ and the union of all the terms $x\sigma$ corresponds to a complete set of E-generalizations of $t \triangleq u$, since all the rules are sound and complete. ◀

▶ Remark 14. For the inference system given in Figure 2, an input problem $\{s = t\}$ is such that $s =_E t$ iff all the normal forms reached by this inference system are the empty set.

For the inference system in Figure 3, the set of solutions of an input E-matching problem $\{s \leq t\}$ consists of all the normal forms reached by this inference system corresponding to solved forms $\bigcup_{i=1}^{m}\{x_i \leq t_i\}$ where $x_1, \ldots, x_m$ are pairwise distinct variables.

The class of finite syntactic theories is known to be closed by disjoint union [34]. Actually, for a union of two disjoint finite theories with respective resolvent presentations RP_1 and RP_2, we have that $RP_1 \cup RP_2$ is a resolvent presentation corresponding to a finite theory. Consequently, the inference system in Figure 4 can be directly applied to any union of disjoint finite syntactic theories for which the resolvent presentations are known. However, finding a resolvent presentation may be a difficult task, especially if E-unification is not known to be finitary. In the case we already have a GSI algorithm for a finite theory E, we can combine it directly using the approach introduced in Section 4; we don't have to find a resolvent presentation (if there exists one for E).

Mut$_=$:

$$\{f(\bar{t}) = g(\bar{u})\} \uplus \Gamma \Longrightarrow \{\bar{t} = \bar{l}\theta\} \cup \Gamma$$

where $f(\bar{l}) = g(\bar{r})$ is a fresh renaming of an axiom in RP, and θ is a substitution such that $\bar{r}\theta =_{RP} \bar{u}$.

Dec$_=$:

$$\{f(\bar{t}) = f(\bar{u})\} \uplus \Gamma \Longrightarrow \{\bar{t} = \bar{u}\} \cup \Gamma \qquad \text{where } f \in \Sigma^C.$$

Figure 2 Procedure for Checking Equality in a finite theory with a resolvent presentation RP.

Mut$_\leq$: Mutation rule for matching

$$\{f(\bar{t}) \leq^? g(\bar{u})\} \uplus \Gamma \Longrightarrow \{\bar{t} \leq^? \bar{l}\theta\} \cup \Gamma$$

where $f(\bar{l}) = g(\bar{r})$ is a fresh renaming of an axiom in RP, and θ is a substitution such that $\bar{r}\theta =_{RP} \bar{u}$.

Dec$_\leq$: Decomposition rule for matching

$$\{f(\bar{t}) \leq^? f(\bar{u})\} \uplus \Gamma \Longrightarrow \{\bar{t} \leq^? \bar{u}\} \cup \Gamma \qquad \text{where } f \in \Sigma^C.$$

Mer$_\leq$: Merging rule for matching

$$\{x \leq^? t, x' \leq^? t'\} \uplus \Gamma \Longrightarrow \{x \leq^? t\} \cup \Gamma \qquad \text{if } t =_{RP} t'.$$

Figure 3 Matching Procedure in a finite theory with a resolvent presentation RP.

▶ Remark 15. The empty theory $\emptyset$ is a finite syntactic theory, and its resolvent presentation is empty. In the case $RP = \emptyset$, one can notice that the inference system from Figure 4 corresponds to the classical rule-based generalization algorithm known for the empty theory.

6 White-Box Combination of Rule-Based Generalization Algorithms

This section focuses on the modular construction of generalization algorithms transforming PSS-configurations. Let us now define precisely this kind of rule-based generalization algorithm.

▶ **Definition 16.** *Let E be a regular theory. An E-*generalization algorithm transforming PSS-configurations *is an inference system $\mathfrak{G}$ such that for any initial configuration $(\{x : t \triangleq u\}, \emptyset, Id)$, it derives a finite set FC of final configurations of the form $(\emptyset; S; \sigma)$ for which $\{x\sigma \mid (\emptyset; S; \sigma) \in FC\}$ is a complete set of E-generalizations of (t, u). Furthermore, for any configuration $(A; S; \sigma)$ derived from the initial configuration $(\{x : t \triangleq u\}; \emptyset; Id)$, the following (invariant) properties must hold:*

- *$x\sigma$ is an E-generalization of (t, u) with respective instantiations (ϕ, ψ) where ϕ (resp., ψ) is the left (resp., right) substitution of $A \cup S$,*
- *for any $x' : t' \triangleq u'$ in S, there is no non-variable E-generalization of (t', u').*

▶ Remark 17. An E-generalization algorithm transforming PSS-configurations leads to a GSI-algorithm. Consider an initial configuration $(\{x : t \triangleq u\}, \emptyset, Id)$ and any triple $(r\sigma, \phi_S, \psi_S)$ such that $(\emptyset, S, \sigma) \in FC$ where FC is the set of final configurations given in Definition 16 and ϕ_S (resp., ψ_S) denotes the left (resp., right) substitution of S. Then, $\{(x\sigma, \phi_S, \psi_S) \mid (\emptyset, S, \sigma) \in FC\}$ is actually a $GSI_E(t, u)$.

Mut$_{LR}$: Left-Right Mutation rule for generalization

$$\{x : f(\bar{t}) \triangleq g(\bar{u})\} \uplus A; S; \sigma \Longrightarrow \{y : y\theta \triangleq y\theta' \mid y \in \bar{y}\} \cup A; S; \sigma\{x \mapsto h(\bar{y})\}$$

where $h(\bar{l}) = f(\bar{r}) \in RP$, $h(\bar{l}') = g(\bar{r}') \in RP$, $h(\bar{y})\theta =_{RP} f(\bar{t})$ and $h(\bar{y})\theta' =_{RP} g(\bar{u})$.

Mut$_{R}$: Right Mutation rule for generalization

$$\{x : f(\bar{t}) \triangleq g(\bar{u})\} \uplus A; S; \sigma \Longrightarrow \{y : y\theta \triangleq y\theta' \mid y \in \bar{y}\} \cup A; S; \sigma\{x \mapsto f(\bar{y})\}$$

where $f(\bar{l}') = g(\bar{r}') \in RP$, $\theta = \{\bar{y} \mapsto \bar{t}\}$ and $f(\bar{y})\theta' =_{RP} g(\bar{u})$.

Mut$_{L}$: Left Mutation rule for generalization

$$\{x : f(\bar{t}) \triangleq g(\bar{u})\} \uplus A; S; \sigma \Longrightarrow \{y : y\theta \triangleq y\theta' \mid y \in \bar{y}\} \cup A; S; \sigma\{x \mapsto g(\bar{y})\}$$

where $g(\bar{r}') = f(\bar{l}') \in RP$, $g(\bar{y})\theta =_{RP} f(\bar{t})$ and $\theta' = \{\bar{y} \mapsto \bar{u}\}$.

Dec: Decomposition rule for generalization

$$\{x : f(\bar{t}) \triangleq f(\bar{u})\} \uplus A; S; \sigma \Longrightarrow \{\bar{y} : \bar{t} = \bar{u}\} \cup A; S; \sigma\{x \mapsto f(\bar{y})\}$$

where $f \in \Sigma^C$ and $\bar{y}$ are fresh variables.

Sol: Solving rule for generalization

$$\{x : t \triangleq u\} \uplus A; S; \sigma \Longrightarrow A; S \cup \{x : t \triangleq u\}; \sigma$$

if no rule in $\{\mathbf{Mut}_{LR}, \mathbf{Mut}_{R}, \mathbf{Mut}_{L}, \mathbf{Dec}\}$ applies.

Mer: Merging rule for generalization

$$\emptyset; \{x : t \triangleq u, x' : t' \triangleq u'\} \uplus S; \sigma \Longrightarrow \emptyset; S \cup \{x : t \triangleq u\}; \sigma\{x' \mapsto x\}$$

if $t =_{RP} t'$ and $u =_{RP} u'$.

Figure 4 Generalization Procedure in a finite theory with a resolvent presentation RP.

The inference procedure in Figure 4 corresponds to a generalization algorithm transforming PSS-configurations, thus leading to a GSI algorithm. Conversely, note that a GSI algorithm can be turned into a generalization algorithm transforming PSS-configurations where all derivations are of length 1.

▶ **Corollary 18.** *A theory admits a generalization algorithm transforming PSS-configurations iff it admits a GSI algorithm.*

Generalization algorithms transforming PSS-configurations correspond to GSI algorithms, and so they can be combined using the method given in Figure 1, provided that the respective theories E_1 and E_2 are regular collapse-free. Since this method is given as an inference system manipulating PSS-configurations, it can be reworded as an $E_1 \cup E_2$-generalization algorithm transforming PSS-configurations. In this new presentation, we do not directly use the triples from GSI_{E_i}-sets, but we just apply one step of the inference system corresponding to an E_i-generalization algorithm transforming PSS-configurations, see Figure 5. This is sufficient to move forward with a solution.

Given two generalization algorithms $\mathfrak{G}_1$ and $\mathfrak{G}_2$ transforming PSS-configurations, let $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ be the inference system given by the set of rules $\{$**E_i-Step**, **$E_{1,2}$-Sol**, **$E_{1,2}$-Mer** $\}$ where **E_i-Step** is introduced in Figure 5 while $E_{1,2}$-**Sol** and **$E_{1,2}$-Mer** are from Figure 1.

***E_i*-Step**:

$$\{x : t \triangleq u\} \uplus A; S; \sigma \Longrightarrow (A_i\pi^{-1}) \cup A; (S_i\pi^{-1}) \cup S; \sigma(\sigma_i\pi^{-1})$$

where $t(\epsilon) \in \Sigma_i^C, u(\epsilon) \in \Sigma_i^C$ and $(\{x : t^{\pi_i} \triangleq u^{\pi_i}\}; \emptyset; Id) \longrightarrow_{\mathfrak{G}_i} (A_i; S_i; \sigma_i)$.

Figure 5 Rule calling a generalization algorithm $\mathfrak{G}_i$ transforming PSS-configurations.

▶ **Theorem 19** (Modular Construction for Rule-Based Generalization). *Let E_1 and E_2 be two signature-disjoint regular collapse-free theories. If $\mathfrak{G}_i$ is an E_i-generalization algorithm transforming PSS-configurations for $i = 1, 2$, then $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ is an $E_1 \cup E_2$-generalization algorithm transforming PSS-configurations.*

Proof. The inference system $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ is terminating since $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are terminating, and there are finitely many alternations of calls to $\mathfrak{G}_1$ and to $\mathfrak{G}_2$ (see Lemma 10). Actually, $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ corresponds to an $E_1 \cup E_2$-generalization algorithm transforming PSS-configurations since the ***E_i*-Step** rule from Figure 5 is sound and complete by Lemma 10 and Corollary 11. ◀

▶ Remark 20. In both rules ***E_i*-Step** and ***E_i*-Gen**, the constant abstraction $(\cdot)^{\pi_i}$ and the term concretization $(\cdot)\pi^{-1}$ should be considered as transparent operations. With $(\cdot)^{\pi_i}$, aliens are viewed as free constants, with the restriction that two $E_1 \cup E_2$-equal aliens must be viewed as the same free constant. Conversely, with $(\cdot)\pi^{-1}$, these free constants are viewed back as terms in the combined theory $E_1 \cup E_2$. By slightly abusing notation, if we omit to apply these transformations, then the rules ***E_i*-Step** and ***E_i*-Gen** become easy to express.

Note that for free, associative, and commutative theories, all four algorithms for the combined theories $(\emptyset)(A)$, $(\emptyset)(C)$, $(A)(C)$, $(\emptyset)(A)(C)$ described in [2] can be obtained as a combination of generalization algorithms transforming PSS-configurations.

▶ **Example 21.** Consider $E_1 = \{f(x) = g(x)\}$ and $E_2 = AC(+)$. For E_1, one can show the existence of an E_1-generalization algorithm transforming PSS-configurations, say $\mathfrak{G}_1$, given by the set of rules $\{E_i$-**Mut**, **Dec**, **Sol**, **Mer**$\}$, where **Dec**, **Sol**, **Mer** can be found in Figure 4, and the mutation rules E_1-**Mut** is defined as follows:

***E_1*-Mut**:

$$\{x : f(t) \triangleq g(u)\} \uplus A; S; \sigma \Longrightarrow \{y : t \triangleq u\} \cup A; S; \sigma\{x \mapsto f(y)\}$$

For E_2, we call $\mathfrak{G}_2$ the E_2-generalization algorithm transforming PSS-configurations given in Figure 4, with the following resolvent presentation of $AC(+)$ shown in [29]:

$$RP = \left\{ \begin{array}{l} x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 \\ x_1 + x_2 = x_2 + x_1 \\ x_1 + (x_2 + x_3) = (x_1 + x_3) + x_2 \\ (x_1 + x_2) + x_3 = (x_1 + x_3) + x_2 \\ x_1 + (x_2 + x_3) = x_3 + (x_1 + x_2) \\ (x_1 + x_3) + (x_2 + x_4) = (x_1 + x_4) + (x_2 + x_3) \end{array} \right\}$$

Let us now detail how the combined generalization algorithm $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ is applied to solve

$$x : f(a + (f(a) + b)) \triangleq g((a + c) + f(a)).$$

We focus on two particular derivations leading to final PSS-configurations. The derived PSS-configurations are listed in the table below, where the rightmost column shows the configuration number, while the first column indicates how the configuration in that row is obtained (which generalization algorithm is applied to which configuration):

	$(\{x : f(a + (f(a) + b)) \triangleq g((a + c) + f(a))\}; \emptyset; \mathit{Id})$	(0)
$\mathfrak{G}_1(0)$	$(\{y_1 : a + (f(a) + b) \triangleq (a + c) + f(a)\}; \emptyset; \sigma_0\{x \mapsto f(y_1)\})$	(1)
$\mathfrak{G}_2(1)$	$(\{y_2 : a \triangleq a + c, y_3 : f(a) + b \triangleq f(a)\}; \emptyset; \sigma_1\{y_1 \mapsto y_2 + y_3)\})$	(2)
$\mathfrak{G}_2(2)$	$(\{y_3 : f(a) + b \triangleq f(a)\}; \{y_2 : a \triangleq a + c\}; \sigma_2)$	(3)
$\mathfrak{G}_2(3)$	$(\emptyset; \{y_2 : a \triangleq a + c, y_3 : f(a) + b \triangleq f(a)\}; \sigma_3)$	(4)
$\mathfrak{G}_2(1)$	$(\{y_4 : f(a) \triangleq f(a), y_5 : b + a \triangleq a + c\}; \emptyset; \sigma_1\{y_1 \mapsto y_4 + y_5)\})$	(5)
$\mathfrak{G}_1(5)$	$(\{y_6 : a \triangleq a, y_5 : b + a \triangleq a + c\}; \emptyset; \sigma_5\{y_4 \mapsto f(y_6)\})$	(6)
$\mathfrak{G}_1(6)$	$(\{y_5 : b + a \triangleq a + c\}; \emptyset; \sigma_6\{y_6 \mapsto a\})$	(7)
$\mathfrak{G}_2(7)$	$(\{y_7 : a \triangleq a, y_8 : b \triangleq c\}; \emptyset; \sigma_9\{y_5 \mapsto y_7 + y_8\})$	(8)
$\mathfrak{G}_1(8)$	$(\{y_8 : b \triangleq c\}; \emptyset; \sigma_8\{y_7 \mapsto a\})$	(9)
$\mathfrak{G}_1(9)$	$(\emptyset; \{y_8 : b \triangleq c\}; \sigma_9)$	(10)

To get (5), $\mathbf{Mut}_{LR}$ is applied with the axioms $x_1 + (x_2 + x_3) = x_3 + (x_1 + x_2)$ and $x_1 + x_2 = x_2 + x_1$ in RP. To get (8), $\mathbf{Mut}_L$ is applied with the axiom $x_1 + x_2 = x_2 + x_1$ in RP. Notice that $\mathfrak{G}_2$ could be applied instead of $\mathfrak{G}_1$ on (6), (8), (9) since both $\mathfrak{G}_1$ and $\mathfrak{G}_2$ can handle any AUT between two constants in C. Conversely, $\mathfrak{G}_1$ could be applied instead of $\mathfrak{G}_2$ on (3) since both $\mathfrak{G}_1$ and $\mathfrak{G}_2$ include a **Sol** rule. Eventually, (4) and (10) are two final configurations:

- $(\emptyset; \{y_2 : a \triangleq a + c, y_3 : f(a) + b \triangleq f(a)\}; \{x \mapsto f(y_2 + y_3), \dots\}$,
- $(\emptyset; \{y_8 : b \triangleq c\}; \{x \mapsto f(f(a) + (a + y_8)), \dots\})$.

The resulting generalizations are:

- $f(y_2 + y_3)$, with respective instantiations $(\{y_2 \mapsto a, y_3 \mapsto f(a) + b\}$, $\{y_2 \mapsto a + c$, $y_3 \mapsto f(a)\})$,
- $f(f(a) + (a + y_8))$, with respective instantiations $(\{y_8 \mapsto b\}$, $\{y_8 \mapsto c\})$.

The combined generalization algorithm $\mathfrak{G}_1 \oplus \mathfrak{G}_2$ derives additional final configurations. Still, one can check that the corresponding generalizations are always more general than $f(f(a) + (a + y_8))$, just like $f(y_2 + y_3)$ is more general than $f(f(a) + (a + y_8))$.

7 Conclusion

We have introduced a combination method for the generalization problem in any union of disjoint regular collapse-free theories. Our framework requires regularity to ensure that any matching problem only has ground solutions. With this property, we have only to consider the generalization of ground terms involving possibly free constants. Collapse-freeness is also very useful for solving easily the generalization of two terms rooted in distinct theories. In future work, we plan to investigate the possibility of relaxing this regular and collapse-free assumption.

The study of non-disjoint unions of theories would also be a natural continuation. In particular, we plan to consider the case of shared constructors, relying on the combination results available for the matching problem in unions of theories sharing only constructors [23]. When the constructors do not appear in the root of either side of the axioms, it might be possible to solve the combined generalization problem similarly to the disjoint case. More generally, allowing only shared constants at the root of the axioms could also be helpful to solve the problem.

For the class of regular collapse-free theories, we have investigated the disjoint combination of rule-based generalization algorithms transforming PSS-configurations. For the subclass of finite syntactic theories, we have shown how the axioms of a resolvent presentation can be used to construct a generalization algorithm transforming PSS-configurations. More broadly, we are interested in identifying examples of non-finite (yet regular collapse-free) theories that admit a generalization algorithm transforming PSS-configurations, to which our combination approach would be applicable. Finally, we conjecture that this combination approach can be naturally extended to generalization procedures that are not necessarily terminating.

References

1 María Alpuente, Demis Ballis, Angel Cuenca-Ortega, Santiago Escobar, and José Meseguer. ACUOS^2: A High-Performance System for Modular ACU Generalization with Subtyping and Inheritance. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 171–181. Springer, 2019. doi:10.1007/978-3-030-19570-0_11.

2 María Alpuente, Santiago Escobar, Javier Espert, and José Meseguer. A modular order-sorted equational generalization algorithm. *Inf. Comput.*, 235:98–136, 2014. doi:10.1016/J.IC.2014.01.006.

3 Nino Amiridze and Temur Kutsia. Anti-unification and natural language processing. EasyChair Preprint no. 203, EasyChair, 2018. doi:10.29007/fkrh.

4 Mauricio Ayala-Rincón, David M. Cerna, Andres Felipe Gonzalez Barragan, and Temur Kutsia. Equational anti-unification over absorption theories. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*, volume 14740 of *Lecture Notes in Computer Science*, pages 317–337. Springer, 2024. doi:10.1007/978-3-031-63501-4_17.

5 Franz Baader. Unification, weak unification, upper bound, lower bound, and generalization problems. In Ronald V. Book, editor, *Rewriting Techniques and Applications, 4th International Conference, RTA-91, Como, Italy, April 10-12, 1991, Proceedings*, volume 488 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 1991. doi:10.1007/3-540-53904-2_88.

6 Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998. doi:10.1017/CBO9781139172752.

7 Franz Baader and Klaus U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symb. Comput.*, 21(2):211–243, 1996. doi:10.1006/JSCO.1996.0009.

8 Johannes Bader, Andrew Scott, Michael Pradel, and Satish Chandra. Getafix: learning to fix bugs automatically. *Proc. ACM Program. Lang.*, 3(OOPSLA):159:1–159:27, 2019. doi:10.1145/3360585.

9 Adam D. Barwell, Christopher Brown, and Kevin Hammond. Finding parallel functional pearls: Automatic parallel recursion scheme detection in Haskell functions via anti-unification. *Future Gener. Comput. Syst.*, 79:669–686, 2018. doi:10.1016/j.future.2017.07.024.

10 Hans-Jürgen Bürckert, Alexander Herold, and Manfred Schmidt-Schauß. On equational theories, unification, and (un)decidability. *J. Symb. Comput.*, 8(1/2):3–49, 1989. doi:10.1016/S0747-7171(89)80021-5.

11 Jochen Burghardt. E-generalization using grammars. *Artif. Intell.*, 165(1):1–35, 2005. doi:10.1016/J.ARTINT.2005.01.008.

12 David Cao, Rose Kunkel, Chandrakana Nandi, Max Willsey, Zachary Tatlock, and Nadia Polikarpova. babble: Learning Better Abstractions with E-Graphs and Anti-unification. *Proceedings of the ACM on Programming Languages*, 7(POPL):396–424, 2023. doi:10.1145/3571207.

13 David M. Cerna. Anti-unification and the theory of semirings. *Theor. Comput. Sci.*, 848:133–139, 2020. doi:10.1016/J.TCS.2020.10.020.

14 David M. Cerna and Temur Kutsia. Idempotent anti-unification. *ACM Trans. Comput. Log.*, 21(2):10:1–10:32, 2020. doi:10.1145/3359060.

15 David M. Cerna and Temur Kutsia. Unital anti-unification: Type and algorithms. In Zena M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, volume 167 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.FSCD.2020.26.

16 David M. Cerna and Temur Kutsia. Anti-unification and generalization: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 6563–6573. ijcai.org, 2023. doi:10.24963/IJCAI.2023/736.

17 Hubert Comon, Marianne Haberstrau, and Jean-Pierre Jouannaud. Syntacticness, cycle-syntacticness, and shallow theories. *Inf. Comput.*, 111(1):154–191, 1994. doi:10.1006/inco.1994.1043.

18 Andrew Cropper, Sebastijan Dumancic, Richard Evans, and Stephen H. Muggleton. Inductive logic programming at 30. *Mach. Learn.*, 111(1):147–172, 2022. doi:10.1007/S10994-021-06089-1.

19 Reudismam Rolim de Sousa, Gustavo Soares, Rohit Gheyi, Titus Barik, and Loris D'Antoni. Learning quick fixes from code repositories. In Cristiano D. Vasconcellos, Karina Girardi Roggia, Vanessa Collere, and Paulo Bousfield, editors, *35th Brazilian Symposium on Software Engineering, SBES 2021, Joinville, Santa Catarina, Brazil, 27 September 2021 - 1 October 2021*, pages 74–83. ACM, 2021. doi:10.1145/3474624.3474650.

20 Eric Domenjoud, Francis Klay, and Christophe Ringeissen. Combination techniques for non-disjoint equational theories. In Alan Bundy, editor, *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings*, volume 814 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 1994. doi:10.1007/3-540-58156-1_19.

21 Rui Dong, Zhicheng Huang, Ian Iong Lam, Yan Chen, and Xinyu Wang. WebRobot: web robotic process automation using interactive programming-by-demonstration. In Ranjit Jhala and Isil Dillig, editors, *PLDI '22: 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, San Diego, CA, USA, June 13 - 17, 2022*, pages 152–167. ACM, 2022. doi:10.1145/3519939.3523711.

22 Serdar Erbatur, Andrew M. Marshall, and Christophe Ringeissen. Non-disjoint combined unification and closure by equational paramodulation. In Boris Konev and Giles Reger, editors, *Frontiers of Combining Systems - 13th International Symposium, FroCoS 2021, Birmingham, UK, September 8-10, 2021, Proceedings*, volume 12941 of *Lecture Notes in Computer Science*, pages 25–42. Springer, 2021. doi:10.1007/978-3-030-86205-3_2.

23 Serdar Erbatur, Andrew M. Marshall, and Christophe Ringeissen. Combined Hierarchical Matching: the Regular Case. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 6:1–6:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FSCD.2022.6.

24 Boris Galitsky. *Developing Enterprise Chatbots - Learning Linguistic Structures*. Springer, 2019. doi:10.1007/978-3-030-04299-8.

25 Peter Graf. Substitution tree indexing. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Kaiserslautern, Germany, April 5-7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 117–131. Springer, 1995. doi:10.1007/3-540-59200-8_52.

26 Alexander Herold. *Combination of unification algorithms in equational theories*. PhD thesis, Kaiserslautern University of Technology, Germany, 1987. URL: https://d-nb.info/880327979.

27 Jean-Pierre Jouannaud. Syntactic theories. In Branislav Rovan, editor, *Mathematical Foundations of Computer Science 1990, MFCS'90, Banská Bystrica, Czechoslovakia, August 27-31, 1990, Proceedings*, volume 452 of *Lecture Notes in Computer Science*, pages 15–25. Springer, 1990. doi:10.1007/BFB0029593.

28 Claude Kirchner. *Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles.* Thèses d'état, Université de Nancy I, 1985.

29 Claude Kirchner and Francis Klay. Syntactic theories and unification. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 270–277. IEEE Computer Society, 1990. doi:10.1109/LICS.1990.113753.

30 Hélène Kirchner and Christophe Ringeissen. Combining symbolic constraint solvers on algebraic domains. *J. Symb. Comput.*, 18(2):113–155, 1994. doi:10.1006/JSCO.1994.1040.

31 Boris Konev and Temur Kutsia. Anti-unification of concepts in description logic EL. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016*, pages 227–236. AAAI Press, 2016. URL: http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12880.

32 Christopher Lynch and Barbara Morawska. Basic syntactic mutation. In Andrei 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. doi:10.1007/3-540-45620-1_37.

33 Sonu Mehta, Ranjita Bhagwan, Rahul Kumar, Chetan Bansal, Chandra Shekhar Maddila, B. Ashok, Sumit Asthana, Christian Bird, and Aditya Kumar. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In Ranjita Bhagwan and George Porter, editors, *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 435–448. USENIX Association, 2020. URL: https://www.usenix.org/conference/nsdi20/presentation/mehta.

34 Tobias Nipkow. Proof transformations for equational theories. In *Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990*, pages 278–288. IEEE Computer Society, 1990. doi:10.1109/LICS.1990.113754.

35 Tobias Nipkow. Combining matching algorithms: The regular case. *J. Symb. Comput.*, 12(6):633–654, 1991. doi:10.1016/S0747-7171(08)80145-9.

36 Julian Parsert and Elizabeth Polgreen. Reinforcement learning and data-generation for syntax-guided synthesis. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10670–10678. AAAI Press, 2024. doi:10.1609/AAAI.V38I9.28938.

37 Brigitte Pientka. Higher-order term indexing using substitution trees. *ACM Trans. Comput. Log.*, 11(1):6:1–6:40, 2009. doi:10.1145/1614431.1614437.

38 Gordon D. Plotkin. A note on inductive generalization. *Machine Intel.*, 5(1):153–163, 1970.

39 Stanislaw J. Purgal, David M. Cerna, and Cezary Kaliszyk. Learning higher-order logic programs from failures. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2726–2733. ijcai.org, 2022. doi:10.24963/IJCAI.2022/378.

40 Mohammad Raza, Sumit Gulwani, and Natasa Milic-Frayling. Programming by Example Using Least General Generalizations. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 283–290. AAAI Press, 2014. doi:10.1609/AAAI.V28I1.8744.

41 John C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intel.*, 5(1):135–151, 1970.

42 Manfred Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *J. Symb. Comput.*, 8(1/2):51–99, 1989. doi:10.1016/S0747-7171(89)80022-7.

43 Erik Tidén. Unification in combinations of collapse-free theories with disjoint sets of function symbols. In Jörg H. Siekmann, editor, *8th International Conference on Automated Deduction, Oxford, England, July 27 - August 1, 1986, Proceedings*, volume 230 of *Lecture Notes in Computer Science*, pages 431–449. Springer, 1986. doi:10.1007/3-540-16780-3_110.

44 Emily Rowan Winter, Vesna Nowack, David Bowes, Steve Counsell, Tracy Hall, Sæmundur Óskar Haraldsson, John R. Woodward, Serkan Kirbas, Etienne Windels, Olayori McBello, Abdurahman Atakishiyev, Kevin Kells, and Matthew W. Pagano. Towards developer-centered automatic program repair: findings from Bloomberg. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim, editors, *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE, Singapore, November 14-18, 2022*, pages 1578–1588. ACM, 2022. doi:10.1145/3540250.3558953.

45 Kaiyu Yang and Jia Deng. Learning symbolic rules for reasoning in quasi-natural language. *Trans. Mach. Learn. Res.*, 2023, 2023. URL: https://openreview.net/forum?id=gwRwHUZUgz.

46 Katherine A. Yelick. Unification in combinations of collapse-free regular theories. *J. Symb. Comput.*, 3(1/2):153–181, 1987. doi:10.1016/S0747-7171(87)80025-1.

47 Gonzague Yernaux and Wim Vanhoof. Anti-unification of unordered goals. In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 37:1–37:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CSL.2022.37.

48 Gonzague Yernaux and Wim Vanhoof. On detecting semantic clones in constraint logic programs. In *16th IEEE International Workshop on Software Clones, IWSC 2022, Limassol, Cyprus, October 2, 2022*, pages 32–38. IEEE, 2022. doi:10.1109/IWSC55060.2022.00014.

49 Dongjun Youn, Wonho Shin, Jaehyun Lee, Sukyoung Ryu, Joachim Breitner, Philippa Gardner, Sam Lindley, Matija Pretnar, Xiaojia Rao, Conrad Watt, and Andreas Rossberg. Bringing the WebAssembly Standard up to Speed with SpecTec. *Proc. ACM Program. Lang.*, 8(PLDI):1559–1584, 2024. doi:10.1145/3656440.