Noname manuscript No.
(will be inserted by the editor)

Order-sorted Equational Generalization Algorithm Revisited

María Alpuente · Santiago Escobar · José Meseguer · Julia Sapiña

the date of receipt and acceptance should be inserted later

Abstract Generalization, also called anti-unification, is the dual of unification. A generalizer of two terms t and t' is a term t'' of which t and t' are substitution instances. The dual of most general equational unifiers is that of least general equational generalizers, i.e., most specific anti-instances modulo equations. In a previous work, we extended the classical untyped generalization algorithm to: (1) an order-sorted typed setting with sorts, subsorts, and subtype polymorphism; (2) work modulo equational theories, where function symbols can obey any combination of associativity, commutativity, and identity axioms (including the empty set of such axioms); and (3) the combination of both, which results in a modular, order-sorted equational generalization algorithm. However, Cerna and Kutsia showed that our algorithm is generally incomplete for the case of identity axioms and a counterexample was given. Furthermore, they proved that, in theories with two identity elements or more, generalization with identity axioms is generally nullary, yet it is finitary for both the linear and one-unital fragments, i.e., either solutions with repeated variables are disregarded or the considered theories are restricted to having just one function symbol with an identity or unit element.

This work has been partially supported by the EC H2020-EU grant agreement No. 952215 (TAILOR), the EU (FEDER) and the Spanish MCIU under grant RTI2018-094403-B-C32, by Generalitat Valenciana under grant PROMETEO/2019/098, and by NRL under contract number N00173-17-1-G002. Julia Sapiña has been supported by the Generalitat Valenciana APOSTD/2019/127 grant.

M. Alpuente
Universitat Politècnica de València, Spain
E-mail: alpuente@upv.es

S. Escobar
Universitat Politècnica de València, Spain
E-mail: sescobar@upv.es

J. Meseguer
University of Illinois at Urbana-Champaign, USA
E-mail: meseguer@illinois.edu

J. Sapiña
Universitat Politècnica de València, Spain
E-mail: jsapina@upv.es

In this work, we show how we can easily extend our original inference system to cope with the non-linear fragment and identify a more general class than one–unit theories where generalization with identity axioms is finitary.

Keywords least general generalization · rule–based languages · equational reasoning · order-sorted · associativity · commutativity · identity

1 Introduction

Computing generalizations is relevant in a wide spectrum of automated reasoning areas where analogical reasoning and inductive inference are needed, such as analogy making, case-based reasoning, web and data mining, ontology learning, machine learning, theorem proving, program derivation, and inductive logic programming, among others (Armengol, 2007; Muggleton, 1999; Ontañón and Plaza, 2012).

Roughly speaking, in a pure syntactic and untyped setting, the syntactic generalization problem for two or more expressions consists in finding their *least general generalizer (lgg)*, i.e., the least general expression t such that all of the given expressions are instances of t under appropriate substitutions. For instance, the expression *sibling(x,y)*, where x and y are variables, is a generalizer of both *sibling(john,sam)* and *sibling(tom,sam)*, but their least general generalizer is *sibling(x,sam)*.

In (Alpuente et al., 2014b), the notion of least general generalization is extended to the order-sorted modulo axioms setting, where any function symbol f can obey any combination of the associativity axiom ($f(f(x, y), z) = f(x, f(y, z))$), the commutativity axiom ($f(x, y) = f(y, x)$), and the identity axiom ($f(x, \mathtt{e}) = x$ and $f(\mathtt{e}, x) = x$, for unit element $\mathtt{e}$) including the empty set of such axioms. For instance, the least general generalizer of *sibling(sam,john)* and *sibling(tom,sam)* is still *sibling(x,sam)* when *sibling* is a commutative symbol. In general, there is no unique least general generalizer in the framework of (Alpuente et al., 2014b) due to both the order-sortedness (Alpuente et al., 2009) and to the equational axioms (Alpuente et al., 2008). However, it is often the case that a minimal and complete set of such least general generalizers exists so that any other generalizer has at least one of them as an instance modulo the considered set E of axioms. The incomparable elements in such a set are called E-lggs. For instance, for the case of a set B of equations consisting of any combination of associativity and commutativity axioms for different function symbols, the set of B-lggs is finite. However, as shown in (Cerna and Kutsia, 2020), this may not be the case when B contains identity axioms, i.e., ACU, AU, CU, and U.

The generalization type of an equational theory is defined similarly (but dually) to the unification types, i.e., based on the existence and cardinality of a minimal and complete set of B-lggs (Cerna and Kutsia, 2020):

Unitary (type 1) Any generalization problem in the theory has one single B-lgg.

Finitary (type ω) Any generalization problem in the theory has a finite, minimal, and complete set of B-lggs whose cardinality is greater than one for at least one problem.

Infinitary (type ∞) There exists a generalization problem in the theory which has a minimal, infinite, and complete set of B-lggs.

Nullary (type 0) There exists a generalization problem in the theory which does not have a minimal and complete set of B-lggs, i.e., every complete set of generalizers for this problem contains two distinct generalizers such that one is less general modulo B than the other.

In the motivating example of Section 2 of (Cerna and Kutsia, 2020), it was shown that our inference system in (Alpuente et al., 2014b) is incomplete for the case of the identity.

Example 1 Let us assume the unsorted case where all symbols correspond to a unique sort S, and consider two binary function symbols f and g such that[1] f has an identity element e_f (i.e., for all x, $f(x, e_f) = x$ and $f(e_f, x) = x$). Given three extra constants a, b, and c, and the generalization problem $g(f(a, c), a) \overset{w}{\triangleq} g(c, b)$, where w stands for a variable denoting the computed generalizer, the algorithm of (Alpuente et al., 2014b) computes the generalizer given by $\{w \mapsto g(f(x, c), y)\}$, where x and y are new variables. However, the algorithm of (Cerna and Kutsia, 2020) computes the more specific, non-linear solution $g(f(x, c), f(x, y))$, where x and y are new variables, which is actually the (only) least general B-generalizer for the input problem, proving that our original algorithm was generally incomplete. Nevertheless, completeness holds for linear generalization problems as shown in (Cerna and Kutsia, 2020).

In (Cerna and Kutsia, 2020), it was also proved that generalization with identity axioms is nullary, meaning that a complete and minimal set of least general generalizers for a given generalization problem may not exist, not even infinite.

Example 2 Consider again the signature of Example 1, where the function symbol g is now given an additional identity element e_g. As shown in (Cerna and Kutsia, 2020), $f(g(x, y), x)$ is a least general generalizer for the generalization problem $e_f \overset{w}{\triangleq} e_g$ since both e_f and e_g are substitution instances of $f(g(x, y), x)$ by $\{x \mapsto e_f, y \mapsto e_g\}$ and $\{x \mapsto e_g, y \mapsto e_f\}$, respectively. Actually, (Cerna and Kutsia, 2020) shows that there is an infinite number of generalizers that are obtained by instantiating variable x of each generalizer to the term $f(g(x, y), x)$ itself, yielding the sequence of generalizers x, $f(g(x, y), x)$, $f(g(f(g(x, y), x), y), f(g(x, y), x))$, etc. Note that each generalizer is less general (i.e., more specific) than the previous one so that a minimal and complete set of B-lggs cannot be distilled from this infinite sequence of generalizers, not even infinite.

Order-sorted specifications include many-sorted ones and these, in turn, include unsorted ones as special cases. Furthermore, the case of equational axioms B includes the free case $B = \emptyset$ as a special instance. In such a highly general setting, it might appear that finding practical and widely applicable conditions under which order-sorted generalization modulo axioms B containing any combination of associativity, commutativity, *and* identity axioms is *finitary* could be a challenging problem. Yet, paradoxically, we have found out that the very generality of the *order-sorted* setting makes solving this problem much easier than in the unsorted

[1] Function g has also an identity element e_g in Cerna and Kutsia's counter-example but it is unnecessary for the counter-example to work. We introduce e_g only in Example 2 to illustrate the nullary type.

case, and substantially reduces the chances of the problem appearing in actual practice. This paper identifies such a widely applicable condition in a syntactic way and provides the desired finitary order-sorted least general equational generalization algorithm under such an easily checkable assumption. Below, we explain our syntactic condition and show why it is widely applicable in practice by means of some examples.

Recall that an order-sorted signature is a pair $\Sigma = ((S, \leq), F)$, where $(S, \leq)$ is a partially ordered set of sorts/types, where $s \leq s'$ specifies a sort/type inclusion, and F are function symbols whose argument sorts and result sort belong to S. For example, the sorts `Nat`, `Int`, and `Rat` for, respectively, naturals, integers and rationals have natural subsort inclusion `Nat` < `Int` < `Rat`, which semantically correspond to the set inclusions $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q}$. Since the nullarity problem of Example 2 is caused by (non-linear) B-generalizers modulo identity of binary operators f and g, note that these problems are less likely to occur in a many-sorted setting for the simple reason that, when the respective unit elements e_f and e_g have different sorts, no generalizers exist at all. The order-sorted setting is even more flexible. Our easily checkable syntactic condition is as follows: a signature $\Sigma = ((S, \leq), F)$ is called *U-tolerant* modulo axioms B, where $U \subseteq B$ denotes the set of identity axioms, if and only if for any two different identity constants e_f and e_g with identity axioms for f and g in U, their respective *least sorts*[2] $LS(e_f)$ and $LS(e_g)$ are *incomparable* at the kind level, that is, $[LS(e_f)] \neq [LS(e_g)]$. The naturalness of this property is later illustrated in Example 5.

The contributions of this paper are the following:

(1) We identify U-tolerance as a mild syntactic condition on signatures frequently achievable in practice, as explained above and in Section 3.2. As further evidence of the wide applicability of this concept, we have verified that the large collection of examples considered in (Alpuente et al., 2019) —where an efficient implementation of (Alpuente et al., 2014b) called ACUOS2 was provided— are all U-tolerant except for just one of the (untyped) generalization problems, which is both non-linear and not U-tolerant and combines A and U.
(2) We complete the generalization calculi of (Alpuente et al., 2014b) with an extra inference rule so that completeness holds for both linear and non-linear order-sorted equational generalizations under the U-tolerance assumption.
(3) We show that, under the U-tolerance assumption, the completed calculus provides a finitary, minimal and complete set of order-sorted generalizers modulo axioms B of associativity and/or commutativity and/or identity for any generalization problem.
(4) We provide a new implementation for our extended generalization calculus.

To the best of our knowledge, this is the first finite, minimal, and complete procedure for order-sorted least general equational generalization modulo any combinations of associativity, commutativity, and identity axioms (i.e., A, C, U, AC, AU, CU, and ACU) for different function symbols. Furthermore, it not only works for unsorted and many-sorted specifications, but it also works for the more general and expressive case of order-sorted specifications.

[2] An easily checkable condition on $\Sigma = ((S, \leq), F)$ called *pre-regularity* ensures that any Σ-term t has a least sort $LS(t) \in S$ in the subsort ordering (see Section 2).

2 Preliminaries

We follow the classical notation and terminology from (TeReSe, 2003) for term rewriting and from (Goguen and Meseguer, 1992; Meseguer, 1997) for order-sorted equational logic.

We assume an *order-sorted signature* $\Sigma = (S, F, \leq)$ that consists of a finite poset of sorts $(\mathsf{S}, \leq)$ and a family F of function symbols of the form $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s}$, with $\mathsf{s}_1, \ldots, \mathsf{s}_n, \mathsf{s} \in \mathsf{S}$. Two sorts s and s' belong to the same connected component if either $\mathsf{s} \leq \mathsf{s}'$ or $\mathsf{s}' \leq \mathsf{s}$. We assume a *kind-completed signature* such that: (i) each connected component in the poset ordering has a top sort, and, for each $\mathsf{s} \in \mathsf{S}$, we denote by $[\mathsf{s}]$ the top sort in the connected component of s (*i.e.,* if s and s' are sorts in the same connected component, then $[\mathsf{s}] = [\mathsf{s}']$); and (ii) for each operator declaration $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s}$ in Σ, there is also a declaration $f : [\mathsf{s}_1] \times \ldots \times [\mathsf{s}_n] \to [\mathsf{s}]$ in Σ. A given term t in an order-sorted term algebra can have many different sorts. Specifically, if $t \in T_\Sigma$ has sort s, then it also has sort s' for any $\mathsf{s}' \geq \mathsf{s}$; and because a function symbol f can have different sort declaration $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s}$, a term $f(t_1, .., t_n)$ can have sorts that are not directly comparable (Goguen and Meseguer, 1992).

We assume a fixed S-sorted family $\mathcal{X} = \{\mathcal{X}_\mathsf{s}\}_{\mathsf{s} \in \mathsf{S}}$ of pairwise disjoint variable sets (*i.e.,* $\forall \mathsf{s}, \mathsf{s}' \in \mathsf{S} : \mathcal{X}_\mathsf{s} \cap \mathcal{X}_{\mathsf{s}'} = \emptyset$), with each $\mathcal{X}_\mathsf{s}$ being countably infinite. We write the sort associated to a variable explicitly with a colon and the sort, *i.e.,* x:Nat. A *fresh* variable is a variable that appears nowhere else. The set $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ denotes all Σ-terms of sort s defined by $\mathcal{X}_\mathsf{s} \subseteq \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ and $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ if $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_n \to \mathsf{s} \in \Sigma$, $n \geq 0$ and $t_1 \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}_1}, \ldots, t_n \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}_\mathsf{n}}$. Furthermore, if $t \in \mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ and $s \leq s'$, then $t \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}'}$. For a term t, we write $\mathit{Var}(t)$ for the set of all variables in t. $\mathcal{T}_{\Sigma,\mathsf{s}}$ is the set of ground terms of sort s, *i.e.,* t is a Σ-term of sort s and $\mathit{Var}(t) = \emptyset$. We write $\mathcal{T}_\Sigma(\mathcal{X})$ and $\mathcal{T}_\Sigma$ for the corresponding term algebras. We assume that $\mathcal{T}_{\Sigma,\mathsf{s}} \neq \emptyset$ for every sort s.

We assume *pre-regularity* of the signature Σ: for each operator declaration $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_\mathsf{n} \to \mathsf{s}$, and for the set S_f containing all sorts s' that appear in operator declarations of the form $f : \mathsf{s}'_1, \ldots, \mathsf{s}'_n \to \mathsf{s}'$ in Σ such that $\mathsf{s}_i \leq \mathsf{s}'_i$ for $1 \leq i \leq n$, then the set S_f has a least sort. Thanks to pre-regularity of Σ, each Σ-term t has a *unique least sort* that is denoted by $LS(t)$. The top sort in the connected component of $LS(t)$ is denoted by $[LS(t)]$. Since the poset $(\mathsf{S}, \leq)$ is finite and each connected component has a top sort, given any two sorts s and s' in the same connected component, the set of least upper bound sorts of s and s' always exists (although it might not be a singleton set) and is denoted by $LUBS(\mathsf{s}, \mathsf{s}')$.

Throughout this paper, we assume that Σ has no *ad-hoc operator overloading*, *i.e.,* any two operator declarations for the same symbol f with equal number of arguments, $f : \mathsf{s}_1 \times \ldots \times \mathsf{s}_\mathsf{n} \to \mathsf{s}$ and $f : \mathsf{s}'_1 \times \ldots \times \mathsf{s}'_n \to \mathsf{s}'$, must necessarily have $[\mathsf{s}_1] = [\mathsf{s}'_1], \ldots, [\mathsf{s}_n] = [\mathsf{s}'{}_n], [\mathsf{s}] = [\mathsf{s}']$.

The set of positions of a term t, written $\mathit{Pos}(t)$, is represented as a sequence of natural numbers, *e.g.,* 1.2.1. The set of non-variable positions is written $\mathit{Pos}_\Sigma(t)$. The root position of a term is Λ. The subterm of t at position p is $t|_p$, and $t[u]_p$ is the term obtained from t by replacing $t|_p$ by u. By $\mathit{root}(t)$, we denote the symbol occurring at the root position of t.

A *substitution* $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ is a mapping from variables to terms which is almost everywhere equal to the identity except over a finite set of variables $\{x_1, \ldots, x_n\}$, written $\mathit{Dom}(\sigma) = \{x \in \mathcal{X} \mid x\sigma \neq x\}$. Substitutions are *sort–preserving*, *i.e.,* for any substitution σ, if $x \in \mathcal{X}_{\mathsf{s}}$, then $x\sigma \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\mathsf{s}}$. We assume substitutions are idempotent, *i.e.,* $x\sigma = (x\sigma)\sigma$ for any variable x. The set of variables introduced by σ is $VRan(\sigma) = \bigcup\{Var(x\sigma) \mid x\sigma \neq x\}$. The identity substitution is *id*. Substitutions are homomorphically extended to $\mathcal{T}_{\Sigma}(\mathcal{X})$. Substitutions are written in suffix notation (*i.e.,* $t\sigma$ instead of $\sigma(t)$), and, consequently, the composition of substitutions must be read from left to right, formally denoted by juxtaposition, *i.e.,* $x(\sigma\sigma') = (x\sigma)\sigma'$ for any variable x. The restriction of σ to a set of variables V is $\sigma|_V$. We call a substitution σ a *renaming* if there is another substitution σ^{-1} such that $(\sigma\sigma^{-1})|_{Dom(\sigma)} = id$.

A *Σ-equation* is an unoriented pair $t \doteq t'$, where t and t' are Σ-terms for which there are sorts s, s' with $t \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\mathsf{s}}$, $t' \in \mathcal{T}_{\Sigma}(\mathcal{X})_{\mathsf{s}'}$, and s, s' are in the same connected component of the poset of sorts $(S, \leq)$. An *equational theory* (Σ, B) is a set B of Σ-equations. An *equational theory* (Σ, B) over a kind-completed, pre-regular, and order-sorted signature $\Sigma = (S, F, \leq)$ is called kind-completed, pre-regular, and order-sorted equational theory. Given an equational theory (Σ, B), order-sorted equational logic induces a congruence relation $=_B$ on terms $t, t' \in \mathcal{T}_{\Sigma}(\mathcal{X})$, see (Goguen and Meseguer, 1992; Meseguer, 1997).

The *B-subsumption* preorder $\leq_B$ (simply $\leq$ when B is empty) holds between $t, t' \in \mathcal{T}_{\Sigma}(\mathcal{X})$, denoted $t \leq_B t'$ (meaning that t is more general than t' modulo B), if there is a substitution σ such that $t\sigma =_B t'$; such a substitution σ is said to be a *B-matcher* for t' in t. The equivalence relation $\equiv_B$ (or $\equiv$ if B is empty) induced by $\leq_B$ is defined as $t \equiv_B t'$ if $t \leq_B t'$ and $t' \leq_B t$. The B-renaming equivalence $t \simeq_B t'$ (or $\simeq$ if B is empty) holds if there is a renaming substitution θ such that $t\theta =_B t'$. In general, the relations $=_B$, $\equiv_B$ and $\simeq_B$ do not coincide; actually $=_B \subseteq \simeq_B \subseteq \equiv_B$.

Example 3 Consider terms $t = f(f(a, X), Y)$ and $t' = f(a, Z)$ where f is associative and commutative with identity symbol 0 (ACU), and a and b are two constants. We have that $t \equiv_{ACU} t'$, *i.e.,* $t \leq_{ACU} t'$ and $t' \leq_{ACU} t$ since $f(f(a, X), Y)\sigma_1 =_{ACU} f(a, Z)$ with $\sigma_1 = \{X \mapsto 0, Y \mapsto Z\}$ and $f(a, Z)\sigma_2 =_{ACU} f(f(a, X), Y)$ with $\sigma_2 = \{Z \mapsto f(X, Y)\}$. However, $t \neq_{ACU} t'$, and moreover $t \not\simeq_{ACU} t'$, since they are not even equal up to ACU-renaming.

For the sake of simplicity, we follow the common approach of order-sorted equational languages such as Maude where the signature Σ is considered to be aware of $f's$ axioms by attaching to its sort declaration special attributes denoting that the function f obeys associativity (`assoc`), commutativity (`comm`), and identity (`id: e`), for unit element `e`.

3 Order-Sorted Least General Generalizations modulo Axioms

In this section, we complete the inference system for order-sorted, equational least general generalization presented in (Alpuente et al., 2014b) by introducing an extra inference rule. We ascertain a suitable requirement (called *U-tolerance*) ensuring that generalization modulo identity is finitary, and moreover, that our extended algorithm computes a finite, minimal, and complete set of B-lggs modulo any combinations of associativity, commutativity and identity axioms.

3.1 Recovering completeness of the order-sorted equational least general generalization calculus

In the following, we consider that each function symbol f in the signature Σ obeys a subset of axioms $ax(f) \subseteq \{A_f, C_f, U_f(e)\}$ where e is the identity symbol for the function f. Note that f may not satisfy any such axioms, *i.e.,* $ax(f) = \emptyset$.

A term t is a generalizer modulo B of t_1 and t_2 if there are two substitutions σ_1 and σ_2 such that $t\sigma_1 =_B t_1$ and $t\sigma_2 =_B t_2$.

We represent a generalization problem between terms t and t' as a *constraint* $t \overset{x}{\triangleq} t'$, where x is a fresh variable that stands for a generalizer of t and t', that becomes more and more instantiated as the computation proceeds until becoming a least general generalizer modulo the considered axioms. Given a constraint $t \overset{x}{\triangleq} t'$, any generalizer w of t and t' is given by a suitable substitution θ such that $x\theta = w$.

A set of constraints is represented by $s_1 \overset{x_1}{\triangleq} t_1 \wedge \ldots \wedge s_n \overset{x_n}{\triangleq} t_n$, or $\emptyset$ for the empty set. Given a constraint $t \overset{x}{\triangleq} t'$, we call x an *index variable.* We define the set of index variables of a set C of constraints as $\mathit{Index}(C) = \{y \in \mathcal{X} \mid \exists u \overset{y}{\triangleq} v \in C\}$.

Note that, although it is natural to consider that a constraint $t \overset{x}{\triangleq} t'$ is commutative, the inference rules that are described in this paper do not admit that commutativity property for $\triangleq$ since we need to keep track of the origin of new generated generalization subproblems to avoid non-termination. However, the constructor symbol $\wedge$ that we use to build a set (conjunction) of constraints is *associative* and *commutative* in the inference rules described in this paper.

Definition 1 A configuration $\langle C \mid S \mid \theta\rangle$ consists of three components: (i) the *constraint component* C, which represents the *set of unsolved constraints*; (ii) the *store component* S, which records the *set of already solved constraints*, and (iii) the *substitution component* θ, which binds some of the index variables previously met during the computation.

We consider any two terms t and t' in a constraint $t \overset{x}{\triangleq} t'$ having the same top sort; otherwise, they are incomparable and no generalizer exists. Starting from the initial configuration $\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \mid \emptyset \mid id\rangle$ where $[\mathsf{s}] = [LS(t)] = [LS(t')]$, configurations are transformed until a terminal configuration $\langle \emptyset \mid S \mid \theta\rangle$ is reached. When different function symbols are considered that satisfy distinct combinations of associativity and/or commutativity and/or identity axioms given by B, the inference rules of Figures 1, 2, 3, 5, 6, and 7 must be used altogether to compute the set of B-lggs. The new inference rule solving the generalization problem of Example 1 is given in Figure 3; all of the other rules are taken from (Alpuente et al., 2014b) and are included here for completeness. The transition relation $\to$ on configurations is given by the smallest relation satisfying all of the rules.

Roughly speaking, given an equational theory (Σ, B) and the generalization problem $t \overset{x}{\triangleq} t'$, the basic rules **Decompose**$_B$, **Solve**$_B$, and **Recover**$_B$ in Figure 1 extend to the (order-sorted) equational setting the standard, syntactic generalization of (Huet, 1976; Plotkin, 1970; Reynolds, 1970) by handling the constraints modulo B. The meaning of the basic rules in Figure 1 is as follows.

$$\textbf{Decompose}_B \qquad \frac{f \in (\Sigma \cup \mathcal{X}) \wedge A_f \notin ax(f) \wedge C_f \notin ax(f) \wedge f : [\mathsf{s_1}] \times \ldots \times [\mathsf{s_n}] \to [\mathsf{s}]}{\begin{array}{c}\langle f(t_1, \ldots, t_n) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1, \ldots, t'_n) \wedge C \mid S \mid \theta \rangle \to \\ \langle t_1 \overset{x_1:[\mathsf{s_1}]}{\triangleq} t'_1 \wedge \ldots \wedge t_n \overset{x_n:[\mathsf{s_n}]}{\triangleq} t'_n \wedge C \mid S \mid \theta\sigma \rangle\end{array}}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s_1}], \ldots, x_n{:}[\mathsf{s_n}])\}$, $x_1{:}[\mathsf{s_1}], \ldots, x_n{:}[\mathsf{s_n}]$ are fresh variables, and $n \geq 0$

$$\textbf{Solve}_B \qquad \frac{\begin{array}{c}f = root(t) \wedge g = root(t') \wedge f \neq g \wedge U_f(e) \notin ax(f) \wedge U_g(e') \notin ax(g) \wedge \\ s' \in LUBS(LS(t), LS(t')) \wedge \nexists y \nexists \mathsf{s}'' : t \overset{y:\mathsf{s}''}{\triangleq} t' \in^B S\end{array}}{\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \wedge C \mid S \mid \theta \rangle \to \langle C \mid S \wedge t \overset{z:\mathsf{s}'}{\triangleq} t' \mid \theta\sigma \rangle}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto z{:}\mathsf{s}'\}$, $z{:}\mathsf{s}'$ is a fresh variable, and $(t \overset{y}{\triangleq} t') \in^B S$ means that there exists $u \overset{y}{\triangleq} u' \in S$ such that $t =_B u$ and $t' =_B u'$.

$$\textbf{Recover}_B \qquad \frac{root(t) \neq root(t') \wedge \exists y \exists \mathsf{s}' : t \overset{y:\mathsf{s}'}{\triangleq} t' \in^B S}{\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \wedge C \mid S \mid \theta \rangle \to \langle C \mid S \mid \theta\sigma \rangle}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto y{:}\mathsf{s}'\}$, and $\in^B$ stands for membership modulo axioms.

Fig. 1: Basic inference rules for order-sorted least general B–generalization (Alpuente et al., 2014b)

- The **Decompose**$_B$ rule decomposes a constraint $f(t_1, \ldots, t_n) \overset{x}{\triangleq} f(t'_1, \ldots, t'_n)$ into new constraints $t_1 \overset{x_1}{\triangleq} t'_1 \wedge \ldots \wedge t_n \overset{x_n}{\triangleq} t'_n$ to be solved provided that f does not obey either associativity or commutativity axioms. If $n = 0$, then no constraints are generated. Note that this rule can be applied if f has an identity symbol e, *i.e.,* $U_f(e) \in ax(f)$, and even if f is the identity element of another symbol. Additional, specialized decomposition rules are given in Figures 5, 6, and 7 for the case when the root function symbol f obeys C, A, or AC, respectively. Note that there is no inference rule for the ACU case, it is just a combination of the rules that are applicable to AC and U.
- The **Solve**$_B$ rule moves a constraint $t \overset{x}{\triangleq} t'$, with $root(t) \neq root(t')$, to the store only when there is no constraint in S of the form $u \overset{y}{\triangleq} u'$ for two terms u and u' that are respectively B-equal to t and t'. Note that rule **Solve**$_B$ does not apply to a constraint $t \overset{x}{\triangleq} t'$ such that either t or t' are rooted by a function symbol f with $U_f(e) \in ax(f)$, since it is given a more specialized treatment in the rule **Expand**$_B$ of Figure 2. If f or g are identity symbols of other symbols, the rule is applied in the same way.
- The **Recover**$_B$ rule checks whether there is an already solved constraint $u \overset{y}{\triangleq} u'$ in S for two terms u and u' that are respectively B-equal to t and t'. Then, the previously computed generalizer given by the variable y is reused. This allows us to handle common generalization subproblems that may appear more than once, *e.g.,* the least general generalizer of $f(f(a, a), a)$ and $f(f(b, b), a)$ is $f(f(y, y), a)$. Note that this rule may overlap with the specialized **Recover**$_U$ rule of Figure 3 and both rules would be non-deterministically applied.

$$\textbf{Expand}_U \quad \dfrac{\begin{array}{c} c \equiv (t \overset{x:[\mathsf{s}]}{\triangleq} t') \text{ (resp. } c \equiv (t' \overset{x:[\mathsf{s}]}{\triangleq} t)) \wedge \\ f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge U_f(e) \in ax(f) \wedge root(t) \equiv f \wedge t' \neq_B e \wedge root(t') \neq f \wedge \\ t'' \in \{f(e,t'), f(t',e)\} \wedge \\ c' \equiv (t \overset{x:[\mathsf{s}]}{\triangleq} t'') \text{ (resp. } c' \equiv (t'' \overset{x:[\mathsf{s}]}{\triangleq} t)) \end{array}}{\langle c \wedge C \mid S \mid \theta \rangle \to \langle c' \wedge C \mid S \mid \theta \rangle}$$

Fig. 2: Order-sorted inference rule for expanding a term t using a function f with identity element e (Alpuente et al., 2014b)

$$\textbf{Recover}_U \quad \dfrac{\begin{array}{c} c \equiv (t \overset{x:[\mathsf{s}]}{\triangleq} t') \text{ (resp. } c \equiv (t' \overset{x:[\mathsf{s}]}{\triangleq} t)) \wedge \\ \exists y \exists \mathsf{s}' : t \overset{y:\mathsf{s}'}{\triangleq} e \in^B S \text{ (resp. } e \overset{y:\mathsf{s}'}{\triangleq} t \in^B S) \wedge U_f(e) \in ax(f) \wedge f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge \\ c' \equiv (e \overset{z:[\mathsf{s}]}{\triangleq} t') \text{ (resp. } c' \equiv (t' \overset{z:[\mathsf{s}]}{\triangleq} e)) \end{array}}{\langle c \wedge C \mid S \mid \theta \rangle \to \langle c' \wedge C \mid S \mid \theta\sigma \rangle}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(y{:}\mathsf{s}', z{:}[\mathsf{s}])\}$ or $\sigma = \{x{:}[\mathsf{s}] \mapsto f(z{:}[\mathsf{s}], y{:}\mathsf{s}')\}$

Fig. 3: Order-sorted inference rule for recovering a partially computed generalizer for a term t and the identity element e of f

Special rules are introduced for dealing with constraints that may involve an identity axiom:

- The **Expand**$_U$ rule in Figure 2 allows any constraint $f(t_1,t_2) \overset{x}{\triangleq} t$ to be reduced to the constraint $f(t_1,t_2) \overset{x}{\triangleq} f(t,e)$ (or $f(t_1,t_2) \overset{x}{\triangleq} f(e,t)$) whenever f has an identity element e and $root(t) \neq f$. The rule is applied also when the constraint has the form $t \overset{x}{\triangleq} f(t_1,t_2)$.
- The new inference rule **Recover**$_U$ in Figure 3 applies to any constraint $t \overset{x}{\triangleq} t'$ such that either $t \overset{y}{\triangleq} e$ or $e \overset{y}{\triangleq} t$ has been respectively stored in S, with e being the identity element of a given function f of the signature. Then, the new constraint $e \overset{z:[\mathsf{s}]}{\triangleq} t'$ (resp. $t' \overset{z:[\mathsf{s}]}{\triangleq} e$) is added to C in order to be solved in a subsequent step, while the index variable y from the recovered, previously solved common subproblem is reused to instantiate x into $f(y,z)$ (resp. $f(z,y)$). Note that this is equivalent to first introduce a new, intermediate constraint $f(t,e) \overset{x}{\triangleq} f(e,t')$ (resp. $f(e,t) \overset{x}{\triangleq} f(t',e)$) by exploiting $U_f(e)$, and then derive the added constraints $e \overset{z:[\mathsf{s}]}{\triangleq} t'$ (resp. $t \overset{z:[\mathsf{s}]}{\triangleq} e$) by decomposing this intermediate constraint.

Let us briefly discuss why the original calculus (without the **Recover**$_U$ rule of Figure 3) is incomplete.

Example 4 Let us again consider the generalization problem $g(f(a,c),a) \overset{w}{\triangleq} g(c,b)$ of Example 1. Our extended algorithm computes the least general generalizer by means of the computation sequence of Figure 4, which applies the new rule **Recover**$_U$ at the penultimate step by exploiting the fact that $g(f(a,c),a)$ is equal modulo identity to $g(f(a,c),f(a,e_f))$. Note that, without the new rule **Recover**$_U$, there is no way to generalize the constants a and b in the second argument of g to its least general generalizer $f(w_{11}, w_{22})$ modulo identity. Also, for the generalization problem $g(f(a,c), f(a,e_f)) \overset{w}{\triangleq} g(f(e_f,c), f(e_f,b))$, these rules allow us to identify the generalization sub-problem $a \overset{w_{11}}{\triangleq} e_f$ at two different locations, assigning them the same variable w_{11}.

$$\begin{array}{lr}
\langle g(f(a,c),a) \overset{w}{\triangleq} g(c,b) \mid \emptyset \mid id\rangle & \\
\to \langle f(a,c) \overset{w_1}{\triangleq} c \wedge a \overset{w_2}{\triangleq} b \mid \emptyset \mid \{w \mapsto g(w_1,w_2)\}\rangle & (\mathbf{Decompose}_B) \\
\to \langle f(a,c) \overset{w_1}{\triangleq} f(e_f,c) \wedge a \overset{w_2}{\triangleq} b \mid \emptyset \mid \{w \mapsto g(w_1,w_2)\}\rangle & (\mathbf{Expand}_U) \\
\to \langle a \overset{w_{11}}{\triangleq} e_f \wedge c \overset{w_{12}}{\triangleq} c \wedge a \overset{w_2}{\triangleq} b \mid \emptyset \mid \{w \mapsto g(f(w_{11},w_{12}),w_2)\}\rangle & (\mathbf{Decompose}_B) \\
\to \langle c \overset{w_{12}}{\triangleq} c \wedge a \overset{w_2}{\triangleq} b \mid a \overset{w'_{11}}{\triangleq} e_f \mid \{w \mapsto g(f(w'_{11},w_{12}),w_2)\}\rangle & (\mathbf{Solve}_B) \\
\to \langle a \overset{w_2}{\triangleq} b \mid a \overset{w'_{11}}{\triangleq} e_f \mid \{w \mapsto g(f(w'_{11},c),w_2)\}\rangle & (\mathbf{Decompose}_B) \\
\to \langle e_f \overset{w_{22}}{\triangleq} b \mid a \overset{w'_{11}}{\triangleq} e_f \mid \{w \mapsto g(f(w'_{11},c),f(w'_{11},w_{22}))\}\rangle & (\mathbf{{*}Recover}_U) \\
\to \langle \emptyset \mid a \overset{w'_{11}}{\triangleq} e_f \wedge e_f \overset{w'_{22}}{\triangleq} b \mid \{w \mapsto g(f(w'_{11},c),f(w'_{11},w'_{22}))\}\rangle & (\mathbf{Solve}_U)
\end{array}$$

Fig. 4: Execution of our inference rules for the generalization problem $g(f(a,c),a) \overset{w}{\triangleq} g(c,b)$

Termination was straightforward in the original calculus of (Alpuente et al., 2014b), but the addition of the new rule $\mathbf{Recover}_U$ in Figure 3 breaks down termination unless suitable restrictions are imposed. Indeed, (Cerna and Kutsia, 2020) proves that generalization with more than one identity symbol becomes nullary, i.e., the existence of a minimal and complete set of least general generalizers is not guaranteed (not even infinite), and two shortcut solutions are proposed: restricting to linear generalization problems for completeness (i.e., to compute a finite, minimal and complete set of generalizers) and restricting to (unsorted) one-unit signatures (i.e., with a unique function obeying identity) for finiteness of the set of B-lggs, even though their algorithm may generate an infinite complete set of generalizers. In the following section, we revisit the one-unit restriction and ascertain a more general notion that not only guarantees non-nullarity but also finiteness of the set of solutions so that our extended algorithm computes a finite, complete and minimal set of least general equational generalizers.

3.2 Ensuring finiteness of least general generalization modulo identity

The only inference rule that can be applied to the generalization problem $e_f \overset{w}{\triangleq} e_g$ of Example 2 using our original order-sorted, equational least general generalization calculus in (Alpuente et al., 2014b) is the rule **Solve**$_B$ in Figure 1. However, the extended calculus proposed in Subsection 3.1 does not terminate on this problem as shown by any of the following non-terminating computations that infinetely apply the inference rule **Recover**$_U$: $\langle e_f \overset{x}{\triangleq} e_g \mid e_f \overset{y}{\triangleq} e_g \mid \theta\rangle \to \langle e_f \overset{x'}{\triangleq} e_g \mid e_f \overset{y}{\triangleq} e_g \mid \theta\{x \mapsto f(y,x')\}\rangle \cdots$, and $\langle e_f \overset{x}{\triangleq} e_g \mid e_f \overset{y}{\triangleq} e_g \mid \theta\rangle \to \langle e_f \overset{x'}{\triangleq} e_g \mid e_f \overset{y}{\triangleq} e_g \mid \theta\{x \mapsto g(y,x')\}\rangle \cdots$.

In the following, we generalize to our order-sorted equational setting the restriction to one-unit signatures of (Cerna and Kutsia, 2020). This is done by formalizing a new notion, called U-tolerance, which focuses on the generalization problems themselves so that U-tolerance may hold for equational theories that do not satisfy Cerna and Kutsia's condition provided the connected component of sorts that corresponds to the problem subsignature is one-unit.

Note that for any binary function symbol obeying any combination of A, C, and U axioms (except for just C), the top sort of both arguments coincides with the top sort of the result.

Definition 2 (U-tolerant signature of generalization problems) Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B), and a generalization problem Γ, consider the restriction $\Sigma_\Gamma = (S, F, \leq)$ of Σ to the function symbols of Γ. Σ_Γ is called *U-tolerant* if it does not contain two different function symbols $f : \mathsf{s_f} \times \mathsf{s_f} \to \mathsf{s_f}$ and $g : \mathsf{s_g} \times \mathsf{s_g} \to \mathsf{s_g}$ with different identity symbols e_f and e_g, respectively, such that $[LS(e_f)] = [LS(e_g)]$.

We may simply say that the whole signature Σ is *U-tolerant* when no generalization problem Γ is made explicit and Σ satisfies the U-tolerance condition.

Note that the one-unit requirement of (Cerna and Kutsia, 2020) implies U-tolerance. In the following we show how our more relaxed, yet syntactic, U-tolerance condition is widely applicable in practice by means of some examples.

Example 5 Consider the following Maude functional module that defines lists and multisets of natural numbers:

```
fmod LIST+MSET-NO-U-tolerant is
 sorts Nat List MSet Top .
 subsorts Nat < List < Top .
 subsorts Nat < MSet < Top .
 op nil : -> List [ctor] .
 op _;_ : List List -> List [ctor assoc id: nil] .
 op null : -> MSet [ctor] .
 op _,_ : MSet MSet -> MSet [ctor assoc comm id: null] .
 op 0 : -> Nat [ctor] .
 op s : Nat -> Nat [ctor] .
endfm
```

where the `ctor` declarations specify that all of these operators are *data constructors*, as opposed to defined functions such as list reverse, or multiset cardinality, and the axioms B are specified by the `assoc`, `comm`, and `id:` keywords. Note that, since `Nat < List` and `Nat < MSet`, 0 is both a list of length one and a singleton multiset, but the list `0 ; s(0) ; s(s(0))` and the multiset `0 , s(0) , s(s(0))` have incomparable least sorts `List` and `MSet`. Moreover these are also the least sorts of `nil` and `null`, respectively. However, this signature is not *U-tolerant*, since the kind of `List` and `MSet` coincide, i.e., the sort `Top` added for kind-completeness.

However, it is easy to repair it by using extra symbols [_] and {_} to encapsulate a natural as an element of a list or multiset, respectively.

```
fmod LIST+MSET-U-tolerant is
 sorts Nat List MSet .
 op [_] : Nat -> List .
 op {_} : Nat -> MSet .
 op nil : -> List [ctor] .
 op _;_ : List List -> List [ctor assoc id: nil] .
 op null : -> MSet [ctor] .
 op _,_ : MSet MSet -> MSet [ctor assoc comm id: null] .
 op 0 : -> Nat [ctor] .
 op s : Nat -> Nat [ctor] .
endfm
```

The former list and multiset are now represented as [0] ; [s(0)] ; [s(s(0))] and {0} , {s(0)} , {s(s(0))}. They have incomparable least sorts List and MSet and kinds [List] and [MSet], which coincide with the least sorts and kinds of nil and null. Therefore, this signature is not one-unital in the sense of (Cerna and Kutsia, 2020) yet it is *U-tolerant*, and thus any generalization problem in this theory has a finite set of least general generalizers.

Besides order-sortedness, there are two additional reasons why, in practice, the U-tolerance requirement is a mild one.

First of all, most reasoning about generalization modulo B happens in the context of algebraic specifications in which operators are naturally classified into *constructor symbols* and *defined function symbols*. The point is that the use of axioms B is much more important for constructor symbols —allowing us to define data structures such as the lists and multisets above— than for defined functions, where they are not really needed, although they may be useful to have. This distinction can help us totally avoid lacks of U-tolerance that would seem unavoidable, such as the possible conflict between identity elements 0 and 1 for $+$ and $*$ in the natural numbers as illustrated in the following example.

Example 6 Consider the following Maude functional module that defines addition and multiplication of natural numbers:

```
fmod NAT-ACU .
 sort Nat .
 ops 0 1 : -> Nat [ctor] .
 op _+_ : Nat Nat -> Nat [ctor assoc comm id: 0] .
 op _*_ : Nat Nat -> Nat [assoc comm] .
 vars N M : Nat .
 eq N * 0 = 0 .
 eq N * (1 + M) = N + (N * M) .
endfm
```

The constructor terms for this example are the additive monoid with identity 0 generated by 1. This means that any binary function f can be defined by recursive equations of the form $f(u_1, u_2) = t$, where the u_1, u_2 are constructor terms. Therefore, f itself will never occur in the u_1, u_2. This implies that even if f has an identity element e_f, specifying such an f with an identity axiom is entirely useless. This is indeed the case for the multiplication function, even though associativity and commutativity are still useful multiplicative axioms, and even if the identity property *is* provable from the above equations:

$$x * 1 =_B x * (1 + 0) = x + (x * 0) = x + 0 =_B x$$

Note that adding `NAT-ACU` to `LIST+MSET-U-tolerant` does not change the U-tolerant property, since there are three connected components `Nat`, `List`, and `MSet`.

The second reason why U-tolerance can be achieved in practice in cases where its failure might seem unavoidable can also be illustrated by a simple example.

Example 7 The U-tolerance of the `LIST+MSET-NO-U-tolerant` of Example 5 can be easily obtained in a *semantics-preserving manner*. This is so for the following reason. Suppose we have an order-sorted equational theory $\mathcal{E} = (\Sigma, E \cup B)$ specified in Maude as a functional module with axioms B and equations E, which, oriented as rules, are confluent and terminating modulo B. Now let $U_0 \subseteq U \subseteq B$ be a set of identity axioms for some of the operators in Σ. Then, as explained in (Durán et al., 2009), there is an automatic theory transformation $\mathcal{E} \mapsto \mathcal{E}_{U_0}$ such that: (i) $\mathcal{E}$ and $\mathcal{E}_{U_0}$ are equivalent equational theories and therefore have isomorphic initial algebras; (ii) the axioms of $\mathcal{E}_{U_0}$ are $B \setminus U_0$, and the U_0 now becomes additional equations in $\mathcal{E}_{U_0}$; (iii) $\mathcal{E}$ is confluent and terminating iff $\mathcal{E}_{U_0}$ is so. In other words, the transformation $\mathcal{E} \mapsto \mathcal{E}_{U_0}$ is *semantics-preserving* in the strongest way possible.

How would we apply this transformation to our `LIST+MSET-NO-U-tolerant` module? We would just choose the identity axioms of `nil` and `null` as U_0. The effect of the $\mathcal{E} \mapsto \mathcal{E}_{U_0}$ transformation for $\mathcal{E}$ in our `LIST+MSET-NO-U-tolerant` example would be as follows: (i) we would drop the `id: nil` attribute; (ii) we would add the following equations, where `X` and `Y` are of sort `List`:

```
eq nil ; X = X .
eq X ; nil ; Y = X ; Y .
eq Y ; nil = Y .
```

(iii) we would drop the `id: null` attribute; and (iv) we would add the following equations, where `Z` is of sort `MSet`:

```
eq null , Z = Z .
```

The resulting module is a semantically equivalent specification that is now U-tolerant.

As we have discussed so far, the U-tolerance condition is much more powerful than it might appear at first sight. Actually, when order-sorted equational generalization is applied in the context of rewriting logic tools such as program transformers or correctors (Alpuente et al., 2020a,b) (where computing B-lggs is key for ensuring correctness and termination of the transformation), only generalization problems that are normalized w.r.t. the equational theory are considered so that any defined function symbols (together with their identity elements) have been evaluated away prior to generalization.

Regarding the benchmark examples in (Alpuente et al., 2019), we have verified that all of the examples are U-tolerant, except for `synthetic`. The examples in (Alpuente et al., 2019) which contain the identity axiom for some function symbol are the following:

- `spouses` (AU) and `children` (ACU), two classical generalization problems borrowed from the logic programming domain that are described in (Alpuente et al., 2014a);
- `only-U`, a generalization problem without associative or commutative axioms;
- `synthetic`, an involved generalization example mixing U and A axioms for two distinct binary symbols;
- `chemical`, a case-based reasoning problem for chemical compounds inspired by (Armengol, 2007);
- `graph`, the leading example of (Baumgartner et al., 2018); and
- `biological`, a cell model for the analysis of biological systems.

All of the generalization problems `spouses`, `children`, `only-U`, `chemical`, and `biological` are one-unit, hence U-tolerant. As for the `graph` example, it is U-tolerant since the two identity elements in the signature (`empty` for the graph constructor and `nil` for the list constructor) have incomparable least sorts, similarly to Example 5. The only generalization problem that does not fulfill U-tolerance is `synthetic`, which can be seen as a more complex version of Example 1 that is both non-linear and not U-tolerant (actually, it contains two functions f and g with identity elements e_f and e_g, respectively, similarly to Example 2) and combines U with A. As we have shown, in our order-sorted setting we can deal with this problem by either introducing incomparable least sorts for e_f and e_g, or in the case when this would not be appropriate, we could easily achieve U-tolerance of the equational theory E by applying the transformation in (Durán et al., 2009) that we sketched in Example 7.

More precisely, in a real application involving an equational specification $\mathcal{E} = (\Sigma, E \uplus B)$, we are interested to compute least general generalizers w.r.t. (Σ, B). The transformation $\mathcal{E} \mapsto \mathcal{E}_{U_0}$ of (Durán et al., 2009) technically relies on computing variants[3] for the equations in E with the axioms of U_0 oriented as rules. In the transformed specification $\mathcal{E}_{U_0}$, the new set of axioms is $B \setminus U_0$ while the equations in U_0 have been oriented as rules. Formally, $\mathcal{E}_{U_0} = (\Sigma, (E_{U_0} \cup U_0) \uplus (B \setminus U_0))$, where the equations that are used as rules are: 1) E_{U_0}, the set of variants that are generated by E using the theory $(\Sigma, U_0 \uplus (B \setminus U_0))$, and 2) the equations of U_0. Note that the equational axioms to be used for generalization in the transformed equational theory $\mathcal{E}_{U_0}$ (which is semantically equivalent to $\mathcal{E}$) are $B \setminus U_0$ so that the problematic identity axioms U_0 have now become oriented equations of the transformed specification, which are executed modulo $B \setminus U_0$.

So far we have provided a modular algorithm for least general generalization in equational theories containing different axioms such as associativity, commutativity, and identify (and their combinations). However, our modular algorithm does not provide, a priori, a minimal set $lgg_E(t, t')$ of least general equational generalizers for t and t' so that it must be filtered out to obtain one of the possible minimal sets of E-lgg's (see (Alpuente et al., 2014b)). That is, first a complete set of E-generalizers is computed by the inference rules of Figures 1, 2, 3, 5, 6 and 7, given above, and then they are filtered to obtain $lgg_E(t, s)$ by using the fact that,

[3] Given $\mathcal{E} = (\Sigma, E \uplus B)$, the *equational variants* (or simply variants) of a term t are the set of all pairs (t', σ), where t' is the canonical form of $t\sigma$ for a substitution σ (Escobar et al., 2012). Intuitively, the variants of t are the "irreducible patterns" to which t can be symbolically evaluated by applying $\vec{E}$ modulo B. An equational theory $\mathcal{E}$ has the *finite variant property* (FVP) (or $\mathcal{E}$ is called a *finite variant theory*) iff there is a finite and complete set of most general equational variants for each term.

Decompose$_C$

$$\frac{f : [\mathsf{s}'] \times [\mathsf{s}'] \to [\mathsf{s}] \wedge C_f \in ax(f) \wedge A_f \notin ax(f) \wedge i \in \{1,2\}}{\langle f(t_1,t_2) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1,t'_2) \wedge C \mid S \mid \theta\rangle \to \langle t_1 \overset{x_1:[\mathsf{s}']}{\triangleq} t'_i \wedge t_2 \overset{x_2:[\mathsf{s}']}{\triangleq} t'_{(i \bmod 2)+1} \wedge C \mid S \mid \theta\sigma\rangle}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s}'], x_2{:}[\mathsf{s}'])\}$, and $x_1{:}[\mathsf{s}'], x_2{:}[\mathsf{s}']$ are fresh variables

Fig. 5: Order-sorted decomposition rule for a commutative function symbol f (Alpuente et al., 2014b)

Decompose$_{A-\mathrm{left}}$

$$\frac{\begin{array}{c} f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge A_f \in ax(f) \wedge C_f \notin ax(f) \wedge \\ n \geq 2 \wedge m \geq 2 \wedge k \in \{1,\ldots,n-1\} \end{array}}{\begin{array}{c} \langle f(t_1,\ldots,t_n) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1,\ldots,t'_m) \wedge C \mid S \mid \theta\rangle \to \\ \langle f(t_1,\ldots,t_k) \overset{x_1:[\mathsf{s}]}{\triangleq} t'_1 \wedge f(t_{k+1},\ldots,t_n) \overset{x_2:[\mathsf{s}]}{\triangleq} f(t'_2,\ldots,t'_m) \wedge C \mid S \mid \theta\sigma\rangle \end{array}}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}])\}$, and $x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}]$ are fresh variables

Decompose$_{A-\mathrm{right}}$

$$\frac{\begin{array}{c} f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge A_f \in ax(f) \wedge C_f \notin ax(f) \wedge \\ n \geq 2 \wedge m \geq 2 \wedge k \in \{1,\ldots,m-1\} \end{array}}{\begin{array}{c} \langle f(t_1,\ldots,t_n) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1,\ldots,t'_m) \wedge C \mid S \mid \theta\rangle \to \\ \langle t_1 \overset{x_1:[\mathsf{s}]}{\triangleq} f(t'_1,\ldots,t'_k) \wedge f(t_2,\ldots,t_n) \overset{x_2:[\mathsf{s}]}{\triangleq} f(t'_{k+1},\ldots,t'_m) \wedge C \mid S \mid \theta\sigma\rangle \end{array}}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}])\}$, and $x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}]$ are fresh variables

Fig. 6: Order-sorted decomposition rules for an associative (non–commutative) function symbol f (Alpuente et al., 2014b)

for all theories E in the parametric family of theories we consider in this paper, there is a matching algorithm modulo E that provides the relation $\leq_E$.

In the following section we establish the correctness and termination of our resulting order-sorted, equational least general generalization algorithm.

4 Termination and Correctness of Order-sorted Equational Least General Generalization

Let us first establish that the extended, order-sorted least general generalization calculus modulo axioms of Section 3 terminates. The proof of this theorem is in Appendix A.1.

Let us call a set B of Σ-equational axioms $A{\vee}C{\vee}U$ iff $B = \bigcup_{f\in\Sigma} ax(f)$ and $ax(f) \subseteq \{A_f, C_f, U_f\}$.

Theorem 1 (Termination) *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A{\vee}C{\vee}U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that the subsignature Σ_Γ is U-tolerant,*

Decompose$_{AC-\text{left}}$

$$\frac{\begin{array}{c} f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge \{A_f, C_f\} \subseteq ax(f) \wedge n \geq 2 \wedge m \geq 2 \wedge \\ \{i_1, \ldots, i_n\} = \bar{\uplus}\{1, \ldots, n\} \ \wedge k_n \in \{1, \ldots, n-1\} \wedge k_m \in \{1, \ldots, m\} \end{array}}{\begin{array}{c} \langle f(t_1, \ldots, t_n) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1, \ldots, t'_m) \wedge C \mid S \mid \theta \rangle \to \\ \langle f(t_{i_1}, \ldots, t_{i_{k_n}}) \overset{x_1:[\mathsf{s}]}{\triangleq} t'_{k_m} \wedge f(t_{i_{(k_n+1)}}, \ldots, t_{i_n}) \overset{x_2:[\mathsf{s}]}{\triangleq} f(t'_1, \ldots, t'_{k_m-1}, t'_{k_m+1}, \ldots, t'_m) \wedge C \mid S \mid \theta\sigma \rangle \end{array}}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}])\}$, and $x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}]$ are fresh variables

Decompose$_{AC-\text{right}}$

$$\frac{\begin{array}{c} f : [\mathsf{s}] \times [\mathsf{s}] \to [\mathsf{s}] \wedge \{A_f, C_f\} \subseteq ax(f) \wedge n \geq 2 \wedge m \geq 2 \wedge \\ \{i_1, \ldots, i_m\} = \bar{\uplus}\{1, \ldots, m\} \ \wedge k_m \in \{1, \ldots, m-1\} \wedge k_n \in \{1, \ldots, n\} \end{array}}{\begin{array}{c} \langle f(t_1, \ldots, t_n) \overset{x:[\mathsf{s}]}{\triangleq} f(t'_1, \ldots, t'_m) \wedge C \mid S \mid \theta \rangle \to \\ \langle t_{k_n} \overset{x_1:[\mathsf{s}]}{\triangleq} f(t'_{i_1}, \ldots, t'_{i_{k_m}}) \wedge f(t_1, \ldots, t_{k_n-1}, t_{k_n+1}, \ldots, t_n) \overset{x_2:[\mathsf{s}]}{\triangleq} f(t'_{i_{(k_m+1)}}, \ldots, t'_{i_m}) \wedge C \mid S \mid \theta\sigma \rangle \end{array}}$$

where $\sigma = \{x{:}[\mathsf{s}] \mapsto f(x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}])\}$, and $x_1{:}[\mathsf{s}], x_2{:}[\mathsf{s}]$ are fresh variables

Fig. 7: Order-sorted decomposition rules for an associative–commutative function symbol f (Alpuente et al., 2014b)

every derivation stemming from an initial configuration $\langle \Gamma \mid \emptyset \mid id \rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7 terminates in a final configuration of the form $\langle \emptyset \mid S \mid \theta \rangle$.

In order to prove correctness (Theorem 2 below) and completeness (Theorem 3 below) of the order-sorted, least general equational generalization procedure, we follow the same proof schema of (Alpuente et al., 2014b) and define order-sorted B-lgg computation by subsort specialization. In other words, we mimick the computation of least general generalizers by first removing sorts (*i.e.,* upgrading variables to top sorts), then computing (unsorted) B-lggs, and finally obtaining the right subsorts by a suitable specialization post-processing. Note that this naïve procedure, formalized in Appendix A.3, is not used in practice but only as a useful reference for the proofs of correctness and completeness of Appendix A.2.

Since we follow the proof scheme of (Alpuente et al., 2014b) and the only change is in the identity case, we only need to modify the following lemma from (Alpuente et al., 2014b), whose proof is in Appendix A.3.

Lemma 1 *Given terms t and t' such that every symbol in t and t' is either free or has an identity element, and a fresh variable x,*

- *if $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle C \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, and 3, then $x\theta$ is a generalizer of t and t' modulo identity;*
- *if u is a generalizer of t and t' modulo identity, then there is a derivation $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle C \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, and 3, such that $u \equiv_B x\theta$.*

In order to precisely state correctness, we need the following. We use flattened versions of terms that use poly-variadic versions of the associative symbols, *i.e.,* given an associative symbol f with n arguments, and $n \geq 2$, flattened terms are canonical forms w.r.t. the set of rules given by the following rule schema

$$f(u_1, \ldots, f(v_1, \ldots, v_n), \ldots, u_m) \to f(u_1, \ldots, v_1, \ldots, v_n, \ldots, u_m) \quad m \geq 1$$

Given an associative symbol f and a term $f(t_1, \ldots, t_n)$, we call *f-alien terms* (or simply *alien terms*) those terms among the $t_1, \ldots, t_n$ that are not rooted by f.

Theorem 2 (Correctness) *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A \vee C \vee U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that t and t' are flattened Σ-terms and the subsignature Σ_Γ is U-tolerant, if $\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle \emptyset \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7, then $(x{:}[\mathsf{s}])\theta$ is a generalizer of t and t'.*

Theorem 3 (Completeness) *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A \vee C \vee U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that t and t' are flattened Σ-terms and the subsignature Σ_Γ is U-tolerant, if u is a least general generalizer of t and t' modulo B, then there is a derivation $\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle C \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7, such that $u \equiv_B (x{:}[\mathsf{s}])\theta$.*

Note that, because of termination, the filtering procedure of (Alpuente et al., 2014b) can be finally applied to get rid of those generalizers that are not least general so that minimality of the set of E-lgg's trivially holds.

5 Conclusions

We have completed the generalization calculus of (Alpuente et al., 2014b) with an extra inference rule so that completeness holds for both linear and non-linear order-sorted equational generalization under a mild syntactic condition on signatures, called U-tolerance, ensuring finiteness of the set of B-lggs that is easily achievable in practice. To our knowledge, this is the first finite, minimal, and complete procedure for order-sorted equational least general generalization modulo any combinations of associativity, commutativity, and identity axioms for different function symbols (i.e., A, C, U, AC, AU, CU, and ACU). Furthermore, it not only works for unsorted and many-sorted specification, but it also works for the more general and expressive case of order-sorted specifications. This allows us to deal with equational theories more expressive than the unsorted ones currently handled by (Cerna and Kutsia, 2020). Furthermore, this is done without resorting to heuristics or tree grammars.

We have implemented the extended, order-sorted, least general generalization algorithm described in this paper in the ACUOS^2 system, which is publicly available at `http://safe-tools.dsic.upv.es/acuos2`. We also endowed the ACUOS^2 system with a new capability to check the U-tolerance property, which can be accessed as a suitable tool option.

References

Alpuente M, Escobar S, Meseguer J, Ojeda P (2008) A modular equational generalization algorithm. In: Hanus M (ed) Logic-Based Program Synthesis and Transformation, 18th International Symposium, LOPSTR 2008, Valencia, Spain, July 17-18, 2008, Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 5438, pp 24–39, DOI 10.1007/978-3-642-00515-2_3, URL https://doi.org/10.1007/978-3-642-00515-2_3

Alpuente M, Escobar S, Meseguer J, Ojeda P (2009) Order-sorted generalization. Electr Notes Theor Comput Sci 246:27–38, DOI 10.1016/j.entcs.2009.07.013, URL https://doi.org/10.1016/j.entcs.2009.07.013

Alpuente M, Escobar S, Espert J, Meseguer J (2014a) ACUOS: A System for Modular ACU Generalization with Subtyping and Inheritance. In: Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA 2014), Springer-Verlag, Berlin, Lecture Notes in Computer Science, vol 8761, pp 573–581

Alpuente M, Escobar S, Espert J, Meseguer J (2014b) A modular order-sorted equational generalization algorithm. Information and Computation 235:98–136, DOI 10.1016/j.ic.2014.01.006, URL https://doi.org/10.1016/j.ic.2014.01.006

Alpuente M, Ballis D, Cuenca-Ortega A, Escobar S, Meseguer J (2019) ACUOS^2: A High-Performance System for Modular ACU Generalization with Subtyping and Inheritance. In: Proceedings of the 16th European Conference on Logics in Artificial Intelligence (JELIA 2019), Springer-Verlag, Berlin, Lecture Notes in Computer Science, vol 11468, pp 171–181

Alpuente M, Ballis D, Sapiña J (2020a) Efficient Safety Enforcement for Maude Programs via Program Specialization in the ÁTAME system. Mathematics in Computer Science 14(3):591–606

Alpuente M, Cuenca-Ortega A, Escobar S, Meseguer J (2020b) A Partial Evaluation Framework for Order-Sorted Equational Programs modulo Axioms 110:1–36

Armengol E (2007) Usages of Generalization in Case-Based Reasoning. In: Proc. of the 7th International Conference on Case-Based Reasoning (ICCBR 2007), Springer-Verlag, Lecture Notes in Computer Science, vol 4626, pp 31–45, DOI 10.1007/978-3-540-74141-1_3

Baumgartner A, Kutsia T, Levy J, Villaret M (2018) Term-graph anti-unification. In: FSCD, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, LIPIcs, vol 108, pp 9:1–9:17

Cerna DM, Kutsia T (2020) Unital anti-unification: Type and algorithms. In: Ariola ZM (ed) 5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, LIPIcs, vol 167, pp 26:1–26:20, DOI 10.4230/LIPIcs.FSCD.2020.26, URL https://doi.org/10.4230/LIPIcs.FSCD.2020.26

Durán F, Lucas S, Meseguer J (2009) Termination modulo combinations of equational theories. In: Ghilardi S, Sebastiani R (eds) FroCos, Springer, Lecture Notes in Computer Science, vol 5749, pp 246–262

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

Goguen J, Meseguer J (1992) Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoretical Computer Science 105:217–273

Huet G (1976) Resolution d'equations dans des langages d'order 1, 2,. . . ,ω. PhD thesis, Univ. Paris VII

Meseguer J (1997) Membership Algebra As a Logical Framework for Equational Specification. In: Parisi-Presicce F (ed) Proc. of 12th International Workshop on Recent Trends in Algebraic Development Techniques, WADT'97, Springer, LNCS, vol 1376, pp 18–61

Muggleton S (1999) Inductive Logic Programming: Issues, Results and the Challenge of Learning Language in Logic. Artif Intell 114(1-2):283–296

Ontañón S, Plaza E (2012) Similarity measures over refinement graphs. Machine Learning 87(1):57–92, DOI 10.1007/s10994-011-5274-3

Plotkin G (1970) A note on inductive generalization. In: Machine Intelligence, vol 5, Edinburgh University Press, pp 153–163

Reynolds J (1970) Transformational systems and the algebraic structure of atomic formulas. Machine Intelligence 5:135–151

TeReSe (ed) (2003) Term Rewriting Systems. Cambridge University Press, Cambridge

A Proofs of technical results

A.1 Proof of Theorem 1

Theorem 1 (Termination). *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A{\vee}C{\vee}U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that the subsignature Σ_Γ is U-tolerant, every derivation stemming from an initial configuration $\langle \Gamma \mid \emptyset \mid id\rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7, terminates in a final configuration of the form $\langle \emptyset \mid S \mid \theta\rangle$.*

Proof Identical to the proof of (Alpuente et al., 2014b) since it is not possible to have a generalization problem of the form $e_f \overset{w}{\triangleq} e_g$ in a U-tolerant signature. □

A.2 An auxiliary least general generalization procedure

In order to prove correctness and completeness (Theorems 2 and 3, respectively, in Appendix A.3) of the order-sorted, equational least general generalization procedure, we follow the same proof schema of (Alpuente et al., 2014b) and define order-sorted B-lgg computation by subsort specialization. In other words, we mimick the computation of least general generalizers by first removing sorts (*i.e.,* upgrading variables to top sorts), then computing (unsorted) B-lggs, and finally obtaining the right subsorts by a suitable specialization post-processing.

First, we recall the notion of a conflict pair.

Definition 3 (Conflict Position/Pair) Given terms t and t', a position $p \in \mathcal{P}os(t) \cap \mathcal{P}os(t')$ is called *a conflict position of t and t'* if $root(t|_p) \neq root(t'|_p)$ and for all $q < p$, $root(t|_q) = root(t'|_q)$. Given terms t and t', the pair (u, v) is called a *conflict pair of t and t'* if there exists at least one conflict position p of t and t' such that $u = t|_p$ and $v = t'|_p$.

The following notions of *pair of subterms* and *conflict pair* are specialized to the case when function symbols obey C, A, AC, and U and are the basis for our overall proof scheme.

Definition 4 (Commutative Pair of Subterms) Given terms t and t' such that every symbol in t and t' is either free or commutative, the pair (u, v) of terms is called a *commutative pair of subterms of t and t'* if and only if there are positions $p \in Pos(t)$ and $p' \in Pos(t')$ such that:

- $t|_p = u$, $t'|_{p'} = v$, $depth(p) = depth(p')$,
- for each $0 \leq i < depth(p)$, $root(t|_{p|_i}) = root(t'|_{p'|_i})$, and
- for each $0 < j \leq depth(p)$:
  - if $root(t|_{p|_{j-1}})$ is free, then $(p)_j = (p')_j$, and
  - if $root(t|_{p|_{j-1}})$ is commutative, $(p)_j = (p')_j$ or $(p)_j = ((p')_j \ mod\ 2) + 1$.

Definition 5 (Commutative Conflict Pair) Given terms t and t' such that every symbol in t and t' is either free or commutative, the pair (u, v) is called a *commutative conflict pair of t and t'* if and only if $root(u) \neq root(v)$ and (u, v) is a commutative pair of subterms of t and t'.

Definition 6 (Associative Pair of Positions) Given flattened terms t and t' such that every symbol in t and t' is either free or associative, and given positions $p \in Pos(t)$ and $p' \in Pos(t')$, the pair (p, p') of positions is called an *associative pair of positions of t and t'* if and only if

- $depth(p) = depth(p')$,
- for each $0 \leq i < depth(p)$, $root(t|_{p|_i}) = root(t'|_{p'|_i})$, and
- for each $0 < j \leq depth(p)$:
  - if $root(t|_{p|_{j-1}})$ is free, then $(p)_j = (p')_j$, and
  - if $root(t|_{p|_{j-1}})$ is associative, then no restriction on $(p)_j$ and $(p')_j$.

Definition 7 (Associative Pair of Subterms) Given flattened terms t and t' such that every symbol in t and t' is either free or associative, the pair (u, v) of terms is called an *associative pair of subterms of t and t'* if and only if either

1. (Regular subterms) for each pair of positions $p \in Pos(t)$ and $p' \in Pos(t')$ such that $t|_p = u$, $t'|_{p'} = v$, then (p, p') is an associative pair of positions of t and t'; or
2. (Associative subterms) there are positions $p \in Pos(t)$, $p' \in Pos(t')$ such that the following conditions are satisfied:
   - (p, p') is an associative pair of positions of t and t',
   - $u = f(u_1, \ldots, u_{n_u})$, $n_u \geq 1$, $v = f(v_1, \ldots, v_{n_v})$, $n_v \geq 1$, f is associative,
   - $t|_p = f(t_1, \ldots, t_{k_1}, u_1, \ldots, u_{n_u}, t_{k_2}, \ldots, t_{n_p})$, $n_p \geq 2$, $t'|_{p'} = f(t'_1, \ldots, t'_{k'_1}, v_1, \ldots, v_{n_v}, t'_{k'_2}, \ldots, t'_{n_{p'}})$, $n_p \geq 2$, and
   - $k_1 = 0$ (no arguments before u_1) if and only if $k'_1 = 0$ (no arguments before v_1), and,
   - $k_2 > n_p$ (no arguments after u_{n_u}) if and only if $k'_2 > n_{p'}$ (no arguments after v_{n_v}).

Definition 8 (Associative Conflict Pair) Given flattened terms t and t' such that every symbol in t and t' is either free or associative, the pair (u, v) is called an *associative conflict pair of t and t'* if and only if $root(u) \neq root(v)$ and (u, v) is an associative pair of subterms of t and t'.

Definition 9 (Associative-commutative Pair of Positions) Given flattened terms t and t' such that every symbol in t and t' is either free or associative-commutative, and given positions $p \in Pos(t)$ and $p' \in Pos(t')$, the pair (p, p') of positions is called an *associative-commutatve pair of positions of t and t'* if it satisfies the conditions for being an associative pair of positions of t and t'.

Definition 10 (Associative-commutative Pair of Subterms) Given flattened terms t and t' such that every symbol in t and t' is either free or associative-commutative, the pair (u, v) of terms is called an *associative-commutative pair of subterms of t and t'* if and only if either

1. (Regular subterms) for each pair of positions $p \in Pos(t)$ and $p' \in Pos(t')$ such that $t|_p = u$, $t'|_{p'} = v$, then (p, p') is an associative-commutative pair of positions of t and t'; or
2. (Associative-commutative subterms) there are positions $p \in Pos(t)$, $p' \in Pos(t')$ such that the following conditions are satisfied:

- (p, p') is an associative-commutative pair of positions of t and t', and
- $u = f(u_1, \ldots, u_{n_u})$, $n_u \geq 1$, $v = f(v_1, \ldots, v_{n_v})$, $n_v \geq 1$, f is associative,
- $t|_p = f(t_1, \ldots, t_{n_p})$, $n_p \geq 2$, $t'|_{p'} = f(t'_1, \ldots, t'_{n_{p'}})$, $n_{p'} \geq 2$,
- for all $1 \leq i \leq n_u$, there exists $1 \leq j \leq n_p$ s.t. $u_i =_B t_j$, and
- for all $1 \leq i \leq n_v$, there exists $1 \leq j \leq n_{p'}$ s.t. $v_i =_B t'_j$.

Definition 11 (Associative-commutative Conflict Pair) Given flattened terms t and t' such that every symbol in t and t' is either free or associative-commutative, the pair (u, v) is called an *associative-commutative conflict pair of t and t'* if and only if $\mathit{root}(u) \neq \mathit{root}(v)$ and (u, v) is an associative-commutative pair of subterms of t and t'.

Definition 12 (Identity Pair of Positions) Given terms t and t' such that every symbol in t and t' is either free or has an identity, and given positions $p \in \mathit{Pos}(t)$ and $p' \in \mathit{Pos}(t')$, the pair (p, p') of positions is called an *identity pair of positions of t and t'* if and only if either

1. (Base case) $p = \Lambda$ and $p' = \Lambda$;
2. (Free symbol) $p = q.i$, $p' = q'.i$, $\mathit{root}(t'|_{q'}) = \mathit{root}(t|_q)$ is a free symbol, and (q, q') is an identity pair of positions of t and t';
3. (Identity on the left) $\mathit{depth}(p) > \mathit{depth}(p')$, $p = q.i$, $\mathit{root}(t|_q)$ has an identity symbol e, and (q, p') is an identity pair of positions of t and t'; or
4. (Identity on the right) $\mathit{depth}(p') > \mathit{depth}(p)$, $p' = q'.i$, $\mathit{root}(t'|_{q'})$ has an identity symbol e, and (p, q') is an identity pair of positions of t and t'.

Definition 13 (Identity Pair of Subterms) Given terms t and t' such that every symbol in t and t' is either free or has an identity, the pair (u, v) of terms is called an *identity pair of subterms of t and t'* if and only if for each pair of positions $p \in \mathit{Pos}(t)$ and $p' \in \mathit{Pos}(t')$ such that $t|_p = u$, $t'|_{p'} = v$, then (p, p') is an identity pair of positions of t and t'.

Definition 14 (Identity Conflict Pair) Given terms t and t' such that every symbol in t and t' is either free or has an identity, the pair (u, v) is called an *identity conflict pair of t and t'* if and only if $\mathit{root}(u) \neq \mathit{root}(v)$ and (u, v) is an identity pair of subterms of t and t'.

We also recall a special notation for subterm replacement when we have associative or associative-commutative conflict pairs and order-sorted information.

Definition 15 (A-Subterm Replacement) Given two flattened terms t and t' and an associative conflict pair (u, v) with a pair of conflict positions $p \in \mathit{Pos}(t)$ and $p' \in \mathit{Pos}(t')$ such that $u = f(u_1, \ldots, u_m)$, $m \geq 1$, $v = f(v_1, \ldots, v_n)$, $n \geq 1$, f is associative, $t|_p = f(w_1, \ldots, w_{k_1}, u_1, \ldots, u_m, w'_1, \ldots, w'_{k_2})$, $k_1 \geq 0$, $k_2 \geq 0$, and $t'|_{p'} = f(w''_1, \ldots, w''_{k_3}, v_1, \ldots, v_n, w'''_1, \ldots, w'''_{k_4})$, $k_3 \geq 0$, $k_4 \geq 0$, we write $t[[x{:}\mathsf{s}]]_p$ and $t'[[x{:}\mathsf{s}]]_{p'}$ to denote the terms $t[[x{:}\mathsf{s}]]_p = t[f(w_1, \ldots, w_{k_1}, x{:}\mathsf{s}, w'_1, \ldots, w'_{k_2})]_p$, and $t'[[x{:}\mathsf{s}]]_{p'} = t[f(w''_1, \ldots, w''_{k_3}, x{:}\mathsf{s}, w'''_1, \ldots, w'''_{k_4})]_{p'}$.

Definition 16 (AC-Subterm Replacement) Given two flattened terms t and t' and an associative-commutative conflict pair (u, v) with a pair of conflict positions $p \in \mathit{Pos}(t)$ and $p' \in \mathit{Pos}(t')$ such that $u = f(u_1, \ldots, u_m)$, $m \geq 1$, $v = f(v_1, \ldots, v_n)$, $n \geq 1$, f is associative-commutative, $t|_p = f(w_1, \ldots, w_{k_1})$ s.t. for each $i \in \{1, \ldots, m\}$, there is $j \in \{1, \ldots, k_1\}$ with $u_i =_B w_j$, and $t'|_{p'} = f(w'_1, \ldots, w'_{k_2})$ s.t. for each $i \in \{1, \ldots, n\}$, there is $j \in \{1, \ldots, k_2\}$ with $v_i =_B w'_j$, then we write $t[[x{:}\mathsf{s}]]_p$ and $t'[[x{:}\mathsf{s}]]_{p'}$ to denote the terms $t[[x{:}\mathsf{s}]]_p = t[f(\overline{w_1}, \ldots, \overline{w_{k_1}}, x{:}\mathsf{s})]_p$ where $\{\overline{w_1}, \ldots, \overline{w_{k_1}}\} = \bigcup\{w_i \mid 1 \leq i \leq k_1, \forall 1 \leq j \leq n, w_i \neq_B u_j\}$, and $t'[[x{:}\mathsf{s}]]_{p'} = t[f(\overline{w'_1}, \ldots, \overline{w'_{k_2}}, x{:}\mathsf{s})]_{p'}$ where $\{\overline{w'_1}, \ldots, \overline{w'_{k_2}}\} = \bigcup\{w'_i \mid 1 \leq i \leq k_2, \forall 1 \leq j \leq n, w'_i \neq_B v_j\}$.

Note that B-pre-regularity is essential here because it ensures that after replacing a subterm by a variable, the least sort does not depend on the chosen representation within the equivalence class of a term, i.e., it does not depend on how the flattened version of the term is obtained.

We recall order-sorted B-lgg computation by subsort specialization using *top-sorted generalization* and *sort-specialized generalization.*

Definition 17 (Top-sorted Equational Generalization) Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A \vee C \vee U$ axioms, and flattened Σ-terms t and t' such that $[LS(t)] = [LS(t')]$, let $(u_1, v_1), \ldots, (u_k, v_k)$ be the B-conflict pairs of t and t', and for each such conflict pair (u_i, v_i), let $(p^i_1, \ldots, p^i_{n_i}, q^i_1, \ldots, q^i_{n_i})$, $1 \leq i \leq k$, be the corresponding B-conflict positions, and let $[\mathsf{s_i}] = [LS(u_i)] = [LS(v_i)]$. We define the term denoting the *top order-sorted equational least general generalization* as

$$tsg_E(t, t') = t[[x^1_1{:}[\mathsf{s_1}], \ldots, x^1_{n_1}{:}[\mathsf{s_1}]]]_{p^1_1,\ldots,p^1_{n_1}} \cdots [[x^k_1{:}[\mathsf{s_k}], \ldots, x^k_{n_k}{:}[\mathsf{s_k}]]]_{p^k_1,\ldots,p^k_{n_k}}$$

where $x^1_1{:}[\mathsf{s_1}], \ldots, x^1_{n_1}{:}[\mathsf{s_1}], \ldots, x^k_1{:}[\mathsf{s_k}], \ldots, x^k_{n_k}{:}[\mathsf{s_k}]$ are fresh variables.

The order-sorted equational lggs are obtained by subsort specialization.

Definition 18 (Sort-specialized Equational Generalization) Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A \vee C \vee U$ axioms, and flattened Σ-terms t and t' such that $[LS(t)] = [LS(t')]$, let $(u_1, v_1), \ldots, (u_k, v_k)$ be the conflict pairs of t and t', and for each such conflict pair (u_i, v_i), let $p^i_1, \ldots, p^i_{n_i}$, $1 \leq i \leq k$, be the corresponding B-conflict positions, let $[\mathsf{s_i}] = [LS(u_i)] = [LS(v_i)]$, and let $x^1_1{:}[\mathsf{s_1}], \ldots, x^1_{n_1}{:}[\mathsf{s_1}], \ldots, x^k_1{:}[\mathsf{s_k}], \ldots, x^k_{n_k}{:}[\mathsf{s_k}]$ be the variable identifiers used in Definition 17. We define

$$\begin{aligned} \textit{sort-down-subs}_E(t, t') = \{\rho \mid\ & Dom(\rho) = \{x^1_1{:}[\mathsf{s_1}], \ldots, x^1_{n_1}{:}[\mathsf{s_1}], \ldots, x^k_1{:}[\mathsf{s_k}], \ldots, x^k_{n_k}{:}[\mathsf{s_k}]\} \wedge \\ & \forall 1 \leq i \leq k, \forall 1 \leq j \leq n_i : \\ & (x^i_j{:}[\mathsf{s_i}])\rho = x_i{:}\mathsf{s'_i} \wedge \mathsf{s'_i} \in LUBS(LS(u_i), LS(v_i))\} \end{aligned}$$

where all the $x_i{:}\mathsf{s'_i}$ are fresh variables. The set of sort-specialized B-generalizers is defined as $ssg_E(t, t') = \{tsg_E(t, t')\rho \mid \rho \in \textit{sort-down-subs}_E(t, t')\}$.

A.3 Proof of Theorems 2 and 3

The auxiliary notions and results in this section are similar to the corresponding ones in (Alpuente et al., 2014b), although the proofs of some of the results were just sketched there and we have completed them.

Let us prove that the range of the substitutions partially computed at any stage of a generalization derivation coincides with the set of the index variables of the configuration, except for the generalization variable x of the original generalization problem $t \overset{x}{\triangleq} t'$. This is stated in the following lemma that is similar to Lemma 28 of (Alpuente et al., 2014b).

Lemma 2 (Range of Substitutions) *Given terms t and t' and a fresh variable x such that $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle C \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7, then $Index(S \cup C) \subseteq VRan(\theta) \cup \{x\}$, and $VRan(\theta) = Var(x\theta)$.*

Proof Immediate by construction. □

The following lemma establishes an auxiliary property that is useful for formulating the notion of an identity conflict pair of terms. It is similar to Lemma 30 of (Alpuente et al., 2014b).

Lemma 3 *Given terms t and t' such that every symbol in t and t' is either free or has an identity, and a fresh variable x, then there is a sequence $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id \rangle \to^* \langle u \overset{y}{\triangleq} v \wedge C \mid S \mid \theta \rangle$ using the inference rules of Figures 1, 2, and 3 such that there is no variable z such that $u \overset{z}{\triangleq} v \in S$ if and only if (u, v) is an identity pair of subterms of t and t'.*

Proof Straightforward by successive application of the inference rule $\textbf{Decompose}_B$ of Figure 1 and the inference rule $\textbf{Expand}_U$ of Figure 2. □

The following lemma expresses the precise connection between the constraints in a derivation and the identity conflict pairs of the initial configuration. It is similar to Lemma 31 of (Alpuente et al., 2014b).

Lemma 4 *Given terms t and t' such that every symbol in t and t' is either free or has an identity, and a fresh variable x, then there is a sequence $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle C \mid u \overset{y}{\triangleq} v \wedge S \mid \theta\rangle$ using the inference rules of Figures 1, 2, and 3 if and only if (u, v) is an identity conflict pair of t and t'.*

Proof ($\Rightarrow$) Since $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle C \mid u \overset{y}{\triangleq} v \wedge S \mid \theta\rangle$, then there must be two configurations $\langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle$, $\langle C_2 \mid u \overset{y}{\triangleq} v \wedge S_2 \mid \theta_2\rangle$ such that

$$\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\ \rangle \to^* \langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle,$$

$$\langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle \to_{\mathbf{Solve}_B} \langle C_2 \mid u \overset{y}{\triangleq} v \wedge S_2 \mid \theta_2\rangle,$$

$$\langle C_2 \mid u \overset{y}{\triangleq} v \wedge S_2 \mid \theta_2\rangle \to^* \langle \emptyset \mid u \overset{y}{\triangleq} v \wedge S \mid \theta\rangle,$$

and, by application of the inference rule $\mathbf{Solve}_B$, $root(u) \neq root(v)$. By using Lemma 3 with the derivation $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\ \rangle \to^* \langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle$, (u, v') is an identity pair of subterms of t and t'. Therefore, (u, v) is an identity conflict pair.

($\Leftarrow$) By Lemma 3, there is a configuration $\langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle$ such that $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\ \rangle \to^* \langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle$, and $root(u) \neq root(v)$. Then, the inference rule $\mathbf{Solve}_B$ is applied, *i.e.,* $\langle u \overset{y}{\triangleq} v \wedge C_1 \mid S_1 \mid \theta_1\rangle \to \langle C_1 \mid u \overset{y}{\triangleq} v \wedge S_1 \mid \theta_1\rangle$ and $u \overset{y}{\triangleq} v$ will be part of S in the final configuration $\langle \emptyset \mid S \mid \theta\rangle$. □

Finally, the following lemma establishes the link between the computed substitution and a proper generalizer. It is similar to the proof of Lemma 32 of (Alpuente et al., 2014b). We have underlined the beginning of the extra necessary cases that allow us to repair the original proof of (Alpuente et al., 2014b).

Lemma 1. *Given terms t and t' such that every symbol in t and t' is either free or has an identity, and a fresh variable x,*

- *if $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle C \mid S \mid \theta\rangle$ using the inference rules of Figures 1, 2, and 3, then $x\theta$ is a generalizer of t and t' modulo identity;*
- *if u is a generalizer of t and t' modulo identity, then there is a derivation $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle C \mid S \mid \theta\rangle$ using the inference rules of Figures 1, 2, and 3, such that $u \equiv_B x\theta$.*

Proof By structural induction on the term $x\theta$ (resp. u). If $x\theta = x$ (resp. u is a variable), then $\theta = id$ and the conclusion follows. If $x\theta = f(u_1, \ldots, u_k)$ (resp. $u = f(u_1, \ldots, u_k)$) and f is free, then the inference rule $\mathbf{Decompose}_B$ of Figure 1 is applied and we have that $t = f(t_1, \ldots, t_k)$ and $t' = f(t'_1, \ldots, t'_k)$. If f has an identity symbol e and $x\theta = f(u_1, u_2)$ (resp. $u = f(u_1, u_2)$), then we have two possibilities: (1) the inference rule $\mathbf{Expand}_U$ of Figure 2 is applied and we have that either: (i) $t = f(t_1, t_2)$ and $t' = f(t'_1, t'_2)$; (ii) $t = f(t_1, t_2)$ and $root(t') \neq f$; or (iii) $root(t) \neq f$ and $t' = f(t'_1, t'_2)$. $\underline{(2)}$ the inference rule $\mathbf{Recover}_U$ of Figure 3 is applied and we have that $root(t) \neq f$ and $root(t') \neq f$.

For the case when f is free, by using the induction hypothesis, u_i is a generalizer of t_i and t'_i, for each i.

For the case when f has an identity symbol e and the inference rule $\mathbf{Expand}_U$ was applied, by using the induction hypothesis, u_1 is a generalizer of either t_1 and t'_1, t_1 and t' (by applying$f(x, e) \doteq x$ to t'), or t and t'_1 (by applying$f(x, e) \doteq x$ to t). Similarly, u_2 is a generalizer of either t_2 and t'_2, t_2 and t' (by applying$f(e, x) \doteq x$ to t'), or t and t'_2 (by applying$f(e, x) \doteq x$ to t).

$\underline{\text{For}}$ the case when f has an identity symbol e and the inference rule $\mathbf{Recover}_U$ was applied, by using the induction hypothesis, either u_1 is a generalizer of t and e and u_2 is a generalizer of t' and e, or u_1 is a generalizer of t' and e and u_2 is a generalizer of t and e. Now, if for each pair of terms in $u_1, \ldots, u_k$ there are no shared variables, then the conclusion follows. Otherwise, for each variable z shared between two different terms u_i and u_j, there is a constraint $w_1 \overset{z}{\triangleq} w_2 \in S$, and, by Lemma 4, there is an identity conflict pair (w_1, w_2) in t_i and t'_i. Thus, the conclusion follows. □

Correctness and completeness are finally proved as follows.

Theorem 2 (Correctness). *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A{\vee}C{\vee}U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that t and t' are flattened Σ-terms and the subsignature Σ_Γ is U-tolerant, if $\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle \emptyset \mid S \mid \theta\rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7,then $(x{:}[\mathsf{s}])\theta$ is a generalizer of t and t'.*

Proof By Lemma 1. □

Theorem 3 (Completeness). *Given a kind-completed, B-pre-regular, order-sorted equational theory (Σ, B) with a set B of $A{\vee}C{\vee}U$ axioms, and a generalization problem $\Gamma = t \overset{x:[\mathsf{s}]}{\triangleq} t'$, with $[\mathsf{s}] = [LS(t)] = [LS(t')]$, such that t and t' are flattened Σ-terms and the subsignature Σ_Γ is U-tolerant, if u is a least general generalizer of t and t' modulo B, then there is a derivation $\langle t \overset{x:[\mathsf{s}]}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle C \mid S \mid \theta\rangle$ using the inference rules of Figures 1, 2, 3, 5, 6 and 7 such that $u \equiv_B (x{:}[\mathsf{s}])\theta$.*

Proof By contradiction. Consider a derivation $\langle t \overset{x}{\triangleq} t' \mid \emptyset \mid id\rangle \to^* \langle \emptyset \mid S \mid \theta\rangle$ such that $x\theta$ is not a least general generalizer of t and t' up to renaming. Since $x\theta$ is a generalizer of t and t' by Lemma 1, there is a substitution ρ which is not a variable renaming such that $x\theta\rho =_B u$. By Lemma 2, $VRan(\theta) = Var(x\theta)$; hence, we can choose ρ with $Dom(\rho) = Var(x\theta)$. Now, since ρ is not a variable renaming, either:

1. there are variables $y, y' \in Var(x\theta)$ and a variable z such that $y\rho = y'\rho = z$, or
2. there is a variable $y \in Var(x\theta)$ and a non-variable term v such that $y\rho = v$.

In case (1), there are two conflict positions p, p' for t and t' such that $u|_p = z = u|_{p'}$ and $x\theta|_p = y$ and $x\theta|_{p'} = y'$. Specifically, this means that $t|_p = t|_{p'}$ and $t'|_p = t'|_{p'}$. However, this is impossible by Lemmas 4 and 2. In case (2), there is a position p such that $x\theta|_p = y$ and p is neither a conflict position of t and t' nor is it under a conflict position of t and t'. Since this is impossible by Lemmas 4 and 2, the claim is proved. □