Order-sorted Homeomorphic Embedding modulo Combinations of Associativity and/or Commutativity Axioms[⋆]

María Alpuente[1], Angel Cuenca-Ortega[3], Santiago Escobar[1], and José Meseguer[2]

[1] VRAIN, Universitat Politècnica de València, Spain. {alpuente,sescobar}@upv.es
[2] University of Illinois at Urbana-Champaign, USA. meseguer@illinois.edu
[3] Universidad de Guayaquil, Ecuador. angel.cuencao@ug.edu.ec

Abstract. The Homeomorphic Embedding relation has been amply used for defining termination criteria of symbolic methods for program analysis, transformation, and verification. However, homeomorphic embedding has never been investigated in the context of order-sorted rewrite theories that support symbolic execution methods *modulo* equational axioms. This paper generalizes the symbolic homeomorphic embedding relation to order–sorted rewrite theories that may contain various combinations of associativity and/or commutativity axioms for different binary operators. We systematically measure the performance of different, increasingly efficient formulations of the homeomorphic embedding relation modulo axioms that we implement in Maude. Our experimental results show that the most efficient version indeed pays off in practice.

1 Introduction

Homeomorphic Embedding is a control mechanism that is commonly used to ensure termination of symbolic methods and program optimization techniques. Homeomorphic embedding is a structural preorder relation under which a term t' is greater than (i.e., it embeds) another term t represented by $t \trianglelefteq t'$ if t can be obtained from t' by deleting some symbols of t'. For instance, $s(0+s(X))*s(X+Y)$ embeds $s(X)*s(Y)$. The usefulness of homeomorphic embedding for ensuring termination is given by the following well-known property of well-quasi-orderings (see [18]): given a finite signature Σ, for every infinite sequence of terms $t_1, t_2, \ldots$, there exist $i < j$ such that $t_i \trianglelefteq t_j$. Therefore, if we iteratively compute a sequence $t_1, t_2, \ldots, t_n$, we can guarantee finiteness of the sequence by using the embedding as a whistle: whenever a new expression t_{n+1} is to be added to the sequence, we first check whether t_{n+1} embeds any of the expressions $t_i, i = 1, \ldots, n$ that are already in the sequence. If that is the case, the computation must be stopped because the whistle ($\trianglelefteq$) signals (potential) non-termination. Otherwise, t_{n+1} can be safely added to the sequence and the computation proceeds.

Maude is a language and a system that efficiently implements rewriting logic [24], which is a logic of change that seamlessly unifies a wide variety of models of concurrency. Recently, an experimental open platform has been developed in [16] that allows the performance of functional and algebraic programming languages to be compared, including CafeOBJ, Clean, Haskell, LNT, LOTOS, Maude, mCRL2, OCaml, Opal, Rascal, Scala, SML, Stratego / XT, and Tom (see references in [16]). In the top 5 of the more efficient tools, Maude ranks second after Haskell. This is remarkable for at least two reasons: (i) Maude is not a compiled language but runs under an interpreter; (ii) Maude has quite sophisticated features (subtype polymorphism, pattern matching modulo associativity, commutativity and identity, reflection, strategies, objects, etc.) that have no equivalent in Haskell or other functional languages. In

[⋆] This work has been partially supported by the EU (FEDER) and the Spanish MINECO under grant TIN 2015-69175-C4-1-R, by Generalitat Valenciana under grant PROMETEO/2019/098, and by NRL under contract number N00173-17-1-G002. Angel Cuenca-Ortega has been supported by the SENESCYT, Ecuador (scholarship program 2013).

[2], an extension of homeomorphic embedding modulo equational axioms was defined as a key component of Maude's symbolic partial evaluator, Victoria, which is based on (variant) narrowing[4] [14]. Unfortunately, the formulation in [2] was done with a concern for simplicity in mind and degrades the tool performance because the proposed implementation of equational homeomorphic embedding did not scale well to realistic problems. This was not unexpected since other equational problems (such as equational matching, equational unification, or equational least general generalization) are typically much more costly than their corresponding "syntactic" counterparts and achieving efficient implementations has required years of significant investigation effort. Furthermore, the equational homeomorphic embedding relation $\trianglelefteq_B$ (modulo a set B of axioms) in [2] did not consider types so that an embedding test such as `X:Bool` $\trianglelefteq_B$ `0 + suc(N:Nat)` succeeds.

Our contributions in this paper include the following. We present five different formulations (one of which is totally novel to this paper and the other significantly improve [3]) of order-sorted homeomorphic embedding modulo axioms in rewrite theories that may contain sorts, subsort polymorphism, overloading, and rewriting with (conditional) rules and equations modulo a set B of equational axioms, and we compare their performance. In particular, we propose an order-sorted, equational homeomorphic embedding formulation $\breve{\trianglelefteq}_B^{kosml}$ that runs up to 6 orders of magnitude faster than the original definition of the homeomorphic embedding modulo equational axioms $\trianglelefteq_B$ in [2]. For this improvement in performance, we take advantage of Maude's powerful capabilities such as the efficiency of deterministic computations with equations versus non-deterministic computations with rewriting rules, or the use of non-strict definitions of the Boolean operators versus more speculative standard Boolean definitions [7].

Comparison with [3]. The equational homeomorphic embedding relation $\trianglelefteq_B$ of [2] was efficiently implemented in the earlier conference paper [3]. The novel contributions with respect to [3] are as follows:

1. A finer treatment of subtype hierarchies is provided that considers the *connected components* of sorts so that two variables whose respective types (sorts) are not related in the suptype hierarchy (e.g., `Bool` and `Nat`) are considered incomparable.
2. Extended definitions (and corresponding extended results) are formalized dealing with sorts and subsorts in such a finer way.
3. Two novel optimizations are proposed that can achieve significant improvements in performance. Each optimization is formulated as a pruning rule that anticipates failure by considering Maude's total order on terms and the compatibility of kinds for any possible instance of all subterms.
4. A fuller treatment of all aspects is given, including more examples and detailed explanations of the key notions, together with full formal proofs of all technical results.
5. The implementation and experimental section have been improved in two ways: (i) the prototype implementation itself has been advanced with a new embedding test $\breve{\trianglelefteq}_B^{kosml}$ that implements the novel optimizations; and (ii) new examples that benchmark these optimizations with increasingly larger terms are provided. One important difference with the smaller set of benchmarks in [3] is that much bigger terms can be now handled by the tool.

Plan of the paper. After some preliminaries in Section 2. Section 3 recalls the pure, "syntactically simpler" homeomorphic embedding relation and extends it to deal with order-sorted non-ground terms. An order-sorted equational extension of the symbolic homeomorphic embedding relation is given in Section 4. Section 5 provides two *goal-driven* formulations for equational homeomorphic embedding: first, a calculus for embeddability goals that directly handles the algebraic axioms in the deduction system, and then a reachability-oriented characterization that cuts down the search space by taking advantage of pattern matching modulo associativity

[4] Narrowing is an extension of term rewriting where pattern matching is replaced with unification [15].

and commutativity axioms. Section 6 is concerned with an efficient meta-level formulation of equational homeomorphic embedding that relies on the classical flattening transformation that canonizes terms w.r.t. associativity and/or commutativity axioms (for instance, $1+(3+(2+0))$ gets flattened to $+(1,2,3)$). This formulation is optimized by replacing the classical Boolean operators by short-circuit, strategic versions of these operators. In Section 7, two new optimizations of the algorithm are defined that can achieve significant speedup by anticipating failure, which is done by considering Maude's total order on terms and the compatibility of kinds for any possible instance of all subterms. In Section 8, we provide an experimental performance evaluation of the proposed formulations showing that we can efficiently deal with realistic embedding problems modulo axioms.

2 Preliminaries

Let us recall some key concepts of order-sorted rewriting logic theories [24]. We consider an order-sorted signature $(\Sigma, \mathsf{S}, \leq)$ that consists of a poset of sorts $(\mathsf{S}, \leq)$ and an $\mathsf{S}^* \times \mathsf{S}$-indexed family of sets $\Sigma = \{\Sigma_{\mathsf{s}_1 \ldots \mathsf{s}_n, \mathsf{s}}\}_{(\mathsf{s}_1 \ldots \mathsf{s}_n, \mathsf{s}) \in \mathsf{S}^* \times \mathsf{S}}$ of function symbols. The poset $(\mathsf{S}, \leq)$ of sorts for Σ is partitioned into equivalence classes $C_1, \ldots, C_n$ (called *connected components*) by the equivalence relation $(\leq \cup \geq)^+$, which is the transitive closure of the union of $\leq$ with the symmetric, supersort relation $\geq$. Throughout this paper, Σ is assumed to be *preregular*, so that each term t has a least sort, denoted $ls(t)$ (see [17]). Σ is also assumed to be *kind-complete*, that is, for each sort $\mathsf{s} \in \mathsf{S}$, its connected component in the poset $(\mathsf{S}, \leq)$ has a top sort under $\leq$, denoted $[\mathsf{s}]$ and called the connected component's *kind*, and for each function symbol $f \in \Sigma_{\mathsf{s}_1 \ldots \mathsf{s}_n, \mathsf{s}}$, there is also an $f \in \Sigma_{[\mathsf{s}_1] \ldots [\mathsf{s}_n], [\mathsf{s}]}$. An order-sorted signature can always be extended to be kind-complete [25]. Maude automatically checks preregularity and adds a new "kind" sort $[\mathsf{s}]$ at the top of the connected component of each sort $\mathsf{s} \in \mathsf{S}$ specified by the user and automatically lifts each operator to the kind level. Given a term t, by abuse we let $\lceil t \rceil$ denote the kind of t, i.e., $[ls(t)]$.

For technical reasons, it is useful to assume that Σ has no ad-hoc overloading; i.e., that for any two typings $f : \mathsf{s}_1 \ldots \mathsf{s}_n \longrightarrow \mathsf{s}$ and $f : \mathsf{s}'_1 \ldots \mathsf{s}'_n \longrightarrow \mathsf{s}' \in \Sigma$, the connected components of each s_i and $\mathsf{s}_i{}'$, $1 \leq i \leq n$, and of s and s' are always the same. However, this assumption entails no real loss of generality: any Σ can be transformed into a semantically equivalent signature with no ad-hoc overloading (by symbol renaming). Similarly, we assume that two different operators $f : \mathsf{s}_1 \ldots \mathsf{s}_n \longrightarrow \mathsf{s}$ and $f : \mathsf{s}'_1 \ldots \mathsf{s}'_m \longrightarrow \mathsf{s}' \in \Sigma$, $n \neq m$, do not exist after the symbol renaming. Finally, Σ is also assumed to be *sensible*, in the sense that for any two typings $f : \mathsf{s}_1 \ldots \mathsf{s}_n \longrightarrow \mathsf{s}$ and $f : \mathsf{s}'_1 \ldots \mathsf{s}'_n \longrightarrow \mathsf{s}'$ of an n-ary function symbol f, if s_i and s'_i are in the same connected component of $(\mathsf{S}, \leq)$ for $1 \leq i \leq n$, then s and s' are also in the same connected component; this provides the right notion of *unambiguous* signature at the order-sorted level.

We assume an S-sorted family $\mathscr{X} = \{\mathscr{X}_s\}_{\mathsf{s} \in \mathsf{S}}$ of disjoint variable sets. $\mathscr{T}_\Sigma(\mathscr{X})_\mathsf{s}$ and $\mathscr{T}_{\Sigma\mathsf{s}}$ denote the sets of terms and of ground terms of sorts s, respectively. We also write $\mathscr{T}_\Sigma(\mathscr{X})$ and $\mathscr{T}_\Sigma$ for the corresponding term algebras.

A *position* p in a term t is represented by a sequence of natural numbers (Λ denotes the empty sequence, i.e., the root position). Positions are ordered by the *prefix* ordering: $p \leq q$ if there exists w such that $p.w = q$. Given a term t, we let $\mathscr{P}os(t)$ and $\mathscr{P}os_\Sigma(t)$ respectively denote the set of positions and the set of non-variable positions of t (i.e., positions where a variable does not occur). $t|_p$ denotes the *subterm* of t at position p, and $t[u]_p$ denotes the result of *replacing the subterm* $t|_p$ by the term u. The set of variables occurring in a term t is denoted by $\mathscr{V}ar(t)$.

A *substitution* σ is a sorted mapping from a finite subset of $\mathscr{X}$ to $\mathscr{T}_\Sigma(\mathscr{X})$. Substitutions are written as $\sigma = \{X_1 : s_1 \mapsto t_1 : s_1, \ldots, X_n : s_n \mapsto t_n : s_n\}$ where the domain of σ is $Dom(\sigma) = \{X_1, \ldots, X_n\}$ and the set of variables introduced by terms $t_1, \ldots, t_n$ is written $Ran(\sigma)$. The identity substitution is *id*. Substitutions are homomorphically extended to $\mathscr{T}_\Sigma(\mathscr{X})$. The application of a substitution σ to a term t is called *an instance* of t and is denoted by $t\sigma$. We define the relative generality ordering $\leq$ between terms as $t \leq t'$ if there exists σ s.t. $t\sigma = t'$, i.e., t is more general than t'. For simplicity, we assume that every substitution is idempotent, i.e., σ satisfies

$Dom(\sigma) \cap Ran(\sigma) = \emptyset$. Substitution idempotency ensures $(t\sigma)\sigma = t\sigma$. The restriction of σ to a set of variables V is denoted $\sigma|_V$. Composition of two substitutions is denoted by $\sigma\sigma'$ so that $t(\sigma\sigma') = (t\sigma)\sigma'$.

A *Σ-equation* is an unoriented pair $t = t'$, where $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}}$ for some sort $\mathsf{s} \in \mathsf{S}$. Given Σ and a set E of Σ-equations, order-sorted equational logic induces a congruence relation $=_E$ on terms $t, t' \in \mathcal{T}_\Sigma(\mathcal{X})$ (see [5]). An *equational theory* (Σ, E) is a pair with Σ being an order-sorted signature and E a set of Σ-equations. We omit Σ when no confusion can arise.

A substitution θ is more (or equally) general than σ modulo B, denoted by $\theta \leq_E \sigma$, if there is a substitution γ such that $\sigma =_E \theta\gamma$, i.e., for all $x \in \mathcal{X}, x\sigma =_E x\theta\gamma$. A substitution σ is called a renaming if $\sigma = \{X_1 \mapsto Y_1, \ldots, X_n \mapsto Y_n\}$ and variables $Y_1, \ldots, Y_n$ are pairwise distinct. The renaming substitution σ is a renaming for expression E if $(\mathcal{V}ar(E) - \{X_1, \ldots, X_n\}) \cap \{Y_1, \ldots, Y_n\} = \emptyset$.

A *rewrite theory* is a triple $\mathcal{R} = (\Sigma, B, R)$, where (Σ, B) is the equational theory that is used for rewriting in R modulo B, and R is a set of rewrite rules. Rules are of the form $l \to r$ where terms $l, r \in \mathcal{T}_\Sigma(\mathcal{X})_{[\mathsf{s}]}$ for some kind $[\mathsf{s}]$ are respectively called the *left-hand side* (or *lhs*) and the *right-hand side* (or *rhs*) of the rule and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The set R of rules is *sort-decreasing*, i.e., for each $t \to t'$ in R, each $\mathsf{s} \in \mathsf{S}$, and each substitution σ, $t'\sigma \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}}$ implies $t\sigma \in \mathcal{T}_\Sigma(\mathcal{X})_{\mathsf{s}}$. Let $\to \subseteq A \times A$ be a binary relation on a set A. We denote its transitive closure by $\to^+$, and its reflexive and transitive closure by $\to^*$.

We define the *one-step rewrite relation* on $\mathcal{T}_\Sigma(\mathcal{X})$ for the set of rules R as follows: $t \to_R t'$ iff there is a position $p \in \mathcal{P}os(t)$, a rule $l \to r$ in R, and a substitution σ such that $t|_p = l\sigma$ and $t' = t[r\sigma]_p$. The relation $\to_{R/B}$ for rewriting modulo B is defined as $=_B \circ \to_R \circ =_B$. A term t is called *R/B-irreducible* iff there is no term u such that $t \to_{R/B} u$. A substitution σ is R/B-irreducible if, for every $x \in \mathcal{X}$, $x\sigma$ is R/B-irreducible. We say that the relation $\to_{R/B}$ is *terminating* if there is no infinite sequence $t_1 \to_{R/B} t_2 \to_{R/B} \cdots t_n \to_{R/B} t_{n+1} \cdots$. We say that the relation $\to_{R/B}$ is *confluent* if, whenever $t \to^*_{R/B} t'$ and $t \to^*_{R/B} t''$, there exists a term t''' such that $t' \to^*_{R/B} t'''$ and $t'' \to^*_{R/B} t'''$. A rewrite theory (Σ, B, R) is convergent if R is sort-decreasing and the relation $\to_{R/B}$ is confluent and terminating. In a convergent order-sorted rewrite theory, for each term $t \in \mathcal{T}_\Sigma(\mathcal{X})$, there is a unique (up to B-equivalence) R/B-irreducible term t' that can be obtained by rewriting t to R/B-irreducible or *normal* form, which is denoted by $t \to^!_{R/B} t'$, or $t!_{R/B}$ when t' is not relevant.

Since B-congruence classes can be infinite, $\to_{R/B}$-reducibility is undecidable in general. Therefore, R/B-rewriting is usually implemented by R,B-rewriting. We define the relation $\to_{R,B}$ on $\mathcal{T}_\Sigma(\mathcal{X})$ by $t \to_{p,R,B} t'$ (or simply $t \to_{R,B} t'$) iff there is a non-variable position $p \in Pos_\Sigma(t)$, a rule $l \to r$ in R, and a substitution σ such that $t|_p =_B l\sigma$ and $t' = t[r\sigma]_p$. To ensure completeness of R,B-rewriting w.r.t. R/B-rewriting, we require *strict coherence* [26] between the rules R and the equational theory B, ensuring that $=_B$ is a bisimulation for R,B-rewriting: for any Σ-terms u, u', v if $u =_B u'$ and $u \to_{R,B} v$, then there exists a term v' such that $u' \to_{R,B} v'$ and $v =_B v'$. Note that, assuming B-matching is decidable, $\to_{R,B}$ is decidable and notions such as convergence, irreducible term, and normalized substitution, are defined for $\to_{R,B}$ straightforwardly. It is worth noting that Maude automatically provides B-coherence completion for rules and equations.

Algebraic structures often involve axioms like associativity (A) and/or commutativity (C) of function symbols, which cannot be handled by ordinary term rewriting but instead are handled implicitly by working with congruence classes of terms. This is why often an equational theory E is decomposed into a disjoint union $E = E_0 \uplus B$, where B is a set of algebraic axioms (which are implicitly expressed in Maude as attributes of their corresponding operator using the `assoc`, `comm`, and `id`: keywords), and E_0 consists of (possibly conditional) equations that are implicitly oriented from left to right as a set $\overrightarrow{E_0}$ of rewrite rules (and operationally used as simplification rules modulo B), i.e., a rewrite theory $(\Sigma, B, \overrightarrow{E_0})$ is defined.

3 Pure homeomorphic embedding

The pure (syntactic) homeomorphic embedding relation known from term algebra [19] was introduced by Dershowitz for variable-arity symbols in [10] and for fixed-arity symbols in [9]. In the following, we consider only fixed-arity symbols.

Definition 1 (Homeomorphic embedding, Dershowitz [9]). *The homeomorphic embedding relation $\trianglelefteq$ over $\mathcal{T}_\Sigma$ is defined as follows:*

$$\frac{\exists i{\in}\{1,\ldots,n\} : s \trianglelefteq t_i}{s \trianglelefteq f(t_1,\ldots,t_n)} \qquad\qquad \frac{\forall i{\in}\{1,\ldots,n\} : s_i \trianglelefteq t_i}{f(s_1,\ldots,s_n) \trianglelefteq f(t_1,\ldots,t_n)}$$

with $n \geq 0$.

Roughly speaking, the left inference rule deletes subterms, while the right inference rule deletes context. We write $s \trianglelefteq t$ if s is derivable from t using the above rules. When $s \trianglelefteq t$, we say that s is (syntactically) *embedded* in t (or t syntactically *embeds* s). Note that $\equiv\, \subseteq\, \trianglelefteq$, where $\equiv$ denotes syntactic identity.

A well-quasi ordering (wqo) $\preceq$ on terms is a transitive and reflexive binary relation such that, for any infinite sequence of terms $t_1,t_2,\ldots$ with a finite number of operators, there exist i,j with $i < j$ and $t_i \preceq t_j$.

Theorem 1 (Tree Theorem, Kruskal [19]). *The embedding relation $\trianglelefteq$ is a well-quasi-ordering on $\mathcal{T}_\Sigma$.*

3.1 Mechanizing the Homeomorphic Embedding

The derivability relation given by $t \trianglelefteq t'$ is mechanized in [27] by introducing a term rewriting system $Emb(\Sigma)$ that is used to rewrite t' to t, i.e., $t' \rightarrow^*_{Emb(\Sigma)} t$. Similarly to Definition 1, order-sorted signatures are not considered.

Definition 2 (Rewrite theory for $\trianglelefteq$, Middeldorp [27]). *Let Σ be an unsorted signature. Homeomorphic embedding can be decided by a rewrite theory $Emb(\Sigma) = (\Sigma,\emptyset,R)$ such that R consists of rewrite rules of the form*

$$f(x_1,\cdots,x_n) \rightarrow x_i$$

where $f \in \Sigma$ is a function symbol of arity $n \geq 1$ and $i \in \{1,\cdots,n\}$.

Lemma 1 ([27]). *Given an unsorted signature Σ and two terms $t_1,t_2 \in \mathcal{T}_\Sigma$, we have $t_1 \trianglelefteq t_2$ iff $t_2 \rightarrow^*_{Emb(\Sigma)} t_1$.*

Definition 1 can be applied to terms of $\mathcal{T}_\Sigma(\mathcal{X})$ by simply regarding the variables in terms as constants. However, this definition cannot be used when existentially quantified variables are considered (as in logic programming or symbolic execution). The following definition from [21, 28] adapts the pure (syntactic) homeomorphic embedding from [10] by adding a simple treatment of logical variables where all variables are treated as if they were identical, which is enough for many symbolic methods such as the partial evaluation of [2]. Some extensions of $\trianglelefteq$ dealing with variadic symbols and infinite signatures are investigated in [22].

3.2 Symbolic Homeomorphic Embedding

The homeomorphic embedding is commonly applied not only to ground terms but also to terms with variables. The extension to variables is given by Leuschel [21].

$$\begin{array}{ccc} \text{Variable} & \text{Diving} & \text{Coupling} \\ \dfrac{}{x \trianglelefteq y} & \dfrac{\exists i \in \{1,\ldots,n\} : s \trianglelefteq t_i}{s \trianglelefteq f(t_1,\ldots,t_n)} & \dfrac{\forall i \in \{1,\ldots,n\} : s_i \trianglelefteq t_i}{f(s_1,\ldots,s_n) \trianglelefteq f(t_1,\ldots,t_n)} \end{array}$$

Fig. 1. Symbolic homeomorphic embedding

Definition 3 (Symbolic homeomorphic embedding, Leuschel [21]). *The extended homeomorphic embedding relation $\trianglelefteq$ over $\mathcal{T}_\Sigma(\mathcal{X})$ is defined in Figure 1, where the Variable inference rule allows dealing with free (unsorted) variables in terms, while the Diving and Coupling inference rules are similar to the pure (syntactic) homeomorphic embedding definition.*

For instance, given variables X and Y, $X \trianglelefteq g(Y)$, $g(X) \trianglelefteq f(g(Y))$, and $f(X) \trianglelefteq f(g(X))$ but $g(0) \ntrianglelefteq g(X)$. Note that the embedding relation $\trianglelefteq$ does not subsume the relative generality ordering $\leq$, e.g., $f(X) \ntrianglelefteq f(g(0))$ even if $f(g(0))$ is an instance of $f(X)$. Also note that the embedding relation $\trianglelefteq$ is not closed under instantiation, e.g., the instance of $g(X) \trianglelefteq f(g(Y))$ with substitution $\{X \mapsto 0, Y \mapsto 0\}$ holds, in symbols $g(0) \trianglelefteq f(g(0))$, whereas the instance of $g(X) \trianglelefteq f(g(Y))$ with substitution $\{X \mapsto g(Z)\}$ does not hold, in symbols $g(g(Z)) \ntrianglelefteq f(g(Y))$.

Theorem 2 ([21,28]). *The embedding relation $\trianglelefteq$ is a well-quasi-ordering on $\mathcal{T}_\Sigma(\mathcal{X})$.*

3.3 Adding sorts and subsorts

When we are interested in adding sorts and subsorts, the extension of the homeomorphic embedding to the order-sorted setting is not completely trivial. Let us provide a motivating example.

$$\begin{tikzcd} a \arrow[r] & A \arrow[loop above, "g"] \arrow[d, dash] \\ a \arrow[r] & B \arrow[loop below, "h"] \end{tikzcd}$$

Fig. 2. Signature graph of Example 1

Example 1. Consider the order-sorted signature depicted in Figure 2 that defines two sorts A and B, with $B < A$. Also, consider an overloaded constant a for sorts A and B, and two operators $g : A \to A$ and $h : B \to B$. Strictly speaking, the no ad-hoc overloading requirement would rule out ad-hoc overloaded constants such as a. However, this requirement can be relaxed in a natural and easy way by allowing constants to be qualified by their sort. For instance, in Maude this is done by enclosing them in parentheses followed by a dot and the sort name. For example, we can disambiguate the term a by writing either $(a).A$ or $(a).B$.

In an unsorted setting, where sorts are simply disregarded, both the embedding goals $a \underline{\blacktriangleleft} g(a)$ and $a \underline{\blacktriangleleft} h(a)$ hold.

In a many-sorted setting, where sorts are considered but subsort hierarchies are disregarded, it is immediate to extend the unsorted homeomorphic embedding relation $\underline{\blacktriangleleft}$ of Definition 1 to the many-sorted case. On the one hand, two constants $(a).A$ and $(a).B$ that belong to different sorts are distinct and hence incomparable by embedding. On the other hand, since we assume there is no ad-hoc overloading (except for constants), no extra check must be added to the Coupling rule because no two symbols f (of different sorts) exist, except for constants, while the Diving rule should be applied regardless of sorts. Thus, by abusing notation, $(a).A \ntrianglelefteq (a).B$ and $(a).B \ntrianglelefteq (a).A$, and hence $(a).A \trianglelefteq g((a).A)$ and $(a).B \trianglelefteq h((a).B)$, whereas $(a).B \ntrianglelefteq g((a).A)$ and $(a).A \ntrianglelefteq h((a).B)$.

Extending $\trianglelefteq$ to the order-sorted case is also easy in kind-complete signatures, where each constant has a kind (and so does any ground term). Therefore, by lifting all symbol declaration to the kind level, two constants $(c).s_1$ and $(c).s_2$ are only incomparable if the kinds $[s_1]$ and $[s_2]$ are different. By abusing notation, this means that all of the four embedding goals $(a).A \trianglelefteq g((a).A)$, $(a).B \trianglelefteq h((a).B)$, $(a).B \trianglelefteq g((a).A)$, and $(a).A \trianglelefteq h((a).B)$ hold (because $[A] = [B]$), whereas none of them would hold if we substitute A (resp. B) by a new sort C such that $[A] \neq [C]$ (resp. $[B] \neq [C]$).

Let us now consider the problem of extending Definition 3 to the order-sorted case, which requires to reason about variables. Fortunately, in kind-complete signatures every variable has a kind (and so does any term). Following Maude syntax, let us qualify any variable X by appending its name with a colon (:) followed by the sort name, e.g. $X{:}A$. Then, given sorts A, B, and C, with $B < A$, variables $X{:}A$ and $X{:}B$ are in the same kind, $[A]$, whereas $X{:}A$ and $X{:}C$ are in different kinds, $[A]$ and $[C]$, respectively.

Let us define the order-sorted symbolic homeomorphic embedding relation as follows.

Definition 4 (Order-sorted symbolic homeomorphic embedding). *The order-sorted symbolic homeomorphic embedding relation* $\breve{\trianglelefteq}$ *over* $\mathcal{T}_\Sigma(\mathcal{X})_\mathsf{s}$ *is defined in Figure 3, where the Variable inference rule allows dealing with free order-sorted variables in terms while the Diving and Coupling inference rules are similar to Definition 3.*

Variable $\quad\dfrac{\lceil x\rceil = \lceil y\rceil}{x \,\breve{\trianglelefteq}\, y}$ $\qquad$ Diving $\quad\dfrac{\exists i\in\{1,\ldots,n\}\ :\ s \,\breve{\trianglelefteq}\, t_i}{s \,\breve{\trianglelefteq}\, f(t_1,\ldots,t_n)}$ $\qquad$ Coupling $\quad\dfrac{\forall i\in\{1,\ldots,n\}\ :\ s_i \,\breve{\trianglelefteq}\, t_i}{f(s_1,\ldots,s_n) \,\breve{\trianglelefteq}\, f(t_1,\ldots,t_n)}$

Fig. 3. Order-sorted extended homeomorphic embedding

It is worth noting that, while it seems natural to consider that $X{:}A \,\breve{\trianglelefteq}\, Y{:}B$ for the case when A is a subsort of B (which holds because $[A] = [B]$, hence $\lceil X\rceil = \lceil Y\rceil$), the practical usefulness of having $X{:}A \,\breve{\trianglelefteq}\, Y{:}B$ is less evident for the case when B is a subsort of A (which holds for the very same reason). However, consider an overloaded operator g, with $g : A \to A$ and $g : B \to B$, with $A > B$. In a context of symbolic execution where logical variables are considered, it could be the case of having a computation sequence $(t_1, \ldots t_i, \ldots, t_j,\ t_{j+1} \ldots)$, with $t_i = g(X{:}A)$, $t_j = g(Y{:}B)$, and $t_{j+1} = g(X'{:}A)$, where t_{j+1} derives from t_j by a symbolic execution step (e.g., think of a narrowing step from $g(Y{:}B)$ by using a program rule $g(g(X'{:}A)) \to g(X'{:}A)$). Hence, the fact that $g(X{:}A) \,\breve{\trianglelefteq}\, g(Y{:}B)$ allows the risk of non-termination to be detected at t_j, i.e., before generating t_{j+1}; that is, this prevents an infinite sequence like $g(X{:}A), \ldots, g(X'{:}A), \ldots, g(X''{:}A), \ldots$ from being generated by applying the given rule.

Example 2. Consider the order-sorted signature depicted in Figure 4 which defines three sorts A, B, and C, with $B < A$. Also, consider a constant a of sort A, an operator $f : A \to B$, an operator $d : B \to A$, the (subsort) overloaded operator g for sorts A and B, with $g : A \to A$ and $g : B \to B$, a constant c of sort C, and an operator $h : C \to A$. Note that there are two different kinds $[A]$ and $[C]$, where $[A]$ contains sorts A and B,

Now consider three variables $X{:}A$, $Y{:}B$, and $Z{:}C$. By applying first the coupling rule and then the diving rule, the embedding goal $g(Y) \breve{\trianglelefteq} g(f(X))$ holds because variables X and Y are in

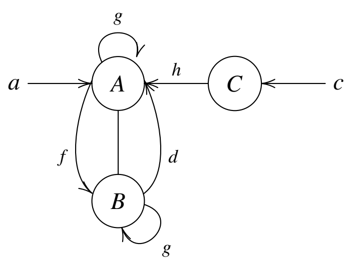

Fig. 4. Signature graph of Example 2

the same kind. However, $g(Y)\breve{\trianglelefteq}g(h(Z))$ (resp. $g(X)\breve{\trianglelefteq}g(h(Z))$) does not hold because Y (resp. X) and Z are in different kinds.

Note that the embedding goal $g(X)\breve{\trianglelefteq}g(Y)$ does also hold even though $g(X)$ is of sort A, $g(Y)$ is of sort B, and $B < A$. Similarly, the embedding goal $Y\breve{\trianglelefteq}d(X)$ does also hold even if the term $d(X)$ is not well-typed but admissible in Maude[5]; it belongs to the kind $[A]$ but not to sorts A and B. Actually, $d(X)$ has well-typed instances such as $d(Y)$, with the sort-decreasing substitution $\{X \mapsto Y\}$. Furthermore, the embedding goal $Z\breve{\trianglelefteq}d(X)$ does not hold, which is consistent with the fact that X cannot be instantiated with a term of sort C.

Similarly to $\trianglelefteq$, it is immediate to see that the order-sorted symbolic embedding relation $\breve{\trianglelefteq}$ is a well-quasi-ordering on the set of terms $\mathcal{T}_\Sigma(\mathcal{X})$.

Theorem 3. *The embedding relation* $\breve{\trianglelefteq}$ *is a well-quasi-ordering on* $\mathcal{T}_\Sigma(\mathcal{X})$.

Proof. The proof is similar to that of [20]. We need the following concept from [10] that we adapt to fixed-arity symbols. Let $\lesssim$ be a relation on a set $\mathcal{S}$ of symbols. Then the embedding extension of $\lesssim$ is a relation $\lesssim_{emb}$ on terms, constructed (only) from the symbols in $\mathcal{S}$, which is inductively defined as follows:

1. $t \lesssim_{emb} f(t_1,...,t_n)$ if $t \lesssim_{emb} t_i$ for some i;
2. $f(s_1,\ldots,s_n) \lesssim_{emb} g(t_1,\ldots,t_n)$ if $f \lesssim g$ and $\forall i \in \{1,\ldots,n\} : s_i \lesssim_{emb} t_i$.

We define the relation $\lesssim$ on the set $\mathcal{S} = (\Sigma \cup \mathcal{X})$ of symbols as the least relation satisfying:

1. $x \lesssim y$ if $x \in \mathcal{X}$, $y \in \mathcal{X}$, and $\lceil x \rceil = \lceil y \rceil$;
2. $f \lesssim f$ if $f \in \Sigma$.

This relation is a wqo on $\mathcal{S}$ because Σ is finite. Therefore, by the Higman-Kruskal's theorem (see e.g., [10]), its embedding extension to terms, $\lesssim_{emb}$, (which is by definition identical to $\breve{\trianglelefteq}$) is a wqo on $\mathcal{T}_\Sigma(\mathcal{X})$. $\square$

3.4 Getting rid of Variables

In the following, we show how it is possible to reformulate the homeomorphic embedding calculus for order-sorted terms with variables without the hassle of dealing with variables explicitly, which is easier to formalize and more efficient to be checked.

We need the following notation. Given the set $\mathcal{K} = \{[\mathsf{s}] \mid \mathsf{s} \in \mathsf{S}\}$ of all kinds in the equational theory (Σ, E), for each kind $k \in \mathcal{K}$ we consider a fresh constant symbol $\sharp_k$. Let $\Pi_{\mathcal{K}}$ be

[5] This is because we are assuming that, as in Maude, each signature is extended to a kind-complete one (recall Section 2).

the set of all such symbols, $\Pi_{\mathscr{K}} = \{\sharp_k \mid k \in \mathscr{K}\}$. Given an order-sorted signature Σ, we define $\Sigma^\sharp$ as the extension of Σ with the constants in $\Pi_{\mathscr{K}}$. For a term $t \in \mathscr{T}_\Sigma(\mathscr{X})$, let $t^\sharp \in \mathscr{T}_{\Sigma^\sharp}$ denote the (ground) instance of t where every variable x of sort s is replaced by the corresponding $\sharp_k$, with $k = [\mathsf{s}]$ being the kind of sort s.

The following result extends [3, Lemma 1] to the order-sorted case.

Lemma 2 (Variable-less characterization of $\breve{\trianglelefteq}$). *Given two terms $t_1, t_2 \in \mathscr{T}_\Sigma(\mathscr{X})$, we have $t_1 \breve{\trianglelefteq} t_2$ iff $t_1^\sharp \trianglelefteq t_2^\sharp$ where $t_1^\sharp, t_2^\sharp \in \mathscr{T}_{\Sigma^\sharp}$.*

Proof. Immediate by structural induction. The only relevant case is the base case, i.e., when the Variable inference rule $\frac{\lceil x \rceil = \lceil y \rceil}{x \breve{\trianglelefteq} y}$ is applied to the problem $t_1 \breve{\trianglelefteq} t_2$ and t_1 and t_2 are variables of the same kind. In the corresponding grounded homeomorphic embedding problem $t_1^\sharp \trianglelefteq t_2^\sharp$, we just have that $t_1^\sharp = t_2^\sharp = \sharp_k$ for the same kind $k = \lceil x \rceil = \lceil y \rceil$. But then, the result follows directly from the trivial problem $\sharp_k \trianglelefteq \sharp_k$. □

By abuse of notation, from now on, we sometimes consider terms with variables as ground terms with corresponding $\sharp_k$ symbols, since they are equivalent in our formulation.

Example 3. Consider again the order-sorted signature of Example 2. Also consider the embedding goals $g(Y) \breve{\trianglelefteq} g(f(X))$ and $g(Y) \breve{\trianglelefteq} g(h(Z))$, which were respectively proved to hold (because variables X and Y are in the same kind, i.e., $\lceil X \rceil = \lceil Y \rceil = [A]$) and not to hold (because Y and Z are in different kinds). For the corresponding grounded goals, it is immediate to see that $g(\sharp_{[A]}) \trianglelefteq g(f(\sharp_{[A]}))$ holds whereas $g(\sharp_{[A]}) \trianglelefteq g(h(\sharp_{[C]}))$ does not hold.

4 Homeomorphic embedding modulo equational axioms

Let us extend the order-sorted homeomorphic embedding relation on nonground terms $\breve{\trianglelefteq}$ to the case when we compare terms *modulo* a set of axioms B.

In [2], an equational extension of the "syntactically simpler" homeomorphic embedding relation $\trianglelefteq$ on nonground terms, called the B–embedding relation $\trianglelefteq_B$ (or embedding modulo B), was defined as follows: $\trianglelefteq_B \equiv (\overset{ren}{=}_B).(\trianglelefteq).(\overset{ren}{=}_B)$, where $v \overset{ren}{=}_B v'$ iff there is a renaming substitution σ for v' such that $v =_B v'\sigma$. We define the corresponding order-sorted extension $\breve{\trianglelefteq}_B$ of $\trianglelefteq_B$ in the natural way..

Definition 5 ((Order-sorted) homeomorphic embedding modulo B). *The order-sorted B–embedding relation $\breve{\trianglelefteq}_B$ is $(\overset{ren}{=}_B).(\breve{\trianglelefteq}).(\overset{ren}{=}_B)$.*

Example 4. Consider the following rewrite theory (written in Maude syntax) that defines the signature of natural numbers. The defined sort hierarchy has top sort `Nat` and (disjoint) subsorts `Zero` and `NzNat` (for non-zero natural numbers). The sort `Nat` is generated from the constant `0` (of sort `Zero`) and the successor operator `suc`[6] (of sort `NzNat`). We also define the associative and commutative natural addition operator symbol `_+_` for sort `Nat` but add two extra subsort-overloaded definitions.

```
fmod NAT is
  sorts Zero NzNat Nat .
  subsorts Zero NzNat < Nat .
  op 0 : -> Zero .
  op suc : Nat -> NzNat .
  op _+_ : Zero Zero -> Zero [assoc comm] .
  op _+_ : NzNat Zero -> NzNat [assoc comm] .
  op _+_ : Nat Nat -> Nat [assoc comm] .
endfm
```

[6] For simplicity, we represent natural numbers in normal decimal notation; e.g., 2 for `suc(suc((0))`.

Then, we have $1 + X{:}Nat \breve{\trianglelefteq}_B Y{:}Nat + (1+2)$ because $Y{:}Nat + (1+2)$ is equal to $1 + (2 + Y{:}Nat)$ modulo the associativity and commutativity of _+_, and $1 + X{:}Nat$ is homeomorphically embedded into $1 + (2 + Y{:}Nat)$. Similarly, $1 + X{:}Zero \breve{\trianglelefteq}_B Y{:}Nat + (1+2)$ and $1 + X{:}NzNat \breve{\trianglelefteq}_B Y{:}Nat + (1+2)$ hold. Note that $suc(X{:}Nat) \breve{\trianglelefteq}_B suc(Y{:}Zero)$ holds despite the fact that $Zero < Nat$; this is a deliberate decision as explained in Example 2.

The following result extends Kruskal's Tree Theorem for the equational theories B considered in this paper, which we restrict to *class–finite* equational theories. B is called *class-finite* if all of the B-equivalence classes of terms in the quotient term algebra $\mathcal{T}_\Sigma(\mathcal{X})/\!=_B$ are finite. This includes the class of permutative equational theories: an equational theory $\mathcal{E}$ is permutative if for all terms $t,\ t'$, the fact that $t =_{\mathcal{E}} t'$ implies that the terms t and t' contain the same symbols with the same number of occurrences [6]. Permutative theories include any theory with any combination of symbols obeying any combination of associativity and commutativity axioms.

Theorem 4. *For class-finite theories, the embedding relation $\breve{\trianglelefteq}_B$ is a well-quasi-ordering on the set $\mathcal{T}_\Sigma(\mathcal{X})$ for finite Σ.*

Proof. A binary relation $\succ$ is Noetherian (i.e., well-founded) on a set X if and only if its dual relation $\preceq$ (defined as $u \preceq v$ iff $u \not\succ v$) is a well order[7] on X: in every sequence $(x_i)_{i\in\mathbb{N}}$ of elements of X, there exist $i < j$ such that $x_i \preceq x_j$. Since $\breve{\trianglelefteq}$ is a wqo (by Theorem 3), the result follows for class-finite theories from the fact that $>_B$ is well-founded [6], and $\breve{\trianglelefteq}$ is compatible with $(\overset{ren}{=}_B)$. $\square$

Similarly to Lemma 2, in the equational case, we can also get rid of variables and consider only ground terms.

Lemma 3. *Given two terms $t_1,t_2 \in \mathcal{T}_\Sigma(\mathcal{X})$, we have $t_1 \breve{\trianglelefteq}_B t_2$ iff $t_1^\sharp \breve{\trianglelefteq}_B t_2^\sharp$ where $t_1^\sharp,t_2^\sharp \in \mathcal{T}_{\Sigma^\sharp}$.*

Function symbols with variable arity are sometimes seen as associative operators. Let us briefly discuss by means of an example the homeomorphic embedding modulo axioms $\breve{\trianglelefteq}_B$ of Definition 5 when compared with the variadic extension $\underline{\blacktriangleleft}^v$ of Definition 1 as given in [10]:

$$\text{Diving}\quad \frac{\exists i{\in}\{1,\ldots,n\} : s \;\underline{\blacktriangleleft}^v\; t_i}{s \;\underline{\blacktriangleleft}^v\; f(t_1,\ldots,t_n)} \qquad\qquad \text{Coupling}\quad \frac{\forall i{\in}\{1,\ldots,m\} : s_i \;\underline{\blacktriangleleft}^v\; t_{j_i}, \text{with } 1{\leq}j_1{<}j_2{<}\cdots{<}j_m{\leq}n}{f(s_1,\ldots,s_m) \;\underline{\blacktriangleleft}^v\; f(t_1,\ldots,t_n)}$$

Example 5. Consider a variadic version of the addition symbol $+$ of Example 4 that allows any number of natural numbers to be used as arguments; for instance, $+(1,2,3)$. On the one hand, $+(1) \;\underline{\blacktriangleleft}^v\; +(1,2,3)$ whereas $+(1) \not\breve{\trianglelefteq}_B +(1,2,3)$, with B consisting of the associativity and commutativity axioms for the operator $+$ (actually, $+(1)$ is ill-formed for our relation $\breve{\trianglelefteq}_B$). On the other hand, we have both $+(1,2) \;\underline{\blacktriangleleft}^v\; +(1,0,3,2)$ and $+(1,2) \breve{\trianglelefteq}_B +(1,0,3,2)$. This is because any well-formed term that consists of the addition (in any order) of the constants 0, 1, 2, and 3 (for instance, $+(+(1,0),+(3,2))$ can be given a flat representation $+(1,0,2,3)$. Note that there are many other equivalent terms due to associativity and commutativity, e.g., $+(+(1,2),+(3,0))$ or $+(+(1,+(3,2)),0)$, all of which are represented by the flattened term $+(0,1,2,3)$. Actually, because of the associativity and commutativity of symbol $+$, flattened terms like $+(1,0,2,3)$ can be further simplified into a single[8] *canonical representative* $+(0,1,2,3)$, hence also $+(1,2) \breve{\trianglelefteq}_B +(0,1,2,3)$. A more detailed explanation of flat terms can be found in Section 6. However, note that $+(2,1) \breve{\trianglelefteq}_B +(1,0,3,2)$ but $+(2,1) \;\not\underline{\blacktriangleleft}^v\; +(1,0,3,2)$ because the relation $\underline{\blacktriangleleft}^v$ does not consider the commutativity of symbol $+$. In contrast, if $+$ was associative (and not commutative), then $+(2,1) \not\breve{\trianglelefteq}_B +(1,0,3,2)$.

[7] A well order is a wqo that is a proper ordering relation, i.e., it is antisymmetric.

[8] Maude uses a term lexicographic order for the arguments of flattened terms [11].

Roughly speaking, in the worst case, the homeomorphic embedding modulo axioms B of Definition 5, $t \,\breve{\trianglelefteq}_B\, t'$, amounts to considering all of the elements in the B-equivalence classes of t and t' and then checking for standard homeomorphic embedding, $u \,\breve{\trianglelefteq}\, u'$, every pair u and u' of such terms, one term from each class. Unfortunately, the enumeration of all terms in a B-equivalence class is impractical, as shown in the following example.

Example 6. Consider the AC binary symbol $+$ of Example 4 and the terms $t = +(1,2)$ and $t' = +(2,+(3,1))$. The AC-equivalence class of t contains two terms whereas the AC-equivalence class of t' contains 12 terms. This implies checking 24 embedding problems $u \,\breve{\trianglelefteq}\, u'$ in order to decide $t \,\breve{\trianglelefteq}_{AC}\, t'$, in the worst case. Moreover, we know a priori that half of these embedding tests will fail (those in which 1 and 2 occur in different order in u' and u; for instance $u' = +(1,+(2,3))$ and $u = +(2,1)$.

A more effective rewriting characterization of $\breve{\trianglelefteq}_B$ can be achieved by lifting Definition 2 to both the order-sorted and the *modulo* case in a natural way. However, ill-formed terms can be produced by naïvely applying the rules $f(x_1,\ldots,x_n) \to x_i$ of Definition 2 to typed (i.e., order-sorted) terms. For example, "$(\mathtt{0} \leq \mathtt{1})$ `or` `true`" $\to$ "`0` `or` `true`".

In the order-sorted context, we can overcome this drawback as follows. We extend Σ to a new signature $\Sigma^{\mathscr{U}}$ by adding a new top sort $\mathscr{U}$ that is bigger than all other sorts. Now, for each $f : A_1,\ldots,A_n \to A$ in Σ, we add the rules $f(x_1{:}\mathscr{U},\ldots,x_n{:}\mathscr{U}) \to x_i{:}\mathscr{U}$, $1 \leq i \leq n$. In this way, rewriting with $\to^*_{Emb(\Sigma^{\mathscr{U}})/B}$ becomes a relation between well-formed $\Sigma^{\mathscr{U}}$-terms, as first proposed in [2].

The order-sorted homeomorphic embedding modulo B can be decided by the following rewrite theory that extends the definition in [2] with sorts.

Definition 6 (Rewrite theory for $\breve{\trianglelefteq}_B$). *Let Σ be an order-sorted signature and B be a set of axioms. Let us introduce the following signature transformation $\Sigma \ni (f : \mathsf{s}_1 \ldots \mathsf{s}_n \to \mathsf{s}) \mapsto (f : \mathscr{U} \overset{n}{\ldots} \mathscr{U} \to \mathscr{U}) \in \Sigma^{\mathscr{U}}$, where $\mathscr{U}$ conceptually represents a universal supersort of all sorts in Σ. Since there is no ad-hoc overloading, any term $t \in \mathscr{T}_\Sigma$ is well-typed for $\Sigma^{\mathscr{U}}$.*

We define the rewrite theory $Emb(\Sigma) = (\Sigma^{\mathscr{U}}, B, R)$ such that R consists of all rewrite rules

$$f(x_1{:}\mathscr{U},\ldots,x_n{:}\mathscr{U}) \to x_i{:}\mathscr{U}$$

for each $f : A_1,\ldots,A_n \to A$ in Σ and $i \in \{1,\ldots,n\}$.

Proposition 1. *Given Σ, B, and terms $t,t' \in \mathscr{T}_\Sigma(\mathscr{X})$, $t \,\breve{\trianglelefteq}_B\, t'$ iff $t'^\sharp \to^*_{Emb(\Sigma^\sharp)/B} t^\sharp$.*

Proof. By Lemma 3, $t \,\breve{\trianglelefteq}_B\, t'$ iff $t^\sharp \,\breve{\trianglelefteq}_B\, t'^\sharp$. We prove $t^\sharp \,\breve{\trianglelefteq}_B\, t'^\sharp$ iff $t'^\sharp \to^*_{Emb(\Sigma^\sharp)/B} t^\sharp$.

($\Rightarrow$) We prove that $t^\sharp \,\breve{\trianglelefteq}_B\, t'^\sharp$ implies $t'^\sharp \to^*_{Emb(\Sigma^\sharp)/B} t^\sharp$. By Definition 5, there are two terms w,w' such that $t^\sharp =_B w$, $t'^\sharp =_B w'$, and $w \,\breve{\trianglelefteq}\, w'$. By Lemma 2 $w \,\breve{\trianglelefteq}\, w'$ iff $w \trianglelefteq w'$. By Lemma 1, $w \trianglelefteq w'$ implies $w' \to^*_{Emb(\Sigma^\sharp)} w$. And this implies $t'^\sharp \to^*_{Emb(\Sigma^\sharp)/B} t^\sharp$.

($\Leftarrow$) We prove that $t'^\sharp \to^*_{Emb(\Sigma^\sharp)/B} t^\sharp$ implies $t^\sharp \,\breve{\trianglelefteq}_B\, t'^\sharp$ by induction on the length k of the rewriting sequence. If $k = 0$, then there is a constant a such that $t'^\sharp = a = t^\sharp$ and $a \to^*_{Emb(\Sigma^\sharp)/B} a$. Thus, $a \,\breve{\trianglelefteq}_B\, a$ by using the Coupling inference rule of Figure 3 with $n = 0$. If $k > 0$, then there exists w such that $t'^\sharp \to^{k-1}_{Emb(\Sigma^\sharp)/B} w \to_{Emb(\Sigma^\sharp)/B} t^\sharp$. Given the form $f(x_1{:}\mathscr{U},\ldots,x_n{:}\mathscr{U}) \to x_i{:}\mathscr{U}$ of the rewrite rules in $Emb(\Sigma^\sharp)$, we have that $w =_B C[f(t_1,\ldots,t_n)]$ and there exists $i \in \{1,\ldots,n\}$ s.t. $C[t_i] =_B t^\sharp$. Hence, it is immediate to prove $t^\sharp \,\breve{\trianglelefteq}_B\, w$ by a straightforward combination of several applications of the Coupling inference rule to remove the context $C[\,]$ from both terms, $t^\sharp$ and w, one application of the Diving inference rule to extract t_i from $f(t_1,\ldots,t_n)$ in w, and the remaining applications of the Coupling inference rule that are needed to recognize the embedding of t_i inside $t^\sharp$. Now, by induction hypothesis we also have that $w \,\breve{\trianglelefteq}_B\, t'^\sharp$, which yields $t^\sharp \,\breve{\trianglelefteq}_B\, w \,\breve{\trianglelefteq}_B\, t'^\sharp$. □

Example 7. Consider the order-sorted signature for natural numbers of Example 4. Let us represent by sort U in Maude the unique (top) sort of the transformed signature:

```
fmod NAT-U is
  sort U .
  op 0 : -> U .
  op suc : U -> U .
  op _+_ : U U -> U [assoc comm] .
endfm
```

Since we have no ad-hoc overloading, there is no need to transform a ground term of module NAT into a ground term of module NAT-U since any ground term of sorts Zero, NzNat, and Nat is well typed in the transformed signature. The associated rewrite theory $Emb(\Sigma^\sharp)$ contains the following two rules for the operator $+$:

$$+(x_1{:}U, x_2{:}U) \to x_1{:}U$$
$$+(x_1{:}U, x_2{:}U) \to x_2{:}U$$

However, when the rules of $Emb(\Sigma^\sharp)$ are used for rewriting modulo the commutativity of symbol $+$ (as in Maude), in practice, we can get rid of one of the two rules above since only one of them is required.

Example 8. Following Example 6, instead of comparing pairwisely all terms in the equivalence classes of t and t', we use the rewrite rule $+(x_1{:}U, x_2{:}U) \to x_2{:}U$ to prove the rewrite step $+(2,+(3,1)) \to_{Emb(\Sigma^\sharp)/B} +(2,1)$, and finally, we check that $+(2,1) =_B +(1,2)$. Recall that B contains associativity and commutativity of $+$. However, there are six alternative rewriting steps stemming from the initial term $+(2,+(3,1))$. Five of these rewrite steps are useless for proving the considered embedding (the selected redex is underlined):

$$\begin{array}{ll}
+(2,\underline{+(3,1)}) \to_{Emb(\Sigma^\sharp)/B} +(2,1) & \underline{+(2,+(3,1))} \to_{Emb(\Sigma^\sharp)/B} 1 \\
+(2,\underline{+(3,1)}) \to_{Emb(\Sigma^\sharp)/B} +(2,3) & \underline{+(2,+(3,1))} \to_{Emb(\Sigma^\sharp)/B} 2 \\
\underline{+(2,+(3,1))} \to_{Emb(\Sigma^\sharp)/B} +(3,1) & \underline{+(2,+(3,1))} \to_{Emb(\Sigma^\sharp)/B} 3
\end{array}$$

For a term with k addends, we have $(2^k)-2$ rewriting steps. This leads to a huge combinatorial explosion when considering the complete rewrite search tree.

In conclusion, there are three problems when trying to use Definition 6 to mechanize the order-sorted homeomorphic embedding relation modulo axioms $\breve{\trianglelefteq}_B$ of Definition 5. First, the intrinsic non-determinism of the rules may unnecessarily produce an extremely large search space. Second, as shown in Example 8, this intrinsic non-determinism in the presence of axioms is intolerable, that is, unfeasible to handle. Third, the associated reachability problems do not scale up to complex embedding problems so that a suitable search strategy must be introduced. We address these problems stepwise in the sequel.

5 Goal-driven homeomorphic embedding modulo *B*

The formulation of homeomorphic embedding as a reachability problem by using the rewrite rules of Definition 6 generates a blind search that does not take advantage of the actual terms t and t' being compared for embedding. In this section, we provide a more refined formulation of homeomorphic embedding modulo axioms that is *goal driven* in the sense that, given an embedding problem for t and t', it inductively processes the terms t and t' in a top-down manner.

First, in the following section, we introduce a calculus that extends the homeomorphic embedding relation of Definition 3 to the order-sorted equational case.

5.1 An order-sorted homeomorphic embedding calculus modulo B

Let us introduce an order-sorted calculus for embeddability goals $t \breve{\trianglelefteq}^{gd}_B t'$ that directly handles the algebraic axioms of B in the deduction system, with B being any combination of A and/or C axioms for the theory operators. Roughly speaking, this is achieved by specializing the Coupling rule of Definition 3 w.r.t. B. This calculus extends with sorts (kinds) a similar definition given in [2], where kinds were not considered in the Variable inference rule of [2].

Definition 7 (Goal-driven (order-sorted) homeomorphic embedding modulo B). *The homeomorphic embedding relation modulo B is defined as the smallest relation that satisfies the inference rules of Definition 3 together with the new inference rules given in Figure 5. These are:*

1. *the three inference rules (Variable, Diving, and Coupling) of Definition 3 for any function symbol;*
2. *one extra coupling rule for the case of a commutative symbol with or without associativity (Coupling$_C$);*
3. *two extra coupling rules for the case of an associative symbol with or without commutativity (Coupling$_A$); and*
4. *two extra coupling rules for the case of an associative-commutative symbol (Coupling$_{AC}$).*

Coupling$_C$

$$\frac{s_0 \breve{\trianglelefteq}^{gd}_B t_1 \ \wedge \ s_1 \breve{\trianglelefteq}^{gd}_B t_0}{f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B f(t_0,t_1)}$$

Coupling$_A$

$$\frac{f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B t_0 \ \wedge \ s_2 \breve{\trianglelefteq}^{gd}_B t_1}{f(s_0,f(s_1,s_2)) \breve{\trianglelefteq}^{gd}_B f(t_0,t_1)} \qquad \frac{s_0 \breve{\trianglelefteq}^{gd}_B f(t_0,t_1) \ \wedge \ s_1 \breve{\trianglelefteq}^{gd}_B t_2}{f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B f(t_0,f(t_1,t_2))}$$

Coupling$_{AC}$

$$\frac{f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B t_1 \ \wedge \ s_2 \breve{\trianglelefteq}^{gd}_B t_0}{f(s_0,f(s_1,s_2)) \breve{\trianglelefteq}^{gd}_B f(t_0,t_1)} \qquad \frac{s_1 \breve{\trianglelefteq}^{gd}_B f(t_0,t_1) \ \wedge \ s_0 \breve{\trianglelefteq}^{gd}_B t_2}{f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B f(t_0,f(t_1,t_2))}$$

Fig. 5. Extra coupling rules for A, C, and AC symbols

Proposition 2. *Given Σ, B, and terms $t,t' \in \mathcal{T}_\Sigma(\mathcal{X})$, $t \trianglelefteq_B t'$ iff $t^\sharp \breve{\trianglelefteq}^{gd}_B t'^\sharp$ where $t^\sharp, t'^\sharp \in \mathcal{T}_{\Sigma^\sharp}$.*

Proof. Immediate by considering that each inference rule of Figure 5 explicitly combines the Coupling inference rule with the corresponding A and/or C axioms. For example, the inference rule Coupling$_C$ of Figure 5 is equivalent to the inference rule Coupling of Figure 3, but when the former considers an embedding problem $f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B f(t_0,t_1)$, the latter considers both $f(s_1,s_0) \breve{\trianglelefteq}^{gd}_B f(t_0,t_1)$ and $f(s_0,s_1) \breve{\trianglelefteq}^{gd}_B f(t_1,t_0)$. □

Example 9. Consider the binary symbol $+$ obeying associativity and commutativity axioms, and the terms $t = +(1,2)$ and $t' = +(2,+(3,1))$ of Example 8. We can prove $t \breve{\trianglelefteq}^{gd}_B t'$ by

$$\frac{\dfrac{1 \breve{\trianglelefteq}^{gd}_B 1}{1 \breve{\trianglelefteq}^{gd}_B +(3,1)} \qquad 2 \breve{\trianglelefteq}^{gd}_B 2}{+(1,2) \breve{\trianglelefteq}^{gd}_B +(2,+(3,1))}$$

We can also prove a more complex embedding goal by first using the right inference rule for AC of Figure 5 and then the generic Coupling and Diving inference rules.

$$\frac{\dfrac{\dfrac{2 \breve{\trianglelefteq}^{gd}_B 2}{2 \breve{\trianglelefteq}^{gd}_B +(4,2)} \qquad 3 \breve{\trianglelefteq}^{gd}_B 3}{+(2,3) \breve{\trianglelefteq}^{gd}_B +(+(4,2),3)} \qquad 1 \breve{\trianglelefteq}^{gd}_B 1}{+(1,+(2,3)) \breve{\trianglelefteq}^{gd}_B +(+(4,2),+(3,1))}$$

It is immediate to see that, when the size of the involved terms t and t' grows, the improvement in performance of $\breve{\trianglelefteq}_B^{gd}$ w.r.t. $\breve{\trianglelefteq}_B$ can be significant (just compare these two embedding proofs with the corresponding search trees for $\breve{\trianglelefteq}_B$).

5.2 Reachability-based, (order-sorted) goal-driven homeomorphic embedding formulation

Let us provide a more operational goal-driven characterization of the order-sorted homeomorphic embedding modulo B by means of the relation $\breve{\trianglelefteq}_B^{rbgd}$. We formalize it in the reachability style of Definition 6. The main challenge here is how to generate a suitable rewrite theory $R^{rbgd}(\Sigma,B)$ that can decide embedding modulo B by running a reachability goal.

Definition 8 (Goal-driven homeomorphic embedding rewrite rules modulo B). *Given Σ and B, a rewrite theory $R^{rbgd}(\Sigma,B) = (\Sigma,B,R)$ is defined as follows*

1. *For each particular instance of the inference rules of the form $\frac{}{u\breve{\trianglelefteq}_B^{gd} v}$ given in Definition 7 (e.g., the Variable Inference Rule from Definition 3 or the Coupling Inference Rule from Definition 3, for the case of a constant symbol c), we include in R a rewrite rule of the form $u\breve{\trianglelefteq}_B^{rbgd} v \to true$.*
2. *For each particular instance of the inference rules of the form $\frac{u_1\breve{\trianglelefteq}_B^{gd} v_1 \wedge\cdots\wedge u_k\breve{\trianglelefteq}_B^{gd} v_k}{u\breve{\trianglelefteq}_B^{gd} v}$ given in Definition 7, we include in R a rewrite rule of the form $u\breve{\trianglelefteq}_B^{rbgd} v \to u_1\breve{\trianglelefteq}_B^{rbgd} v_1 \wedge\cdots\wedge u_k\breve{\trianglelefteq}_B^{rbgd} v_k$.*

Proposition 3. *Given Σ, B, and terms $t,t' \in \mathcal{T}_{\Sigma^\sharp}$, $t \breve{\trianglelefteq}_B^{gd} t'$ iff $(t\breve{\trianglelefteq}_B^{rbgd} t') \to^*_{R^{rbgd}(\Sigma^\sharp,B)/B} true$.*

Proof. Immediate by a straightforward transformation of inference rules into rewrite rules. □

Example 10. Consider the binary symbol $+$ of Example 4. According to Definition 7 and the use of only ground terms, there are eleven inference rules for $\breve{\trianglelefteq}_B^{gd}$:

Diving

$$\frac{t\breve{\trianglelefteq}_B^{gd} t'}{t\breve{\trianglelefteq}_B^{gd} suc(t')} \qquad \frac{t\breve{\trianglelefteq}_B^{gd} t_1}{t\breve{\trianglelefteq}_B^{gd} +(t_1,t_2)} \qquad \frac{t\breve{\trianglelefteq}_B^{gd} t_2}{t\breve{\trianglelefteq}_B^{gd} +(t_1,t_2)}$$

Coupling

$$\frac{}{0\breve{\trianglelefteq}_B^{gd} 0} \qquad \frac{t\breve{\trianglelefteq}_B^{gd} t'}{suc(t)\breve{\trianglelefteq}_B^{gd} suc(t')} \qquad \frac{t_1\breve{\trianglelefteq}_B^{gd} t'_1 \wedge t_2\breve{\trianglelefteq}_B^{gd} t'_2}{+(t_1,t_2)\breve{\trianglelefteq}_B^{gd} +(t'_1,t'_2)}$$

Coupling$_C$

$$\frac{t_1\breve{\trianglelefteq}_B^{gd} t'_2 \wedge t_2\breve{\trianglelefteq}_B^{gd} t'_1}{+(t_1,t_2)\breve{\trianglelefteq}_B^{gd} +(t'_1,t'_2)}$$

Coupling$_A$

$$\frac{+(t_0,t_1)\breve{\trianglelefteq}_B^{gd} t'_1 \wedge t_2\breve{\trianglelefteq}_B^{gd} t'_2}{+(t_0,+(t_1,t_2))\breve{\trianglelefteq}_B^{gd} +(t'_1,t'_2)} \qquad \frac{t_1\breve{\trianglelefteq}_B^{gd} +(t'_0,t'_1) \wedge t_2\breve{\trianglelefteq}_B^{gd} t'_2}{+(t_1,t_2)\breve{\trianglelefteq}_B^{gd} +(t'_0,+(t'_1,t'_2))}$$

Coupling$_{AC}$

$$\frac{+(t_0,t_1)\breve{\trianglelefteq}_B^{gd} t'_2 \wedge t_2\breve{\trianglelefteq}_B^{gd} t'_1}{+(t_0,+(t_1,t_2))\breve{\trianglelefteq}_B^{gd} +(t'_1,t'_2)} \qquad \frac{t_2\breve{\trianglelefteq}_B^{gd} +(t'_0,t'_1) \wedge t_1\breve{\trianglelefteq}_B^{gd} t'_2}{+(t_1,t_2)\breve{\trianglelefteq}_B^{gd} +(t'_0,+(t'_1,t'_2))}$$

However, the corresponding TRS $R^{rbgd}(\Sigma,B)$ only contains seven rewrite rules because, due to pattern matching modulo associativity and commutativity in rewriting logic, the other rules are redundant:

$$
\begin{array}{rll}
\text{(Diving)} & T \mathrel{\check{\trianglelefteq}}_B^{rbgd} suc(T') & \to T \mathrel{\check{\trianglelefteq}}_B^{rbgd} T' \\
 & T \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(T_1,T_2) & \to T \mathrel{\check{\trianglelefteq}}_B^{rbgd} T_1 \\
\text{(Coupling)} & \sharp \mathrel{\check{\trianglelefteq}}_B^{rbgd} \sharp & \to true \\
 & 0 \mathrel{\check{\trianglelefteq}}_B^{rbgd} 0 & \to true \\
 & suc(T) \mathrel{\check{\trianglelefteq}}_B^{rbgd} suc(T') & \to T \mathrel{\check{\trianglelefteq}}_B^{rbgd} T' \\
\text{(Coupling}_{\emptyset,C,A,AC}\text{)} & +(T_1,T_2) \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(T_1',T_2') & \to T_1 \mathrel{\check{\trianglelefteq}}_B^{rbgd} T_1' \wedge T_2 \mathrel{\check{\trianglelefteq}}_B^{rbgd} T_2'
\end{array}
$$

For example, the rewrite sequence proving $+(1,+(2,3)) \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(+(4,2),+(3,1))$ is:

$$
\begin{array}{rl}
+(1,+(2,3)) \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(+(4,2),+(3,1)) & \to_{R^{rbgd}(\Sigma^\sharp,B)/B} +(2,3)) \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(+(4,2),3) \wedge 1 \mathrel{\check{\trianglelefteq}}_B^{rbgd} 1 \\
 & \to_{R^{rbgd}(\Sigma^\sharp,B)/B} 2 \mathrel{\check{\trianglelefteq}}_B^{rbgd} +(4,2) \wedge 3 \mathrel{\check{\trianglelefteq}}_B^{rbgd} 3 \\
 & \to_{R^{rbgd}(\Sigma^\sharp,B)/B} 2 \mathrel{\check{\trianglelefteq}}_B^{rbgd} 2 \\
 & \to_{R^{rbgd}(\Sigma^\sharp,B)/B} true
\end{array}
$$

Although the improvement in performance achieved by using the rewriting relation $\to_{R^{rbgd}(\Sigma^\sharp,B)/B}$ versus the rewriting relation $\to^*_{Emb(\Sigma^\sharp)/B}$ is important, the search space is still huge since the expression $+(1,+(2,3)) \mathrel{\check{\trianglelefteq}}_B^{gd} +(+(4,2),+(3,1))$ matches the left-hand side $+(T_1,T_2) \mathrel{\check{\trianglelefteq}}_B^{gd} +(T_1',T_2')$ in many different ways (e.g., $\{T_1 \mapsto 1, T_2 \mapsto +(2,3),\ldots\}$, $\{T_1 \mapsto 2, T_2 \mapsto +(1,3),\ldots\}$, $\{T_1 \mapsto 3, T_2 \mapsto +(1,2),\ldots\}$).

In the following section, we provide a more efficient calculus of homeomorphic embedding modulo axioms by considering equational (deterministic) normalization (thus avoiding search) and by exploiting the meta-level features of Maude (thus avoiding any theory generation).

6 Meta-Level deterministic (order-sorted) goal-driven homeomorphic embedding modulo B

The implementation of rewriting modulo equational axioms in high-performance languages such as Maude relies on a meta-level, flattened representation of terms that is rooted by polyvariadic versions of the associative (or associative-commutative) symbols of the term [7, Chapter 14]. For instance, given an associative symbol f with n arguments and $n \geq 2$, flattened terms rooted by f are canonical forms w.r.t. the set of rules given by the following rule schema

$$f(x_1,\ldots,f(t_1,\ldots,t_n),\ldots,x_m) \to f(x_1,\ldots,t_1,\ldots,t_n,\ldots,x_m) \quad n,m \geq 2$$

The flattened version of a term t is represented by $\underline{t}$. Given an associative symbol f and a term $f(t_1,\ldots,t_n)$, those terms among the $t_1,\ldots,t_n$ that are not rooted by f are called *f-alien terms* (or simply *alien terms*). In the following, we implicitly consider that all terms are in canonical form.

Our first improvement for implementing the order-sorted homeomorphic embedding modulo equational axioms of Definition 7 is based on handling flattened terms explicitly. Roughly speaking, for a homeomorphic problem of the form $f(s_1,\ldots,s_n) \mathrel{\check{\trianglelefteq}}_B^{ml} f(t_1,\ldots,t_m)$, with f being associative, we just search for an argument t_j in the right-hand term that embeds s_1 and reduce the previous problem to the new problem $f(s_2,\ldots,s_n) \mathrel{\check{\trianglelefteq}}_B^{ml} f\left(t_{j+1},\ldots,t_m\right)$, since all the arguments $t_1,\ldots,t_{j-1}$ can be discarded. However, in the case when f is associative and commutative, the arguments $t_1,\ldots,t_{j-1}$ cannot be discarded and we reduce the original problem $f(s_1,\ldots,s_n) \mathrel{\check{\trianglelefteq}}_B^{ml} f(t_1,\ldots,t_m)$ to the new problem $f(s_2,\ldots,s_n) \mathrel{\check{\trianglelefteq}}_B^{ml} f\left(t_1,\ldots,t_{j-1},t_{j+1},\ldots,t_m\right)$.

Definition 9 (Flattened homeomorphic embedding modulo B). *The homeomorphic embedding relation modulo B, $\breve{\trianglelefteq}^{ml}_{B}$,* for terms in flattened form *is defined as the smallest relation that satisfies the following inference rules:*

1. *the three inference rules (Variable, Diving, and Coupling) of Definition 3 for any function symbol;*
2. *the extra coupling rule of Figure 5 for the case of a commutative symbol with or without associativity (Coupling$_C$);*
3. *the extra coupling rule of Figure 6 for the case of an associative symbol with or without commutativity (Coupling$_A$); and*
4. *the extra coupling rule of Figure 6 for the case of an associative-commutative symbol (Coupling$_{AC}$).*

$$\mathit{Coupling}_A\,\frac{\exists j \in \{1,\ldots,m-n+1\} : s_1 \breve{\trianglelefteq}^{ml}_{B} t_j \wedge f\left(s_2,\ldots,s_n\right) \breve{\trianglelefteq}^{ml}_{B} f\left(t_{j+1},\ldots,t_m\right) \wedge \forall k < j : s_1 \not\breve{\trianglelefteq}^{ml}_{B} t_k}{f\left(s_1,\ldots,s_n\right) \breve{\trianglelefteq}^{ml}_{B} f\left(t_1,\ldots,t_m\right)}$$

$$\mathit{Coupling}_{AC}\,\frac{\exists j \in \{1,\ldots,m\} : s_1 \breve{\trianglelefteq}^{ml}_{B} t_j \wedge f\left(s_2,\ldots,s_n\right) \breve{\trianglelefteq}^{ml}_{B} f\left(t_1,\ldots,t_{j-1},t_{j+1},\ldots,t_m\right)}{f\left(s_1,\ldots,s_n\right) \breve{\trianglelefteq}^{ml}_{B} f\left(t_1,\ldots,t_m\right)}$$

Fig. 6. Coupling rules for flattened terms with associative and associative-commutative symbols

Proposition 4. *Given Σ and B, for terms t and t' in $\mathcal{T}_{\Sigma}(\mathcal{X})$, $t \breve{\trianglelefteq}^{gd}_{B} t'$ iff $\underline{t} \breve{\trianglelefteq}^{ml}_{B} \underline{t}$'.*

Proof. Consider an embedding goal $t \breve{\trianglelefteq}^{gd}_{B} t'$ where both t and t' are rooted by an associate (resp. an associative-commutative) symbol f. Let us assume that the flattened versions are $\underline{t} = f(t_1,\ldots,t_n)$ and $\underline{t}\text{'} = f(t'_1,\ldots,t'_m)$, with $m \geq n$. For the embedding goal $t \breve{\trianglelefteq}^{gd}_{B} t'$, the Coupling inference rule of Figure 1 and the Coupling$_A$ (resp. Coupling$_{AC}$) inference rule of Figure 5 can be applied to all of the elements in the equivalence class of t, i.e., the terms resulting from all possible rearrangements of t due to associativity (resp. associativity and commutativity). However, after several applications of the inference rules of $\breve{\trianglelefteq}^{gd}_{B}$, all of them end up with the form $t_i \breve{\trianglelefteq}^{gd}_{B} t'_j$ where $i \leq j$ (resp. $1 \leq i \leq n$ and $1 \leq j \leq m$). For the embedding goal $\underline{t} \breve{\trianglelefteq}^{ml}_{B} \underline{t}$', the Coupling$_A$ (resp. Coupling$_{AC}$) inference rule of Figure 6 can be applied and after several applications of the inference rules, all of them end up with the same form. □

Flattened terms are only accessible in Maude through its META-LEVEL functional module. In the following, we provide a characterization of the order-sorted homeomorphic embedding relation $\breve{\trianglelefteq}^{ml}_{B}$ for flattened terms by (i) using a set of equations instead of rules and (ii) using meta-representations of terms instead of explicit terms of the given signature Σ. Rewriting logic is reflective [8], in the sense that some commands of Maude at the user level can be represented at the object level in a consistent way. In other words, the meta-level representation correctly simulates the relevant metatheoretic features of Maude such as loading a module, evaluating a term, or computing the least sort of a term. Reflection is systematically used in the design and implementation of the Maude language, making the metatheory of rewriting logic accessible to the user in a clear, principled, and efficient way. In the sequel, a variable x of sort s is meta-represented as $\bar{x} =$ 'x:s and a non-variable term $t = f(t_1,\ldots,t_n)$, with $n \geq 0$, is meta-represented as $\bar{t} =$ '$f[\bar{t_1},\ldots,\bar{t_n}]$. See [7, Chapter 14] for further details.

Definition 10 (Meta-level (order-sorted) homeomorphic embedding modulo B). *The meta-level (order-sorted) homeomorphic embedding modulo B, $\breve{\trianglelefteq}^{ml}_{B}$, of Definition 9 is defined for* *term meta-representations by means of the equational theory E^{ml} given in Figure 7, where the auxiliary meta-level functions **any** and **all** implement the existential and universal tests in the Diving and Coupling inference rules of Figure 1, and we introduce two new meta-level functions **all_A** and **all_AC** that implement existential tests of Figure 6, which are specific to A and AC symbols. Note that the function **all_AC** uses an auxiliary function **all_AC_Aux** to search for an element in a list and we also introduce the function **remove(U,L)** that eliminates the element **U** from the list **L**.*

$$
\begin{array}{rll}
\sharp \ \breve{\trianglelefteq}_B^{ml}\ \sharp & = \textbf{true} & \textit{for each } \sharp \textit{ in } \Pi_{\mathscr{X}} \\
F[TermList]\ \breve{\trianglelefteq}_B^{ml}\ \sharp & = \textbf{false} & \text{if } F[TermList] \neq \sharp, \text{for } \sharp \in \Pi_{\mathscr{X}} \\
T\ \breve{\trianglelefteq}_B^{ml} F[TermList] & = \textbf{any}(T, TermList) & \text{if } root(T) \neq F \\
F[Term1]\ \breve{\trianglelefteq}_B^{ml} F[Term2] & = Term1 \breve{\trianglelefteq}_B^{ml} Term2 & \\
F[TermList1]\ \breve{\trianglelefteq}_B^{ml} F[TermList2] & = \textbf{any}(F[TermList1], TermList2) & \\
& \quad \textbf{or all}(TermList1, TermList2) & \text{if } length(TermList_i) \neq 1, \\
& & \quad i \in \{1,2\}, Ax(F) = \emptyset \\
F[U,V]\ \breve{\trianglelefteq}_B^{ml} F[X,Y] & = \textbf{any}(F[U,V],[X,Y]) & \text{if } Ax(F) = \{C\} \\
& \quad \textbf{or}(\ U \breve{\trianglelefteq}_B^{ml} X \textbf{ and } V \breve{\trianglelefteq}_B^{ml} Y\ ) & \\
& \quad \textbf{or}\ (\ U \breve{\trianglelefteq}_B^{ml} Y \textbf{ and } V \breve{\trianglelefteq}_B^{ml} X\ ) & \\
F[TermList1]\ \breve{\trianglelefteq}_B^{ml} F[TermList2] & = \textbf{any}(F[TermList1], TermList2) & \text{if } Ax(F) = \{A\} \\
& \quad \textbf{or all\_A}(TermList1, TermList2) & \\
F[TermList1]\ \breve{\trianglelefteq}_B^{ml} F[TermList2] & = \textbf{any}(F[TermList1], TermList2) & \text{if } Ax(F) = \{A, C\} \\
& \quad \textbf{or all\_AC}(TermList1, TermList2) &
\end{array}
$$

$$
\begin{array}{rl}
\textbf{any}(U, nil) & = \textbf{false} \\
\textbf{any}(U, V:L) & = U \breve{\trianglelefteq}_B^{ml} V \textbf{ or any}(U, L) \\[1ex]
\textbf{all}(nil, nil) & = \textbf{true} \\
\textbf{all}(nil, U:L) & = \textbf{false} \\
\textbf{all}(U:L, nil) & = \textbf{false} \\
\textbf{all}(U:L1, V:L2) & = U \breve{\trianglelefteq}_B^{ml} V \textbf{ and all}(L1, L2) \\[1ex]
\textbf{all\_A}(nil, L) & = \textbf{true} \\
\textbf{all\_A}(U:L, nil) & = \textbf{false} \\
\textbf{all\_A}(U:L1, V:L2) & = (U \breve{\trianglelefteq}_B^{ml} V \textbf{ and all\_A}(L1, L2)) \textbf{ or all\_A}(U:L1, L2)) \\[1ex]
\textbf{all\_AC}(nil, L) & = \textbf{true} \\
\textbf{all\_AC}(U:L1, L2) & = \textbf{all\_AC\_Aux}(U:L1, L2, L2) \\
\textbf{all\_AC\_Aux}(U:L1, nil, L3) & = \textbf{false} \\
\textbf{all\_AC\_Aux}(U:L1, V:L2, L3) & = (U \breve{\trianglelefteq}_B^{ml} V \textbf{ and all\_AC}(L1, \textbf{remove}(V, L3))) \\
& \quad\ \textbf{or all\_AC\_Aux}(U:L1, L2, L3)) \\[1ex]
\textbf{remove}(U, nil) & = \textbf{nil} \\
\textbf{remove}(U, V:L) & = \textbf{if } U = V \textbf{ then } L \textbf{ else } V : \textbf{remove}(U, L)
\end{array}
$$

Fig. 7. Meta-level order-sorted homeomorphic embedding modulo axioms

Example 11. Given the embedding problem for terms $+(1,+(2,3))$ and $+(+(4,2),+(3,1))$, the corresponding call to the meta-level homeomorphic embedding $\breve{\trianglelefteq}_B^{ml}$ of Definition 10 is $\text{'}+[\text{'}1, \text{'}2, \text{'}3] \breve{\trianglelefteq}_B^{ml} \text{'}+[\text{'}4, \text{'}2, \text{'}3, \text{'}1]$ and its evaluation sequence is given in Figure 8, according to the equational theory E^{ml} of Figure 7.

Proposition 5. *Given Σ and B, for terms t and t' in $\mathscr{T}_{\Sigma}(\mathscr{X})$, $t \breve{\trianglelefteq}_B^{gd} t'$ iff $(\bar{t} \breve{\trianglelefteq}_B^{ml} \bar{t'})!_{\overrightarrow{E^{ml}/B}} = true$.*

$$
\begin{aligned}
\text{'}{+}[\text{'}1,\text{'}2,\text{'}3]\,\breve{\trianglelefteq}^{ml}_{B}\,\text{'}{+}[\text{'}4,\text{'}2,\text{'}3,\text{'}1] \rightarrow_{E^{ml}/B}\; & \mathbf{any}(\text{'}{+}[\text{'}1,\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
& \mathbf{or} \\
& \mathbf{all\_AC}([\text{'}1,\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{false} \\
& \mathbf{or} \\
& \mathbf{all\_AC}([\text{'}1,\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow_{E^{ml}/B}\; & \mathbf{all\_AC}([\text{'}1,\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],[\text{'}1],[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow_{E^{ml}/B}\; & \text{'}1\,\breve{\trianglelefteq}^{ml}_{B}\,\text{'}1 \\
& \;\mathbf{and} \\
& \;\mathbf{all\_AC}([\text{'}2,\text{'}3],\mathbf{remove}(\text{'}1,[\text{'}4,\text{'}2,\text{'}3,\text{'}1])) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{all\_AC}([\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{all\_AC\_Aux}([\text{'}2,\text{'}3],[\text{'}2,\text{'}3],[\text{'}4,\text{'}2,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow_{E^{ml}/B}\; & \quad\text{'}2\,\breve{\trianglelefteq}^{ml}_{B}\,\text{'}2 \\
& \quad\mathbf{and} \\
& \quad\mathbf{all\_AC}([\text{'}3],\mathbf{remove}(\text{'}2,[\text{'}4,\text{'}2,\text{'}3])) \\
& \;\mathbf{or} \\
& \;\mathbf{all\_AC\_Aux}([\text{'}2,\text{'}3],[\text{'}3],[\text{'}4,\text{'}2,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{all\_AC}([\text{'}3],[\text{'}4,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}2,\text{'}3],[\text{'}3],[\text{'}4,\text{'}2,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \quad\text{'}3\,\breve{\trianglelefteq}^{ml}_{B}\,\text{'}3 \\
& \quad\mathbf{and} \\
& \quad\mathbf{all\_AC}(\mathbf{nil},\mathbf{remove}(\text{'}3,[\text{'}4,\text{'}3])) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}2,\text{'}3],[\text{'}3],[\text{'}4,\text{'}2,\text{'}3]) \\
& \mathbf{or} \\
& \mathbf{all\_AC\_Aux}([\text{'}1,\text{'}2,\text{'}3],nil,[\text{'}4,\text{'}2,\text{'}3,\text{'}1]) \\
\rightarrow^{*}_{E^{ml}/B}\; & \mathbf{true}
\end{aligned}
$$

Fig. 8. Evaluation sequence for $\text{'}{+}[\text{'}1,\text{'}2,\text{'}3]\,\breve{\trianglelefteq}^{ml}_{B}\,\text{'}{+}[\text{'}4,\text{'}2,\text{'}3,\text{'}1]$

Proof. Immediate by realizing that the inference rules of Definition 9 can be easily transformed into a rewrite theory $R^{ml}(\Sigma,B)$, as we did in Definition 8 for the inference rules of Definition 7. This rewrite theory is terminating and can be easily transformed into a set of rules that are confluent, terminating, and coherent modulo associativity and commutativity axioms. And then those rewrite rules can be straightforwardly translated into the equational theory E^{ml} of Definition 10 which manipulates flattened terms at the meta-level. □

A further optimized version of Definition 10 can be obtained by replacing the Boolean conjunction (*and*) and disjunction (*or*) operators with the computationally more efficient Maude Boolean operators `and-then` and `or-else` which avoid evaluating the second argument when the result of evaluating the first one suffices to compute the result.

Definition 11 (Strategic meta-level deterministic embedding modulo B). *We define $\breve{\trianglelefteq}_B^{sml}$ as the strategic version of relation $\breve{\trianglelefteq}_B^{ml}$ that is obtained by replacing the Boolean operators* `and` *and* `or` *with Maude's* `and-then` *operator for the* short-circuit *version of the conjunction and the* `or-else` *operator for the short-circuit version of the disjunction [7, Chapter 9.1], respectively.*

The optimization idea behind Definition 11 was first proposed in [3]. In the following section, we propose two novel optimizations of order-sorted homeomorphic embedding modulo equational axioms B that can achieve further speedups.

7 Optimizations based on the term B-ordering and reachable kinds

Typically hidden inside the B-matching algorithms, some pertinent term transformations allow terms that contain operators obeying equational axioms to be rewritten into convenient B-normal forms that facilitate the matching modulo B. In the case of AC-theories, these transformations not only include translating the term to flattened form (as shown in Section 6) but also allowing terms to be reordered and suitably parenthesized in order to enable subsequent rewrite steps. Basically, this is achieved by producing a single, auxiliary representative of their AC congruence class (i.e., the AC-normal form). An AC-normal form is typically generated by replacing nested occurrences of the same AC operator by a flattened argument list under a variadic symbol, sorting these arguments under a fixed linear ordering and combining equal arguments using multiplicity superscripts [12]. For example, the congruence class containing $f(f(\alpha, f(\beta,\alpha)), f(f(\gamma,\beta),\beta))$ where f is an AC symbol and subterms α, β, and γ belong to alien theories might be represented by $f^*(\alpha^2, \beta^3, \gamma)$, where f^* is a variadic symbol that replaces nested occurrences of f. A more formal account of this transformation is given in [13, 12]. For any set B of axioms, Maude's total ordering on terms is written $\preceq_B$. This ordering is closed under context, i.e., $t \preceq_B t' \Rightarrow C[t] \preceq_B C[t']$ for any context C, and size-compatible, i.e., $t \preceq_B t' \Rightarrow size(t) \leq size(t')$, where $size(t)$ is the number of symbols in the unflattened version of t.

Let us formulate a novel optimization $\breve{\trianglelefteq}_B^{order+sml}$ of the order-sorted homeomorphic embedding modulo B that takes advantage of Maude's total order on terms $\preceq_B$. The optimization is based on adding a pruning equation that negatively concludes the test whenever $T' \prec_B T$ without actually running the call $T \breve{\trianglelefteq}_B^{sml} T'$.

Definition 12 (Early pruning based on term order $\preceq_B$). *We extend the equational theory E^{ml} of Figure 7 (but with the strategic optimization given in Definition 11) with the following pruning equation that determines the (meta-level) embedding problem $T \breve{\trianglelefteq}_B^{order+sml} T'$ negatively whenever $T' \prec_B T$.*

$$T \breve{\trianglelefteq}_B^{order+sml} T' = \textbf{\textit{if}}\ T' \prec_B T\ \textbf{\textit{then false else}}\ T \breve{\trianglelefteq}_B^{sml} T'$$

The correctness of the above optimization derives from the following result.

Lemma 4. *If $T' \prec_B T$ then $T \breve{\trianglelefteq}_B T'$ does not hold.*

Proof. If $T' \prec_B T$, then $T \neq T'$. Since $T' \prec_B T$ is equivalent to $T' \preceq_B T$ and $T' \neq_B T$, and since $\preceq_B$ is size-compatible, then $T' \prec_B T$ implies $T \neq T'$ and $size(T') \leq size(T)$. Let us consider the cases when $size(T') = size(T)$ and $size(T') < size(T)$ separately. If $T \neq T'$ and $size(T') = size(T)$, then there is at least one clash position, i.e., a position in t and t' whose root symbol differ, hence the successive application of the coupling inference rule can never succeed, which implies $T \not\breve{\trianglelefteq}_B T'$.

For the case when $size(T') < size(T)$, the proof follows directly from the following facts: (i) B is any combination of associativity and commutativity axioms for the binary function symbols in Σ; (ii) every associativity or commutativity axiom $l = r$ in B is size-preserving, i.e., $size(l) = size(r)$; and (iii) $\trianglelefteq$ is size-compatible, i.e., $T \trianglelefteq T'$ implies that $size(T) \leq size(T')$. From (i), (ii) and (iii) it follows that $\breve{\trianglelefteq}_B$ is also size-compatible, i.e., $T \breve{\trianglelefteq}_B T'$ implies that $size(T) \leq size(T')$. Therefore, $size(T') < size(T)$ implies that $T \not\breve{\trianglelefteq}_B T'$ and the result follows.

□

Example 12. Consider the (flattened) terms $t = +(1,2,3)$ and $t' = +(0,1,2)$ for the signature of Example 4, and let T, T' be their corresponding meta-level representation. In Maude's total order for terms, $+(0,1,2) \prec_B +(1,2,3)$ (and correspondingly $T' \prec_B T$ as well). According to Definition 12, the embedding problem $T \breve{\trianglelefteq}_B^{order+sml} T'$ immediately fails.

Let us formalize a second optimization $\breve{\trianglelefteq}_B^{kinds+sml}$ of $\breve{\trianglelefteq}_B^{sml}$ that anticipates failure whenever there is a subterm of t whose kind is not in the set of all kinds that can be obtained (by instantiation) from the subterms of t'.

Definition 13 (Reachable kinds). *Given a term t, we let $kinds(t)$ denote the set of kinds of all of the subterms of t. Given the signature Σ, we compute $kinds(t)$ as the solution to a reachability problem using the following rewrite theory $Kinds(\Sigma) = (\Sigma', \emptyset, R)$, where Σ' contains the kinds of Σ as constants of a universal sort $\mathscr{U}$ and R consists of all rewrite rules*

$$k \to k_i$$

*where $(f : s_1,\ldots,s_n \to s)$ in Σ, $k = [s]$, $k_i = [s_i]$, $k \neq k_i$ and $i \in \{1,\ldots,n\}$. By using $\mathscr{R} = Kinds(\Sigma)$, we define **kinds**$(t) = \{k \mid \lceil t \rceil \to^*_{\mathscr{R}} k\}$.*

The intuition behind Definition 13 is that $kinds(t)$ represents the set of kinds of the subterms of any instance of t.

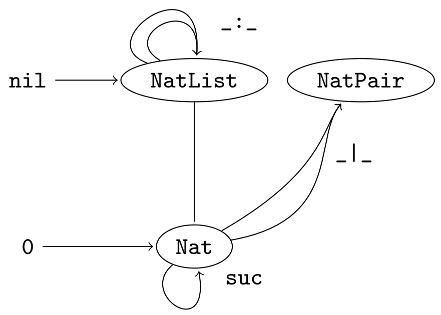

Fig. 9. Signature graph of Example 13

Example 13. Consider the following order-sorted signature Σ (graphically depicted in Figure 9), which extends the signature of natural numbers with two new sorts `NatPair` and `NatList`, which are aimed at representing pairs of natural numbers and sequences of natural numbers, respectively. Pairs are constructed by using the pair concatenation operator `_|_`, whereas lists are constructed by using `nil` and the (associative) list concatenation operator `_:_`. Note that there are two different kinds, `[NatPair]` and `[NatList]`, where `[NatList]` contains the sorts `Nat` and `NatList` because `Nat < NatList`.

```
sorts Nat NatPair NatList .
subsort  Nat < NatList .
op 0 : -> Nat .
op suc : Nat -> Nat .
op _|_ : Nat Nat -> NatPair .
op nil -> NatList .
op _:_ : NatList NatList -> NatList [assoc] .
```

The rewrite theory $Kinds(\Sigma)$ consists of:

```
sort U .
ops [NatList] [NatPair] : -> U .
rl [NatPair] => [NatList] .
```

Now, given the term Z:`NatList`, we have $kinds(Z{:}\texttt{NatList}) = \{\texttt{[NatList]}\}$, whereas given the term X:`Nat` | Y:`Nat`, we have $kinds(X{:}\texttt{Nat} \mid Y{:}\texttt{Nat}) = \{\texttt{[NatPair]}, \texttt{[NatList]}\}$.

Definition 14 (Early pruning based on reachable kinds). *We extend the equational theory E^{ml} of Figure 7 (with the strategic optimization given in Definition 11) with the following pruning equation that decides the (meta-level) embedding problem $T \breve{\trianglelefteq}_B^{kinds+sml} T'$ negatively whenever the set of all reachable kinds from T' does not contain all of the reachable kinds from T.*

$$T \breve{\trianglelefteq}_B^{kinds+sml} T' = \textbf{\textit{if}}\ kinds(T) \not\subseteq kinds(T')\ \textbf{\textit{then false else}}\ T \breve{\trianglelefteq}_B^{sml} T'$$

Example 14. Consider again the order-sorted signature Σ for pairs and sequences of natural numbers of Example 13. The embedding goal $(X{:}\texttt{Nat} \mid Y{:}\texttt{Nat}) \breve{\trianglelefteq}_B Z{:}\texttt{NatList}$ does not hold because the kind `[NatPair]` of the left term $(X{:}\texttt{Nat} \mid Y{:}\texttt{Nat})$ is not contained in the set $kinds(Z{:}\texttt{NatList})$ of reachable kinds from the right term, which only contains `[NatList]`. However, for the inverse goal we cannot conclude $Z{:}\texttt{NatList} \not\breve{\trianglelefteq}_B (X{:}\texttt{Nat} \mid Y{:}\texttt{Nat})$ because the set $kinds(X{:}\texttt{Nat} \mid Y{:}\texttt{Nat}) = \{\texttt{[NatPair]}, \texttt{[NatList]}\}$ does contain the only reachable kind `[NatList]` for the left term Z:`NatList`. And actually, $Z{:}\texttt{NatList} \breve{\trianglelefteq}_B (X{:}\texttt{Nat} \mid Y{:}\texttt{Nat})$ holds according to Definition 5 because $Z{:}\texttt{NatList} \breve{\trianglelefteq}_B$ `X:Nat`.

The correctness of the above optimization derives from the following result.

Lemma 5. *If $kinds(T) \not\subseteq kinds(T')$, then $T \breve{\trianglelefteq}_B T'$ does not hold.*

Proof. (By contradiction). Assume that there is a subterm t of T such that $\lceil t \rceil \notin kinds(T')$ and that $T \breve{\trianglelefteq}_B T'$ holds. Since $T \breve{\trianglelefteq}_B T'$ then also $t \breve{\trianglelefteq}_B T'$. Let us consider separately the case when t is ground and when t is non-ground. By Definition of $\breve{\trianglelefteq}_B$, if t is ground, then every symbol of t must appear in T', hence it cannot happen that $\lceil t \rceil \notin kinds(T')$. If t is non-ground then every non-variable symbol of t must appear in T' and for each variable X of t there must be a variable Y in T' such that $\lceil X \rceil = \lceil Y \rceil$; hence, it cannot happen that $\lceil t \rceil \notin kinds(T')$. □

In the following section, we demonstrate that the two optimizations $\breve{\trianglelefteq}_B^{order+sml}$ and $\breve{\trianglelefteq}_B^{kinds+sml}$ pay off in practice. We consider the two optimizations $\breve{\trianglelefteq}_B^{order+sml}$ and $\breve{\trianglelefteq}_B^{kinds+sml}$ separately as well as their natural combination in our last optimal order-sorted equation embedding relation $\breve{\trianglelefteq}_B^{kosml}$ that (lazily) first checks the pruning condition based on reachable kinds of Definition 14 and then checks the pruning condition based on the term order $\preceq_B$ of Definition 12. In other words,

Definition 15 (Early pruning based on combining term order $\preceq_B$ and reachable kinds). *We extend the equational theory E^{ml} of Figure 7 (with the strategic optimization given in Definition 11) with the following pruning equation that combines the pruning based on term order $\preceq_B$ and the pruning based on reachable kinds.*

$$T \breve{\trianglelefteq}_B^{kosml} T' = \textbf{\textit{if}}\ kinds(T) \not\subseteq kinds(T')\ \textbf{\textit{or-else}}\ T' \prec_B T\ \textbf{\textit{then false else}}\ T \breve{\trianglelefteq}_B^{sml} T'$$

Benchmark	$\sharp$ Axioms				$\breve{\trianglelefteq}_B$			$\breve{\trianglelefteq}_B^{rbgd}$			$\breve{\trianglelefteq}_B^{ml}$, $\breve{\trianglelefteq}_B^{sml}$, $\breve{\trianglelefteq}_B^{kosml}$		
	$\emptyset$	A	C	AC	$\sharp E$	$\sharp R$	GT(ms)	$\sharp E$	$\sharp R$	GT(ms)	$\sharp E$	$\sharp R$	GT(ms)
Kmp	9	0	0	0	0	15	1	0	57	2	21	0	0
NatList	5	1	1	2	0	10	1	0	26	1	21	0	0
Maze	5	1	0	1	0	36	7	0	787	15	21	0	0
Dekker	16	1	0	2	0	59	8	0	823	18	21	0	0

Table 1. Size of generated theories for naïve and goal-driven definitions vs. meta-level definitions

Note that we do not implement a recursive optimization that tries to verify the pruning conditions recursively within $\breve{\trianglelefteq}_B^{sml}$. The reason is that the recursive checking of the pruning conditions would introduce a dramatic, exponential overhead so that the recursive optimization would prove ineffective.

8 Experiments

We have implemented in Maude all five equational homeomorphic embedding formulations $\breve{\trianglelefteq}_B$ (Definition 6), $\breve{\trianglelefteq}_B^{rbgd}$ (Definition 8), $\breve{\trianglelefteq}_B^{ml}$ (Definition 10), $\breve{\trianglelefteq}_B^{sml}$ (Definition 11), and $\breve{\trianglelefteq}_B^{kosml}$ (Definition 15) of the previous sections. The implementation consists of approximately 2.5K lines of Maude source code and is publicly available[9]. In this section, we provide an experimental comparison of the five equational homeomorphic embedding implementations by running a significant number of equational embedding goals. In order to compare the performance of the different implementations in the worst possible scenario, all benchmarked goals return false, which ensures that the whole search space for each goal has been completely explored, while the execution times for succeeding goals whimsically depend on the particular node of the search tree where success is found.

We tested our implementations on a 3.3GHz Intel Xeon E5-1660 with 64 GB of RAM running Maude v2.7.1, and we considered the average of ten executions for each test. We have chosen four representative programs: (i) *KMP*, the classical KMP string pattern matcher [4]; (ii) *NatList*, a Maude implementation of lists of natural numbers; (iii) *Maze*, a non-deterministic Maude specification that defines a maze game in which multiple players must reach a given exit point by walking or jumping, where colliding players are eliminated from the game [1]; and (iv) *Dekker*, a Maude specification that models a faulty version of Dekker's protocol, one of the earliest solutions to the mutual exclusion problem that appeared in [7]. As testing benchmarks, we considered a set of representative embeddability problems for the four programs that are generated during the execution of Maude's partial evaluator, Victoria [2].

Tables 1, 2, 3, and 4 below analyze different aspects of the implementation. In Table 1, we compare the size of the generated rewrite theories for the naïve and the goal-driven definitions versus the meta-level definitions. For $\breve{\trianglelefteq}_B^{ml}$, $\breve{\trianglelefteq}_B^{sml}$, and $\breve{\trianglelefteq}_B^{kosml}$, there are the same number (21) of generated equations ($\sharp$E), whereas the number of generated rules ($\sharp$R) is zero because both definitions are purely equational (deterministic) and just differ in the version of the Boolean operators being used. As for the generated rewrite theories for computing $\breve{\trianglelefteq}_B$ and $\breve{\trianglelefteq}_B^{rbgd}$, they contain no equations, while the number of generated rules increases with the complexity of the program (that heavily depends on the equational axioms that the function symbols obey). The number of generated rules is much bigger for $\breve{\trianglelefteq}_B^{rbgd}$ than for $\breve{\trianglelefteq}_B$ (for instance, $\breve{\trianglelefteq}_B^{rbgd}$ is encoded by 823 rules for the Dekker program versus the 59 rules of $\breve{\trianglelefteq}_B$). Columns $\emptyset$, A, C, and AC summarize the number of free, associative, commutative, and associative-commutative symbols, respectively, for each benchmark program. The generation times (GT) are negligible for all rewrite theories.

[9] At `http://safe-tools.dsic.upv.es/victoria/jsp-pages/embedding.jsp`.

To make the comparison on equal terms, in Tables 2 and 3, we compare the equational embedding implementations $\check{\trianglelefteq}_B$, $\check{\trianglelefteq}_B^{rbgd}$, $\check{\trianglelefteq}_B^{ml}$, and $\check{\trianglelefteq}_B^{sml}$ without adding the pruning conditions of Section 7 as they could be applicable to any of them. Our figures demonstrate that the implementation of $\check{\trianglelefteq}_B^{sml}$ is the most efficient one among the four formulations. The performance improvement of the two optimizations of Section 7 w.r.t. $\check{\trianglelefteq}_B^{sml}$ is evaluated in Table 4 below.

For all benchmarks $T1 \trianglelefteq_B^{\alpha} T2$ in Table 2, we have fixed the size of $T1$, which is measured in the depth of (the non-flattened version of) the term, to five. As for $T2$, we have considered terms with increasing depths: 5, 10, 100, 500, 1000, and 5000. The ♯ `Symbols` column records the number of A (resp. AC) symbols occurring in the benchmarked goals.

Benchmark	**♯ Symbols**		**Size**		$\check{\trianglelefteq}_B$	$\check{\trianglelefteq}_B^{rbgd}$	$\check{\trianglelefteq}_B^{ml}$	$\check{\trianglelefteq}_B^{sml}$
	A	**AC**	**T1**	**T2**	**Time(ms)**	**Time(ms)**	**Time(ms)**	**Time(ms)**
Kmp	0	0	5	5	10	6	1	1
				10	150	125	4	1
				100	TO	TO	280	95
				500	TO	TO	714	460
				1000	TO	TO	2184	1324
				5000	TO	TO	7528	5152
NatList	1	2	5	5	2508	2892	1	1
				10	840310	640540	1	1
				100	TO	TO	8	2
				500	TO	TO	60	5
				1000	TO	TO	102	10
				5000	TO	TO	210	85
Maze	1	1	5	5	40	25	1	1
				10	TO	20790	4	1
				100	TO	TO	256	2
				500	TO	TO	1980	10
				1000	TO	TO	10148	20
				5000	TO	TO	1057912	46
Dekker	1	1	5	5	50	40	1	1
				10	111468	110517	2	1
				100	TO	TO	5	3
				500	TO	TO	20	13
				1000	TO	TO	45	20
				5000	TO	TO	80	38

Table 2. Performance of equational homeomorphic embedding implementations w.r.t. problem size

The figures in Table 2 confirm our expectations regarding $\check{\trianglelefteq}_B$ and $\check{\trianglelefteq}_B^{rbgd}$ that the search space is huge and increases exponentially with the size of $T2$ (discussed for $\check{\trianglelefteq}_B$ in Example 8 and for $\check{\trianglelefteq}_B^{rbgd}$ in Example 9). Actually, when the size of $T2$ is 100 (and beyond), a given timeout (represented by TO in the tables) is reached that is set for 3.6e+6 milliseconds (1 h). The reader can also check that the more A,C, and AC symbols occur in the original program signature, the bigger the execution times. An odd exception is the Maze example, where the timeout is already reached for the size 10 of $T2$ even if the number of equational axioms is comparable to the other programs. This is because the AC-normalized, flattened version of the terms is much smaller than the original term size for the NatList and Dekker benchmarks but not for Maze, where the flattened and original terms have similar size. On the other hand, our experiments demonstrate that both $\check{\trianglelefteq}_B^{ml}$ and $\check{\trianglelefteq}_B^{sml}$ bring impressive speedups, with $\check{\trianglelefteq}_B^{sml}$ working outstandingly well in practice even for really complex terms where, taking into account

T1						T2						$\breve{\trianglelefteq}_B$	$\breve{\trianglelefteq}_B^{rbgd}$	$\breve{\trianglelefteq}_B^{ml}$	$\breve{\trianglelefteq}_B^{sml}$
Size		**♯ Symbols**				**Size**		**♯ Symbols**				**Time(ms)**	**Time(ms)**	**Time(ms)**	**Time(ms)**
OT	**FT**	$\emptyset$	**C**	**A**	**AC**	**OT**	**FT**	$\emptyset$	**C**	**A**	**AC**				
5	5	5	0	0	0	100	100	100	0	0	0	165	70	1	1
5	5	3	2	0	0	100	100	50	50	0	0	TO	TO	24	1
5	2	4	0	1	0	100	2	50	0	50	0	TO	TO	108035	3
5	2	4	0	0	1	100	2	50	0	0	50	TO	TO	42800	4
5	3	8	0	1	2	100	3	50	0	25	25	TO	TO	22796	5
5	5	5	0	0	0	500	500	500	0	0	0	48339	34000	12	4
5	5	3	2	0	0	500	500	250	250	0	0	TO	TO	1360	10
5	2	4	0	1	0	500	2	250	0	250	0	TO	TO	TO	30
5	2	4	0	0	1	500	2	250	0	0	250	TO	TO	TO	27
5	3	8	0	1	2	500	3	250	0	125	125	TO	TO	TO	50
5	5	5	0	0	0	1000	1000	1000	0	0	0	202224	88000	18	8
5	5	3	2	0	0	1000	1000	500	500	0	0	TO	TO	10184	40
5	2	4	0	1	0	1000	2	500	0	500	0	TO	TO	TO	50
5	2	4	0	0	1	1000	2	500	0	0	500	TO	TO	TO	56
5	3	8	0	1	2	1000	3	500	0	250	250	TO	TO	TO	87
5	5	5	0	0	0	5000	5000	1000	0	0	0	TO	TO	27	15
5	5	3	2	0	0	5000	5000	2500	2500	0	0	TO	TO	1159236	80
5	2	4	0	1	0	5000	2	2500	0	2500	0	TO	TO	TO	114
5	2	4	0	0	1	5000	2	2500	0	0	2500	TO	TO	TO	240
5	3	8	0	1	2	5000	3	2500	0	1250	1250	TO	TO	TO	368

Table 3. Performance of equational homeomorphic embedding implementations w.r.t. axiom entanglement for the NatList example

the generous one hour time-out, the achieved speedup is at least 10^5 (and in Table 4 is further improved by one additional order of magnitude by enabling the optimizations of Section 7).

The reader may wonder how big is the impact of having A, C, or AC operators. In order to compare the relevance of these symbols, in Table 3 we fix one single benchmark program (NatList) that contains all three kinds of operators: two associative operators (list concatenation ; and natural division $/$), a commutative (natural pairing) operator ($||$), and two associative-commutative arithmetic operators ($+, *$). With regard to the size of the considered terms, we compare the size of the original term (OT) with the size of its flattened version (FT); e.g., 500 versus 2 for the size of $T2$ in the last row. We have included the execution times of $\breve{\trianglelefteq}_B$ and $\breve{\trianglelefteq}_B^{rbgd}$ for completeness, but they do not reveal a dramatic improvement of $\breve{\trianglelefteq}_B^{rbgd}$ with respect to $\breve{\trianglelefteq}_B$ for the benchmarked (false) goals, contrary to what we initially expected. This means that $\breve{\trianglelefteq}_B^{rbgd}$ cannot be generally used in real applications due to the risk of intolerable embedding test times, even if $\breve{\trianglelefteq}_B^{rbgd}$ may be far less wasteful than $\breve{\trianglelefteq}_B$ for succeeding goals, as discussed in Section 5. For $\breve{\trianglelefteq}_B^{ml}$ and $\breve{\trianglelefteq}_B^{sml}$, the figures show that the more A and AC operators comparatively occur in the problem, the bigger the improvement achieved. This is due to the following: (i) these two embedding definitions manipulate flattened meta-level terms; (ii) they are equationally defined, which has a much better performance in Maude than doing search; and (iii) our definitions are highly optimized for lists (that obey associativity) and sets (that obey both associativity and commutativity).

Homeomorphic embedding has been extensively used in Prolog for different purposes, such as termination analysis and partial deduction. In Figure 10, we have compared on a log-

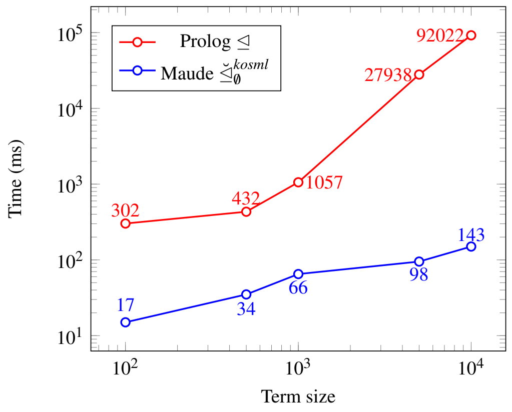

Fig. 10. Comparison of $\trianglelefteq$ in Prolog vs. $\check{\trianglelefteq}_{\emptyset}^{kosml}$ for the NatList example (no axioms in goals)

arithmic scale our best embedding definition[10], $\check{\trianglelefteq}_B^{kosml}$, with a standard meta-level Prolog[11] implementation of the (syntactic) pure homeomorphic embedding $\trianglelefteq$ of Definition 3. We chose the NatList example and terms $T1$ and $T2$ which do not contain symbols obeying equational axioms as this is the only case that can be handled by the syntatic Prolog implementation. Our experiments show that our refined deterministic formulation $\check{\trianglelefteq}_B^{kosml}$ (i.e. without search) outperforms the Prolog version, so no penalty is incurred (actually, performance is increased) when syntactic embeddability tests are run in our equational implementation.

T1	T2	Case	$\check{\trianglelefteq}_B^{sml}$	$\check{\trianglelefteq}_B^{kinds+sml}$	$\check{\trianglelefteq}_B^{order+sml}$	$\check{\trianglelefteq}_B^{kosml}$
Size	**Size**		**Time(ms)**	**Time(ms)**	**Time(ms)**	**Time(ms)**
1000	1000	$kinds(T1) \not\subseteq kinds(T2)$ & $(T2 \not\prec_B T1)$	12	2	15	2
5000	5000	$kinds(T1) \not\subseteq kinds(T2)$ & $(T2 \not\prec_B T1)$	80	10	105	11
10000	10000	$kinds(T1) \not\subseteq kinds(T2)$ & $(T2 \not\prec_B T1)$	240	15	270	18
50000	50000	$kinds(T1) \not\subseteq kinds(T2)$ & $(T2 \not\prec_B T1)$	550	28	580	33
1000	1000	$kinds(T1) \subseteq kinds(T2)$ & $(T2 \prec_B T1)$	10	12	4	6
5000	5000	$kinds(T1) \subseteq kinds(T2)$ & $(T2 \prec_B T1)$	40	50	8	12
10000	10000	$kinds(T1) \subseteq kinds(T2)$ & $(T2 \prec_B T1)$	140	160	14	20
50000	50000	$kinds(T1) \subseteq kinds(T2)$ & $(T2 \prec_B T1)$	290	330	40	60

Table 4. Comparison of $\check{\trianglelefteq}_B^{sml}$ vs. the optimizations $\check{\trianglelefteq}_B^{kinds+sml}$, $\check{\trianglelefteq}_B^{order+sml}$, and $\check{\trianglelefteq}_B^{kosml}$

Finally, Table 4 evaluates the two optimizations $\check{\trianglelefteq}_B^{order+sml}$ and $\check{\trianglelefteq}_B^{kinds+sml}$ separately as well as their combination with respect to our best implementation of order-sorted equational embedding, $\check{\trianglelefteq}_B^{kosml}$. In order to identify the optimization that achieves the best improvement, we have chosen eight pairs of terms $T1$ and $T2$ of (increasingly) equal size (1000, 5000, 10000, and 50000) and such that the embedding test fails because either (i) $kinds(T1) \not\subseteq kinds(T2)$ although $T2 \not\prec_B T1$ or (ii) $kinds(T1) \subseteq kinds(T2)$ but $T2 \prec_B T1$. In case (i), the optimization $\check{\trianglelefteq}_B^{kinds+sml}$ is enabled and the failure is anticipated without running $\check{\trianglelefteq}_B^{sml}$, whereas $\check{\trianglelefteq}_B^{order+sml}$ fruitlessly checks its pruning condition $T2 \prec_B T1$ and then runs $\check{\trianglelefteq}_B^{sml}$ anyway. In case (ii), the optimization $\check{\trianglelefteq}_B^{kinds+sml}$ fruitlessly checks its pruning condition $kinds(T1) \not\subseteq kinds(T2)$

[10] For this comparison, note that $\check{\trianglelefteq}_B^{kosml}$ boils down to $\check{\trianglelefteq}_B^{sml}$ since Prolog terms are unsorted and functors obey no axioms, hence no optimization based on sorts or term order is applicable.

[11] To avoid any bias, we took the Prolog code for the homeomorphic embedding of the ECCE system [23] that is available at `https://github.com/leuschel/ecce`, and we run it in SWI-Prolog 7.6.3.

and then runs $\breve{\trianglelefteq}_B^{sml}$ anyway, whereas the optimization $\breve{\trianglelefteq}_B^{order+sml}$ is enabled and the failure is anticipated without running $\breve{\trianglelefteq}_B^{sml}$. Our figures reveal that the penalty that is incurred by checking the pruning condition of $\breve{\trianglelefteq}_B^{order+sml}$ is slightly worse than for $\breve{\trianglelefteq}_B^{kinds+sml}$, with the difference being more evident as terms grow (e.g., it takes twice as long for the term size of 5000: 580 ms in $\breve{\trianglelefteq}_B^{order+sml}$ versus 330 ms in $\breve{\trianglelefteq}_B^{kinds+sml}$). This justifies our final choice to first check $kinds(T1) \not\subseteq kinds(T2)$ and then eventually $T2 \prec_B T1$ in the definition of the optimal embedding relation formulation, $\breve{\trianglelefteq}_B^{kosml}$, which achieves the best overall speedup.

9 Concluding remarks

Homeomorphic embedding has been extensively used in Prolog, but it has never been investigated in the context of expressive rule-based languages like Maude, CafeOBJ, OBJ, ASF+SDF, and ELAN that support symbolic reasoning methods modulo equational axioms. We have introduced a new equational and order-sorted definition of homeomorphic embedding with a remarkably good performance for theories with symbols satisfying any combination of associativity and/or commutativity axioms. We have also compared different definitions of embedding, identifying some key conclusions: (i) definitions of equational homeomorphic embedding based on (non-deterministic) search in Maude perform dramatically worse than their equational counterparts and are not feasible in practice; (ii) definitions of equational homeomorphic embedding based on generated theories perform dramatically worse than meta-level definitions; and (iii) the flattened meta-representation of terms is crucial for homeomorphic embedding definitions dealing with A and AC operators in order to pay off in practice. As future work, we plan to extend our results to the case when the equational theory B may contain the identity axiom, which is non-trivial since B is not class-finite.

References

1. M. Alpuente, D. Ballis, F. Frechina, and J. Sapiña. Exploring Conditional Rewriting Logic Computations. *Journal of Symbolic Compututation*, 69:3–39, 2015.
2. M. Alpuente, A. Cuenca-Ortega, S. Escobar, and J. Meseguer. Partial Evaluation of Order-Sorted Equational Programs Modulo Axioms. In *Proc. of 26th Int'l Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2016*, volume 10184 of *LNCS*, pages 3–20. Springer, 2017.
3. M. Alpuente, A. Cuenca-Ortega, S. Escobar, and J. Meseguer. Homeomorphic embedding modulo combinations of associativity and commutativity axioms. In Fred Mesnard and Peter J. Stuckey, editors, *Logic-Based Program Synthesis and Transformation - 28th International Symposium, LOPSTR 2018, Frankfurt/Main, Germany, September 4-6, 2018, Revised Selected Papers*, volume 11408 of *Lecture Notes in Computer Science*, pages 38–55. Springer, 2018.
4. M. Alpuente, M. Falaschi, and G. Vidal. Partial Evaluation of Functional Logic Programs. *ACM TOPLAS*, 20(4):768–844, 1998.
5. A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and Proof in Membership Equational Logic. *Theor. Comput. Sci.*, 236(1-2):35–132, 2000.
6. H.J. Bürckert, A. Herold, and M. Schmidt-Schauß. On Equational Theories, Unification, and (Un)decidability. *Journal of Symbolic Computation*, 8(1–2):3–49, 1989.
7. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer-Verlag, 2007.
8. M. Clavel and J. Meseguer. Reflection in conditional rewriting logic. *Theoretical Computer Science*, 285(2):245–288, 2002.
9. N . Dershowitz. A Note on Simplification Orderings. *Information Processing Letters*, 9(5):212–215, 1979.
10. N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 243–320. Elsevier, Amsterdam, 1990.
11. S. Eker. Single Elementary Associative-Commutative Matching. *J. Autom. Reasoning*, 28(1):35–51, 2002.

12. S. Eker. Associative-Commutative Rewriting on Large Terms. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Proceedings*, volume 2706 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2003.
13. S. M. Eker. Associative-commutative matching via bipartite graph matching. *The Computer Journal*, 38(5):381, 1995.
14. S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. *J. Log. Algebr. Program.*, 81(7-8):898–928, 2012.
15. M. Fay. First Order Unification in an Equational Theory. In *Proc of 4th Int'l Conf. on Automated Deduction, CADE'79*, pages 161–167, 1979.
16. H. Garavel, M. Tabikh, and I. Arrada. Benchmarking implementations of term rewriting and pattern matching in algebraic, functional, and object-oriented languages - the 4th rewrite engines competition. In Vlad Rusu, editor, *Rewriting Logic and Its Applications - 12th International Workshop, WRLA 2018, Held as a Satellite Event of ETAPS, Thessaloniki, Greece, June 14-15, 2018, Proceedings*, volume 11152 of *Lecture Notes in Computer Science*, pages 1–25. Springer, 2018.
17. J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.
18. J. B. Kruskal. The Theory of Well-Quasi-Ordering: A Frequently Discovered Concept. *J. Comb. Theory, Ser. A*, 13(3):297–305, 1972.
19. J.B. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95:210–225, 1960.
20. M. Leuschel. *Advanced Techniques for Logic Program Specialisation*. PhD thesis, 1997.
21. M. Leuschel. On the Power of Homeomorphic Embedding for Online Termination. In G. Levi, editor, *Proc. of 5th International Symposium on Static Analysis, SAS'98*, volume 1503 of *LNCS*, pages 230–245. Springer, 1998.
22. M. Leuschel. Homeomorphic Embedding for Online Termination of Symbolic Methods. In T. Æ. Mogensen, D. A. Schmidt, and I. Hal Sudborough, editors, *The Essence of Computation, Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones on occasion of his 60th birthday)*, volume 2566 of *LNCS*, pages 379–403. Springer, 2002.
23. M. Leuschel, B. Martens, and D. De Schreye. Controlling Generalization and Polyvariance in Partial Deduction of Normal Logic Programs. *ACM TOPLAS*, 20(1):208–258, 1998.
24. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
25. J. Meseguer. Order-Sorted Rewriting and Congruence Closure. In Bart Jacobs and Christof Löding, editors, *Foundations of Software Science and Computation Structures - 19th International Conference, FOSSACS 2016*, volume 9634 of *Lecture Notes in Computer Science*, pages 493–509. Springer, 2016.
26. J. Meseguer. Strict Coherence of Conditional Rewriting Modulo Axioms. *Theor. Comput. Sci.*, 672:1–35, 2017.
27. A. Middeldorp and B. Gramlich. Simple Termination is Difficult. *Applicable Algebra in Engineering, Communication and Computing*, 6(2):115–128, 1995.
28. M.H. Sørensen and R. Glück. An Algorithm of Generalization in Positive Supercompilation. In J.W. Lloyd, editor, *Proc. of International Symposium on Logic Programming, ILPS'95*, pages 465–479. MIT Press, 1995.